摘要

图像拼接是计算机视觉的重要组成部分，如何高效、高质量的完成图像拼接一直以来都是研究的热门话题。同时图像拼接也是许多其他计算机视觉问题的基础。

本文首先介绍图像拼接的基本步骤及其典型实现，在图像拼接中，变换矩阵的求得对于图像拼接的最终结果有着很大的影响，其传统方式是利用RANSAC 算法。遗传算法是进化算法中的一种，借鉴了生物进化学中的一些现象来解决最佳化问题。本文则着力于利用遗传算法改进变换矩阵的求值过程来优化图像拼接，利用遗传算法的传承性来改进迭代优化过程。

在本文应用中，我们利用遗传算法优化了变换矩阵的求取过程，在原来的算法中，变换矩阵的求取只考虑其覆盖的内点数量，但本文介绍的算法同时将内点距离偏差考虑了进去，从而在传统算法的基础上得到更高质量的变换矩阵，并提高拼接的质量。

关键词： 图像拼接 遗传算法 计算机视觉 图像变换矩阵 随机抽样一致算法

Abstract

Image stitching is an important part of computer vision, and it is a heated topic of how to do it more efficiently with high quality. In the meanwhile, image stitching is the basis of many other computer vision problems. The paper first introduces the basic conception of image stitching, in which how to calculate the transformation matrix is an important procedure. In practice, RANSAC is widely used in calculating the transform matrix. Genetic algorithm is useful to optimize problem. In this paper, the author will introduce a new way of using generic algorithm to get a better performance in calculating transformation matrix.

The new method uses the genetic algorithm to improve the process of calculating transformation matrix. It not only counts the number of interior points, but also takes standard error into consideration, so that it gets a higher quality transformation matrix and improves the result of the stitching.

Keywords: Image Stitching, Genetic Algorithm, Computer Vision, Transform Matrix, RANSA

目录

第一章 绪论

第 1 节 选题背景

图像拼接是计算机视觉中的重要研究部分，也是许多其他图像处理的基础。高效、高质量的图像拼接算法一直是计算机视觉中研究的重点。在以神经网络为代表的机器学习大热的今天，各种针对于图像的机器学习算法层出不穷，如利用卷积神经网络（CNN）进行的图像识别等。而作为这些算法的基础，如何获取到一个优质的可供下一步计算的图像也愈发重要，其中图像拼接作为一个代表，也是研究的重点之一。图像拼接的作用在于将众多零散的图片以尽可能还原的手段拼接成一副大的图形，而在这过程中，如何高效、高质量的完成图像拼接是一个重要的研究方向。

图像拼接主要分为图像特征识别、特征匹配、图像变换和图像融合等几个步骤。其中特征识别是在图像中提取出可供比较的特征，特征匹配是将相同或相似特征筛选出来，而变换则是将待拼图像进行变换，融合则是为了消除拼接接缝。

在特征匹配和图像变换的过程中，经常会出现由于误匹配而造成的图像重影或者匹配失败的现象，针对这一过程进行优化对图像拼接有着非常重要的影响。对这一过程的优化方式包括误匹配点的剔除，以及求取高质量的变换矩阵。

第 2 节 论文的主要工作

论文的主要工作是总结图像拼接的主要步骤及其主要实现方式，并介绍一些图像拼接方面的前沿技术，针对其中的某些算法进行重点的介绍和总结。

并且对于现有的拼接算法进行优化改进，提出了一种以遗传算法来对图像变换进行改进的方式，以提高其拼接质量或者拼接速度。并对最终结果进行对比测试。

第 3 节 论文的创新性

传统上对于特征点筛选以及变换矩阵的求得是利用 RANSAC 算法，本文在这种算法的基础上，利用遗传算法对其进行改进，对于最终拼接结果的质量有明显的提高。

第 4 节　论文的组织结构

第一章为绪论，主要是介绍本文的背景以及创新方向。

第二章为图像拼接的主要过程及典型算法。

第三章为图像拼接的实践及应用。

第四章为利用遗传算法改进的图像拼接手段，与传统算法结果进行对比，并对最终结果的测试及评判。

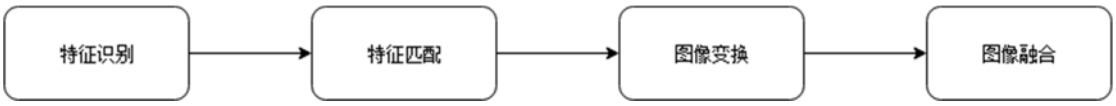
第五章对论文作了总结和展望。

第二章 图像拼接综述

本章主要介绍图像拼接的过程以及其典型算法。

第 1 节 图像拼接过程

图像拼接主要分为图像特征识别、特征匹配、图像变换和图像融合等几个步骤。



图片 1 图像拼接流程

图像特征识别主要是利用特征点提取算法，在图像中提取出特征点，特征点即标示着图像在某点处的特征，常用的算法有 SIFT, SURF, ORB 等。

特征匹配则是根据特征点的相似程度将其配对，即选出源图像和目标图像中的对应点，在这过程中需要剔除一些误匹配点，匹配常用的算法有 FLANN 和 Brute Force 算法，剔除误匹配常用 RANSAC 算法。

图像变换则是在已知匹配点的情况下，计算出将目标图像映射到源图像的变换矩阵，而变换方式主要又分为透视变换、仿射变换等。

图像融合则是将两幅图像尽可能无痕迹的拼合在一起，尽可能的消除接缝和光线差异等问题。

第 2 节 特征识别

特征识别的作用是从图片中提取出一些特殊位置，并且要求在不同图片中相同的特殊位置会有相同或近似的特征值，通过这些特征点之间的对应关系将不同图片匹配在一起。常用的特征识别有点与块的识别、边缘的识别和线条的识别。其共同点即要求可重复性，即前文所讲相同的特殊位置能求出相同或者相似的特征值。

在本项目里主要应用了点与块的特征识别方法，故本节主要介绍这种识别手段。

2.1. 特征识别的要求

特征识别除了对可重复性有要求外，还有其他几个方面的要求，即对于相

同特殊位置在不同尺度下所求出的特征值应该近似——尺度不变性；对于相同特殊位置在不同旋转角度下所求出的特征值应该近似——旋转不变性；只有在这两点满足的基础上，才能做出优秀的特征匹配。

2.2. 特征识别常用算法

在图像拼接中，常用的基于点与块特征识别算法有 SIFT (Scale Invariant Feature Transform, 尺度不变特征变换)、SURF (Speeded Up Robust Features, 加速稳健特征) 和 ORB (Oriented FAST and rotated BRIEF, 定向 FAST 及旋转 BRIEF)等。SURF 算法是一种类似 SIFT 的算法，其主要区别是特征点的筛选方式不同；而 ORB 则利用了改进的 FAST (Features From Accelerated Segment Test, 加速分割特征测试) 算法和 BRIEF (Binary Robust Independent Elementary Features, 二进制鲁棒独立基础特征) 算法来加速计算。

关于三种算法的性能对比，我们选取了三幅图片，并分别利用三种特征提取算法对其进行提取，重复十次记录平均时间，见表格 1。可以看出对不同图片进行特征提取时， SIFT 和 SURF 总体所花时间近似，而 ORB 则比它们快一个数量级。

表格 1 特征提取速度对比

图片	SURF 花费时间(s)	SIFT 花费时间(s)	ORB 花费时间(s)
Road-left	1.1208	0.4324	0.0408
Lake-left	0.7911	0.5234	0.0412
Tree-left	0.7272	0.4532	0.0394

在实践中，比较特征提取的质量来看， SIFT 的提取质量最高，其次是 SURF，提取质量最差的为 ORB；比较时间来看，SURF 所花费时间相对最高，SIFT 其次，而 ORB 在速度上可以比其他两个快一个数量级，所以许多应用选取 ORB 作为特征提取的手段。同时由于 SIFT 和 SURF 受专利保护，盈利应用需要购买许可，所以在很多应用中 ORB 算法的应用更多。

鉴于此，本文选取 ORB 为例，介绍特征识别的方式。

2.3. ORB 算法简介

ORB 特征是将 FAST 特征点的检测方法- 43 -[3]与 BRIEF 特征描述子[4]结合起来，并在它们原来的基础上做了改进与优化。

1. 利用 FAST 特征点检测的方法来检测特征点；
2. 利用 Harris[2] 角点的度量方法，从 FAST[2] 特征点中挑选出 Harris 角点响

应值最大的 N 个特征点；

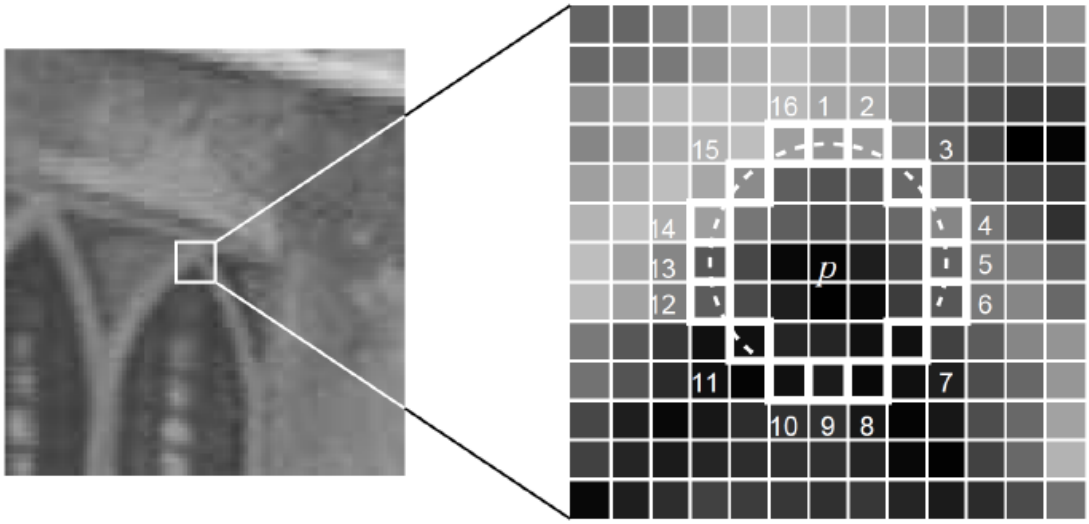
3. 解决局部不变性（尺度不变性和旋转不变性），即选择从不同高斯金字塔进行检测和对特征点添加角度特征；
4. 为特征点增加特征描述即描述子；

2.3.1 特征点查找

特征点查找即在图像中筛选出潜在的特征点，供下一步分析。

(1) FAST 检测

FAST 特征点的检测目标是角点，即在直观视觉中的“角”，其典型的检测方式即观察一个像素点周围半径为 3 的圆，从而判断该点是否为角点，如图示：



图片 2 角点示意图 (a) 角点 (b) 角点放大

而其判断方式则是在这个以点 p 为圆心，半径为 3 的圆上，若有连续 9 个点的灰度值与中心点之差大于或小于特定阈值，即点 p 与周围点的差异比较大，则点 p 为一个候选的 FAST 角点。此方法被称为 FAST-9，类似的还有 FAST-10、FAST-11、FAST-12 等。

(2) Harris 评判

Harris 方法是一种特征点评判方式，利用 Harris 方法将 FAST 方法选出的角点进行筛选，即从中挑出若干个 Harris 角点响应最大的角点作为特征点。

其中 Harris 角点的响应函数定义为：

$$R = det\mathbf{M} - \alpha(trace\mathbf{M})^2$$

(其中 det 指的是矩阵行列式的值；$trace$ 指的是矩阵的迹，即对角线元素之和；M 指的是像素矩阵)

2.3.2 描述子计算

在筛选出潜在的特征点之后，我们需要为其建立不同的描述方式用以区分和比较，描述子即用来区分比较特征点的方式。

由上文可以看出，利用 FAST 和 Harris 筛选出的角点并没有旋转不变性和尺度不变性，即其描述和旋转角度有关，也和其缩放程度有关，为解决这一问题，我们分别利用高斯金字塔和构建角度特征来解决。

(1) 尺度不变性[5]

尺度不变性在图像处理中的简单定义是无论图像处于何种状态（放大或者缩小），其特性应该不变。一个简单的例子就是对于一幅图片，无论人眼距离其远或者近，都能认出图片中的内容。而对于计算机而言，图像并不天然自带尺度不变性特征。尺度不变性的解决是通过构建不同高斯金字塔（即对图像进行缩放）分别进行特征检测，在实践中常选取 1.2 的缩放倍数缩放七层，在这种情况下不同半径大小的角点均可被查找，即可保证同一特征在不同缩放下均可被发现。

(2) 旋转不变性[2]

而旋转不变性的解决则是为每个特征点增加角度特征，使得其有统一的角度描述值，角度的计算公式即：

$$m_{pq} = \sum_{x,y} x^p \, y^q I(x,y)$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

