2020

数字图像处理第二次报告

学生姓名：任泽华

班级：自动化 71

学号：2171411498

提交日期：2020-3-5

摘要 ：

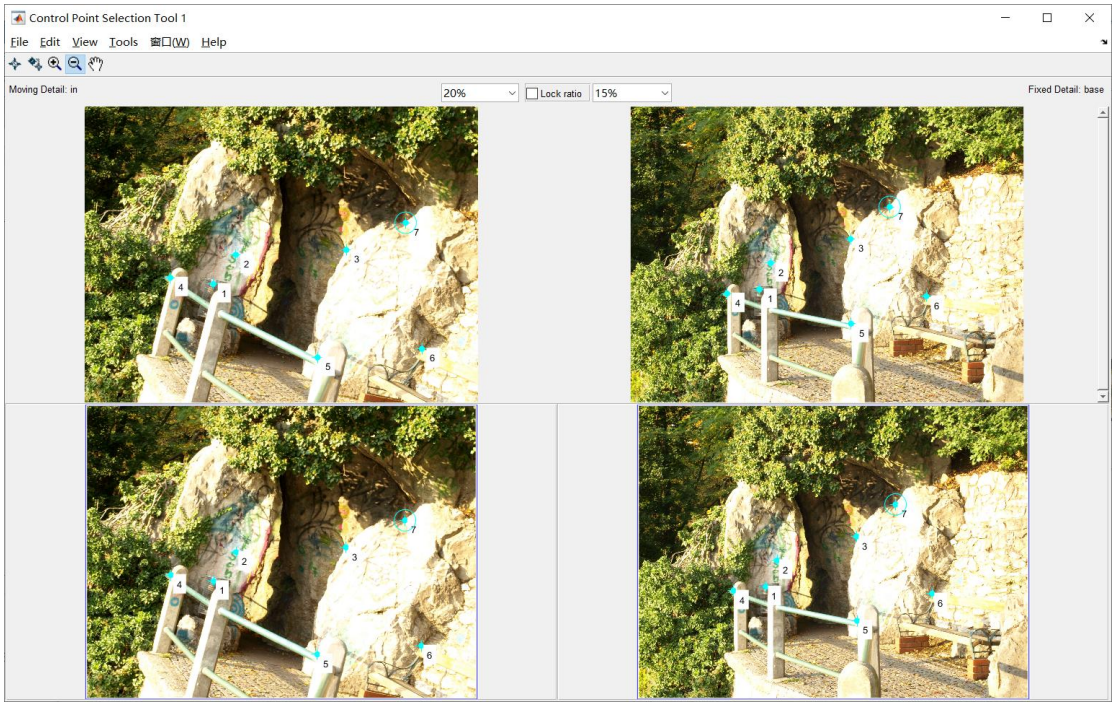
本报告主要工作：进行了图像配准，采用图片选取 7 个点进行配准，并且将 4 点配准与 7 点配准进行了对比；利用自己的图片实现了配准操作，并且学习了正交、相似、仿射、射影三种变换，对比了几种变换效果的优劣，发现利用射影变换匹配变形较严重的图像效果会比较好；最后利用网上的帖子尝试制作了一下全景图和艺术图。本报告软件运行环境为 MATLAB R2018b，所有代码均为自己编写，在编写过程中主要参考了 CSDN 相关帖子与 MATLAB 官方网站。（参考文献）

一、 要求根据已给的两幅图像，在各幅图像中随机找出 7 个点，计算出两幅图像之间的转换矩阵 H，并且输出转换之后的图像。

1. 找七个点实现配准

（1） 标注基准点

此处用到了 matlab 的工具箱 cpselect，输入 cpselect(in,base)，in 和 base 分别为待配准图像和基准图像。此时会弹出一个窗口，如下图所示，可以标注相同位置的七个点，点击生成坐标数组。



在 file 中选择保存，存为 movingpoints1 和 fixedpoints1，分别是待配准坐标和基准坐标。

（2） 使用 cp2tform 函数构建 tform 矩阵

调用格式为：

tform = cp2tform(movingPoints1,fixedPoints1,'affine');%affine 表示采用仿射变换，返回的不是一个数组，而是一个结构体，其中包含了其他的参考信息。

