图像匹配建模
Image Matching Modeling

Abstract

In the collection of images, there may exist some deviations between two images. If we find the lost points and false point to conduct the image matching, the pressure of procession and storage will be largely released.

The key problem for image matching is to figure out the transform regulation for the two images. This paper proposes a method of constructing congruent triangles in finding match points. The object is to include as many points as possible in accordance with this regulation, finding out most of the lost points and false points. Furthermore, concept “feature point set” is introduced to value the extent of matching.

In the end, the algorithm is validated and assessed. Suggestions are proposed.

摘要:

在图像采集中，两幅图像可能存在偏差，可能存在缺失点和伪点。

我们认为找到图像之间的变换关系是解决图像匹配问题的关键本。文通过在两幅图中找全等三角形的方法找到匹配点，从而确定两幅图的变换关系，使尽可能多的点符合此变换，找出缺失点和伪点，并根据匹配程度认定是否为一个特征点集。

最后我们给出了该算法的评价，提出了改进意见。

1） 问题的提出：

在图像采集中，不同时间采集同一景物的两幅图象，或者同一时间由不同传感器采集的两幅图象可能存在一定偏差。为了获得准确清晰地图像，现在要在两幅图象中采用一定的方法抽取特征点，对特征点进行匹配。

图像匹配在检测变化方面（森林学、医学、国防）、注册与验证（卫星图象、地图、指纹）以及立体成像方面都有重要应用。

2） 基本假设：

现设第一个点集有 m 个点，第二个点集有 n 个点，第二个点集是第一个点集经过某个平移和旋转得到，但由于噪声的作用，点的相对位置有微小的变化，而且第一个点集中可能有部分点（称为缺少点）在第二个点集中找不到对应点，第二个点集中可能随机出现一些新的点（称为伪点），通常 m 和 n 的大小范围为从 30 到 400 之间。试就只有平移和既有平移又有旋转两种情形，设计模型与算法，解决以下三个问题：

（ⅰ）找到第一个点集中所有的在第二个点集中没有匹配的点；

（ⅱ）找到第二个点集中所有的在第一个点集中没有匹配的点；

（ⅲ）对有匹配的点，找出正确的对应关系。

(ⅳ)每组数据的两个点集是否可以认为是一个特征点集？

假设 1：

任意三个点形成一个三角形（排除三点共线的情况）。每一个点集中不存在两个全等的三角形。

假设 2：

点集中的点数量不过于少，两个点集保证能出现至少一对全等三角形。

假设 3：

假设只存在线性变换。两个点集的大部分点之间的相对位置没有变化，即图形不存在畸变。第一幅图像中的一条直线经变换后，映射至第二幅图像上仍然是直线 。

假设 4：

实际情况允许一定的误差，确定匹配点的精度在长度的平方小于 100, 点集 1 中的点经理想变换后的对应点为圆心，半径的平方小于 100 的邻域内没有匹配点存在时，即认为该点在点集 2 中没有匹配的点。为点集 1 中的所有点寻找过其匹配点后，点集 2 中若存在没有被匹配的点，则认为该点在点集 1 中没有匹配点。