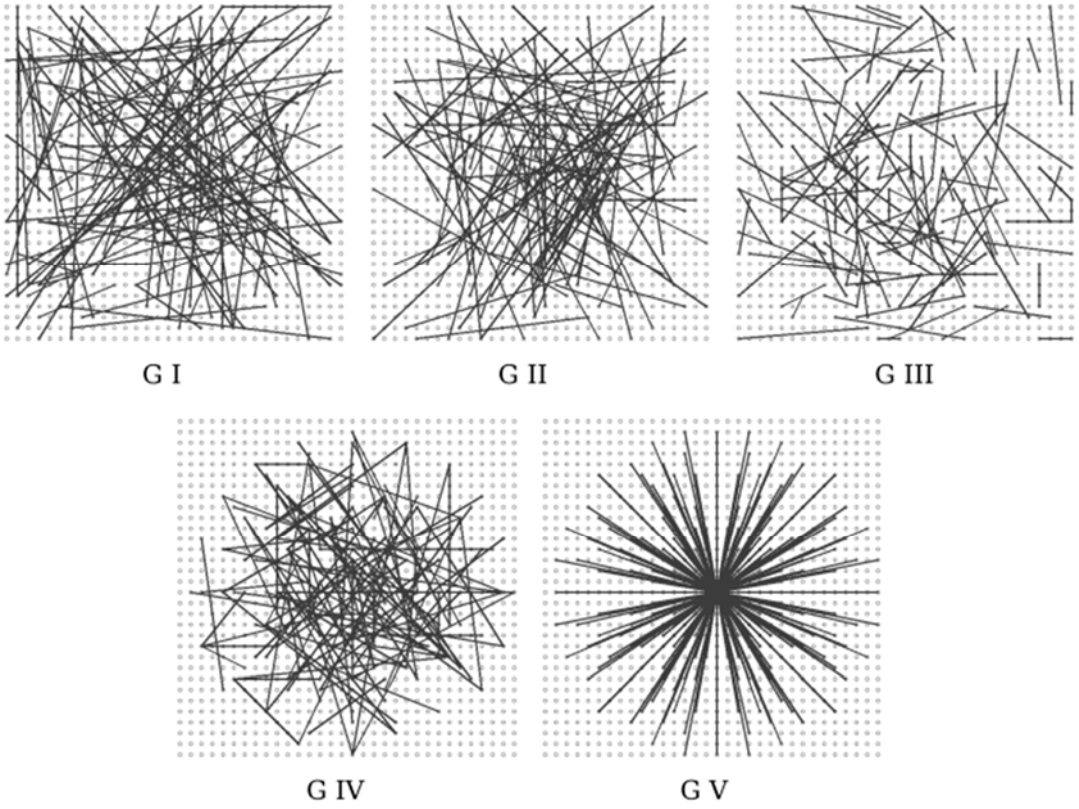
$$\theta = arctan(m_{01}, m_{10}) \qquad \text{公式 1}$$

其中 M_{pq} 的定义是在以特征点为中心的一个圆内，对所有像素值（以灰度为单位，其中I(x,y)为像素）以 x，y 坐标为权进行的加权和，即算出像素的“强度图心”。从而可利用公式 1 来计算出特征点的方向。强度图心假设一个角点的强度是它距离中心的偏差，并且这个向量可以用于计算方向。上面公式中的 C 式表示图心。后文将介绍如何利用这个角度来解决特征点的旋转不变特性。

(3) BRIEF 特征描述[4]

在 ORB 中，描述子使用 BRIEF 方法来进行的特征描述方式，其原理即在特征点一定邻域内按照一定的规则选取一些点对，并对这些点对的灰度值进行比较，将结果的布尔值储存下来，最终结果为长度为 128/256/512 长度的比特串，常用长度为 256 比特的二进制串来保存。在匹配时可通过比较不同二进制串之间的汉明距离来进行相似度比较。

下图为一些点对的选择方法：



图片 3 BRIEF 点对选择方式：I x, y 方向平均分布采样 II. x, y 均服从Gauss$(0, S^{2/25})$各向同性采样 III. x 服从Gauss$(0, S^{2/25})$，y 服从Gauss$(0, S^{2/100})$采样 IV. x, y 从网格中随机获取 V. x 一直在(0,0)，y 从网格中随机选取

所以我们可以得到特征点的 BRIEF 描述方式：

$$S = \begin{pmatrix} x_1 & x_2 & \cdots & x_{2n} \\ y_1 & y_2 & \cdots & y_{2n} \end{pmatrix}$$

其中 x_n 与 y_n 是一组点对里两个点的灰度值。

而这种简单的描述方式并不能解决旋转不变性，即在图像旋转时对相同的特征点并不能得到相同的描述。而为了解决这种问题，一个简单的做法是对于源图像分别进行多次旋转并进行多次的描述值计算，即 rBRIEF [6]。而显然这种做法会耗费大量的系统资源，一种更好的方法是 Steer BRIEF[2] ，即对于一

个特征点只计算一次描述值，其旋转不变性的解决则利用了第 二 章 2.3.2(2)所介绍的角度特征，从而综合 BRIEF 特征子的描述方式我们可以得到带角度特征的描述子：

$$S_\theta = R_\theta S$$

其中 R_θ 为旋转不变性（公式 1）求得的旋转角度:

$$R_\theta = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

利用这种方式，我们即完成了对一个特征点的描述，在这种描述方式下，不同尺度以及旋转方向的特征值均可以被该方法筛选出来。

第 3 节　特征匹配

3.1. 特征匹配要求

特征匹配即将来自不同图片的特征点进行匹配的过程，这一过程要求尽可能的将相同的点匹配，同时应尽力消去误匹配点。一个典型的匹配过程如下图：

图片 4 匹配示意图

两幅图上的圆圈代表利用 SIFT 方法检测出的特征点，两图间匹配的点利用线段相连。由于此图中的两个图是平移后的拍摄结果，所以其理想的匹配连线应平行。可以看出在在这个图片中所选取的匹配是合格的，同时观察到很多特

征点并未被选为匹配点。

所以一个优秀的匹配算法应在保证正确性的同时覆盖尽量多的点对。主流的匹配算法主要有两种：Brute Force (暴力匹配)[10] 和 FLANN (fast library for approximate nearest neighbors, 快速最近邻) 算法[9][11]。Brute Force 匹配即枚举所有结果，通过穷尽所有可能性来取得最佳结果，但相应的时间花费会较大。 FLANN 匹配则利用了优化的算法，在获取近乎最优的结果同时，极大的提高了匹配效率。

FLANN 由于其优秀的速度特性，广泛应用在图像拼接以及其他匹配领域，本文将对其进行简单介绍。

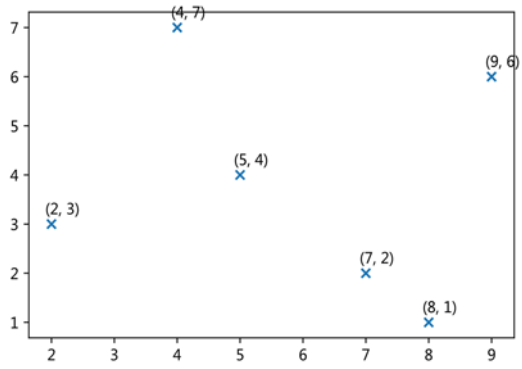
3.2. FLANN 算法简介[11]

FLANN 是一种高维数组的快速最近邻算法，其主要算法为随机 k-d (k-dimensional, k-元) 树算法和优先搜索 k-means (k 均值) 树算法。本文先介绍 k-d 树匹配法。

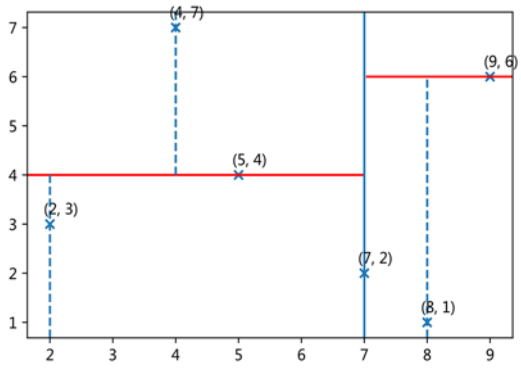
k-d 树是一种高维上的二叉树，其将高维数组按照一定的方式进行分类，从而使得查找比较由 $O(n)$ 级别降为了$O(log(n))$ 级别，从而大大加快了其查找速度。

3.2.1 k-d 树的构建

我们以一个例子来介绍一个典型的 k-d 树构建方式，这里我们用二维点做例子，有[2,3], [5,4], [9,6], [4,7], [8,1], [7,2]几个点，其在笛卡尔坐标系中的分布如下左图，而 k-d 树的构建即二分的过程。首先由 x 轴开始，点 [7,2]作为 x 坐标上的中间元素，选取其将坐标空间分为 $x < 7$ 和 $x > 7$两个部分。然后分别对左右部分利用 y 坐标进行同样的分类，如此重复，直到将所有空间进行分割。分割完成的图如右图。

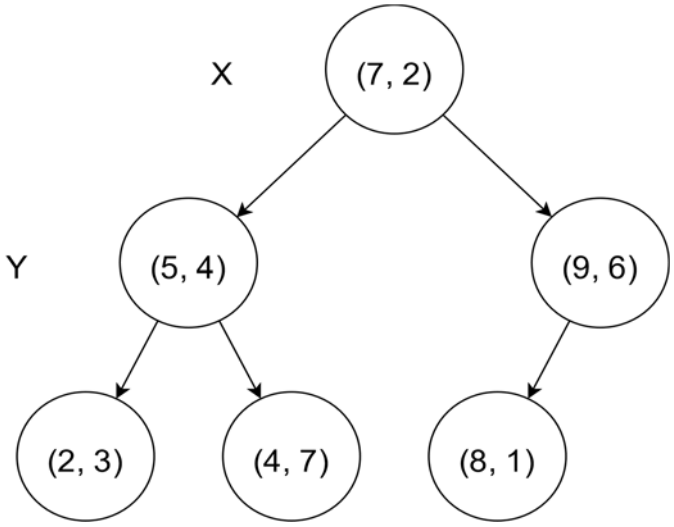


图片 5 k-d 树示意图



图片 6 k-d 树示意图 2

进行这样的分割后，我们即可得到如下所示的二叉树：



图片 7 k-d 二叉树示意图

3.2.2 k-d 树上的最近邻查找

在 k-d 树上进行数据的查找也是特征匹配的一个重要环节。

对于图像拼接中的应用来讲，我们的目的是寻找目标图像中与源图像中特征点最相似的点。所以我们先对源图像中的特征点建立 k-d 树，之后分别对目标图像中的特征点进行最近邻查找。

最近邻查找的方式是检索在已生成的 k-d 树上查询与某点最近的数据点，首先在其所在的区间中查找父节点，然后再查找其父节点对应的叶节点中距离最近的邻居。

由于这种查找为近似的二分查找，从而可以大大加快搜索速度。特别是在一张图片与多张图片进行匹配时，我们只需要对源图片建立一次 k-d 树即可以应用于之后的每次查询。

第 4 节　图像变换

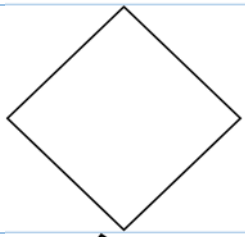
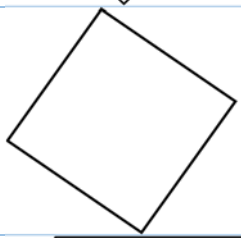
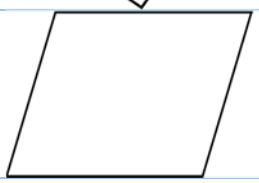
图像变换即将目标图片通过平移旋转等方式对应到源图片上的过程，这一过程的关键点在于利用已得到的匹配点间的匹配关系，得到合适的变换矩阵，并利用变换矩阵将图片进行正确的变换。

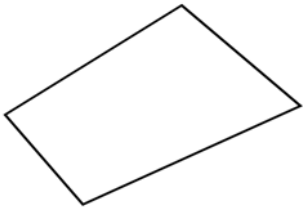
4.1.　图像变换方式概览

图像变换分为多种，在这里均讨论的是在单一变换矩阵下的变换，即利用同一个矩阵将源图片映射到目标图片上。针对其变换方式或者变换矩阵特性的不同，可分为平移变换、刚性变换、相似变换、仿射变换和透视变换（单应性变换）等[1]。其不同点在于不同的自由度。见表格 2，表中图示一栏即为一个正方形经过不同变换后的典型示意。

其中仿射变换和透视变换在图像变换中应用最多，其不同之处在于经过仿射变换后，源图像中平行的边仍是平行的，而经过透视变换后只能保证直线仍是直线。可以说仿射变换是透视变换的一个特例。

表格 2 常用图像变换特点[1]

变换	自由度数	保持	图示
平移	2	方向	
刚性（欧式）	3	长度	
相似	4	夹角	
仿射	6	平行性	

透视	8	直线性	

在应用中，由于透视变换自由度大，所以在图像拼接中对其应用也较多。本文的变换将以其为重点进行介绍。

4.2. 透视变换

透视变换的自由度为 8，即用 8 个参量进行图形变换，而在实践中，通常使用 3×3 矩阵来进行变换处理，其中最后一个元素恒为一，即在矩阵 M 中，$m_{33} = 1$ 。

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

关于透视变换矩阵的计算，由于其有八个参量，所以需要四组点对才能得出一个变换矩阵；相对的，仿射变换矩阵只需要三组点对即可求出。

4.2.1 透视变换矩阵[12]

假设点 $[x, y]$ 在 M的变换下的对应点 $[x', y']$ ，其计算公式为：

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} \quad \text{公式 2}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{u'}{w'} \\ \frac{v'}{w'} \\ 1 \end{bmatrix} \quad \text{(接上行) 公式 3}$$

透视变换矩阵中共有八个参量，其中各个参量的意义如下：

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

为旋转变换部分，用于实现图像的旋转。

$$\begin{bmatrix} m_{13} \\ m_{23} \end{bmatrix}$$

是平移变换部分，控制图片的平移。

$$[m_{31} \quad m_{32}]$$

则是透视变换部分，其参量控制着图像的透视变形。当这部分参数为 0

时，透视变换即退化为仿射变换。

4.2.2 透视矩阵的求解

由上小节我们可以将公式 2 和公式 3 改写为：

$$\begin{bmatrix} m_{11}x + m_{12}y + m_{13} \\ m_{21}x + m_{22}y + m_{23} \\ m_{31}x + m_{32}y + m_{33} \end{bmatrix} = \begin{bmatrix} x' \cdot (m_{31}x + m_{32}y + m_{33}) \\ y' \cdot (m_{31}x + m_{32}y + m_{33}) \\ m_{31}x + m_{32}y + m_{33} \end{bmatrix}$$

