北京理工大学
数字图像处理项目作业报告

基于 SIFT 的全景拼接技术的研究与实现

专　　业：计算机科学与技术

学生姓名：穆凯代斯·木合塔尔

班　　号：07111506

学　　号：1120151909

完成时间：2018 年 2 月 2 日

目录

1 项目简介

1.1 项目内容

1.1.1 题目要求

Implement a panorama maker:

1. Given several overlapped images;
2. Detect feature point in each imag;
3. Find corresponding pairs;
4. Use those pairs to align images;
5. Stitching the images together, to make a panorama;

实现一个全景图生成器：

1. 输入一系列部分重叠的图像；
2. 检测每张图的特征点；
3. 找出各对应的匹配点对；
4. 利用点对使各图片成同一线；
5. 将对应的图片拼接在一起，形成全景图。

1.1.2 全景拼接简介

全景拼接是计算机视觉领域取得的一项早期成就。在 2007 年，Brown 和 Lowe 发表了著名的图像拼接论文。自打那以后，自动全景拼接技术受到广泛应用，例如 Google 街景地图、智能手机上的全景照片、拼接软件比如 Photosynth 和 AutoStitch。

在机器视觉应用领域里，特征检测和匹配是一个很重要的算法，比如图像配准、跟踪和目标检测。在本项目中，也是用基于特征点的方法，从多幅图像中匹配 SIF 关键点，完成单幅的全景图。

1.2 项目背景

本项目是在斯坦福大学计算机视觉专业课程作业的 SIFT 模板的基础上，完成主体程序内容实现的。参考文献：Stanford University CS 131 Computer Vision(Foundations and Applications) PA1

2 实现方案

2.1 解题思路

2.1.1 整体流程

在本项目中，全景拼接的实现程序基于 SIFT 算法，多幅图像中匹配 SIFT 关键点，来构建单幅全景图片。其整体流程可表示为：

（1）使用高斯差分（DoG）检测器找关键点，返回它的坐标位置和尺度。

（2）给一副图像的每个关键点构建 SIFT 描述子。

（3）从两幅不同的图像中比较两组 SIFT 描述子，找到匹配点。

（4）给定一个关键点匹配的列表，使用最小二乘法找到仿射变换矩阵，这个矩阵能将 iamge1 上的位置映射到 image2 上的位置。

（5）使用 RANSAC 使仿射变换矩阵的估计具有更好的鲁棒性。

（6）给定变换矩阵，使用它变换（平移、尺度、或者倾斜）image1，将它覆盖到 image2 上面，构建一个全景图。

（7）完成全景拼接。在真实世界场景中的特定例子中，把多幅图像拼接在一起。

2.1.2 SIF 算法简述

尺度不变特征转换(Scale-invariant feature transform 或 SIFT)是一种电脑视觉的算法用来侦测与描述影像中的局部性特征，它在空间尺度中寻找极值点，并提取出其位置、尺度、旋转不变量，此算法由 David Lowe 在 1999 年所发表，2004

年完善总结。其应用范围包含物体辨识、机器人地图感知与导航、影像缝合、3D 模型建立、手势辨识、影像追踪和动作比对。

局部影像特征的描述与侦测可以帮助辨识物体，SIFT 特征是基于物体上的一些局部外观的兴趣点而与影像的大小和旋转无关。对于光线、噪声、些微视角改变的容忍度也相当高。基于这些特性，它们是高度显著而且相对容易撷取，在母数庞大的特征数据库中，很容易辨识物体而且鲜有误认。使用 SIFT 特征描述对于部分物体遮蔽的侦测率也相当高，甚至只需要 3 个以上的 SIFT 物体特征就足以计算出位置与方位。在现今的电脑硬件速度下和小型的特征数据库条件下，辨识速度可接近即时运算。SIFT 特征的信息量大，适合在海量数据库中快速准确匹配。