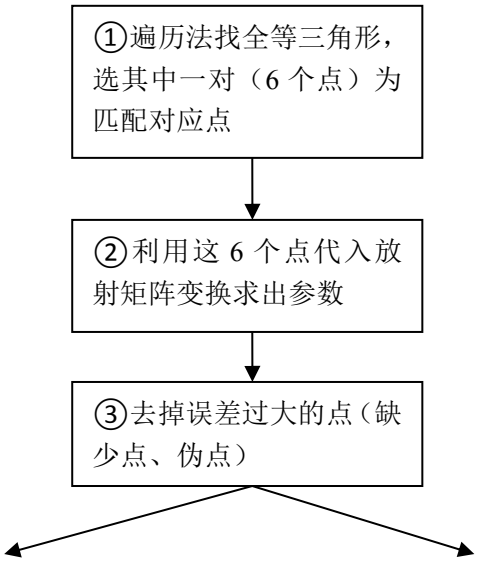
假设 5：

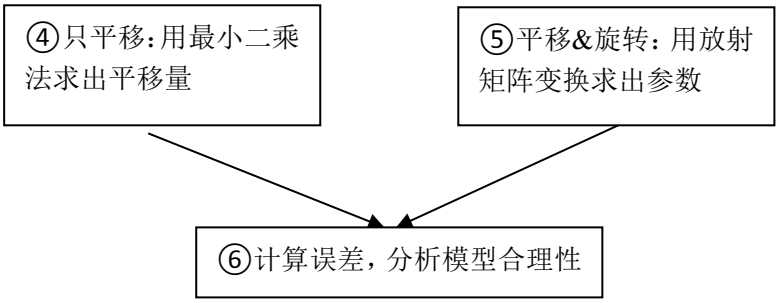
定义匹配度=匹配点对数*2/（点集 1 点数+点集 2 点数）

若两图匹配度超过 60%，即可认为两点集是一个特征点集。

3） **建立模型：**

建模的主要思想：





①遍历法找全等三角形

首先分别在 A 和 B 两个点集中用遍历法找出所有各自点集中的点构成的边，并用 cell 矩阵将其存储起来，分别记为 distance1 和 distance2,distance1 和 distance2 的结构都为$\begin{bmatrix}\text{distance}\\ i\\ j\end{bmatrix}$,distance 为 A 点集或 B 点集中任意两个不相等的点构成的边的长度，i 为第一个点的序号，j 为第二个的序号，这样 distance1 和 distance2 中的每个元素都有这条边的边长和构成这条边的两个点的序号这些信息，然后在再一次运用遍历法找出 distance1 和 distance2 中边长在误差允许的范围内（误差为长度的平方小于 100）对应相等的一组边，并再一次用 cell 矩阵将其存储记为 edge_xiangdeng,edge_xiangdeng 的结构为$\begin{bmatrix}edge1\\ edge2\end{bmatrix}$,edge1 为$\begin{bmatrix}\text{distance}_1\\ i_1\\ j_1\end{bmatrix}$,edge2 为$\begin{bmatrix}\text{distance}_2\\ i_2\\ j_2\end{bmatrix}$，edge1 为一组相等的边中 A 点集中点构成的边，edge2 为一组相等的边中 B 点集中点构成的边,最后运用遍历法找出 edge_xiangdeng 中 edge1 和 edge2 都可以构成三角形的三组边，记为 triangle_quandeng，其结构为$\begin{bmatrix}triangle1\\ triangle2\end{bmatrix}$，triangle1 为 A 点集中点构成的三角形，其结构为$\begin{bmatrix}edge1\\ edge2\\ edge3\end{bmatrix}$,triangle2 为 B 点集中点构成的三角形，其结构为$\begin{bmatrix}edge1'\\ edge2'\\ edge3'\end{bmatrix}$，这样就找出了全部的 A 点集中的点和 B 点集中的点构成的全等三角形并记录了其全部信息。

②利用仿射矩阵求变换参数：

首先分别将 A 和 B 两个点集的横纵坐标用矩阵存储起来，记为 A 和 B，其结构为$\begin{bmatrix} x_1 & x_2 & & x_n \\ y_1 & y_2 & & y_n \end{bmatrix}$和$\begin{bmatrix} x_1 & x_2 & & x_m \\ y_1 & y_2 & & y_m \end{bmatrix}$，然后利用已近得到的仿射变化矩阵求得 A 点集中点的对应点的横纵坐标并将其存储为 A1，其结构为$\begin{bmatrix} x'_1 & x'_2 & & x'_n \\ y'_1 & y'_2 & & y'_n \end{bmatrix}$

设坐标点(x,y) 经仿射变换后的坐标为(x',y') 则仿射变换的一般表达式为

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = k\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

式中k 是比例变化因子，θ 为旋转角，t 是平移矢量。由于此问题只涉及平移和旋转，不涉及图像的伸缩变换，因此$k=1$ 。

为确定仿射变换的相关参数，可以将变换式改写为

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

此变换式中有 6 个待定系数，因此需要列出 6 个线性方程。由于每一对匹配点可以列出两个线性方程，所以只需要三对匹配点即可求出一组仿射变换的参数（前提是列出的方程组线性无关）。

列出的六个方程分别为

$$\begin{cases} x_1'*a_{11} + y_1'*a_{12} + 0*a_{21} + 0*a_{22} + t_1 + 0*t_2 = x_1; \\ 0*a_{11} + 0*a_{12} + x_1'*a_{21} + y_1*a_{22} + 0*t_1 + t_2 = y_1; \\ x_2'*a_{11} + y_2'*a_{12} + 0*a_{21} + 0*a_{22} + t_1 + 0*t_2 = x_2; \\ 0*a_{11} + 0*a_{12} + x_2'*a_{21} + y_2*a_{22} + 0*t_1 + t_2 = y_2; \\ x_3'*a_{11} + y_3'*a_{12} + 0*a_{21} + 0*a_{22} + t_1 + 0*t_2 = x_3; \\ 0*a_{11} + 0*a_{12} + x_3'*a_{21} + y_3*a_{22} + 0*t_1 + t_2 = y_3; \end{cases}$$

解此线性方程组，即可得到仿射变换的参数a_{11} ， a_{12} ， a_{21} ， a_{22} ， t_1 ， t_2

③去掉误差过大的点（缺少点、伪点）：

在误差允许的范围内（误差定位两点之间距离的平方小于 100）判断点集 B 中点是否与 A1 中的点重合，若 A1 点集中的点(x'_i,y'_i)与 B 点集中的点(x_j,y_j)重合，则认为与(x'_i,y'_i)对应的 A 点集中的点 (x_i,y_i)与 B 点集中的点(x_j,y_j)相匹

配，并将(x_i, y_i)，(x_j, y_j)，(x'_i, y'_i)用 cell 矩阵存储在同一单元中方便第④和⑤步查看及调用。

④情况 1：只有平移：

记第一幅图像的各点重心（重心由各点坐标的平均值表示）坐标为（x_1,y_1），平移后的图像重心坐标为（x_1+m, x_1+n）,假设图 1 中任意一点 p1（x_{1i},y_{1i}）平移后为图 2 中的理想点 p2（x_{1i} +m,y_{1i}+n）