将上式移项整理可得：

$$a\,\vec{m} = b$$

其中：

$$a = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -x'y \\ 0 & 0 & 0 & x & y & 1 & -xy' & -yy' \end{bmatrix}$$

$$\vec{m} = [m_{11} \quad m_{12} \quad m_{13} \quad m_{21} \quad m_{22} \quad m_{23} \quad m_{31} \quad m_{32}]^T$$

$$b = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

而为了求得我们的变换矩阵，也即求出这里的 $\vec{m}$，我们需要四对点组进行求解。在有四对点组的情况下，我们可以将 a_n 和 b_n 竖向拼合形成新矩阵 (n=1，2，3，4)：

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

其中：

$$a_n = \begin{bmatrix} x_n & y_n & 1 & 0 & 0 & 0 & -x_n x_n' & -x_n' y_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_n y_n' & -y_n y_n' \end{bmatrix} \qquad \text{公式 4}$$

$$b_n = \begin{bmatrix} x_n' \\ y_n' \end{bmatrix} \qquad \text{公式 5}$$

$\vec{m}$ 的定义与之前相同，故有：

$$A\,\vec{m} = B$$

此时对于 m 的求解即变成了矩阵的除法：

$$m = A^{-1}B$$

将 $\vec{m}$ 添加 1作为尾元素并重新改变其形状为 3×3 矩阵，即可获得透视变换矩阵 M。

4.3. 多点变换矩阵求解

在变换过程中，如何获取透视矩阵是一个很重要的过程。比较典型的方法有 RANSAC (Random sample consensus, 随机抽样一致算法)，其主要思想是在众多匹配点中选取四对最有代表性的点来求算变换矩阵。

其主要思路是：

1. 随机从数据集中随机抽出 4 个样本数据 (此 4 个样本之间不能共线)，计算出变换矩阵 M；
2. 计算数据集中所有数据与模型 M 的投影误差，若误差小于阈值，将该点称为内点，将其加入内点集 I；
3. 如果当前内点集 I 元素个数大于最优内点集 I_{best}，则更新 $I_{best} = I$，同时更新迭代次数 k；
4. 如果迭代次数大于 k，则退出；否则迭代次数加 1，并重复上述步骤；

注：迭代次数 k 在不大于最大迭代次数的情况下，是在不断更新而不是固定的；

$$k = \frac{log(1-p)}{log(1-w^m)}$$

其中，p 为置信度，一般取 0.995；w 为当前“内点”占总点数的比例；m 为计算模型所需要的最少样本数，在此例中为 4；

在计算投影误差时，其计算方式为：

$$\sum_{i=1}^{n}[(x_i' - \frac{m_{11}x_i + m_{12}y_i + m_{13}}{m_{31}x_i + m_{32}y_i + m_{33}})^2 + (y_i' - \frac{m_{w1}x_i + m_{22}y_i + m_{23}}{m_{31}x_i + m_{32}y_i + m_{33}})^2]$$

RANSAC 算法的优点在于利用随机抽取而不是枚举，大大减少了总的计算量，提高了计算速度。但其相应的问题是由于抽取的简单随机性，会导致偏差的发生。

4.4. 图像卷绕[1]

在得到变换矩阵后，将源图像映射到变换的过程叫做图像卷绕 (Warp)。

前向卷绕是最简单的卷绕，即对源图像的每个像素位置进行变换，将源像素值复制到目标位置。但前向卷绕有一定的制约，例如当变换后的坐标不是整数时会导致一定的问题，若直接取整并复制会导致偏移和混叠，或者会有裂缝和空洞的出现。

所以在实践中更多的使用反向卷绕，反向卷绕的主要逻辑是由变换后的图

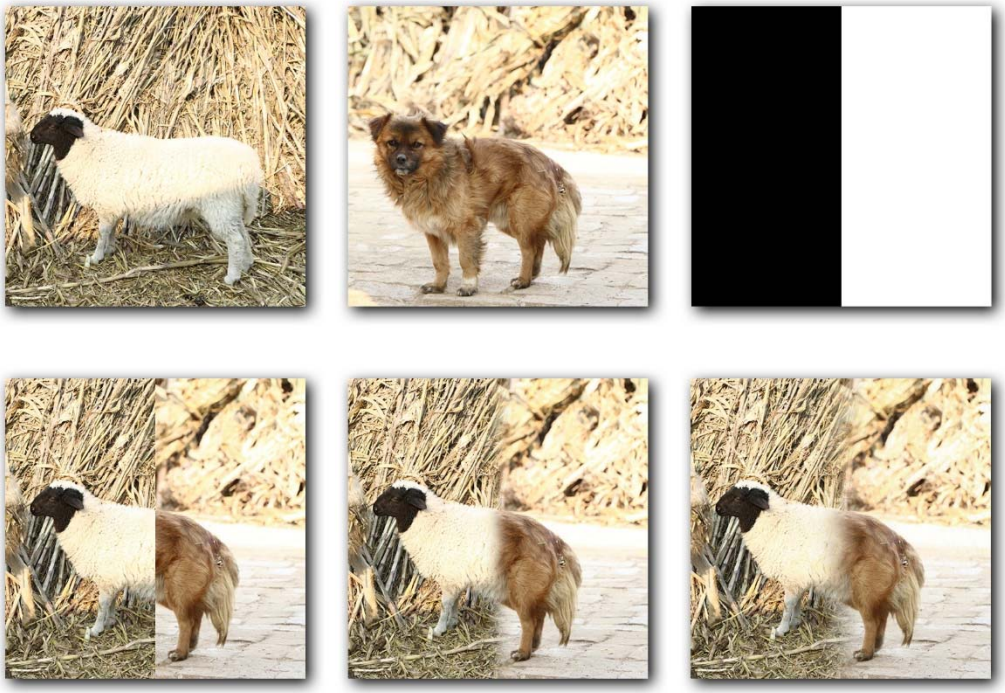
像像素出发，分别计算其在源图像上的对应位置，然后在相应位置进行重采样。由于对于目标图像每个像素都有了定义，所以不会再出现空洞。而重采样过程可以采用差值法来进行，常用的差值滤波器有最近邻、双线性、双三次窗和 sinc 函数等。双线性常用于对速度要求较高的地方，而双三次窗和 sinc 函数常用于对质量要求更高的地方。

第 5 节　图像融合

图像融合的目的是消除图像拼接中的接缝，虽然在优秀的匹配后，图像应大部分为对准的。但是在接缝处无可避免的会产生一些微小的偏差，而这种偏差在接缝处则会影响到图像观感，同时由于图片的曝光度的不同，接缝处也会显现出十分明显的差别，图像融合的目的就在于消除这种明显的接缝。

融合大致有几种技术手段：直接拼接、透明度拼接(Alpha 拼接)、高斯金字塔融合和泊松融合等。这些手法的融合效果依次提升，而同时所消耗的资源也越来越大。

下面图示即对比直接拼接、透明度拼接和高斯融合的不同之处，前两幅图片为原图片，第三幅黑白图像则为融合时使用的蒙版。



图片 8 融合方式示意图 (a) 左图 (b) 右图 (c) 蒙版 (d) 直接融合结果 (e) 透明度融合 (f) 高斯金字塔融合

5.1. 图像融合方式简介

最简单的图像拼合方式为直接拼合，在这一层面并不能算得上是真正的融合，其方法即直接将两张图片拼合，接缝会较明显。

相较于直接拼合，利用透明度拼接的图像接缝就不那么明显，其原理是首先准备一张作为蒙版的灰度图片，全黑处代表透明度为 1，全白代表透明度为 0，通过选择合适的蒙版，可得到最终的融合结果。我们假设有两张图片 A 和B 还有一个蒙版图片 M（这里假设它们拥有一样的尺寸），则融合结果可以由这样求得 $R = A \cdot M + B \cdot (1 - M)$。我们可以通过设置合理的蒙版让过渡更加柔和。

高斯金字塔的融合复杂度就较高，其简单原理是将图像拆分成拉普拉斯金字塔，然后在拉普拉斯金字塔上进行融合，最后再将拉普拉斯金字塔重建为图像，其优点是在拉普拉斯金字塔上各层的操作对于各个尺度分量都有比较柔和的拼接效果，最终效果也更优。

泊松融合则是利用了图像的梯度场进行融合，这种方式在两幅图像颜色有差异时效果最佳。其原理是利用图像的梯度场，将图像分为颜色部分和结构部分，在图像拼接时将一副图的结构梯度场复制到另一个图像上，这样对于图像的细节来说过度十分柔和，而颜色部分则由于继承了源图像，不至于产生太大的相差。但是由于其梯度场的计算和还原需要大量的运算资源，所以应用范围较小。

利用高斯金字塔进行融合在性能和效果上达到了均衡，本文重点对其进行介绍。

5.2. 高斯金字塔融合[13]

5.2.1 高斯金字塔的构建

高斯金字塔本质上为信号的多尺度表示法，即将同一信号或图片多次的进行高斯模糊，并且向下取样。其构建较为简单，对于一个图片来讲，对其进行连续降采样（即缩放），并将每次降采样后的结果储存下来，即构建成了一个关于该图片的高斯金字塔。

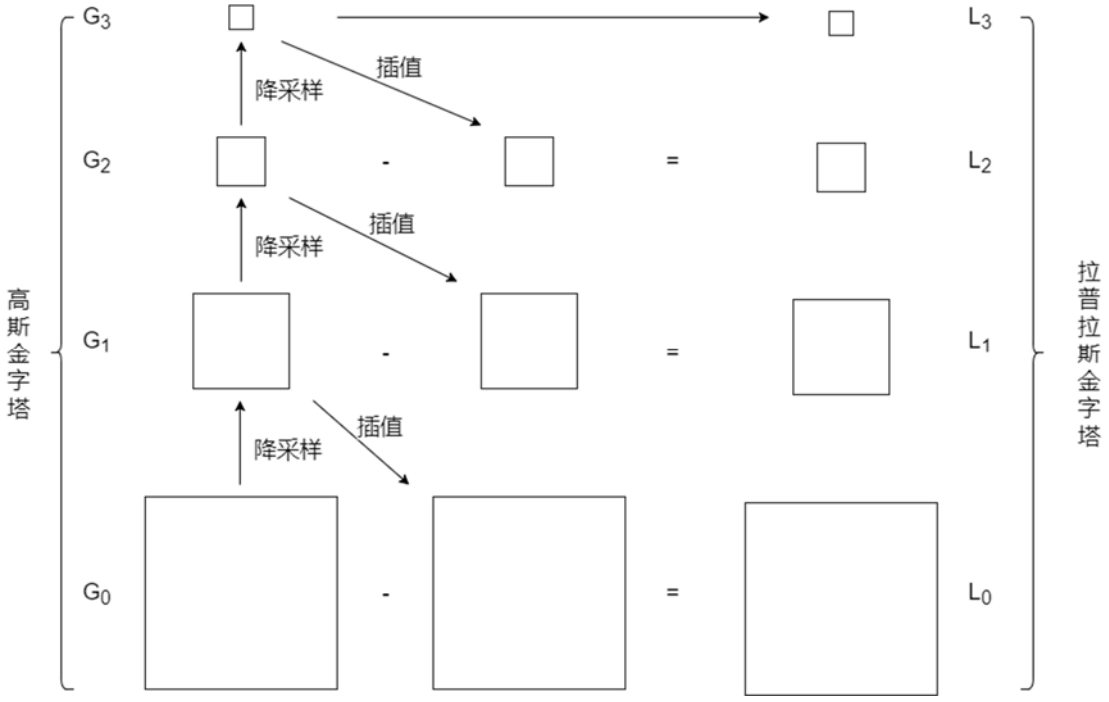
5.2.2 拉普拉斯金字塔的构建

计算机视觉中最有名以及应用最广的金字塔是 Burt and Adlson 的拉普拉斯

金字塔[1]。这个金字塔的构建过程如下：

1. 首先构建高斯金字塔——将图片降采样，将每一层降采样后的图片存为高斯金字塔的下一层，如此反复直到构建若干层；
2. 对于高斯金字塔的每一层进行插值重建，得到其原图的一个低通版本；
3. 对于每个重建层，对其依次相减，储存为新的金字塔，即为高斯金字塔；

下图所示即为一个四层的拉普拉斯金字塔的构建过程。



图片 9 高斯金字塔和拉普拉斯金字塔的构建

而得到拉普拉斯金字塔后，重建就相对简单，只需将拉普拉斯金字塔每层叠加即可得到重建的图形。

5.2.3 高斯金字塔的融合

在掌握如何构建拉普拉斯金字塔后，我们就可以利用其进行图像融合了。我们假设有两张图片 A 和 B 还有一个蒙版图片 M，其融合过程如下：

1. 对 A 和 B 分别建立 k 层的拉普拉斯金字塔 L_A 和 L_B；
2. 对蒙版建立 k 层的高斯金字塔 G_M；
3. 对于金字塔的每一层 n，都对应一个拉普拉斯金字塔 $L_{Rn} = L_{An} \cdot G_{Mn} + L_{Bn} \cdot (1 - G_{Mn})$；
4. 将得到的拉普拉斯金字塔累加得 $R = \sum_{i=0}^{k} L_{Ri}$；

最后即可得到我们最终的融合结果 R。在这种方法下，我们由于对图像的不同尺度均进行了一种类似透明度融合的操作，从而使图片的过渡更柔和。

第三章 图像拼接实践

第 1 节 软件环境

在本文实践中，我们选用了 OpenCV 库进行基础的运算。OpenCV 库是一个历史悠久的计算机视觉图形与界面库，其源代码由 C++ 完成，同时支持 Python、Java、JavaScript、C 等多个语言的接口。利用 OpenCV 库，我们可以方便的实现图像拼接中的特征识别、特征匹配和图像变换等基础部分。