SIFT 特征对旋转、尺度缩放、亮度变化等保持不变性，是非常稳定的局部特征，现在应用很广泛。而 SIFT 算法是将 Blob 检测，特征矢量生成，特征匹配搜索等步骤结合在一起优化。其特点有：①SIFT 特征是图像的局部特征，其对旋转、尺度缩放、亮度变化保持不变性，对视角变化、仿射变换、噪声也保持一定程度的稳定性；②独特性，信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配；③ 多量性，即使少数的几个物体也可以产生大量的 SIFT 特征向量；④高速性，经优化的 SIFT 匹配算法甚至可以达到实时的要求；⑤可扩展性，可以很方便的与其他形式的特征向量进行联合。

2.2 各模块简述

根据全景拼接实现的整体流程，可将该项目分为多个不同的模块，分工完成各个函数，再最后合作实现拼接。

2.2.1 DOG 定位关键点

该模块在文件 SIFTDescriptor.m 中实现。

DoG 其实是对高斯拉普拉斯 LoG 的近似，也就是对$\sigma^2\Delta^2G$的近似。SIFT 算法建议，在某一尺度上的特征检测可以通过对两个相邻高斯尺度空间的图像相减，

得到 DoG 的响应值图像 D(x,y,σ)。然后仿照 LoG 方法，通过对响应值图像 D(x,y,σ)进行局部最大值搜索，在空间位置和尺度空间定位局部特征点。其中：

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))\otimes I(x,y)$$
$$=L(x,y,k\sigma)-L(x,y,\sigma)$$

k 为相邻两个尺度空间倍数的常数。

为了得到 DoG 图像，先要构造高斯金字塔。高斯金字塔在多分辨率金字塔简单降采样基础上加了高斯滤波，也就是对金字塔每层图像用不同参数的σ做高斯模糊，使得每层金字塔有多张高斯模糊图像。金字塔每层多张图像合称为一组（Octave），每组有多张（也叫层 Interval）图像。降采样时，金字塔上边一组图像的第一张图像（最底层的一张）是由前一组（金字塔下面一组）图像的倒数第三张隔点采样得到。高斯尺度空间金字塔中每组有五层不同尺度图像，相邻两层相减得到四层 DoG 结果。关键点搜索就在这四层 DoG 图像上寻找局部极值点。寻找 DoG 极值点时，每一个像素点和它所有的相邻点比较，当其大于（或小于）它的图像域和尺度域的所有相邻点时，即为极值点。

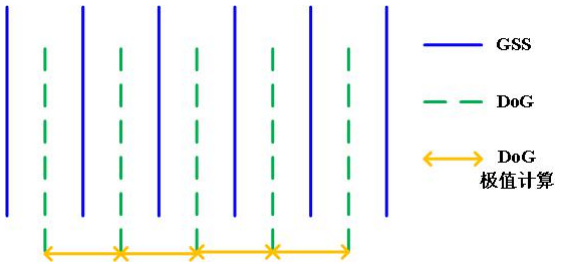


图 2-1 高斯金字塔，DoG 图像及极值计算的相互关系

极值点的搜索是在离散空间进行搜索的，但是在离散空间找到的极值点不一定是真正意义上的极值点，可以通过对尺度空间 DoG 函数进行曲线拟合寻找极值点来减小这种误差。利用 DoG 函数在尺度空间的 Taylor 展开式：

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

则极值点为：

$$\hat{X}=(x,y,\sigma)^T$$

接下来再去掉一些 DoG 响应较低的点，还有一些响应较强的点也不是稳定

的特征点。DoG 对图像中的边缘有较强的响应值，所以落在图像边缘的点也不是稳定的特征点。即完成消除边缘响应，便可最终实现关键点的精确定位。

2.2.2 特征点方向赋值

该模块在文件 SIFTDescriptor.m 中实现。

定位关键点之后，为了图像旋转不变性，需要根据检测到的局部图像结构为特征点方向赋值。本项目中使用图像的梯度直方图法求关键点局部结构的稳定方向，精确定位关键点后也找到改特征点的尺度值σ，根据这一尺度值，得到最接近这一尺度值的高斯图像：