该对匹配点的误差由两点之间距离衡量：

Ei^2= （x_{1i} - x_{2i}）^{2}

设图 1 中的 1 到 m 个点中有 k 个点成功地在图 2 中找到匹配点，则这 k 点的总误差为$\sum^k_{i=1}Ei^2$

用一元线性回归（直线拟合）的方法确定平移量 m 和 n

由于 x_{2i} 和 x_{1i} 之间的关心是线性关系，则函数形势可写成 x_{2i}=a+ bx_{1i}

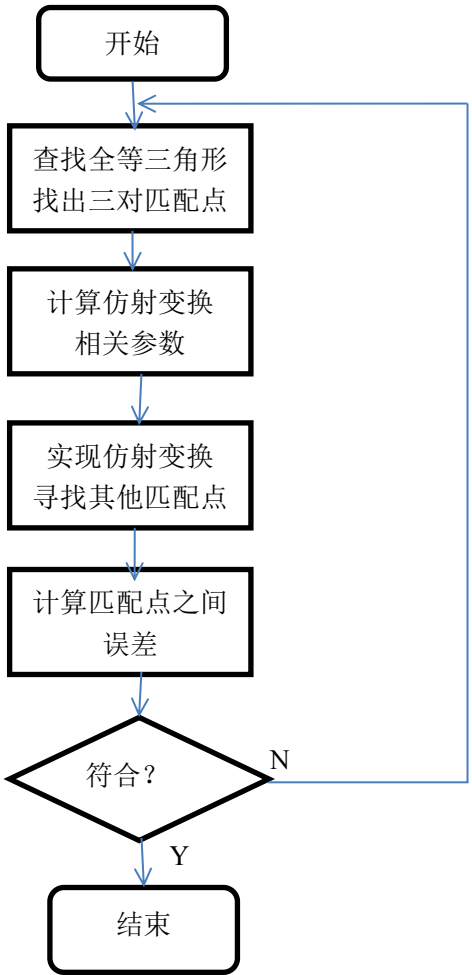
又由于“仅平移”，则 x_{2i}=a+x1i，即 b=1

按最小二乘原理，a 的最佳值应满足$\sum^k_{i=1}Ei^2=\sum^k_{i=1}$（$x_{2i}$-a-x1i）^{2}=min

公式：

$$a=\frac{x*xy-y*x^2}{x^2-x^2}$$

即可求出平移量 m 和 n



⑤情况 2：既存在平移，又存在旋转：

根据找出的三个点求出相应的仿射变换的参数，对第一个点集中的每一个点实现仿射变换。

将变换的象集中的所有点和第二个点集中的点进行匹配。计算误差，如果误差过大（比如出现同一点集有全等三角形的极端情况，则再返回寻找另外的一对全等三角形。

⑥计算误差：
误差采用匹配点的平均偏移距离来衡量，即

$$\delta = \frac{1}{n}\sum\sqrt{(x''-x')^2+(y''-y)^2}$$

4)计算模型：

①遍历法找全等三角形

（程序代码见附录 1）

算法原理图（同时也是第一组数据的散点图）如下：
（注：黑色是点集 1，红色是点集 2）

②利用仿射矩阵求变换参数：

第一组数据：
选取的一对全等三角形（6 个点）为：

x1=79;y1=70.3333;
x2=103.6667;y2=94.6667;
x3=153.6667;y3=83.6667;

x1’=38.6667;y1’=86;
x2’=61.3333;y2’=112.3333;
x3’=113.6667;y3’=104.6667;

求出的仿射矩阵参数为：

D ==[a11 a12; a21 a22]= 1.0234 -0.1059
0.0693 1.0119
E =[t1 t2]=[-34.7326, 9.3521]

观察矩阵D可以看出，D与单位矩阵误差很小，说明第一组数据几乎不存在旋转。

第二组数据：
选取的一对全等三角形（6 个点）为：

x1=887;y1=329;
x2=911;y2=371;
x3=687;y3=239;

x1’=1159;y1’=51248;
x2’=1153;y2’=51200;
x3’=1309;y3’=51408;

求出的仿射矩阵参数为：

D=[a11 a12; a21 a22]= -0.9231 0.3846
0.3846 -0.9231

E=[t1 t2]=[0.1851, 5.1893]

③去掉误差过大的点（缺少点、伪点）：

第一组数据求出的匹配点（共 17 对）：
（程序见附录 2）

点集 1（x1,y1）	点集 2（x2,y2）	110 114	63 163
62 62	22 72	114 81	73 99
70 125	23 142	123 120	76 139
81 32	46 46	140 109	96 129
86 54	47 70	14 146	97 168
88 26	53 42	146 50	108 69
88 92	44 108	150 146	105 167
95 29	59 45	165 55	128 78
106 111	62 129	178 46	140 67

点集 1 中剩余的 9 个点为缺失点：
（19，117），（20，20），（23，111），（28，20），（39，18），（41，140），（53，105），（1，108），（92，170）
点集 2 中剩余的 10 个点为伪点：
（24，20），（34，126），（122，29），（158，28），（161，163），（162，37），（171，53），（171，118），（175，41），（181，102）
第二组数据求出的匹配点（共 93 对）：
（程序见附录 3）

点集 1（x1,y1）	点集 2（x2,y2）	770 497	1332 51138
772 344	1270 51279	445 859	1771 50932
844 154	1130 51425	23 998	2214 50968
510 805	1692 50957	428 203	1541 51541
747 307	1280 51323	883 154	1094 51409
122 631	1988 51267	936 540	1196 51034
871 854	1378 50770	79 83	1806 51788
50 833	2125 51109	866 158	1111 51412
6 123	1889 51779	679 984	1604 50725
706 612	1440 51056	471 660	1670 51105
706 612	1436 51057	377 318	1623 51455
858 336	1188 51252	498 424	1553 51312