此时构造的转换矩阵为：

$$H = \begin{bmatrix} 0.966736409274093 & -0.254642650732802 & 0 \\ 0.259476975229231 & 0.966348751305291 & 0 \\ -7.29010370820616 & 713.921551164015 & 1 \end{bmatrix}$$

（3） 通过转换矩阵转换目标图片

调用 imtransform(in,tform)函数，in 为待配准图片，tform 为刚刚生成

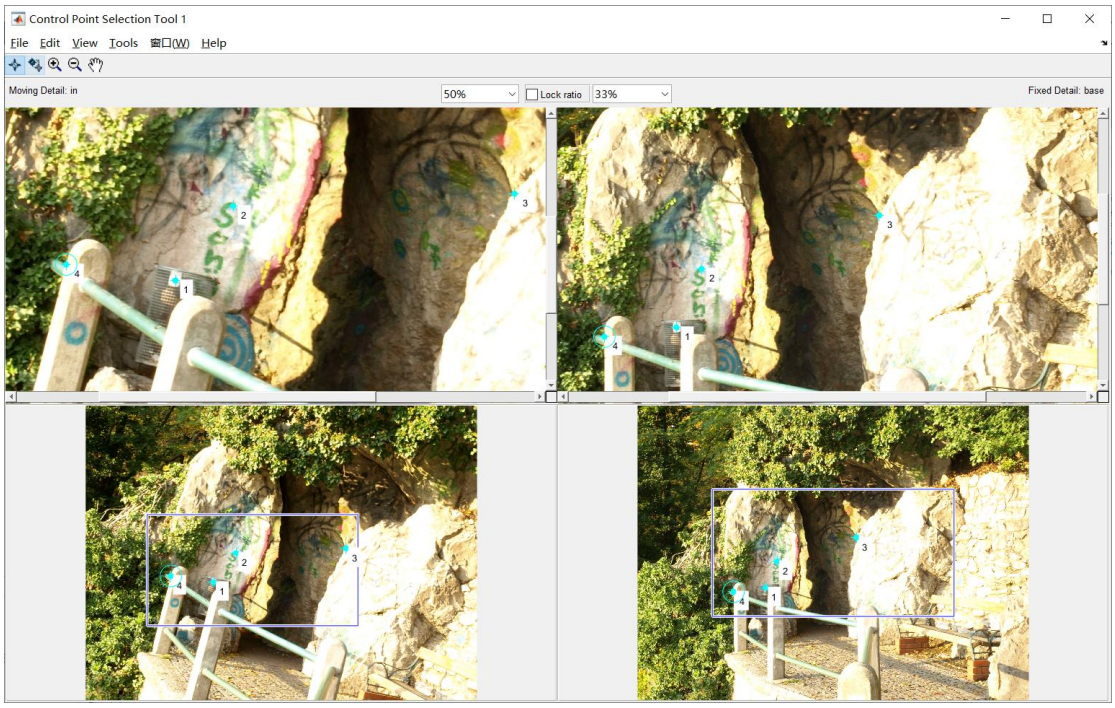
的旋转矩阵结构体。其值可以赋给另一个新变量。

（4） 输出两张图片对比

这样看来似乎效果还不错，那么如果只采四个点能不能实现配准呢？

2. 尝试只找四个点配准

如图所示，只选取四个点，生成的坐标分别为 movingpoints 和 fixedpoints，将四点和七点的坐标存为 data.mat 以便后续使用。



此时生成的转换矩阵为：

$$H = \begin{bmatrix} 0.957140631533280 & -0.247025794315251 & 0 \\ 0.233772164432435 & 0.989820085092475 & 0 \\ 31.4760171401864 & 679.135963517167 & 1 \end{bmatrix}$$

输出的配准图像为：

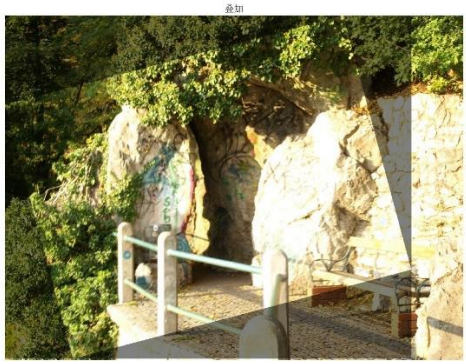
这样看来似乎没有什么问题，但是七个点和四个点真的没有区别吗？我打算把二者做一个对比。