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

使用有限差分，计算以关键点为中心，以 3×1.5σ为半径的区域内图像梯度的幅角和幅值，公式如下：

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

$$\theta(x,y)=\arctan\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right)$$

在完成关键点邻域内高斯图像梯度计算后，使用直方图统计邻域内像素对应的梯度方向和幅值。此处方向直方图的核心是统计以关键点为原点，一定区域内的图像像素点对关键点方向生成所作的贡献。梯度方向直方图的横轴是梯度方向角，纵轴是剃度方向角对应的梯度幅值累加值。梯度方向直方图将 0°~360°的范围分为 36 个柱，每 10°为一个柱。下图是从高斯图像上求取梯度，再由梯度得到梯度方向直方图的例图。

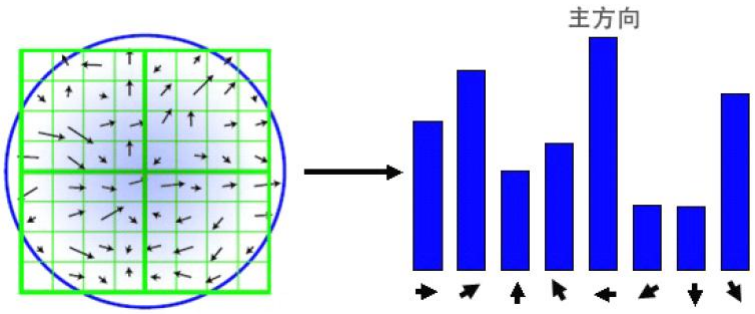


图 2-2 梯度直方图转换

在计算直方图时，每个加入直方图的采样点都使用圆形高斯函数函数进行了

加权处理，也就是进行高斯平滑。这主要是因为 SIFT 算法只考虑了尺度和旋转不变形，没有考虑仿射不变性。通过高斯平滑，可以使关键点附近的梯度幅值有较大权重，从而部分弥补没考虑仿射不变形产生的特征点不稳定。

直方图峰值代表该关键点处邻域内图像梯度的主方向，也就是该关键点的主方向。在梯度方向直方图中，当存在另一个相当于主峰值 80%能量的峰值时，则将这个方向认为是该关键点的辅方向。所以一个关键点可能检测得到多个方向，这可以增强匹配的鲁棒性。David Lowe 关于 SIFT 的论文指出大概有 15%关键点具有多方向，但这些点对匹配的稳定性至为关键。获得图像关键点主方向后，每个关键点有三个信息(x,y,σ,θ)：位置、尺度、方向。由此我们可以确定一个 SIFT 特征区域。通常使用一个带箭头的圆或直接使用箭头表示 SIFT 区域的三个值：中心表示特征点位置，半径表示关键点尺度（r=2.5σ），箭头表示主方向。具有多个方向的关键点可以复制成多份，然后将方向值分别赋给复制后的关键点。

2.2.3 构建描述子

该模块在文件 SIFTDescriptor.m 中实现。

对于每一个关键点，拥有三个信息：位置、尺度以及方向。接下来就是为每个关键点建立一个描述符，用一组向量将这个关键点描述出来，使其不随各种变化而改变，比如光照变化、视角变化等等。这个描述子不但包括关键点，也包含关键点周围对其有贡献的像素点，并且 用来作为目标匹配的依据（所以描述子应该有较高的独特性，以保证匹配率），也可使关键点具有更多的不变特性，如光照变化、3D 视点变化等。