本文选择了 OpenCV 的 Python 绑定及 C++ 代码作为项目实现。 Python 是一门动态语言，其运行前不需要编译，故十分方便修改运行和调试，同时其语法简单易懂，拥有强大的第三方库，与 C++ 的通讯简单，所以选择了其作为本文设计的上层逻辑语言。但 Python 作为一门动态语言，在实现大量代数运算时效率并不如 C++ 高，所以在本次实践中选择了 Python 和 C++ 相结合的方式，即利用 C++ 作为底层代码、加速运算，利用 Python 作为高层逻辑代码，方便设计和调试。

利用 OpenCV 的高性能库，可以省去许多时间和精力去做其他的优化。

第 2 节 代码实现

与本文有关的全部代码均可以在作者本人的代码仓库 https://github.com/zhaobenx/Image-stitcher 中找到。

2.1. 特征识别

利用 Python 版的 OpenCV，我们可以方便的利用其内置特征识别函数，对于常用的 SIFT、 SURF、 ORB 等特征提取算法可分别利用 cv2.xfeatures2d.SIFT(), cv2.xfeatures2d.SURF(), cv2.ORB() 函数来进行特征分析器的构建。以 ORB 为例，其特征提取方式是 keypoints, descriptors = cv2.ORB.detectAndCompute(image, mask)，将图像矩阵和蒙版参数传入函数即可得出特征点数组和描述子数组。返回值中的特征点数组 keypoints 是一个储存了特征点的数组，其中记录了每个特征点的坐标位置、区域、朝向、选取强度、来自第几层金字塔、分类 id 等内容；描述子数组 descriptors 是一个储存了描述子特征的数组，其内容和特征点数组对应，所存内容为用整数储存的特征描述子。

2.2. 特征匹配

在得到两幅图片分别对应的特征点和描述子后，即可进行特征的匹配。特征的匹配可以利用前文所介绍的暴力匹配或者 FLANN 算法进行。OpenCV 同样提供了这两种匹配方式，分别为 cv2.BFMatcher() 和 cv2.FlannBasedMatcher()，其使用方式也是相同的，以 BFMathcer 为例，其方式为 matches = cv2.BFMatcher.match(queryDescriptors, trainDescriptors)，其两个参量分别为源图像和目标图像的描述子数组，最终得到的返回值 matches 是一个关于匹配点的数组，其中记录了匹配的索引和匹配距离。

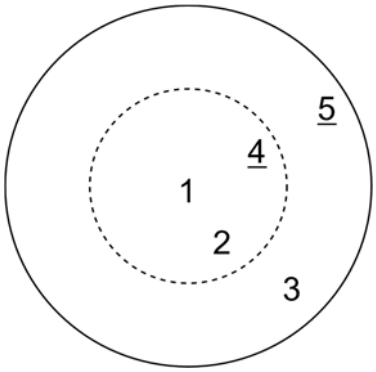
在得到特征匹配后，我们还需要考虑通过一些算法来剔除无用匹配，并保证有用匹配的数量较多。其中一个手段即对与匹配距离进行阈值筛选，只有少于特定阈值的匹配才能被归为有效匹配。

2.2.1 阈值选取

在匹配中，选取合适的阈值是十分重要的，不合适的阈值选取可能会造成误匹配或漏匹配，如图所示，图中 1、2、3 为正确的匹配点，而 ***4***、***5*** 则是误匹配点。在选取内圈虚线作为阈值时，2 为正确的匹配，3 是漏匹配，而 4 则是误匹配；当选择实线圈作为阈值时，可以看出，2 和 3 均被正确匹配，但却引入了 5 作为误匹配。

在实践中选择正确的阈值十分重要，在原则上，我们应该坚持宁缺毋滥，即优先保证避免误匹配发生，同时也应减少漏匹配。所以实践上倾向于选择尽量小的阈值，而在本文的实践中，只要保证至少有四对正确匹配的特征点即能获得正确的变换矩阵，但为了保证其准确度，我们一般会选择大于这个数字的特征点对数。在选择稍大的匹配阈值时，虽然可能会引入一些误匹配点，但是在我们下一步的算法中可以筛选出大多数的正确匹配点，所以通过稍微调大阈值来获取更多的点来保证正确匹配点的个数是十分重要的。故最终我们选出的匹配点对数应大于 4，理想状态至少应有 10 对，以方便我们下一步算法的运算。

在测试中，我们发现选取的特征点数不超过 40 组时，可以获得不错的正确匹配率，同时也能保证众多的特征点可以对图像进行一定的覆盖，具有代表性。在限制匹配点数的同时，本文采取了另一种策略，即阈值设置为最优匹配距离的两倍，在这两种方案的共同作用下，可得到比较优秀的匹配效果。



图片 10 匹配阈值选择示意

2.3. 图像变换

本文全部使用了透视变换矩阵作为图像变换的矩阵，在利用内置的函数进行图像卷绕变换前，需要先求得变换矩阵。

2.3.1 变换矩阵的求得

变换矩阵的传统的求法是利用 RANSAC 算法，在众多匹配点中选择出相对最优的四对解出其对应的变换矩阵。该方法可以利用 OpenCV 的内置函数 cv2.findHomography(srcPoints, dstPoints, method=cv2.RANSAC)来实现，其参量即为由特征匹配后筛选出的特征点数组，在这个函数的调用过程中利用了前文所介绍的 RANSAC 算法。利用该函数，可以在极短时间内得出变换矩阵。

2.3.2 图像的卷绕

图像的卷绕可以使用 OpenCV 库中的 cv2.warpPerspective(src, M, dsize)，其中 src 参量即为我们需要进行变换的图像，M 是求得的变换矩阵，dsize 是变换后图像的尺寸。在本文实践中，我们采用了动态手段计算了图像的尺寸，即利用变换矩阵先求的变换后的图像大小，然后利用该尺寸作为参量进行图像的卷绕。

2.4. 图像融合

在得到变换矩阵并利用内置函数进行图像卷绕后，我们就可以对变形后的目标图像和源图像进行融合。融合首先需要求得蒙版，即计算出源图像和目标

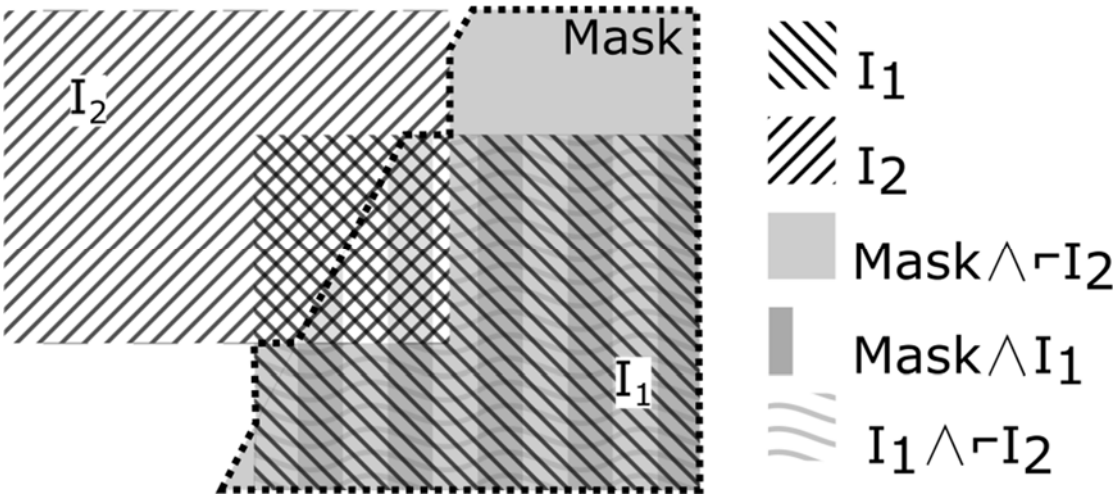
图像各自所占范围。

2.4.1 蒙版计算

在本文实践中，我们选择了一种简单有效的方式来生成蒙版，即利用源图像和变换后图像中点的垂直平分线作为蒙版的分界线，这样可以达到简单有效的结果。同时为了优化部分遮盖与重叠部分，最终的蒙版生成公式如下：

$$Mask = Mask \wedge \neg I_2 + Mask \wedge I_1 + I_1 \wedge \neg I_2$$

其中等式右边的 $Mask$ 为我们由垂直平分线分割计算出的二进制蒙版，I_1 和 I_2 分别为两个待融合图像，其逻辑运算关系的意义是如果图像在某点处有像素，则其为真，其余为非。这样经过运算后即可得到一个合适的蒙版。如图所示，虚线范围内是最终的蒙版，经过修补使得每块像素都有了充足的应用，通过这种算法即可得到最终的蒙版。



图片 11 蒙版生成示意图

2.4.2 融合计算

在 OpenCV 中并没有默认的融合方式，所以这里我们利用了 Python 的 NumPy 库来自行实现图像的融合，其基本原理由前文所讲，即通过对储存了图像像素的矩阵进行不同的运算以求得最后融合结果。

在得到所示蒙版后，即可以利用前文所述方法对图像进行不同方式的融合。其中直接融合和利用透明度融合运算最简单，速度最快，而利用高斯金字塔进行融合的效果最佳。在本文中，所展示例子均为利用高斯金字塔进行融合的结果。

第四章 利用遗传算法进行的图像拼接

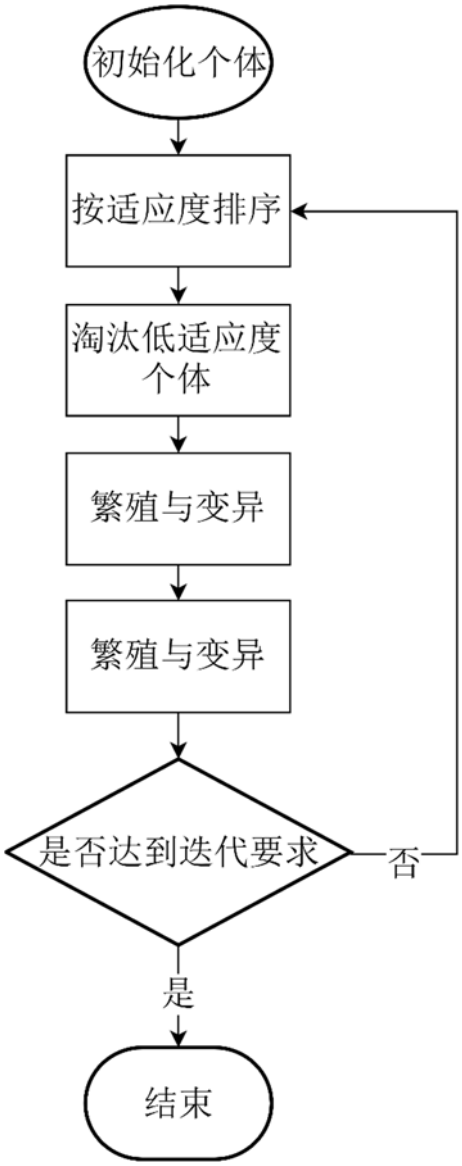
在使用传统拼接技术时，会出现一些图像无法匹配整齐的问题，为了解决这个问题，提高拼接的质量，本文介绍了一种利用遗传算法进行的特征点匹配筛选及变换矩阵计算方式，对拼接结果有着明显的提高。

第 1 节 背景知识

1.1. 遗传算法

遗传算法（英语：Genetic algorithm (GA) ）是计算数学中用于解决最佳化的搜索算法，是进化算法的一种，其主要适用方向是全局最优化问题，利用计算机模拟来进行优化问题的求解。遗传算法并不能保证找到最优解，但是能以较小的开销在众多变量中选择出相对最优解。

其基本思路是模拟遗传与进化的过程，将每个解称为一个个体，其代表一个变量序列，被称为基因或染色体，每个个体的适应度即解的好坏情况。其迭代过程即首先通过一定手段初始化一系列个体，并利用适应度函数计算每个个体的适应度。然后进入遗传进化阶段，即首先按照适应度由高到低排列，根据一定算法选择保留下的个体，然后让保留下的个体进行交配，即交换部分基因产生新个体，同时随机对部分个体进行变异，即改变基因的某部分。如此迭代直到达到预计目标。



图片 12 遗传算法流程图

1.2. 最小二乘法计算变换矩阵

在我们有若干点对需要求其透视变换矩阵时，我们可以利用最小二乘法来进行计算。由前文所介绍，当我们有四对点对时，我们可以利用矩阵乘法计算出精确的变换矩阵，而当我们所得点对数大于四时，其变换公式则变成了：

$$A\vec{m} = B$$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

其中 a_n 和 b_n 的定义与公式 4 和公式 5 相同。

在 $n > 4$ 时，A 不为方阵，此式为超定方程，不能通过直接求逆的方法来求解矩阵，通过矩阵形式最小二乘法我们可以得到解：

$$\overrightarrow{m} = (A^T A)^{-1} A^T B$$

关于超定方程矩阵形式最小二乘法求解的证明如下：

设 $Ax = B$，而我们最小二乘法的目的为求得一 x，使得 $\| Ax - B \|_2$ 取最小值，也即求 $\| Ax - B \|_2{}^2$ 的最小值。由矩阵的范数以及求导知识我们可以得出

$$\| Ax - B \|_2{}^2 = x^T A^T A x - B^T A x - x^T A^T B + B^T B$$

求导得到：

$$\frac{\partial \|Ax - B\|_2{}^2}{\partial x} = 2A^T A x - 2A^T B = 0$$

所以我们便得到超定方程的最小二乘法解（在矩阵 A 满秩的前提下）：

$$x = (A^T A)^{-1} A^T B$$ ■

第 2 节　算法思路

在得到匹配点对后，我们将利用本文介绍的算法替代前文所介绍的 RANSAC 算法去求得变换矩阵。其思路如下：

1. 初始化种群，针对匹配点对随机生成 N 组点对，每组点对所含 M 组点对，每个组称为一个个体；
2. 对每个个体，利用最小二乘法计算其对应的变换矩阵，将该变换矩阵设为该个体的基因；
3. 针对每个个体的基因，计算其对应的内点（符合变换矩阵的点）数量，以及内点偏差平方和，将这两项作为该个体的适应度；
4. 对每个个体的适应度进行筛选，除去除低适应度个体，并保留高适应度个体；
5. 按照适应度排序，个体间随机交配，交换部分基因（变换矩阵的元素）得到新的个体；
6. 随机对于部分个体进行变异操作，即随机改变其基因中的部分参数；