3. 四点和七点的对比

Matlab 函数 imshowpair 可以实现对比功能，

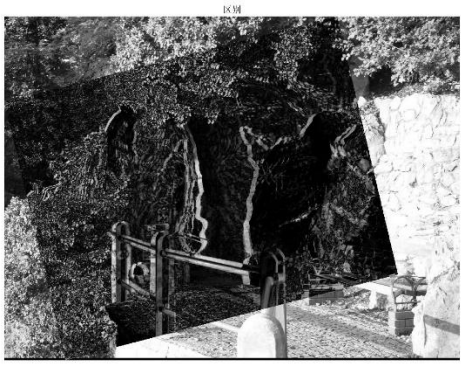
imshowpair(base,out,'diff');前两个分别是要对比的图像，后一个是对比模式（差异、叠加……）将基准图像和配准图像进行对比得到如下图片：

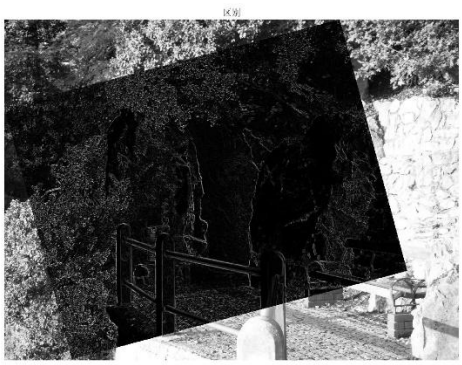
叠加图：



四点匹配　　七点匹配

差异图：





四点匹配　　七点匹配

可以很明显地看出，四点匹配的精度远远不如七点匹配的精度高。

二、　用自己的图实现图像配准

只是用例子怎么能够？为了进一步学习，我自己拍了两张楼下的照片，希望可以实现图片配准。

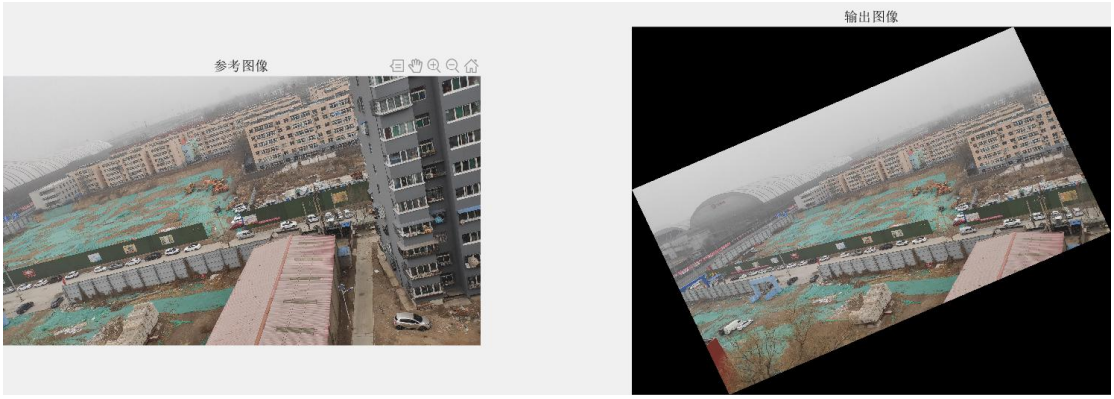
1. 相似变换、仿射变换和射影变换

和一中一样，采用同样的方法选取 7 个点进行标注，坐标保存在 data2.mat 中。

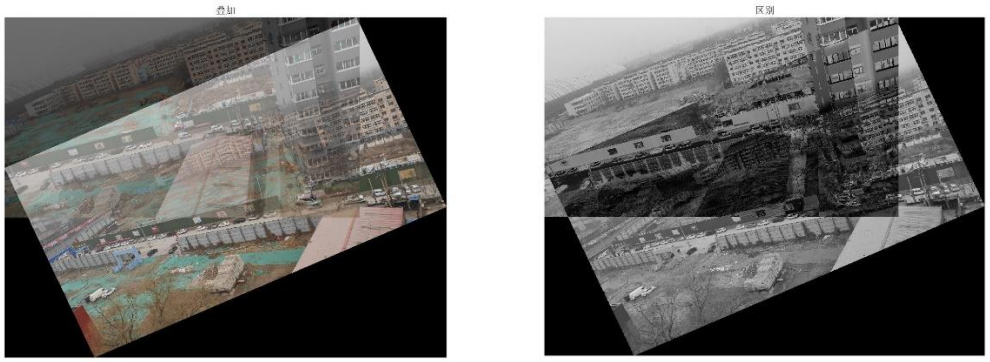
生成的转换矩阵为：

$$H = \begin{bmatrix} 0.991221841732297 & -0.422865558988413 & 0 \\ 0.450668583811623 & 0.946401157713225 & 0 \\ -1433.05523858418 & 926.438096500629 & 1 \end{bmatrix}$$

配准效果如下：



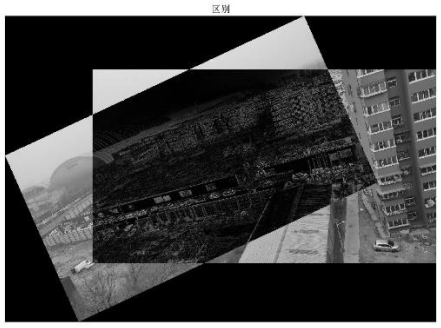
但是在对比时出了问题，因为作业的图片应该是按照原图截取旋转得到的，按照边框即可配准。我的这两幅图片是分别拍的，其中有很多部分不重叠，而且配准后大小不一致，就造成了这种情况：