关键点描述即用用一组向量将这个关键点描述出来，这个描述子不但包括关键点，也包括关键点周围对其有贡献的像素点。SIFT 描述子 h(x,y,θ)是关键点邻域高斯图像梯度统计结果的一种表示。通过对关键点周围图像区域分块，计算块内梯度直方图，生成具有独特性的向量，这个向量是该区域图像信息的一种抽象，具有唯一性。它是一个三维矩阵，但通常用一个矢量来表示。矢量通过对三维矩阵按一定规律排列得到。特征描述子与关键点所在尺度有关，因此对梯度的求取应在特征点对应的高斯图像上进行。将关键点附近划分成 d×d 个子区域，每个子区域尺寸为 mσ个像元（d=4，m=3，σ为尺特征点的尺度值）。考虑到实际计算

时需要双线性插值，故计算的图像区域为 mσ(d+1)，再考虑旋转，则实际计算的图像区域为 $m\sigma(d+1)\sqrt{2}$ 。同时为了保证特征矢量具有旋转不变性，要以特征点为中心，在附近邻域内旋转θ角，即旋转为特征点的方向。旋转后区域内采样点新的坐标为：

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

将旋转后区域划分为 d×d 个子区域（每个区域间隔为 mσ像元），在子区域内计算 8 个方向的梯度直方图，绘制每个方向梯度方向的累加值，形成一个种子点。与计算主方向不同的是，此时每个子区域梯度方向直方图将 0°~360°划分为 8 个方向区间，每个区间为 45°，即每个种子点有 8 个方向区间的梯度强度信息。由于存在 d×d，即 4×4 个子区域，所以最终共有 4×4×8=128 个数据，形成 128 维 SIFT 特征矢量。对特征矢量需要加权处理，加权采用 mσd/2 的标准高斯函数。为了除去光照变化影响，还有一步归一化处理。

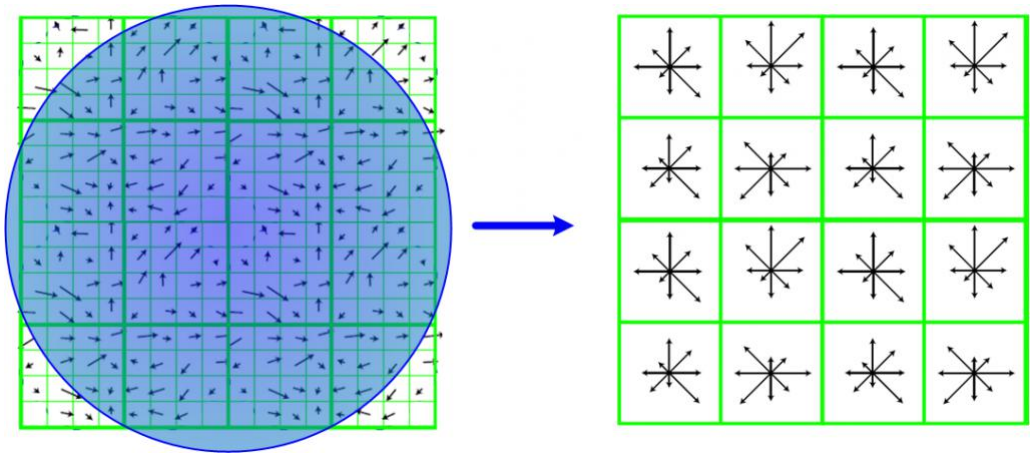


图 2-3 子区域梯度直方图

2.2.4 匹配 SIFT 描述子

该模块在文件 SIFTSimpleMatch.m 中实现。

这一步比较简单，即计算两个向量之间的欧式距离，将结果向量排序，最后筛选出最佳匹配用于拼接图像。给定一个 image1 中 SIFT 描述子和所有 image2 中的 SIFT 描述子，计算它们之间的欧式距离。然后使用它来确定是否是良好的匹配，若与最近邻的向量的距离，比次近邻向量小的多，就认为它是一对匹配。

输出结果是一个数组，每行两个值，分别代表匹配的描述子的索引编号。

欧几里得度量（即欧氏距离）是一个通常采用的距离定义，指在 m 维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