7. 重复步骤 3 - 6，直到若干代后遗传算法结束，选出最优值作为解；

第 3 节 算法可行性

遗传算法的特点是在较大的搜索空间中寻找全局的最优解。这里的问题为在一个特征点对的集合中(例 40 组点对)寻找一个最优变换矩阵，使得该变换矩阵对应的匹配点最多。而传统的 RANSAC 算法利用的是随机挑选方式，在全集中随机的选出四个点，使得这组点对应的匹配点最多，这样会出现一个问题，即最后求出的变换矩阵总是针对某四个点，无法计算平均情况。

在这里介绍的新方法将会由一些点集出发，逐步利用遗传算法迭代逼近最优解，最终选取出一个尽可能覆盖点集最多，同时又能保证覆盖点集偏差距离平方和最小的一个变换矩阵。同时每次迭代是在上次的基础上进行的，充分利用了前面进化的结果，提高了解的有效性。 。

第 4 节 与 RANSAC 的比较

由时间上分析，RANSAC 算法的每次迭代都需要重新进行一次变换矩阵求解，即矩阵求解过程，重复直到找出最优解。本文介绍的新算法则只有在初始化阶段利用最小二乘法得出变换矩阵，之后的遗传与变异则无需再加计算，在矩阵运算开销上花费较小。在考虑内点比例不变时，RANSAC 与本文算法对于输入点对数量的时间复杂度都是 $O(1)$；而在考虑内点数量固定时，由于 RANSAC 算法的随机性，其迭代次数会随输入匹配点数增加而增加，由其限制条件可以算出其时间开销与输入点对数的时间复杂度为 $O(n^4)$，而相比之下由于遗传算法的常数性，其复杂度为 $O(n)$。当然我们也要考虑到利用遗传算法进行的计算在时间复杂度的常数项会稍大，但在数据集较大时此项即可忽略。

在算法上分析，两种算法本身都是迭代算法。RANSAC 算法没有优化过程，而遗传算法有优化过程。即 RANSAC 算法仅仅是通过随机抽取四组点对来得到随机结果，每次迭代之间独立，迭代次数的增加只是提高了相对最优解的得到概率。而对于本文的算法来讲，每次迭代之间都有传承，会将上一次迭代中表现优异的特性转移到下一代中，利用遗传优化了迭代过程。同时，RANSAC 的算法得出的矩阵是由某四组点对计算出的，无法计算出中间情况，而本文算法则是对变换矩阵进行迭代，可以不受点的制约，能生成出针对中间情况的变换矩阵，覆盖范围更广。

同时在对图像匹配点对的要求上来分析，本算法和 RANSAC 算法要求相同——即都要求至少有四对点对，在保证匹配质量的情况下点对越多越好，同时

均要求正确匹配点越多越优。所以两种算法都对于匹配点的筛选有着同样的要求。

第 5 节　实践

5.1. 利用遗传算法的变换矩阵求解

在本项目中，我们以变换矩阵作为基因，基因为八个浮点数，本文采用的优化的方式即对这八个浮点数进行选择和进化，从而选择出最优的变换矩阵。

5.1.1 变量的优化

在实践中，我们会考虑到不同的参数设计，并由实践出发对参数进行优化。主要需要优化的参数有：种群数量、遗传次数、变异率、筛选方式等，针对于具体的应用，我们应该根据实践进行调整以达到最佳状态。

在实践中，我们进行了多次不同的实践，选定了不同的参数。对于我们的应用来讲，我们的初始化变量的是这样 的，匹配点对至多有 40 对，所以在这里我们选择了 40 作为种群数量，同时每个个体由 8 组点对选出，同时我们将变异率设为 0.2。

5.1.2 适应度的选择

适应度函数(Fitness Function)的选取直接关系到种群的进化方向，亦即影响我们优化问题最终解的优劣，选择一个合理的适应度可以加快收敛速度并有助于找到最优解。

在设计适应度函数时，我们应该注意首先设计应尽可能简单，因为适应度函数的计算是整个算法中重复次数最多的部分。同时适应度函数的设计应体现出我们对优化问题的追求方向，由于遗传算法对于解的评价和解的最终结构无关只和我们的适应度函数有关，所以应利用合理设计的适应度函数来显示出我们的进化目标，以体现出遗传算法“优胜劣汰”的特点。[14][15]

求解最佳的点集进行转换矩阵的构造本身可以转化为一个求解一个最大化的问题，但是与通常的遗传算法不同，我们无法对最优解有一个准确的数学描述，即我们无法判断解是否最优，只能以比较来判断。因为在众多拼配点存在的情况下，问题变成了一个多维的最大值求取，同时还需要考虑剔除掉误匹配

点，所以我们无法给出一个准确的最大值判断。尽管如此，我们仍能得出相对优秀的标准，那就是尽可能符合多点的变换矩阵，通过这个标准我们即可以找到进化的方向，而这也正是我们可以利用遗传算法进行优化的原因，即我们无需知道最优解的结构，只需要知道相对优秀解的标准即可利用遗传进化出一个优秀解。

在该问题中，好的解就是尽量匹配出足够多的内点，同时保证选出的内点变换后的距离误差最小、点集分散程度最高。所以在一开始的设计中，我们选择了内点数作为适应度函数。但是仅仅用内点数作为适应度函数出现了一个问题，那就是函数的结果总是整数，无法建立区分度，即在众多个体的内点数相同时，无法确立进化的方向。为了解决这个问题，我们在考虑内点数的同时加入了内点距离的偏差值，偏差值是由源点经变换后和目标点的距离平方差计算得出的。这样做的好处是不仅在内点数相同时确立了进化的方向，而且也可以通过尽力减少这个偏差值来使得变换矩阵的特性更优良。同时在应用于小视差图片时，我们应使特征点的分布范围更大，以避免得到局部最优而不是全局最优的矩阵，造成图片部分拼接良好，而其它部分拼接效果很差的情况，如下文中图片 20 所示，所以在考虑时加入了分布距离作为计算参量。

所以针对本文，我们需要挑选出尽量多的内点，并且使得内点的偏差尽量小，同时使特征点分散较广。在这三点里，内点的数量是最优先考虑的，其次是分布距离以及变换后内点间距离的偏差。综上，我们得到了应用于本文的适应度函数。即设内点数为 N，内点的偏离距离平方和为 D，内点坐标与其重心之间的距离平方和为 E，则得出我们的适应度 $V = N - \frac{D}{N} + \tanh(\frac{\mathrm{E}}{\mathrm{N}})$。这种适应度的公式设计首先是为了突出内点数量的重要性，同时通过双曲正切函数 tanh 对偏离值进行归一化，使其限定在一定范围内，避免其值过大对整个适应度造成影响。

在这种适应度函数的作用下，我们可以观察到代与代之间适应度的提高，即意味着内点数偏差的减少或内点分布更广或内点数的增加。而我们优先考虑的是更多的内点数，内点数增加的影响远大于偏差距离的影响。而本函数也体现出了这种目的，我们的适应度函数对三项参量进行不同的权重分配即可实现对内点数、分散度和内点距离误差的优先程度设置。实践也表明此种方法能得到较为理想的结果。

5.1.3 筛选

在遗传算法应用中，有很多中筛选方式，其中典型的有轮盘赌、精英选择

等，轮盘法即按照适应度高低为每个个体赋予一个生存概率，从而筛选掉适应度低的个体。精英选择即保留最优的个体。在不同应用里，不用的筛选方式有着不同的作用，在本文应用中，我们选择了精英选择方式。

5.1.4 变异

由于我们的基因是数字，所以在这里需要对数字进行变换处理。应用中的变异方式有随机替换等，但在本项目中，由实践证明，随机替换矩阵中一个数字的结果非常不理想，所以这里选择了利用随机分布在原来矩阵的基础上进行乘法的变异方式，即利用正态分布，对于矩阵中的随机值进行乘法，这样会使我们的变异不至于太过激进。本项目里选择了 $\mu = 1, \sigma^2 = 0.1$ 的正态分布作为乘数，并且取得了不错的成果。

5.1.5 交配

交配的目的是使得优秀亲本间的基因可以交换，以期望产生更优个体。交配可以认为是一种剧烈的变异，而这种剧烈的变异在进化初期对种群影响极大，而在本项目中剧烈的变异通常来说不能带来有效的改进，所以在本项目中我们让交配的个数占比降低，以温和的变异为主，交配为辅。

5.1.6 繁殖代数

繁殖的代数由最终结果决定，在本项目里，我们选择 了 20 代作为迭代上限，实践证明选择 20 作为迭代次数可以得到性能与质量的一个平衡点。过多的迭代代数并不能显著的提高变换矩阵的求解质量，反而会耗费掉大量时间。而过少的繁殖代数则会使得进化不充分，不能获得较优的解。

第 6 节　测试

在这里我们选择了八组图片，作为拼接的对象，并分别利用了 SIFT 算法和 ORB 算法，同时进行了传统方式和新方法的结果对比。

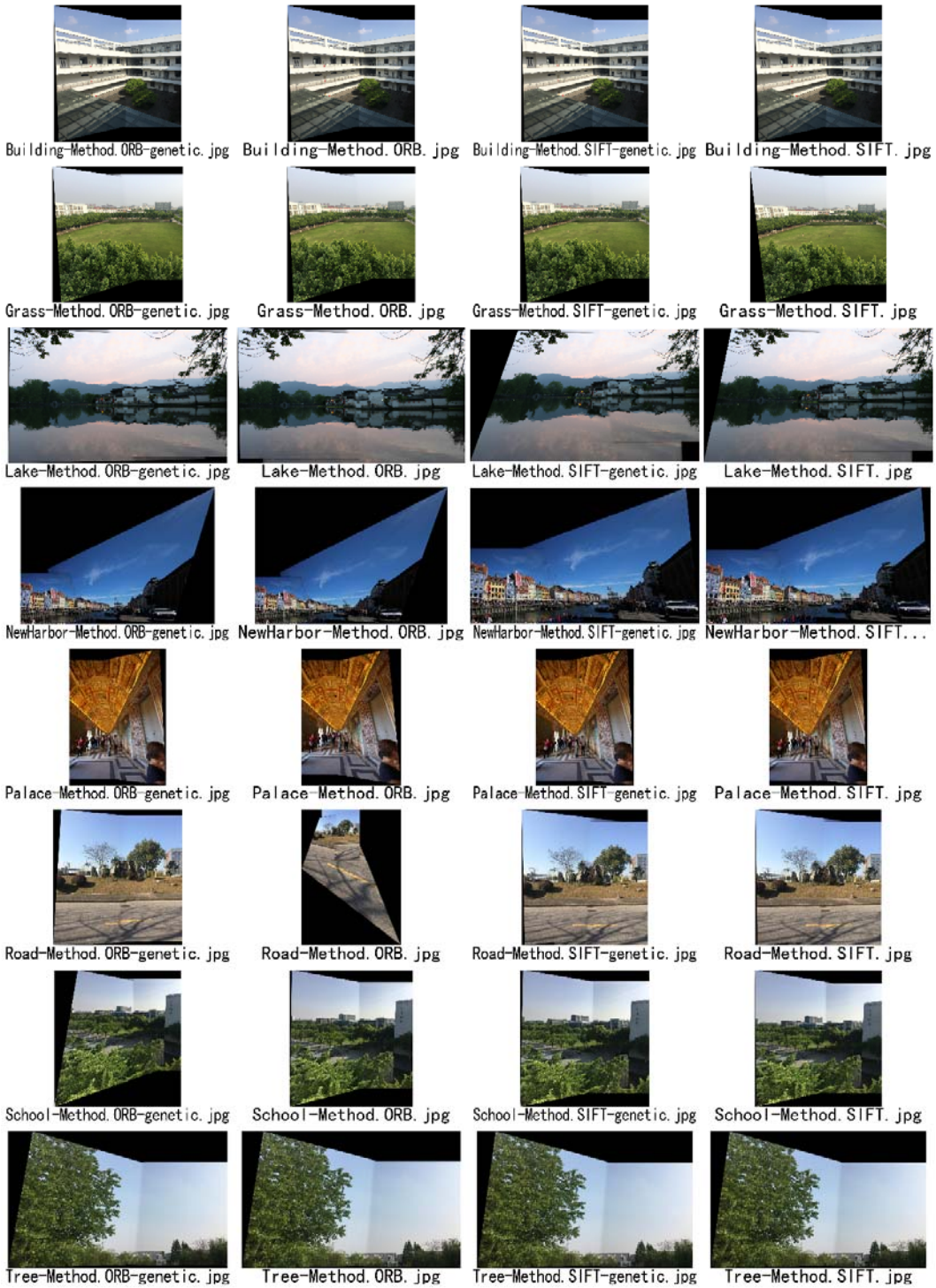
图片 13 拼接原图

6.1. 拼接结果

拼接结果如下图所示，其中每行都是按照每幅图的 ORB 遗传算法，ORB 传统算法， SIFT 遗传算法， SIFT 传统算法的排序方式。观察这些略缩图也可以看到，利用了不同方法的图像拼接基本上都完成了预期目标，唯一只有一张利用了 ORB 与传统算法的 Road 这张图拼接的效果差强人意。图中那些接缝是由