641	282	1367	51387	498	424	1550	51317
990	683	1202	50881	769	804	1452	50856
762	879	1488	50791	95	955	2131	50979
460	865	1760	50920	630	61	1291	51594
887	329	1159	51248	702	222	1287	51417
521	203	1447	51505	184	118	1723	51715
493	377	1539	51357	579	670	1574	51053
277	306	1711	51505	77	228	1864	51655
241	389	1776	51443	311	805	1874	51033
253	685	1880	51167	112	600	1977	51300
427	483	1641	51285	540	101	1390	51592
355	657	1775	51153	477	155	1469	51567
423	216	1541	51531	258	748	1900	51107
531	855	1690	50902	770	497	1332	51138
797	482	1301	51142	445	859	1771	50932
400	371	1623	51399	23	998	2214	50968
778	906	1484	50759	428	203	1541	51541
727	470	1361	51180	883	154	1094	51409
662	192	1313	51460	936	540	1196	51034
886	366	1174	51214	79	83	1806	51788
813	40	1114	51542	866	158	1111	51412
733	215	1256	51412	679	984	1604	50725
444	656	1693	51119	471	660	1670	51105
972	331	1080	51213	377	318	1623	51455
44	940	2172	51013	498	424	1553	51312
155	972	2082	50940	498	424	1550	51317
214	146	1706	51677	769	804	1452	50856
685	578	1442	51097	95	955	2131	50979
883	758	1329	50854	630	61	1291	51594
247	752	1912	51107	702	222	1287	51417
231	875	1975	51000	184	118	1723	51715
702	615	1440	51056	579	670	1574	51053
702	615	1436	51057	77	228	1864	51655
541	197	1426	51503	311	805	1874	51033
78	396	1929	51500	112	600	1977	51300

点集 1 中剩余的 12 个点为缺失点：（有接近重合的点匹配了两次）

（279，26），（508,282），（165,326），（337，597），（389，529），（136，586），（20，624），（297，598），（793，608），（629，490），（752，260），（108，656）

点集 2 中剩余的 3 个点为伪点：

（1256，52364），（1400，51762），（1489，51737）

④情况 1：只有平移：

匹配的 17 个点用最小二乘法在 Matlab 中计算得：

ave_x = 114.3529　　% ave_x 是点集1中x坐标的平均值
ave_y = 73.0588　　% ave_y 是点集2中匹配点x’坐标的平均值
ave_xy = 9.4604e+003　　% ave_xy 是点集1中x坐标与x’坐标的平均值
ave_xx = 1.4186e+004　　% ave_x 是点集1中x坐标的平方的平均值
m=(ave_x.*ave_xy-ave_y.*ave_xx)/(ave_x.^2-ave_xx);
m = -40.9229
即 x_{2i}= -40.9229 +x_{1i}

ave_x = 84　　% ave_x 是点集 1 中 y 坐标的平均值
ave_y = 102.2353
ave_xy = 10376
ave_xx = 8.7978e+003
n=(ave_x.*ave_xy-ave_y.*ave_xx)/(ave_x.^2-ave_xx);
n = 15.9942
即 y_{2i}=15.9942 +y_{1i}

将点集 1 中的点（x_1，y_1）经平移变换（x_1-40.9229，y_1+15.9942）得到的象点与点集 2 中的点进行匹配。

第一组数据匹配后的点如下图：
（注：红色是点集 2，蓝色是点集 1 经变换后的点）

⑤情况 2：既有平移又有旋转：

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

取所有匹配点对。将点集 1 中的点（x,y）坐标取平均值代入仿射矩阵，求出（x’,y’）

第二组数据的原始数据散点图如下：
（注：黑色是第一组点，红色是第二组点）

第二组数据去掉缺失点和伪点点之后匹配的点如下图：
（注：红色是点集 2，蓝色是点集 1 经变换后的点）

⑥计算误差及特征点集评价：

误差=匹配点的平均偏移距离
第一组数据：误差 e=1.2765
第二组数据：误差 e=2.7562

匹配度=匹配点对数*2/（点集 1 点数+点集 2 点数）
第一组数据：匹配度=17*2/（26+27)=64.15% > 60%，可以认为是同一特征点集
第二组数据：匹配度=93*2/（100+96）=94.90%>60%，可以认为是同一特征点集

6）模型评价：

①模型的优点：

1、 通过找两个点集之间的全等三角形来确定整个图形的变换，具有较高的准确性。同时三个点也是最为简便的一种方法。一个点和两个点无法唯一确定既有平移又有旋转的变换。
2、 这种三角形的方法比“用图像所有点的重心确定平移变换”的方法更优，因为此方法并没有使用所有数据点，可避免缺失点和伪点对图像配准的影响。达到了统计学中“舍弃误差过大点”的要求。
3、 情况 2 中，两图像之间既存在平移变换，又存在旋转变换，而旋转中心未知，用仿射矩阵来确定变换系数无需寻找旋转中心。
4、 次方法具有一般性，对被处理图像没有过高要求，不必限制平移和旋转量，不仅能处理两幅图像间的微小改变，也能处理有较大变化的图像。

②模型的缺点：

1、 若对于点分布特别均匀，同一点集出现多对全等三角形或出现等边三角形的情况，此方法效果将不够理想。

2、 在找全等三角形的过程中是在长度相等的边中遍历，进行筛选，得到三边长度相等的全等三角形，虽已经经过改进，但仍类似于穷举，有待进一步改进。

③模型的改进意见：

1、经改进可使之适合非线性变换。若第一幅图像中的一条直线经变换后，映射至第二幅图像上不再是直线，我们把这样的变换称为非线性变换。在二维空间中，点 (x_1, y_1) 经非线性变换至点 (x_2, y_2) 变换公式为：

$$(x_2, y_2) = F(x_1, y_1)$$

F 表示把第一幅图像映射到第二幅图像上的任意一种函数形式。典型的非线性变换如多项式变换，在 2D 空间中，多项式函数可写成如下形式：

$$x_2 = a_{00} + a_{10}x_1 + a_{01}y_1 + a_{20}x_1^2 + a_{11}x_1y_1 + a_{02}y_1^2 + ...$$

$$y_2 = b_{00} + b_{10}x_1 + b_{01}y_1 + b_{20}x_1^2 + b_{11}x_1y_1 + b_{02}y_1^2 + ...$$