欧几里得度量二维空间的公式：

0ρ = sqrt((x1-x2)^2+(y1-y2)^2)　|x| = √(x2 + y2)

欧几里得度量三维空间的公式：

0ρ = √((x1-x2)^2+(y1-y2)^2+(z1-z2)^2)　|x| = √(x2 + y2 + z2)

2.2.5 RANSAC

该模块在文件 RANSACFit.m 中实现。

本模块使用 RANSAC（RANdom SAmple Consensus）选取内点来计算单应矩阵，而不是直接把所有的 SIFT 关键点匹配放进 ComputeAffineMatrix.m 产生结果。在这里，内点是指符合相同的变换矩阵的匹配对。简要的说经典 RANSAC 的目标是不断尝试不同的目标空间参数，使得目标函数 C 最大化的过程。这个过程是随机、数据驱动的过程。通过反复的随机选择数据集的子空间来产生一个模型估计，然后利用估计出来的模型，使用数据集剩余的点进行测试，获得一个得分，最终返回一个得分最高的模型估计作为整个数据集的模型。

经典的 RANSAC 流程中，目标函数 C 可以被看作：在第 k 次迭代过程中，在当前变换参数θ_k作用下，数据集中满 u 足变换参数的点的个数，也就是在当前变换条件下类内点的个数，而 RANSAC 就是最大化 C 的的过程。而判断当前某个点是否为类内需要一个阈值 t。

在迭代的过程中，当前变换参数 θ 的计算需要 u 中的一个子集 I 来计算，RANSAC 是一个随机从 u 中采样一个子集，然后对参数进行“估计-确认”的循环。每一个子集应是一个大小为 m 的最小采样。所谓最小采样，就是 m 的大小刚好满足计算 θ 的个数即可。在置信度为η_0的条件下，在循环过程中，至少有一次采样，使得采样出的 m 个点均为类内点，这样才能保证在循环的过程中，至少有一次采样能取得目标函数的最大值。因此，采样次数 k 应该满足以下条件：

$$k \geq \frac{\log(1-\eta_0)}{\log(1-\varepsilon^m)}$$

这里除了置信度η_0外，m 为子集大小， ε 为类内点在 u 中的比例，其中置信度一般设置为[0.95, 0.99]的范围内。然而在一般情况下 ε 显然是未知的，因此 ε 可以取最坏条件下类内点的比例，或者在初始状态下设置为最坏条件下的比例，然后随着迭代次数，不断更新为当前最大的类内点比例。

阈值 t 是判断在当前的获得的参数 θ 下，u 中某一点是类内点还是类外点的判断依据。在这里，假定类外点是高斯白噪声，其均值为 0，方差为 σ，误差的残差符合 n 维的卡方分布。而误差阈值的选取即可以按照以下的公式计算。

$$t^2 = X_n^{-1}(\alpha)\sigma^2$$

α 为置信概率，若 α =0.95，那么一个真实的类内点被误分类为类外点的概率为 5%。

2.2.6 图像 Wrap 转换

该模块在文件 ComputeAffineMatrix.m 中实现。

采用矩阵的形式描述为：

$$p' = H * p$$

$$其中 H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{21} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

矩阵 H 被称为投影变换矩阵。对上式进行分解化简，即是通常所谓的单应性变换：

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}$$

图像 Warp 变换是一项耗时的操作，相当于对里面每一个像素点进行一次变换，像素之间的操作相互独立，因此操作通常放在 GPU 上来并行处理，对于 CUDA 来讲，透视变换相当于将输入图像的索引坐标值映射到纹理坐标。