于相机拍摄时没有固定曝光，导致拼接两边曝光度不一致而产生的，这种由光线引起的接缝是在我们能接受的合理范围内的。

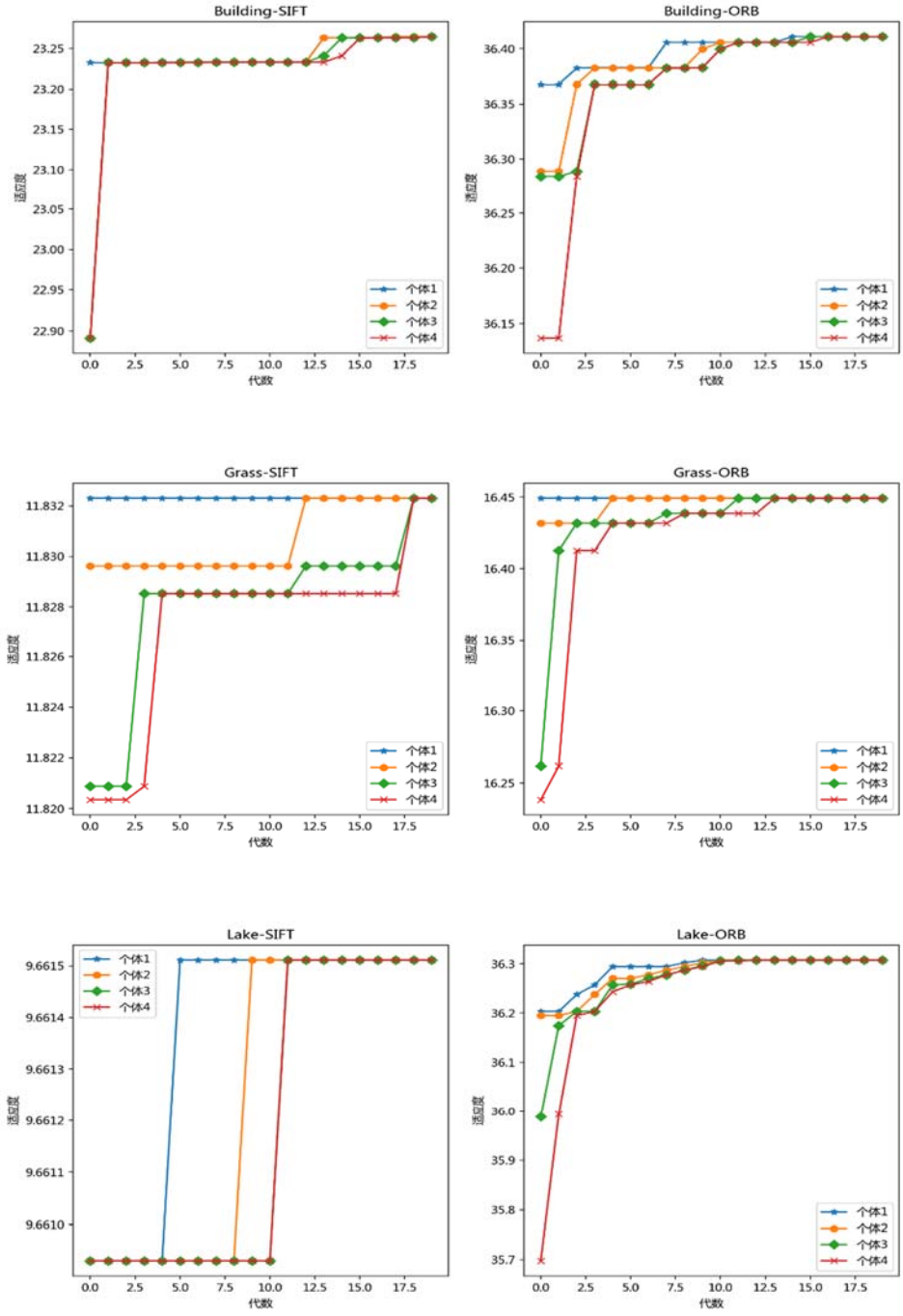
同时总结分析我们可以看出，使用了 SIFT 特征点提取算法的图片拼接效果平均要优于使用了 ORB 算法的图片，同时在细节上使用了遗传算法的会略优于传统算法。

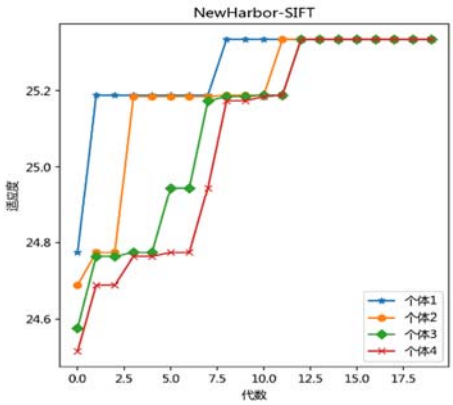


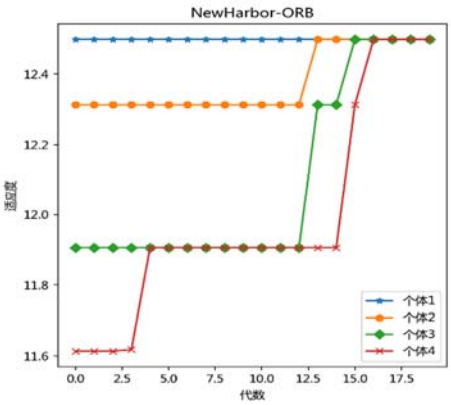
图片 14 拼接结果

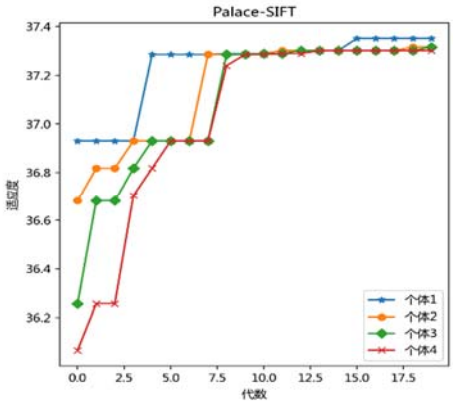
6.2. 遗传算法的有效性

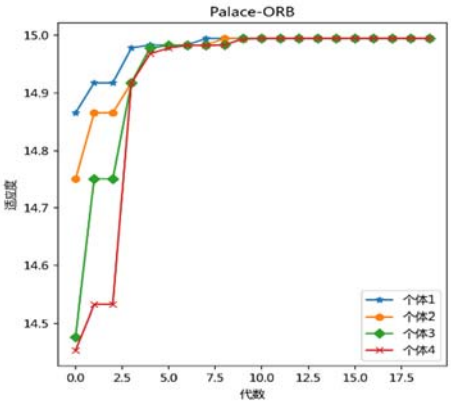
我们在这里对每个进化过程进行了可视化处理，选择了每一代中适应度前四的个体，追踪它们的演进过程，以体现出遗传算法在迭代中的特性。

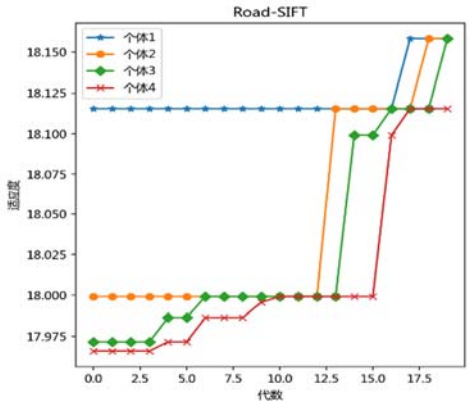


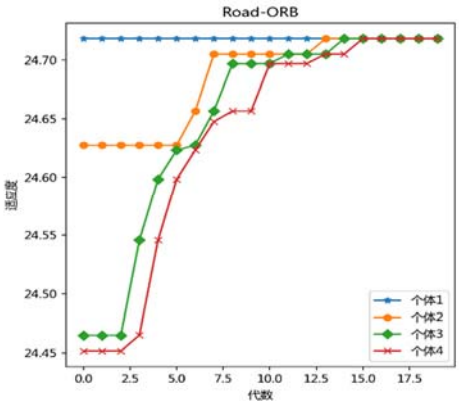

NewHarbor-SIFT
适应度
代数
个体1
个体2
个体3
个体4

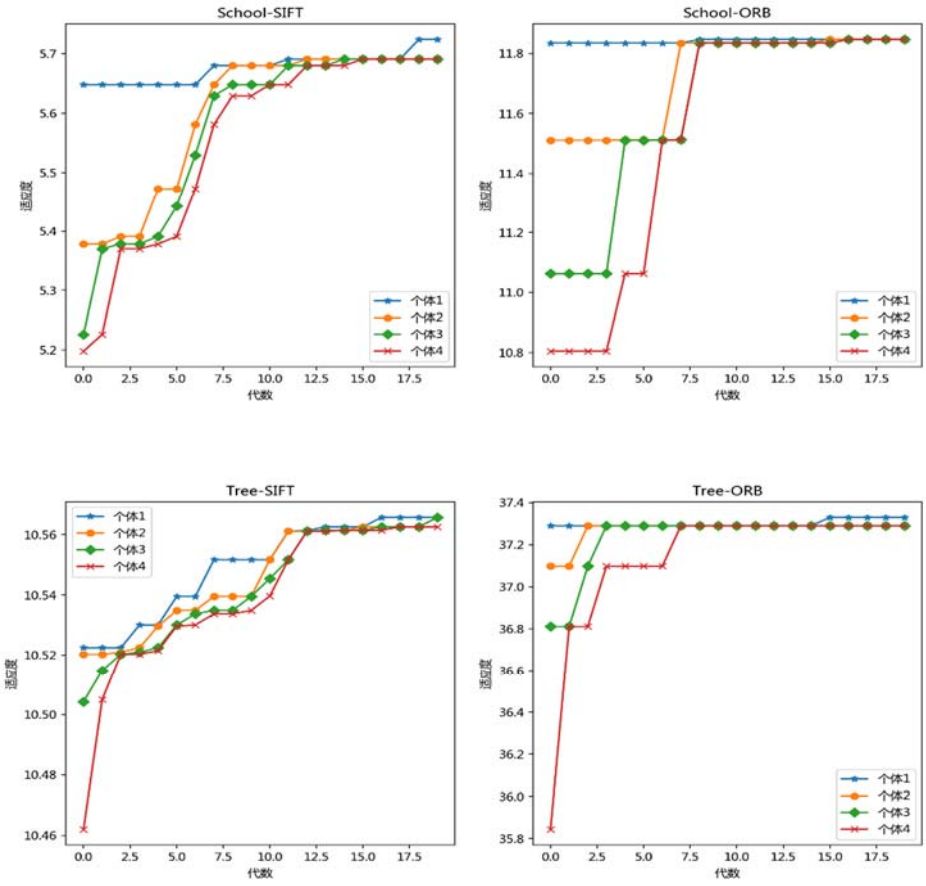

NewHarbor-ORB
适应度
代数
个体1
个体2
个体3
个体4


Palace-SIFT
适应度
代数
个体1
个体2
个体3
个体4


Palace-ORB
适应度
代数
个体1
个体2
个体3
个体4

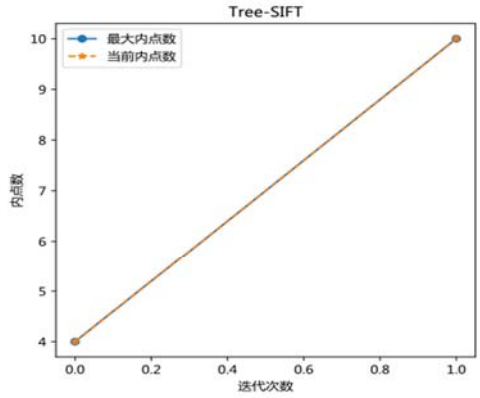

Road-SIFT
适应度
代数
个体1
个体2
个体3
个体4

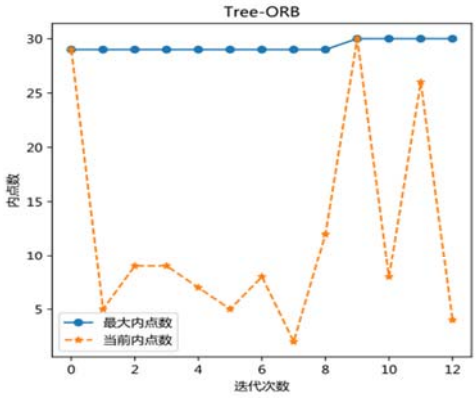

Road-ORB
适应度
代数
个体1
个体2
个体3
个体4

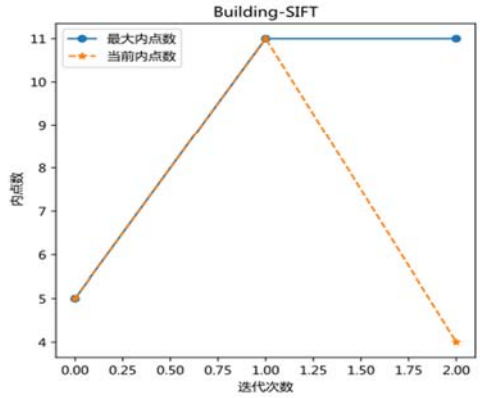


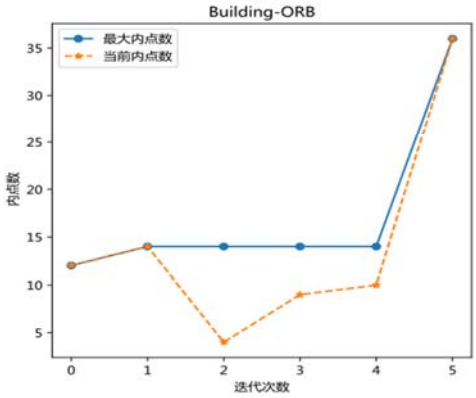
图片 15 遗传算法适应度变化图

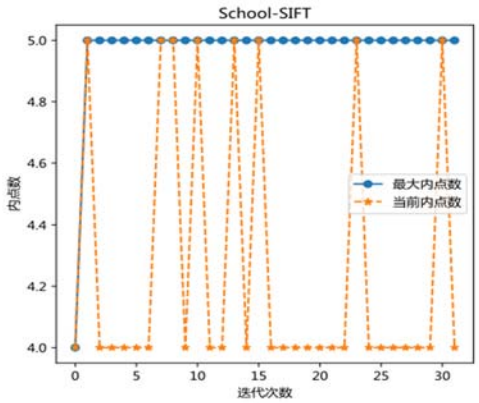
同时我们对于 RANSAC 算法的每一步迭代进行记录，由于 RANSAC 算法中的迭代次数和遗传算法中的代数定义不同，所以我们无法对其采用相同的数据记录，故这里只记录其每次迭代时随机选取的矩阵计算出的内点数和最佳内点数，见图片 16，可以看出，RANSAC 算法的迭代之间内点的变化震荡剧烈，说明后续的迭代并没有从前面的结果中进行学习。