2、实际的图像配准中，图像多是灰度图，需要找到突出点作为特征点。如线交叉点、物体边缘点（灰度在某处偏导数较大的点）。

7）参考文献：

1、刘道海 [1] 孙作龙 [2] 黄樟灿 [3]，图像匹配问题的新算法，2002

2、杨启帆. 何勇. 谈之奕，数学建模竞赛——浙江大学学生获奖论文点评，浙江大学出版社

3、姚列明，结构化大学物理实验，高等教育出版社

4、郑雪梅，图像配准技术

附录:

附录 1：遍历法找全等三角形 Matlab 程序及运行结果：

%录入第一个点集和第二个点集的数据%

```
x1=[772 344;844 154;510 805;747 307;279 26;122 631;871 854;50 833;6 123;706
612;858 336;641 282;990 683;762 879;460 865;508 282;887 329;521 203;493
377;277 306;241 389;253 685;427 483;355 657;423 216;165 326;531 855;797
482;400 371;778 906;727 470;662 192;886 366;813 40; 733 215;444 656;337
597;972 331;44 940;155 972;214 146;685 578;883 758;247 752;231 875;389 529;702
615;541 197;78 396;136 586;770 497;445 859;23 998;428 203;883 154; 20 624;936
540;79 83;866 158;679 984;471 660;377 318;498 424;769 804;95 955;630 61;702
222;184 118;297 598;793 608;579 670;77 228;629 490;311 805; 112 600;540
101;477 155;258 748;147 877; 856 471;681 354;687 239;491 752;759 532;911 371;
159 583;861 375;140 791;434 824;51 820;293 501;499 418;391 189;283 762;832
752;336 648;752 260;108 656;841 982; 885 482];
x2=[1670 51105;1374 50879;1440 51056;2214 50968;1174 51214;1270 51279;1130
51425;1950 51313;1400 51762;1988 51267;1378 50770; 1977 51300; 1975 51000;
1436 51057; 1367 51387;1202 50881; 1760 50920;1489 51737;1159 51248; 1220
51107;1426 51503;1188 51252; 1771 50932;1280 51323;1539 51357;1711
51505;1776 51443;1880 51167; 2082 50940;1641 51285; 1447 51505; 1775
51153;1541 51531;1622 51531;1690 50902;1301 51142;1623 51399;1484
50759;1361 51180;1313 51460; 2125 51109;1256 51412;1693 51119; 1080
51213;2172 51013;1706 51677;1442 51097;1488 50791;1329 50854;1912 51107;
1692 50957; 1929 51500;1332 51138; 1541 51541;1094 51409; 1604 50725;1623
51455;1553 51312; 2131 50979;1889 51779;2026 51113;1291 51594; 2103
51246;1455 50664;1723 51715; 1354 51127; 1359 51200;1874 51033;1390
51592;1469 51567;1900 51107;2053 51031; 1243 51129;1358 51305;1309
51408;1687 51012;1356 51110;1153 51200; 1927 51297;1201 51215; 1768
50968;2119 51121;1772 51320;1550 51317;1559 51571;1883 51084; 1789
51168;1256 52364; 1574 51053; 1111 51412; 1864 51655; 1196 51034;1806 51788;
1452 50856; 1287 51417;1114 51542];
x1=x1';
x2=x2';
%找出由 x1 构成的边集%
length11=0;
for i=1:100
    for j=(i+1):100
        length11=length11+1;
    end
end
distance1=cell(1,length11);
ii=0;
```

```
for i=1:100
    for j=(i+1):100
        d1(1,1)=(x1(1,i)-x1(1,j))^2+(x1(2,i)-x1(2,j))^2;
        d1(2,1)=i;
        d1(3,1)=j;
        ii=ii+1;
        distance1{1,ii}=d1;
    end
end
%找出由 x2 构成的边集%
length22=0;
ii=0;
for i=1:96
    for j=(i+1):96
        length22=length22+1;
    end
end
distance2=cell(1,length22);
for i=1:96
    for j=(i+1):96
        d2(1,1)=(x2(1,i)-x2(1,j))^2+(x2(2,i)-x2(2,j))^2;
        d2(2,1)=i;
        d2(3,1)=j;
        ii=ii+1;
        distance2{1,ii}=d2;
    end
end
%找出对应相等的边集%
length_edge_xiangdeng=0;
for i=1:4950
    for j=1:4560
        if(abs(distance1{1,i}(1,1)^2-distance2{1,j}(1,1)^2)<1000)