2.2.7 拼接多幅图像

该模块是在 MultipleStitch.m 中实现的。

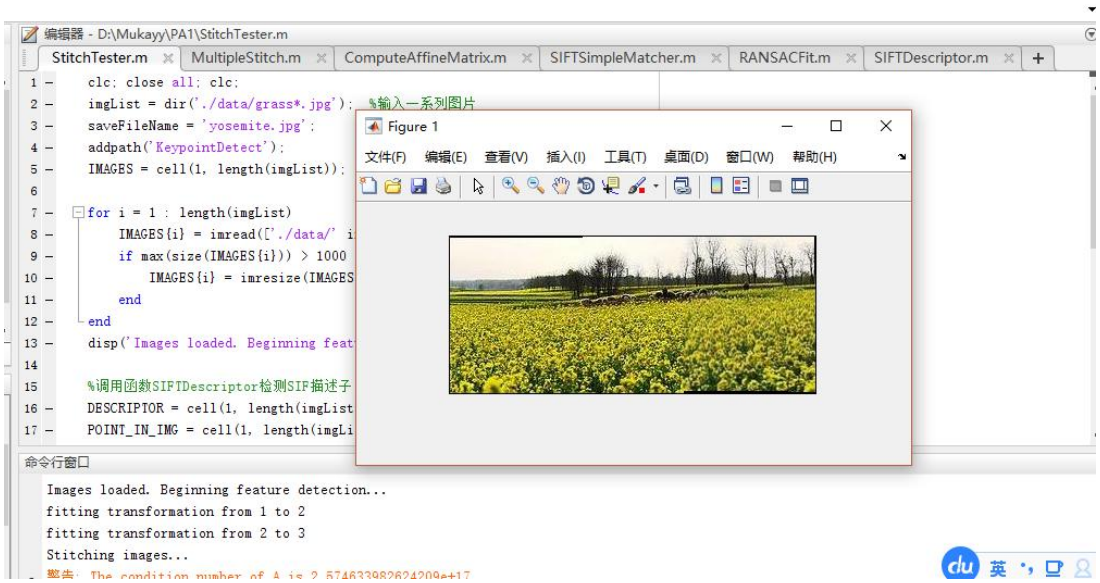
本模块是使用之前编写的各模块函数代码实现高效的拼接多幅图像形成一个有序的图像序列，其中多幅图像是在一条线上拍摄的，每张图像都是最终全景的一部分。给定一个包含 m 张图像的序列， 例如 Img1 ，Img2 ，……Imgm，然后在每一张图像上简单调用之前所编写的各模块函数，使用每相邻两幅图像，计算它们之间的变换矩阵，例如将一个点从 Imgi 的坐标系转换成 Imgi+1 的坐标系。

之后选取一个处在矩阵序列的中间的参考图像 Imgref，让最终的全景图处在 Imgref 的坐标系下。因此对于非参考图像的 Imgi，需要一个变换矩阵，来将第 i 帧图像的点转换到 ref 帧上，再调用 MATLAB 自带的函数，使用该变换矩阵去变换图像。

