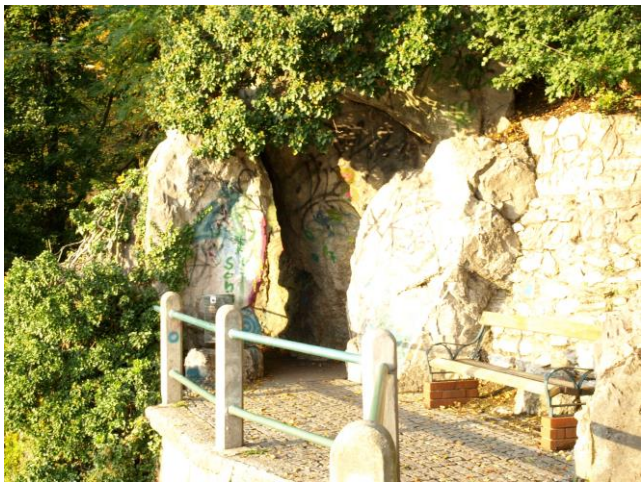

数字图像处理

第二次作业

摘要

图像配准是数字图像处理的一种重要应用，用于对齐两幅或多幅相同场景的图像。图像配置的主要方法之一便是约束点法，即在输入图像和参考图像中选取若干基准点，以此推导变换矩阵。本文尝试利用这种方法实现两幅图片的配准。

刘 昊　自动化 61　2160504016

2019.3.3

1 实验原理

设提取出的 n 对基准点分别组成矩阵 P 和 Q，前者为输入图像的基准点，后者为参考图像的基准点。

$$P = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} u_0 & u_1 & \cdots & u_{n-1} \\ v_0 & v_1 & \cdots & v_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

则利用最小二乘法推导得到的变换矩阵 H 为

$$H = QP^T(PP^T)^{-1}$$

2 实验结果

以 A 图为参考图像，B 图为输入图像，利用 Matlab 提供的 cpselect()函数选择 7 对基准点：

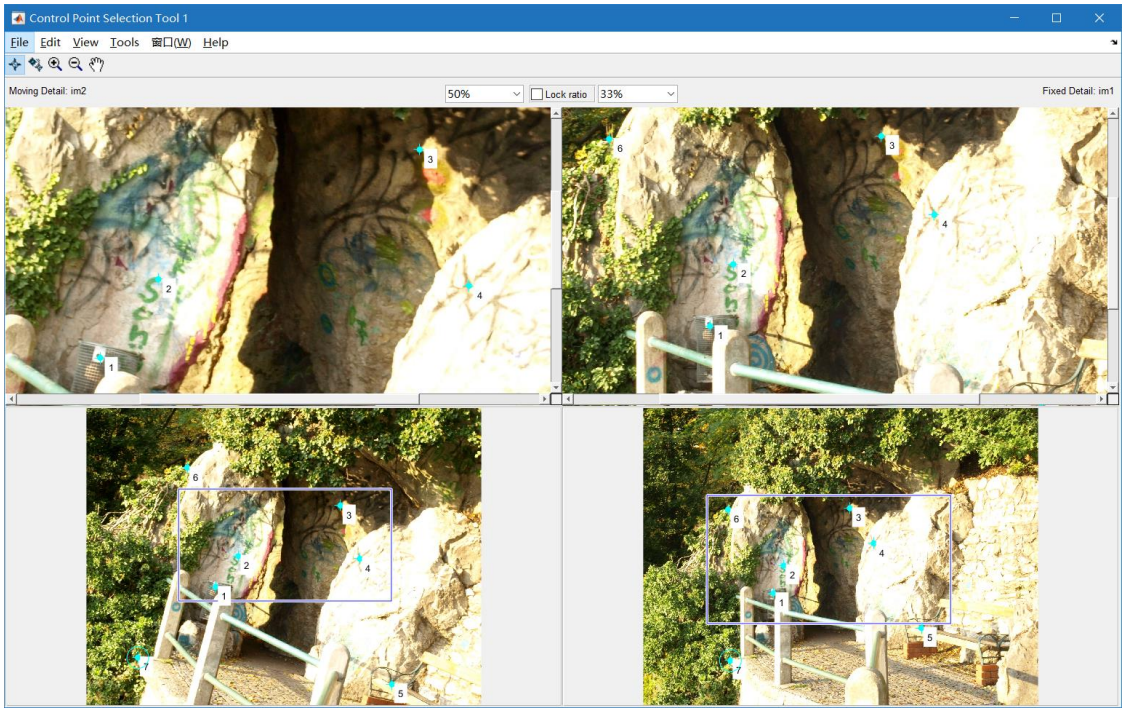

图 1 选择基准点

得到的基准点坐标分别为：

```
fixedPoints =

   1.0e+03 *

   1.202000000000000   1.706000000000000
   1.298000000000000   1.451000000000000
   1.910000000000000   0.923000000000000
   2.129000000000000   1.247000000000000
   2.570000000000000   2.027000000000000
   0.785000000000000   0.935000000000000
   0.805999999999999   2.327000000000000
```