参考了网上的帖子，有一个构建全景图的 matlab 官方帖子，“Feature Based Panoramic Image Stitching”其中就有关于图像翘曲的内容，通过这种办法我把它们同样的图像内容对齐，如图所示：

这样看起来似乎匹配效果不好，经过查阅资料，图片的变换分为以下几种：正交变换、相似变换、仿射变换、射影变换，它们的范围是逐渐变大的，在我这种自己拍的图片，因为变换拍摄角度，单单用仿射变换显得不够，只是进行旋转和平行拉伸不太够用，所以我采用射影

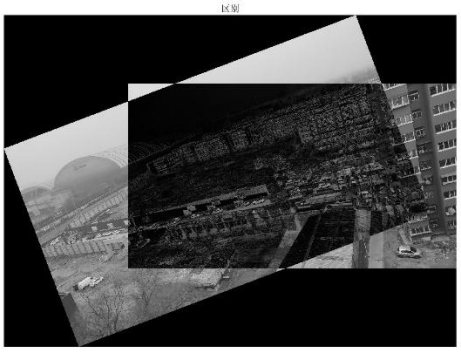
变换再次实验，发现效果还不错。



仿射变换　　射影变换

$$H\begin{bmatrix} 1.25033995829747 & -0.340953322956309 & 1.16141707090573e-04 \\ 0.546240871954637 & 1.11880111826756 & 4.83070232727793e-05 \\ -1879.39136138596 & 683.781992171772 & 0.825098265348853 \end{bmatrix}$$

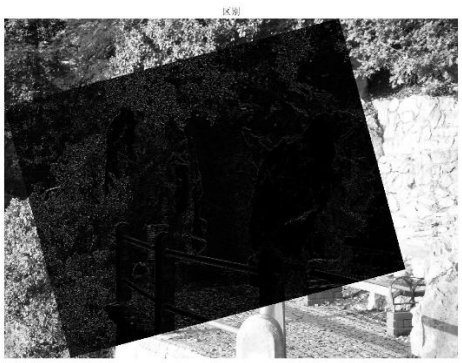
同时我也采用了低一级的相似变换，很明显，相似变换的效果比仿射变换还差：



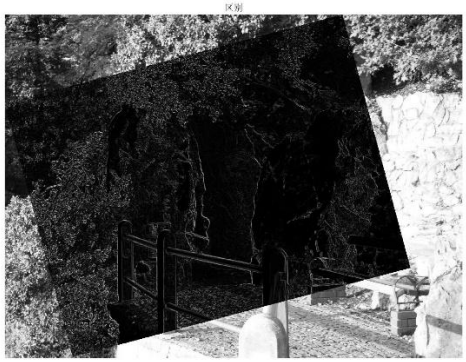
相似变换　　仿射变换

$$H = \begin{bmatrix} 0.975696693502466 & -0.432475331845495 & 0 \\ 0.432475331845495 & 0.975696693502466 & 0 \\ -1379.39855771717 & 909.829416700472 & 1 \end{bmatrix}$$