完成所有模块之后，运行 StitchTester.m 即可最终实现全景拼接。

3 实验结果展示

（1）实验一：三张简单图拼接，特征点易匹配，拼接效果好且运行时间短。



输入图像序列（分别为 grass1，grass2，grass3）：

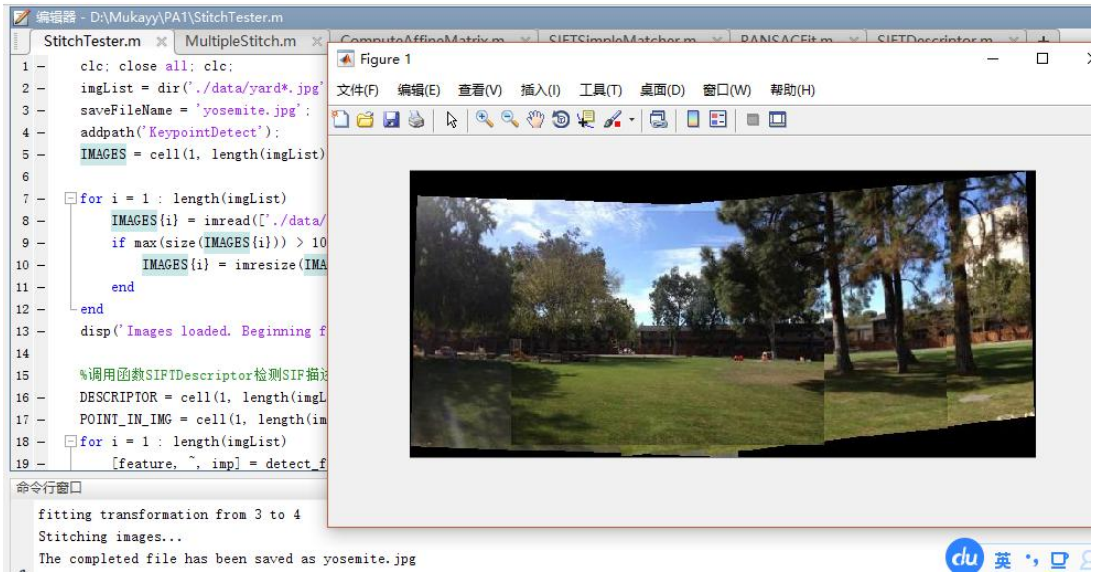
输出拼接结果：

命令窗口（显示匹配方向）：

命令行窗口

```
Images loaded. Beginning feature detection...
fitting transformation from 1 to 2
fitting transformation from 2 to 3
Stitching images...
```

（2）实验二：四张图拼接，拼接成功，但由于各图片特征差异较大，最终输出效果一般。



输入图像序列（分别为 yard1，yard2，yard3，yard4）：

输出拼接结果：

命令行窗口：

```
命令行窗口
  Images loaded. Beginning feature detection...
  fitting transformation from 1 to 2
  fitting transformation from 2 to 3
  fitting transformation from 3 to 4
  Stitching images...
  The completed file has been saved as yosemite.jpg
fx >>
```

（3）实验三：四张图拼接，拼接失败。每张图差异大，无法找到足够的匹配项。

编辑器 - D:\Mukayy\PA1\StitchTester.m

StitchTester.m | MultipleStitch.m | ComputeAffineMatrix.m | SIFTSimpleMatcher.m | RANSACFit.m | SIFTDescr

```
1 -  clc; close all; clc;
2 -  imgList = dir('./data/pine*.jpg');  %输入一系列图片
3 -  saveFileName = 'yosemite.jpg';       %存拼接后的图像
4 -  addpath('KeypointDetect');           %加路径
5 -  IMAGES = cell(1, length(imgList));
6
7 -  for i = 1 : length(imgList)
8 -      IMAGES{i} = imread(['./data/' imgList(i).name]);
9 -      if max(size(IMAGES{i})) > 1000 || length(imgList) > 10
10 -         IMAGES{i} = imresize(IMAGES{i}, 0.6);    %防止图像太大耗时太久 重新定义图像大小
11 -     end
12 -  end
13 -  disp('Images loaded. Beginning feature detection...');
14
15    %调用函数SIFTDescriptor检测SIF描述子
```

```
命令行窗口
  Images loaded. Beginning feature detection...
  fitting transformation from 1 to 2
  fitting transformation from 2 to 3
  fitting transformation from 3 to 4
  错误使用 RANSACFit (line 10)
  not enough matches to produce a transformation matrix
fx 出错 StitchTester (line 30)
```

输入图像序列（分别为 pine1，pine2，pine3，pine4）：

无输出图像。

命令行窗口：

命令行窗口

```
Images loaded. Beginning feature detection...
fitting transformation from 1 to 2
fitting transformation from 2 to 3
fitting transformation from 3 to 4
错误使用 RANSACFit (line 10)
not enough matches to produce a transformation matrix

出错 StitchTester (line 30)
    TRANSFORM{i} = RANSACFit(POINT_IN_IMG{i}, POINT_IN_IMG{i+1}, M);
```