            length_edge_xiangdeng=length_edge_xiangdeng+1;
        end
    end
end
edge_xiangdeng=cell(1,length_edge_xiangdeng);
ii=0;
```

```
for i=1:4950
    for j=1:4560
        if(abs(distance1{1,i}(1,1)^2-distance2{1,j}(1,1)^2)<1000)
            ii=ii+1;
            edge_xiangdeng{1,ii}=distance1{1,i};
            edge_xiangdeng{2,ii}=distance2{1,j};
        end
    end
end
%在全等的边中找出全等三角形%
length_triangle=0;
for i=1:197
    for j=(i+1):197
        for k=(j+1):197
            x=[edge_xiangdeng{1,i}(2,1) edge_xiangdeng{1,i}(3,1)
edge_xiangdeng{1,j}(2,1) edge_xiangdeng{1,j}(3,1) edge_xiangdeng{1,k}(2,1)
edge_xiangdeng{1,k}(3,1)];
            y=[edge_xiangdeng{2,i}(2,1) edge_xiangdeng{2,i}(3,1)
edge_xiangdeng{2,j}(2,1) edge_xiangdeng{2,j}(3,1) edge_xiangdeng{2,k}(2,1)
edge_xiangdeng{2,k}(3,1)];

if((length(find(x==edge_xiangdeng{1,i}(2,1)))==2)&&(length(find(x==edge_xiangd
eng{1,i}(3,1)))==2)&&(length(find(x==edge_xiangdeng{1,j}(2,1)))==2)&&(length(f
ind(x==edge_xiangdeng{1,j}(3,1)))==2)&&(length(find(x==edge_xiangdeng{1,k}(2,
1)))==2)&&(length(find(x==edge_xiangdeng{1,k}(3,1)))==2)&&(length(find(y==ed
ge_xiangdeng{2,i}(2,1)))==2)&&(length(find(y==edge_xiangdeng{2,i}(3,1)))==2)&
&(length(find(y==edge_xiangdeng{2,j}(2,1)))==2)&&(length(find(y==edge_xiangde
ng{2,j}(3,1)))==2)&&(length(find(y==edge_xiangdeng{2,k}(2,1)))==2)&&(length(fi
nd(y==edge_xiangdeng{2,k}(3,1)))==2))
                length_triangle=length_triangle+1;
            end
        end
    end
end
ii=0;
triangle_quandeng=cell(6,length_triangle);
for i=1:197
    for j=(i+1):197
```

```
        for k=(j+1):197
            x=[edge_xiangdeng{1,i}(2,1) edge_xiangdeng{1,i}(3,1)
edge_xiangdeng{1,j}(2,1) edge_xiangdeng{1,j}(3,1) edge_xiangdeng{1,k}(2,1)
edge_xiangdeng{1,k}(3,1)];
            y=[edge_xiangdeng{2,i}(2,1) edge_xiangdeng{2,i}(3,1)
edge_xiangdeng{2,j}(2,1) edge_xiangdeng{2,j}(3,1) edge_xiangdeng{2,k}(2,1)
edge_xiangdeng{2,k}(3,1)];

if((length(find(x==edge_xiangdeng{1,i}(2,1)))==2)&&(length(find(x==edge_xiangd
eng{1,i}(3,1)))==2)&&(length(find(x==edge_xiangdeng{1,j}(2,1)))==2)&&(length(f
ind(x==edge_xiangdeng{1,j}(3,1)))==2)&&(length(find(x==edge_xiangdeng{1,k}(2,
1)))==2)&&(length(find(x==edge_xiangdeng{1,k}(3,1)))==2)&&(length(find(y==ed
ge_xiangdeng{2,i}(2,1)))==2)&&(length(find(y==edge_xiangdeng{2,i}(3,1)))==2)&
&(length(find(y==edge_xiangdeng{2,j}(2,1)))==2)&&(length(find(y==edge_xiangde
ng{2,j}(3,1)))==2)&&(length(find(y==edge_xiangdeng{2,k}(2,1)))==2)&&(length(fi
nd(y==edge_xiangdeng{2,k}(3,1)))==2))
                ii=ii+1;
                triangle_quandeng{1,ii}=edge_xiangdeng{1,i};
                triangle_quandeng{2,ii}=edge_xiangdeng{1,j};
                triangle_quandeng{3,ii}=edge_xiangdeng{1,k};
                triangle_quandeng{4,ii}=edge_xiangdeng{2,i};
                triangle_quandeng{5,ii}=edge_xiangdeng{2,j};
                triangle_quandeng{6,ii}=edge_xiangdeng{2,k};
            end
        end
    end
end
```