但是，利用题目图片，采用相似变换的效果似乎比仿射变换要好一些，可能是由于匹配图就是原图当中用矩形截出来的。在取了七个点以后，在误差允许的前提下，符合其本来生成的方法匹配效果更好。



相似变换

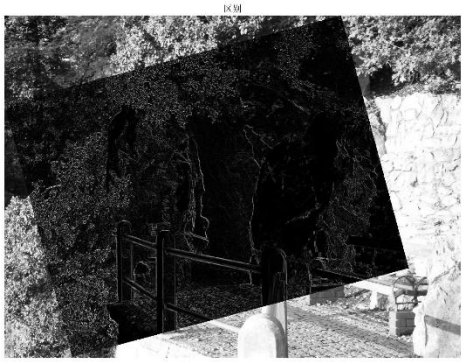


仿射变换

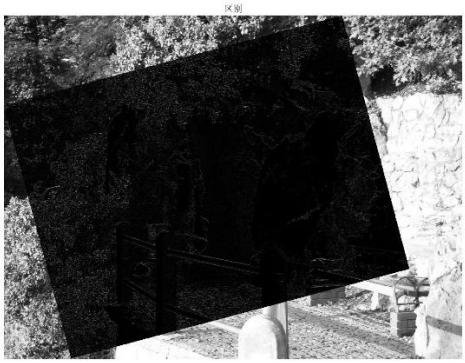
三、 实现自动匹配

前面都是在自己选点，如果图片较多，或者要实现更高精度的要求，要实现快速匹配，能不能利用电脑自动识别，从而解放人的工作呢？在上一问提到的图像翘曲的帖子中，也有关于自动识别的内容，即识别图片的 SURF 特征，即可实现图片的自动识别匹配。