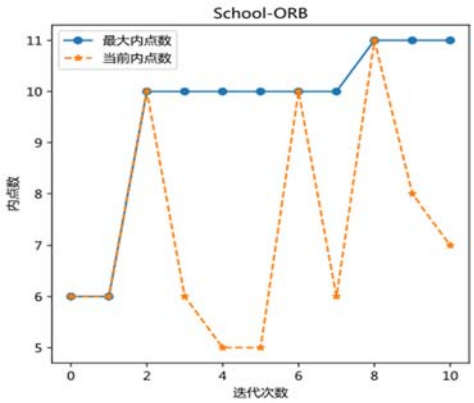

Tree-SIFT
最大内点数
当前内点数
内点数
迭代次数

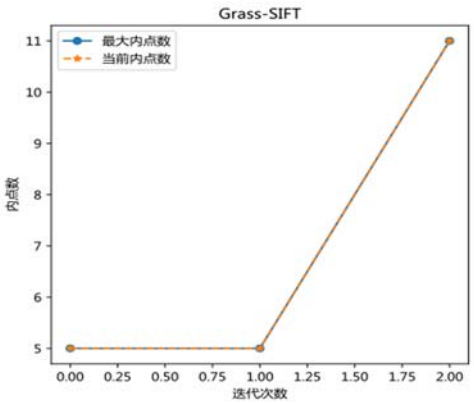

Tree-ORB
最大内点数
当前内点数
内点数
迭代次数

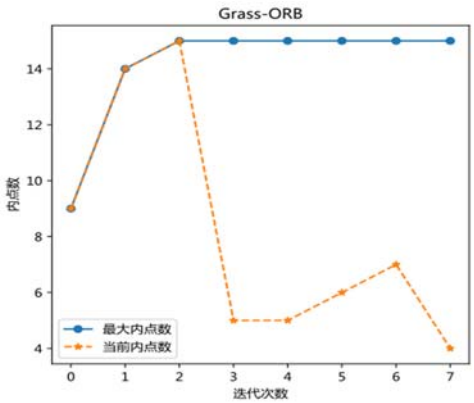

Building-SIFT
最大内点数
当前内点数
内点数
迭代次数

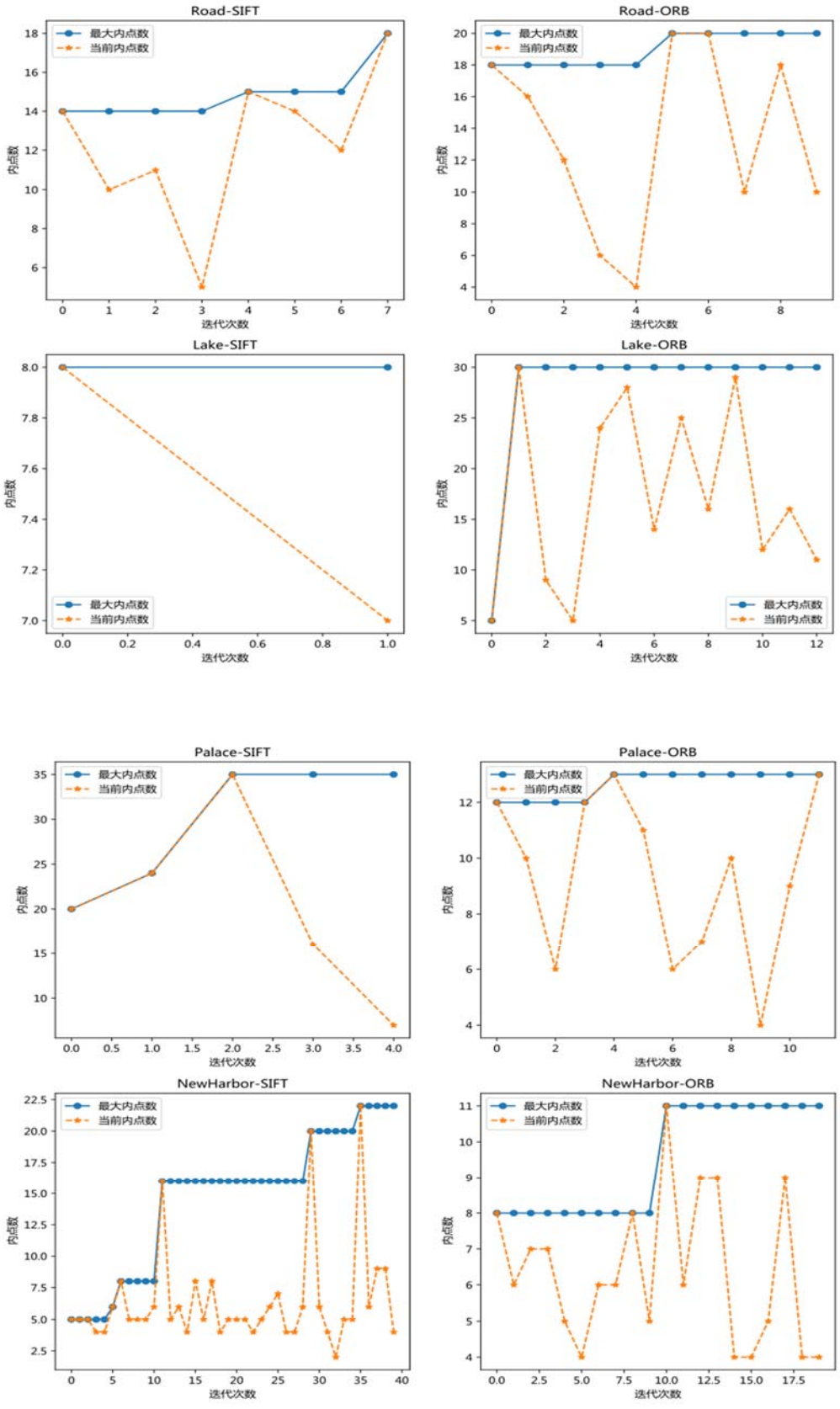

Building-ORB
最大内点数
当前内点数
内点数
迭代次数


School-SIFT
最大内点数
当前内点数
内点数
迭代次数


School-ORB
最大内点数
当前内点数
内点数
迭代次数


Grass-SIFT
最大内点数
当前内点数
内点数
迭代次数


Grass-ORB
最大内点数
当前内点数
内点数
迭代次数



图片 16 RANSAC 算法内点数变化情况

观察遗传算法的迭代的过程，我们可以明显观察到种群适应度的提高，即达到了我们的期望。同时由图表中的数据可以看出，大多数情况下繁殖到 20 代左右时，适应度已无明显提升，这也证明了我们这个代数选择的合理性。利用遗传算法，我们的迭代不再是完全随机的，从而提高了迭代的效率，同时也能使得我们更容易的达到更优的内点数量以及减少内点偏离误差。

观察这些遗传迭代图表可以看出，对于多数情况，遗传效果对于适应度的改进是有一定的作用的，少部分情况下则无明显提升，分析是由于遗传算法有一定的随机性，并不能保证完全的去寻找到最优结果，所以会造成在一些情况下提升不明显的结果。还有一部分情况如在处理 School-ORB 这幅图时，在前步操作后只提取出了 11 对匹配点，而这些匹配点全都符合初始化的变换矩阵，所以其最优解只是在通过改善内点距离偏差来微弱的提高适应度，但这种情况和我们优化的结果是不冲突的。

而与 RANSAC 的结果进行对比则可以明显看出，RANSAC 算法的每次迭代之间都是随机的，其最大内点数是靠随机的当前内点数来进行更新，这样固然可以减少一部分运算，但是会无法保证能得到最优结果。并且在某些情况下一开始的几步迭代即已经获得较优的结果，但在满足迭代次数阈值之前迭代仍然不会停止，并且这些迭代并不会在前面优秀的结果上对结果进行改进，后续迭代只是随机生成变换矩阵，与遗传算法进行对比则少了延续性和继承性。

6.3. 与传统方法对比

6.3.1 图像对比

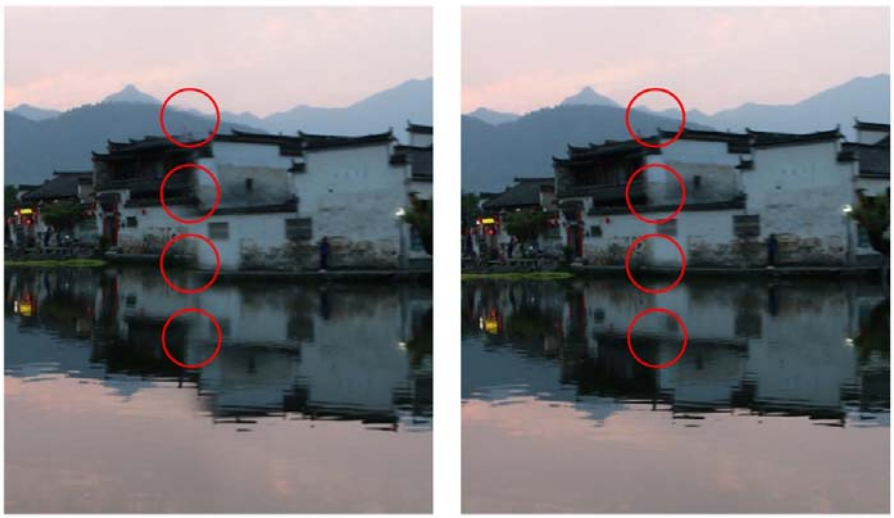
利用遗传算法，我们得出的图片拼接效果不弱于传统算法的结果，并且在一些细节方面还优于传统算法。由于在之前的拼接中，我们已经得出了拼接的结果，下面放一些细节对比的图片来展示一下对比。图片中左图是利用传统算法进行的拼接，右图是利用遗传算法进行的拼接。

由于 SIFT 特征点提取的质量都相对较高，所以在利用 SIFT 时的细节对比不明显，而在利用 ORB 算法时，一些细节上的瑕疵就比较明显，所以下面三幅对比图片都来自于利用 ORB 算法拼接的结果。

(1) 传统算法　　(2) 遗传算法

图片 17 ORB 算法中图 Road 的对比

第一幅图 Road 的拼接结果里，利用传统算法的拼接在整体上有较大的误差，只有上半部分的石头部分是匹配整齐，而下半部分的路的部分则完全变形。而右边利用遗传算法的拼接则保持了相对的整齐性，在传统算法有非常大的偏差时仍能得到相对不错的拼接结果，而这些误差则是来源于原图片的视差，是无法避免的。