运行结果：

```
 triangle_quandeng{1,1}

ans =

        9209
          10
          84
```

```
>> triangle_quandeng{2,1}

ans =

       80485
          10
          92

>> triangle_quandeng{3,1}

ans =

       80596
          84
          92

>> triangle_quandeng{4,1}

ans =

        9209
          14
          77

>> triangle_quandeng{5,1}

ans =

       80485
          77
          84

>> triangle_quandeng{6,1}

ans =

       80596
          14
```

```
            84

>> triangle_quandeng{1,2}

ans =

        48100
           17
           82

>> triangle_quandeng{2,2}

ans =

         2340
           17
           85

>> triangle_quandeng{3,2}

ans =

        67600
           82
           85

>> triangle_quandeng{4,2}

ans =

        48100
           19
           75

>> triangle_quandeng{5,2}

ans =
```

```
        2340
          19
          78

>> triangle_quandeng{6,2}

ans =

       67600
          75
          78
```

附录 2：第一组匹配点的寻找及误差计算 程序及结果：

```
clc;clear all;

A=[19 20  23 28 39 41 53 62 70 81 86 1 88 88 92 95 106 110 114 123 140  142 146
150 165 178 ; 117  20 111 20 18 140 105 62 125 32 54 108 26 92 170 29 111 144 81
120 109 146 50 146 55 46];
B=[22 23 24 34 44 46 47 53 59 62 63 73 76 96 97 105 108 122 128 140 158 161 162
171 171 175 181; 77 142 20 126 108 46 70 42 45 129 163 99 139 129 168 167 69 29
78 67 28 163 37 53 118 41 102];

%使用匹配点
H=[A(1,8),A(1,9),A(1,10),A(1,11),A(1,13),A(1,14),A(1,16),A(1,17),A(1,18),A(1,19),
A(1,20),A(1,21),A(1,22),A(1,23),A(1,24),A(1,25),A(1,26);A(2,8),A(2,9),A(2,10),A(2,
11),A(2,13),A(2,14),A(2,16),A(2,17),A(2,18),A(2,19),A(2,20),A(2,21),A(2,22),A(2,2
3),A(2,24),A(2,25),A(2,26)];
G=[B(1,1),B(1,2),B(1,6),B(1,7),B(1,8),B(1,5),B(1,9),B(1,10),B(1,11),B(1,12),B(1,13),
B(1,14),B(1,15),B(1,17),B(1,16),B(1,19),B(1,20);B(2,1),B(2,2),B(2,6),B(2,7),B(2,8),
B(2,5),B(2,9),B(2,10),B(2,11),B(2,12),B(2,13),B(2,14),B(2,15),B(2,17),B(2,16),B(2,1
9),B(2,20)];
H'
G'
```

```
%用三个全等三角形的重心确定一个三角形，从而确定变换所需的对应点
sum1=0;s1=0;sum2=0;s2=0;sum3=0;s3=0;sum_1=0;s_1=0;sum_2=0;s_2=0;sum_3=0;
s_3=0;
for i=2:4
    sum1=sum1+H(1,i);
    s1=s1+H(2,i);
    sum_1=sum_1+G(1,i);
    s_1=s_1+G(2,i);
end;
for i=7:9;
    sum2=sum2+H(1,i);
    s2=s2+H(2,i);
    sum_2=sum_2+G(1,i);
    s_2=s_2+G(2,i);
end;
for i=14:16
    sum3=sum3+H(1,i);
    s3=s3+H(2,i);
    sum_3=sum_3+G(1,i);
    s_3=s_3+G(2,i);
end;
x1=sum1/3;
y1=s1/3;
x2=sum2/3;
y2=s2/3;
x3=sum3/3;
y3=s3/3;

x_1=sum_1/3;
y_1=s_1/3;
x_2=sum_2/3;
y_2=s_2/3;
x_3=sum_3/3;
y_3=s_3/3;

%计算仿射矩阵的参数
```

```
a11= (x_1*(y2 - y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (x_2*(y1 -
y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) + (x_3*(y1 - y2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
a12= (x_2*(x1 - x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (x_1*(x2 -
x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (x_3*(x1 - x2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
a21= (y_1*(y2 - y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (y_2*(y1 -
y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) + (y_3*(y1 - y2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
a22= (y_2*(x1 - x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (y_1*(x2 -
x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (y_3*(x1 - x2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
t1=    (x_3*(x1*y2 - x2*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) -
(x_2*(x1*y3 - x3*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) +
(x_1*(x2*y3 - x3*y2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
t2=    (y_3*(x1*y2 - x2*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) -
(y_2*(x1*y3 - x3*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) +
(y_1*(x2*y3 - x3*y2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);

%仿射变换矩阵 A1=D*A+E
D=[a11 a12; a21 a22];
E=[t1 t2]';



%得到变换后的矩阵
H1=zeros(size(H));
for i=1:length(H)
    C=[H(1,i),H(2,i)]';
    C1=D*C+E;
    H1(1,i)=C1(1,1);
    H1(2,i)=C1(2,1);
end;

%测定误差,距离的平均
sum=0;
for i=1:length(H)
    sum=sum+sqrt((H1(1,i)-G(1,i))^2+(H1(2,i)-G(2,i))^2);
```

```
end;
e=sum/length(H)



figure
%plot(A(1,:)',A(2,:)','o','markersize',4,'markerfacecolor','k','markeredgecolor','k')
%hold on
plot(G(1,:)',G(2,:)','o','markersize',4,'markerfacecolor','r','markeredgecolor','r')
hold on
plot(H1(1,:)',H1(2,:)','o','markersize',4,'markerfacecolor','b','markeredgecolor','b')
```

运行结果：

```
ans =

    62    62
    70   125
    81    32
    86    54
    88    26
    88    92
    95    29
   106   111
   110   144
   114    81
   123   120
   140   109
   142   146
   146    50
   150   146
   165    55
   178    46


ans =

    22    77
    23   142
```

```
    46    46
    47    70
    53    42
    44   108
    59    45
    62   129
    63   163
    73    99
    76   139
    96   129
    97   168
   108    69
   105   167
   128    78
   140    67
```

e =

```
    1.2765
```

附录 3：第二组匹配点的寻找及误差计算 程序及结果：

```
A=[772 344;844 154;510 805;747 307;279 26;122 631;871 854;50 833;6 123;706
612;858 336;
641 282;990 683;762 879;460 865;508 282;887 329;521 203;493 377;277 306;241
389;
253 685;427 483;355 657;423 216;165 326;531 855;797 482;400 371;778 906;727
470;
662 192;886 366;813 40; 733 215;444 656;337 597;972 331;44 940;155 972;214 146;
685 578;883 758;247 752;231 875;389 529;702 615;541 197;78 396;136 586;770
497;
445 859;23 998;428 203;883 154; 20 624;936 540;79 83;866 158;679 984;471 660;
```