题目图片：

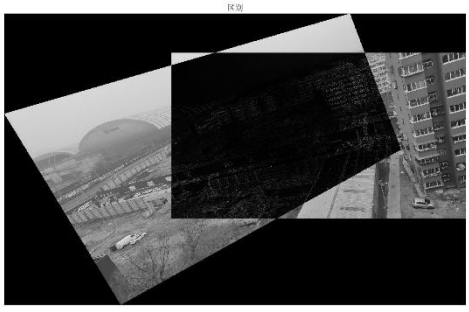


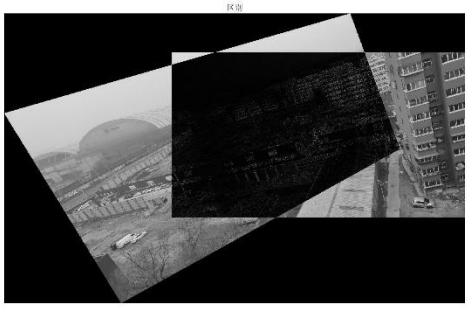
仿射变换



自动识别

自己的图片：





仿射变换　　自动识别

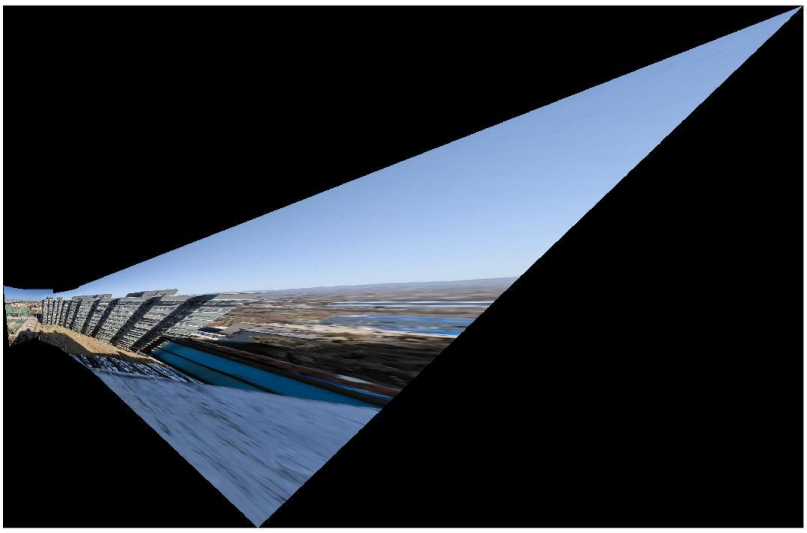
四、　构建全景图和各时段对比图

利用帖子中的代码，打算自己构建小区楼下的全景图，发现失败了

Matlab 显示内存溢出

但是图片不算太大，计算出的图片竟然高达 100G

我减少了几张图片，发现全景图生成出来是这个样子：



经过比较，第一张图片两边视角大于 90°，而第二张图片的视角也将近为 90°，这才造成了图片的严重变形。如果可以水平移动拍摄，应该可以构建比较好的全景图。

于是我把注意力放在了构造一些其他有意思的图片上。

比如说把一天不同时段的照片合成起来，但是我没有三脚架，所以难于实现，不过接住图像配准和拼接的计算，仅仅用手大概拍的照片也能实现比较好的艺术效果：

一天不同时段的拼接：

雪天和晴天的拼接：

晴天->多云->阴->雾霾

附录

1. 参考文献

[1] Rafael C. Gonzalez（拉斐尔 C. 冈萨雷斯）,Richard E. Woods （理查德 E. 伍兹）. 数字图像处理(第三版)（英文版).北京：电子工业出版社. 2017 年.
[2]数字图像处理---图像配准
https://blog.csdn.net/zhengxiaoyang995926/article/details/79417820
[3]图像配准实现（matlab 篇）https://bbs.csdn.net/topics/390975425
[4]Feature Based Panoramic Image Stitching
https://ww2.mathworks.cn/help/vision/examples/feature-based-panoramic-image-stitching.html?s_tid=srchtitle
[5]imshowpair https://ww2.mathworks.cn/help/images/ref/imshowpair.html
[6]Maketform
https://ww2.mathworks.cn/help/images/ref/maketform.html#d118e228471

2. 源代码

（1）四点、七点配准、比较代码

```
clear
base=imread('Image A.jpg');
in=imread('Image B.jpg');
load data.mat
% subplot(1,2,1),imshow(in);
% subplot(1,2,2),imshow(base);
% cpselect(in,base);

% tform = cp2tform(movingPoints,fixedPoints,'affine');
%tform = cp2tform(movingPoints1,fixedPoints1,'affine');
tform = cp2tform(movingPoints1,fixedPoints1,' similarity');
out = imtransform(in,tform);
```

```
% figure
% subplot(121)
% imshow(base);
% title('参考图像');
% subplot(122)
% imshow(out);
% title('输出图像');

figure
diff=imshowpair(base,out,'diff');
title('区别');
saveas(diff,'区别 4.jpg');
% saveas(diff,'区别 2.jpg');
% saveas(diff,'区别 1.jpg');
figure
synthesis=imshowpair(base,out,'blend','Scaling','joint');
title('叠加');
saveas(synthesis,'叠加 4.jpg');
% saveas(synthesis,'叠加 2.jpg');
% saveas(synthesis,'叠加 1.jpg');
% 'falsecolor', 'diff', 'blend', 'montage', 'checkerboard'
```