图 2 参考图像基准点

```
movingPoints =

   1.0e+03 *

   0.911769636015326   1.258102011494253
   1.071846264367816   1.041998563218391
   1.792191091954023   0.685828065134100
   1.928256226053640   1.060007183908046
   2.156274066091954   1.943322557471264
   0.711244795405599   0.419224694903087
   0.363369705671213   1.754745154343144
```

图 3 输入图像基准点

计算得到变换矩阵 H 为

```
H =

   1.0e+02 *

   0.009646162330702  -0.002630001267272   7.273587123619354
   0.002574704532094   0.009624118305363  -0.001276251207804
                   0                   0   0.010000000000000
```

图 4 变换矩阵

配准之后得到的图像为

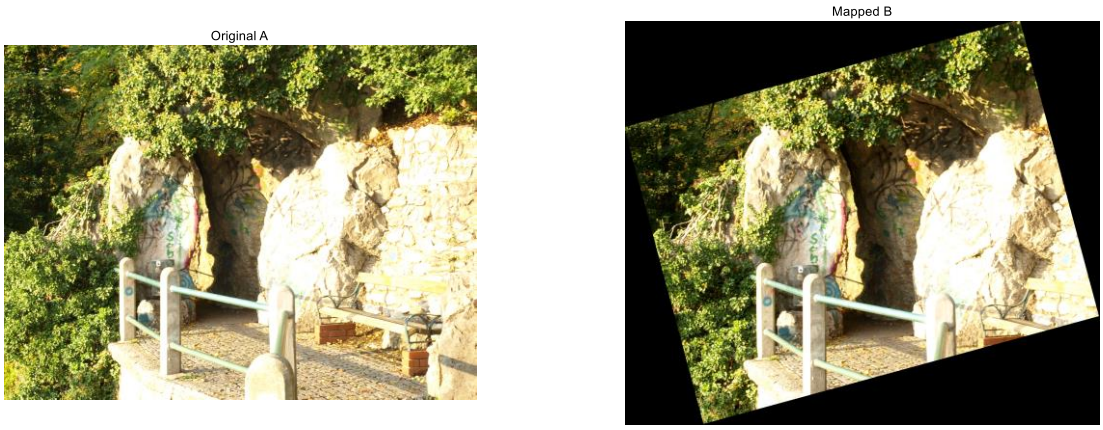

图 5 配准结果

3 代码示例

```
%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
%输出转换之后的图像
im1=imread('Image A.jpg');
im2=imread('Image B.jpg');
[movingPoints,fixedPoints]=cpselect(im2,im1,'Wait',true); %手动选点，程序等待选点结束后再继续运行
P=[movingPoints(:,2)';movingPoints(:,1)';ones(1,7)];      %构造 P、Q 矩阵，注意坐标顺序。
Q=[fixedPoints(:,2)';fixedPoints(:,1)';ones(1,7)];
H=Q*P'/(P*P');
[r1,r2,~]=size(im2);
s1=floor(max(H(1,:)*[0,0,r1,r1;0,r2,0,r2;1,1,1,1]));      %计算输出矩阵维度
s2=floor(max(H(2,:)*[0,0,r1,r1;0,r2,0,r2;1,1,1,1]));
im2_1=zeros(ceil(s1),ceil(s2),3);
Hi=inv(H);
for i=1:s1                                                  %采用反向变换
    for j=1:s2
        x=Hi*[i;j;1];
        if x(1)>=1&&x(1)<=r1&&x(2)>=1&&x(2)<=r2
            im2_1(i,j,:)=im2(floor(x(1)),floor(x(2)),:);
        else
            im2_1(i,j,:)=[0,0,0];
        end
    end
```

```
end
im2_1=uint8(im2_1);
subplot(1,2,1)
imshow(im1);
title('Original A');
subplot(1,2,2)
imshow(im2_1);
title('Mapped B');
```

4 心得体会

通过本实验，我对图像配准的基本思路和流程以及图像仿射变换方法有了更深入的了解，在实际操作中遇到关于横纵坐标顺序、图像尺度等问题是我留意到理论学习中未曾注意到的问题，也使我的编程能力以及对相关函数、工具的熟悉程度有所提高。在自己编写函数的同时，我也尝试使用了 Matlab 自带的图像配准和仿射变换工具，后者在保证的较高配准质量的基础上，运算速度明显快于我自己的代码，可见在相关编程技术上我还有很大的提高空间。