377 318;498 424;769 804;95 955;630 61;702 222;184 118;297 598;793 608;579 670;
77 228;629 490;311 805; 112 600;540 101;477 155;258 748;147 877; 856 471;681
354;
687 239;491 752;759 532;911 371; 159 583;861 375;140 791;434 824;51 820;293
501;
499 418;391 189;283 762;832 752;336 648;752 260;108 656;841 982; 885 482
]';
B=[1670 51105;1374 50879;1440 51056;2214 50968;1174 51214;1270 51279;1130
51425;
1950 51313;1400 51762;1988 51267;1378 50770; 1977 51300; 1975 51000; 1436
51057;
1367 51387;1202 50881; 1760 50920;1489 51737;1159 51248; 1220 51107;1426
51503;
1188 51252; 1771 50932;1280 51323;1539 51357;1711 51505;1776 51443;1880
51167;
2082 50940;1641 51285; 1447 51505; 1775 51153;1541 51531;1622 51531;1690
50902;
1301 51142;1623 51399;1484 50759;1361 51180;1313 51460; 2125 51109;1256
51412;
1693 51119; 1080 51213;2172 51013;1706 51677;1442 51097;1488 50791;1329
50854;
1912 51107; 1692 50957; 1929 51500;1332 51138; 1541 51541;1094 51409; 1604
50725;
1623 51455;1553 51312; 2131 50979;1889 51779;2026 51113;1291 51594; 2103
51246;
1455 50664;1723 51715; 1354 51127; 1359 51200;1874 51033;1390 51592;1469
51567;
1900 51107;2053 51031; 1243 51129;1358 51305;1309 51408;1687 51012;1356
51110;
1153 51200; 1927 51297;1201 51215; 1768 50968;2119 51121;1772 51320;1550
51317;
1559 51571;1883 51084; 1789 51168;1256 52364; 1574 51053; 1111 51412; 1864
51655;
1196 51034;1806 51788; 1452 50856; 1287 51417;1114 51542
]';

%用三个全等三角形的重心确定一个三角形，从而确定变换所需的对应点
x1=887;

```
y1=329;
x2=911;
y2=371;
x3=687;
y3=239;

x_1=1159;
y_1=51248;
x_2=1153;
y_2=51200;
x_3=1309;
y_3=51408;

%计算仿射矩阵的参数
a11= (x_1*(y2 - y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (x_2*(y1 -
y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) + (x_3*(y1 - y2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
a12= (x_2*(x1 - x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (x_1*(x2 -
x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (x_3*(x1 - x2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
a21= (y_1*(y2 - y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (y_2*(y1 -
y3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) + (y_3*(y1 - y2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
a22= (y_2*(x1 - x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (y_1*(x2 -
x3))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (y_3*(x1 - x2))/(x1*y2 -
x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
t1=    (x_3*(x1*y2 - x2*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) -
(x_2*(x1*y3 - x3*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) +
(x_1*(x2*y3 - x3*y2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);
t2=    (y_3*(x1*y2 - x2*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) -
(y_2*(x1*y3 - x3*y1))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) +
(y_1*(x2*y3 - x3*y2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2);

%仿射变换矩阵 A1=D*A+E
D=[a11 a12; a21 a22];
E=[t1 t2]';
```

```
%得到变换后的矩阵 A1
A1=zeros(size(A));
for i=1:length(A)
    C=[A(1,i),A(2,i)]';
    C1=D*C+E;
    A1(1,i)=C1(1,1);
    A1(2,i)=C1(2,1);
end;

%判断 A1，B 中是否有匹配并计数%
length_pipei=0;
for i=1:100
    for j=1:96
        if(((A1(1,i)-B(1,j))^2+(A1(2,i)-B(2,j))^2)<100)
            length_pipei=length_pipei+1;
        end
    end
end

%将匹配的点存储起来%
point_pipei=cell(3,length_pipei);
ii=0;
for i=1:100
    for j=1:96
        if(((A1(1,i)-B(1,j))^2+(A1(2,i)-B(2,j))^2)<100)
            ii=ii+1;
            x1=[A(1,i);A(2,i)];
            x2=[B(1,j);B(2,j)];
            x3=[A1(1,i);A1(2,i)];
            point_pipei{1,ii}=x1;
            point_pipei{2,ii}=x2;
            point_pipei{3,ii}=x3;
        end
    end
end

%输出 A 和 B 中的对应点%
A=[];
```

```
for i=1:93
    B=[point_pipei{1,i}(1,1); point_pipei{1,i}(2,1)];
    A=[A,B];
end
B=[];
for i=1:93
    M=[point_pipei{2,i}(1,1); point_pipei{2,i}(2,1)];
    B=[B,M];
end

%计算误差%
sum=0;
for i=1:93

sum=sum+sqrt((point_pipei{2,i}(1,1)-point_pipei{3,i}(1,1))^2+(point_pipei{2,i}(2,1)
-point_pipei{3,i}(2,1))^2);
end
wucha=sum/length_pipei;

figure
%plot(A(1,:)',A(2,:)','o','markersize',4,'markerfacecolor','k','markeredgecolor','k')
%hold on
plot(B(1,:)',B(2,:)','o','markersize',4,'markerfacecolor','r','markeredgecolor','r')
hold on
plot(A1(1,:)',A1(2,:)','o','markersize',4,'markerfacecolor','b','markeredgecolor','b')
```

误差 e=2.7562