（2）用自己的图实现图像配准

```
clear
base=imread('A.jpg');
in=imread('B.jpg');
grayImage0 = rgb2gray(base);
grayImage = rgb2gray(in);
load data2.mat
tforms(2) = estimateGeometricTransform(movingPoints2, fixedPoints2,...
    'affine', 'Confidence', 99.9, 'MaxNumTrials', 2000);
% tforms(2) = estimateGeometricTransform(movingPoints2, fixedPoints2,...
%     'similarity', 'Confidence', 99.9, 'MaxNumTrials', 2000);
% tforms(2) = estimateGeometricTransform(movingPoints2, fixedPoints2,...
%     'projective', 'Confidence', 99.9, 'MaxNumTrials', 2000);
```

```
imageSize = zeros(2,2);
imageSize(1,:) = size(grayImage0);
imageSize(2,:) = size(grayImage);
for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(i,2)], [1 imageSize(i,1)]);
end
maxImageSize = max(imageSize);
% Find the minimum and maximum output limits
xMin = min([1; xlim(:)]);
xMax = max([maxImageSize(2); xlim(:)]);

yMin = min([1; ylim(:)]);
yMax = max([maxImageSize(1); ylim(:)]);

% Width and height of panorama.
width  = round(xMax - xMin);
height = round(yMax - yMin);

xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);

warpedImage1 = imwarp(base, tforms(1), 'OutputView', panoramaView);
warpedImage2 = imwarp(in, tforms(2), 'OutputView', panoramaView);

figure
diff=imshowpair(warpedImage1,warpedImage2,'diff');
title('区别');
saveas(diff,'区别 3.jpg');
figure
synthesis=imshowpair(warpedImage1,warpedImage2,'blend','Scaling','joint');
title('叠加');
saveas(synthesis,'叠加 3.jpg');
```

（3）实现自动匹配

A.　　题目图自动匹配

```
clear
base=imread('Image A.jpg');
in=imread('Image B.jpg');
grayImage0 = rgb2gray(base);
points = detectSURFFeatures(grayImage0);
[features0, points0] = extractFeatures(grayImage0, points);
grayImage = rgb2gray(in);
points = detectSURFFeatures(grayImage);
[features, points] = extractFeatures(grayImage, points);
indexPairs = matchFeatures(features, features0, 'Unique', true);
matchedPoints = points(indexPairs(:,1), :);
matchedPointsPrev = points0(indexPairs(:,2), :);
tform = estimateGeometricTransform(matchedPoints, matchedPointsPrev,...
    'projective', 'Confidence', 99.9, 'MaxNumTrials', 2000);
T=double(tform.T);
tform=maketform('projective',T);
out = imtransform(in,tform);

figure
diff=imshowpair(base,out,'diff');
title('区别');
saveas(diff,'区别 3.jpg');
figure
synthesis=imshowpair(base,out,'blend','Scaling','joint');
title('叠加');
saveas(synthesis,'叠加 3.jpg');
```

B.　　自己的图自动匹配

```
clear
base=imread('A.jpg');
in=imread('B.jpg');
grayImage0 = rgb2gray(base);
points = detectSURFFeatures(grayImage0);
[features0, points0] = extractFeatures(grayImage0, points);
```

```
grayImage = rgb2gray(in);
points = detectSURFFeatures(grayImage);
[features, points] = extractFeatures(grayImage, points);
indexPairs = matchFeatures(features, features0, 'Unique', true);
matchedPoints = points(indexPairs(:,1), :);
matchedPointsPrev = points0(indexPairs(:,2), :);
tforms(2) = estimateGeometricTransform(matchedPoints, matchedPointsPrev,...
    'projective', 'Confidence', 99.9, 'MaxNumTrials', 2000);

imageSize = zeros(2,2);
imageSize(1,:) = size(grayImage0);
imageSize(2,:) = size(grayImage);
for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(i,2)], [1
imageSize(i,1)]);
end
maxImageSize = max(imageSize);
% Find the minimum and maximum output limits
xMin = min([1; xlim(:)]);
xMax = max([maxImageSize(2); xlim(:)]);

yMin = min([1; ylim(:)]);
yMax = max([maxImageSize(1); ylim(:)]);

% Width and height of panorama.
width  = round(xMax - xMin);
height = round(yMax - yMin);

xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);

warpedImage1 = imwarp(base, tforms(1), 'OutputView', panoramaView);
warpedImage2 = imwarp(in, tforms(2), 'OutputView', panoramaView);

figure
```

```
diff=imshowpair(warpedImage1,warpedImage2,'diff');
title('区别');
saveas(diff,'区别 2.jpg');
figure
synthesis=imshowpair(warpedImage1,warpedImage2,'blend','Scaling','joint');
title('叠加');
saveas(synthesis,'叠加 2.jpg');
```

（4）构建全景图

本题主要采用帖子“Feature Based Panoramic Image Stitching”中的代码，仅为探究，不列出代码。