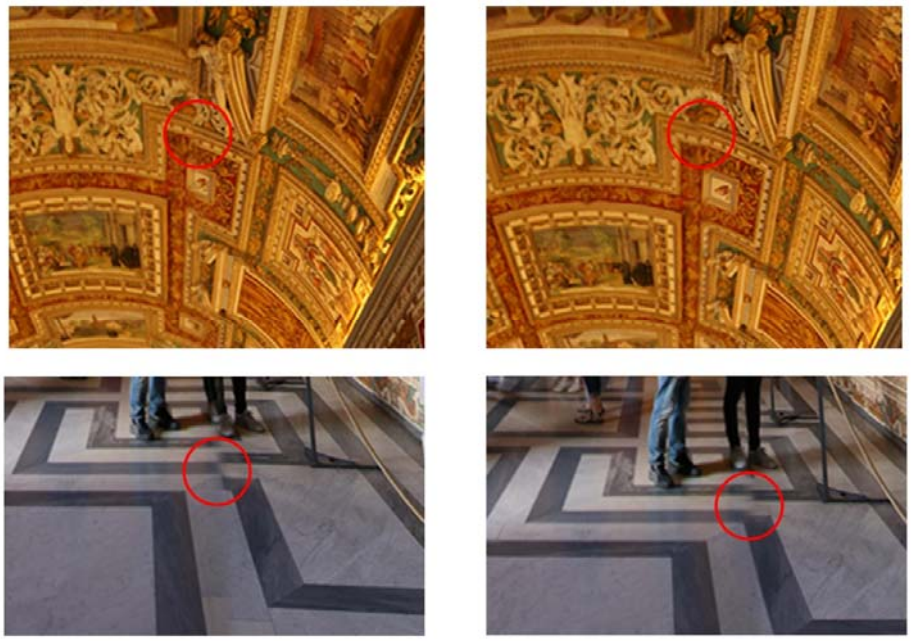


1) 传统算法　　(2) 遗传算法

图片 18 ORB 算法中图 Lake 的对比

第二幅图 Lake 中，对比左图利用传统算法进行拼接的结果，右图明显对齐更准确，观察图中相同位置的圈可以看出，左图中产生了一些明显的错位，即

使在利用 Gauss 金字塔进行融合后，断裂的直线也能被观察出来，而右图则几乎没有偏差。



1) 传统算法　　(2) 遗传算法

图片 19 ORB 算法中图 Palace 的对比

在第三幅图 Palace 中，左图利用 RANSAC 的拼接也是产生了一定的异位，而在右图中则被减轻了很多，拼接效果明显优于左图。

(a) 利用传统算法，内点主要分布在左边　　(b) 利用遗传算法，内点均匀分布

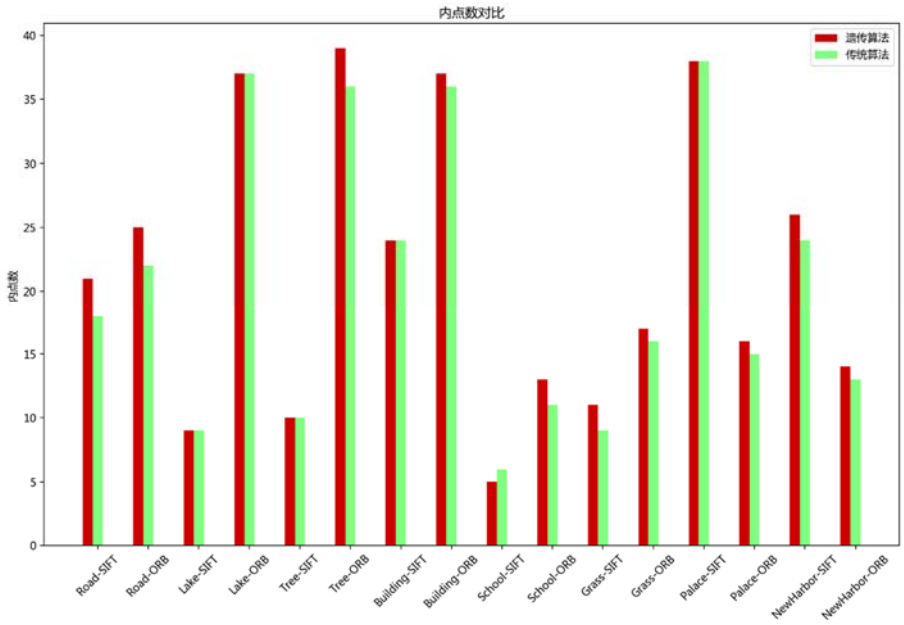
图片 20 特征点分散情况对比

在分散度对于图片拼接的影响中，我们选择了利用 ORB 算法的 Road 拼接图像，如上图所示。在输入特征点完全相同的情况下，利用传统算法的左图的拼接质量差于右图。观察图中标出的特征点可以看出，(a) 图中的内点主要集中在左边部分，右边部分几个特征点有明显的偏离，并未被纳入内点；而 (b) 图中内线的选取较为均匀，虽然中间一些特征点的匹配并非严丝合缝，但是做到了内点的分散，从而获取到了具有全局最优性的变换矩阵。

6.3.2 数据对比

图片只能给我们一个直观的感受，下面将用一些数据对比图来对比新方法和原方法的差异，我们分别选取了最终求得的内点数、分散度和所花费时间这三项来比较分析。

(1) 内点数对比

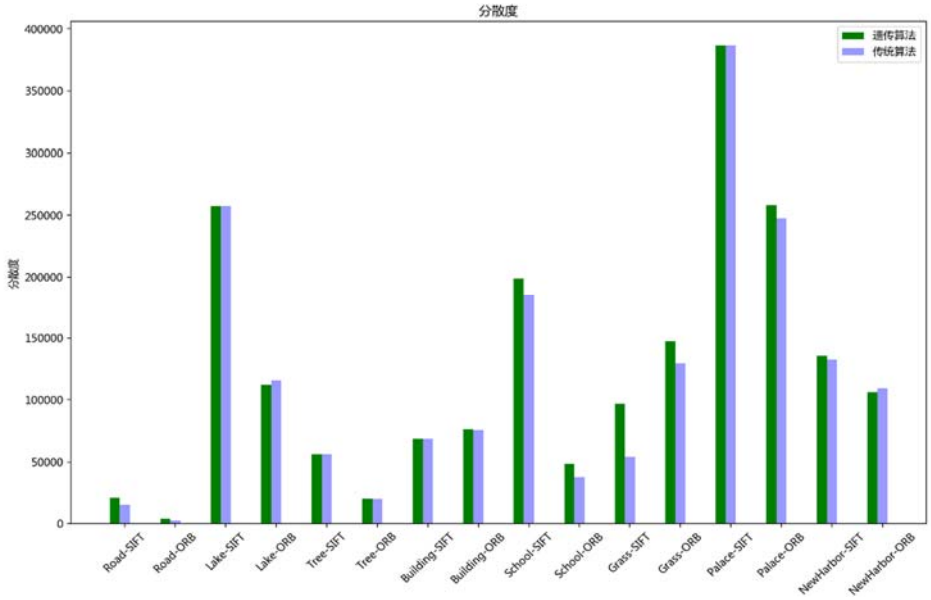


图片 21 RANSAC 与遗传算法所得内点数对比

观察此图表，普遍来看利用了遗传算法后所摘选出的内点数普遍多于传统算法，只有少部分例子出现了持平或者低于传统算法的情况，通过内点数对比也可以看出遗传算法的有效性，即提高了对于内点的拟合能力。在所有例子中，我们注意到只有一项发生了反常现象，这是由于在该图片中输入的误匹配较多（源图像中有多处相似且重复的窗户导致误匹配），导致在种群初始化时难以找到优秀的变换矩阵，但是经过遗传算法处理后，仍然得出了相对理想的变

换矩阵，保证了可以接受的内点数量。

(2) 分散情况对比

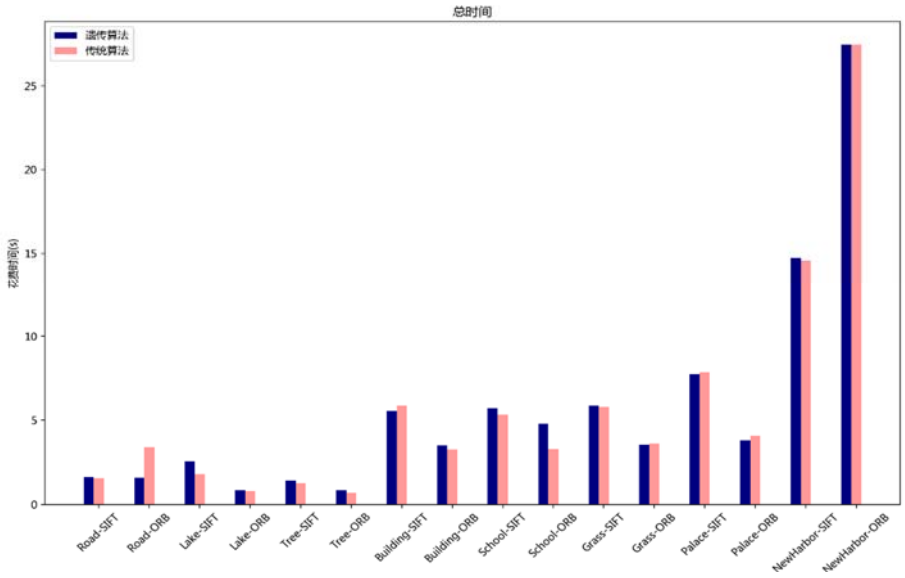


图片 22 RANSAC 与遗传算法所得分散度对比

在这里的分散度即为所选出内点与内点重心距离的平方和，用以表述点的分散情况，其值越大越说明内点分散更广。由图表我们同样可以观察到利用遗传算法后的分散度普遍高于传统算法，只有少部分例外。利用遗传算法可以有效的提高变换矩阵的分散度，而直观到图片上即为图像拼接更自然，接缝处误差更小，这也是我们进行图片拼接所追求的。

虽然大部分情况遗传算法的分散度是高于传统算法的，而那些小部分的反常是不可避免的，这是由随机算法造成的，无论在初始化种群还是繁殖过程中都是随机的过程，因而在这些随机过程中会有一些无法达到更优的效果，但考虑到这些反常结果在所有结果中的占比，我们认为这种偏差是可以接受的。

(3) 时间对比



图片 23 RANSAC 与遗传算法所花时间对比

这里我们对遗传算法和传统算法的拼接时间进行对比。这里统计的是图像拼接的总时间，在实际运算中总时间会受到输入图片以及最终拼接图片尺寸影响，不同图片的拼接时间偏差会较大。但是在使用相同的输入图片和相同的特征检测方式时，其时间差异即主要由变换矩阵求法的不同来决定。

如图表所示，通过时间这项我们可以观察得出最终结果的时间差距和方法之间并没有太大的相关性，利用改进的新方法花费的平均时间是总体上是略优于传统算法的，在这里还需要考虑到本文中使用的传统方法是利用 C++ 实现的，而新的遗传算法是利用 Python 实现的，在性能上 C++ 代码的效率是同样 Python 代码的百倍或者更高，这里的结果也说明即使在如此大的性能劣势下，新方法仍可以达到不凡的速度。

第 五 章 总结与展望

第 1 节 论文总结

传统的图像拼接算法在拼接效果和速度上已经达到了不错的水平，新兴的算法也在不同方面对图片拼接的质量与速度进行着提高。

本文所介绍的新算法主要利用了遗传算法，优化了变换矩阵求解的过程，帮助我们获得更多的内点，同时会优先去选择更加分散的内点，以益于解的全局更优性，并且会去选择矩阵使得其对应的偏差距离更小。

在对比传统算法和利用遗传算法进行的变换矩阵求解后，我们可以看出利用了遗传算法改进后的图像拼接在改善拼接质量上有了很大的进步，达到了我们的预期目标。

第 2 节 展望

通过本文的示例我们可以观察到遗传算法的应用对与图像拼接进行了不错的优化，同时我们还能意识到利用遗传算法对图像拼接进行优化还有着相当大的潜力。本次应用里我们选取的参数都较为保守，而应用中为了提高可靠性可以设置更多的个体以及进行更多次的繁殖，通过牺牲一部分时间来确保更高的质量。或者在应用中可以针对其他参数进行特别的优化，以期望能达到更优的效果。

参考文献

[1] Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[2] Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski. "ORB: An efficient alternative to SIFT or SURF." In *Computer Vision (ICCV), 2011 IEEE international conference on*, pp. 2564-2571. IEEE, 2011.

[3] Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." In *European conference on computer vision*, pp. 430-443. Springer, Berlin, Heidelberg, 2006.

[4] Calonder, Michael, Vincent Lepetit, Christoph Strecha, and Pascal Fua. "Brief: Binary robust independent elementary features." In *European conference on computer vision*, pp. 778-792. Springer, Berlin, Heidelberg, 2010.

[5] Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision*60, no. 2 (2004): 91-110.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *In European Conference on Computer Vision*, 2010. 1, 2, 3, 5

[7] 曲天伟,安波,陈桂兰.改进的 RANSAC 算法在图像配准中的应用[J].计算机应用,2010,30(07):1849-1851+1872.

[8] Brown, Matthew, and David G. Lowe. "Automatic panoramic image stitching using invariant features." *International journal of computer vision* 74, no. 1 (2007): 59-73.

[9] Marius Muja and David G. Lowe: "Scalable Nearest Neighbor Algorithms for High Dimensional Data". *Pattern Analysis and Machine Intelligence* (PAMI), Vol. 36, 2014.

[10] Marius Muja and David G. Lowe: "Fast Matching of Binary Features". *Conference on Computer and Robot Vision* (CRV) 2012.

[11] Marius Muja and David G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration", in *International Conference on Computer Vision Theory and Applications* (VISAPP'09), 2009

[12] Nearing, James (2010). "Chapter 7.3 Examples of Operators" (PDF). *Mathematical Tools for Physics*. ISBN 048648212X. Retrieved January 1, 2012.

[13] Adelson, Edward H., Charles H. Anderson, James R. Bergen, Peter J. Burt, and Joan M. Ogden. "Pyramid methods in image processing." *RCA engineer* 29, no. 6 (1984): 33-41.

[14] 刘英.遗传算法中适应度函数的研究[J].兰州工业高等专科学校学报,2006(03):1-4.

[15] 陈果,邓堰.遗传算法特征选取中的几种适应度函数构造新方法及其应用[J].机械科学与技术,2011,30(01):124-128+132.

致谢

写到这里也就意味着我在复旦的四年本科生涯即将落下帷幕。我在这四年里经历了很多有意义的事情，也收获到了很多的美好。

首先我要感谢我的指导老师荆明娥老师，在她孜孜不倦的教诲下我才能完成这篇文章，在这篇文章的完成过程中她对我提出了很多修改意见，让这篇文章变得更丰满更透彻。同时也要感谢黄宇杰学长，他为我这篇文章的写作提供了很多创意与观点。也要感谢其他的各位任课教师，教给了我那么多宝贵的知识。

还要感谢在复旦认识的那么多可爱的同学们。感谢那位陪伴我让我成长的同学；感谢来耀学长，在我生活学习中给了我许多援助；感谢室友潘健，我们分享了很多快乐；感谢张国一、感谢所有给我带来过欢乐的同学们。还要特别感谢小伙伴翟犇、袁誉萌，谢谢你们对我一路的支持。

还要感谢我的父母和家人，感谢他们，有他们的支持我能顺利的考入复旦大学并取得今天的成绩，他们在生活和精神上对我提供了莫大的支持。

感谢复旦大学，给了我一个成长的平台，让我认识这么多有趣的同学们，教我做一个学术独立思想自由的人。

做了点微小的工作，谢谢大家。

赵凌丰

2018 年 5 月 27 日