基于区域蛙跳搜索与轮廓匹配的显微图像拼接

颜振翔[1*]，王寒迎[1]，石齐双[2]，莫艳红[1]，杨辉华[2,3]

[1]桂林电子科技大学电子工程与自动化学院，广西 桂林 541004；

[2]桂林电子科技大学计算机与信息安全学院，广西 桂林 541004；

[3]北京邮电大学自动化学院，北京 100876

摘要 为解决传统显微图像拼接中产生的几何畸变和错位，及特征稀少造成的正确匹配率低、时效性差等问题，提出基于区域蛙跳搜索和图像轮廓匹配的拼接算法。提取连续采集且有重叠区域的图像轮廓曲线；引入轮廓线索感知相似度和均方误差距离，计算图像轮廓曲线间的相似度或曲线离散距离，并将其作为匹配的衡量指标；在决策域内采用区域蛙跳算法更新鸣叫分贝和蛙跳策略，搜索图像轮廓最优匹配，实现图像快速精确的拼接。结果表明，所提算法不仅具有较高的拼接精度和较强的稳健性，还减小了其简化匹配策略的计算量，具有较强的时效性。

关键词 图像处理；图像拼接；线索感知相似度；蛙跳策略；轮廓曲线

中图分类号 TN911 **文献标识码** A **doi**：10.3788/LOP56.151002

Microscopic Image Stitching Based on Regional Frog Leaping Search Algorithm and Image Contour Matching

Yan Zhenxiang[1*], Wang Hanying[1], Shi Qishuang[2], Mo Yanhong[1], Yang Huihua[2,3]

[1] *School of Electronic Engineering and Automation, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*;

[2] *School of Computer and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*;

[3] *School of Automation, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*

Abstract To solve geometric distortion and misalignment generated in the traditional microscopic image stitching process and avoid low matching rate and high computational burden caused by rare features, a stitching algorithm based on the regional frog leaping search algorithm and image contour matching is proposed. First, the image contour curves of continuous acquisition and overlapping regions were extracted. Then, the clue-aware trajectory similarity or mean square error distance was introduced to calculate the similarity and discrete distance between image contour curves. Finally, the regional frog leaping algorithm was used in the decision domain, and by updating the tweet decibel and frog leaping strategy, optimal contour matching for the image was searched. Consequently, the images were stitched quickly and accurately. Experimental results verify that the proposed algorithm demonstrates high stitching precision and strong robustness. In addition, the simplified matching strategy reduces the amount of calculation and exhibits strong timeliness.

Key words image processing; image stitching; clue-aware similarity; frog leaping strategy; contour curve

OCIS codes 100.2000; 100.2960; 100.3010

收稿日期：2019-01-17；**修回日期**：2019-01-29；**录用日期**：2019-03-04

基金项目：国家自然科学基金(21365008,61105004)

* **E-mail**：670927905@qq.com

1 引 言

高分辨率、宽视场的工业图像对于缺陷检测、毛刺预警、劈裂诊断等非常重要，但受工业镜头传感器的限制，存在视场与分辨率之间矛盾难以平衡的问题。

为了满足焦距和分辨率的双重要求，通常采用机载工业相机对某一区域以不同视角、一定重叠率进行连续采集，然后将采集的图像序列利用图像拼接算法生成高分辨率、宽视场的整张图像。传统的图像拼接算法包括图像配准和图像融合两部分，其中高效精确的配准算法对图像拼接至关重要，图像拼接算法可依据配准方法分为基于区域的方法（如模板匹配算法）和基于特征的方法两类。基于特征点匹配的拼接算法是目前研究的热点之一。仇国庆等[1]提出的基于自适应阈值的 Harris 角点检测改进拼接算法，具有较高的稳健性和可靠性，但可能产生伪角点，难以满足精确定位要求。许佳佳等[2]和徐鑫等[3]提出的 Harris 与尺度不变特征变换（SIFT）算子相结合的图像配准方法，对图像尺度变换和缩放具有较好的稳健性，但由于其计算复度较高，难以满足实际工程的实时性。黄琼丹等[4]提出的基于相位一致性特征点的匹配方法与 Levenberg-Marquardt（LM）优化理论方法相结合的拼接算法，具有较高的准确度，但易受匹配点间距的影响。封靖波等[5]提出的利用两张图像的列梯度最大点寻找两条曲线最匹配区域的图像拼接方法，简化了匹配策略，但特征匹配精度不足。上述拼接算法已在诸多领域得到广泛应用，但是在工业显微图像应用领域具有很大的局限性，其中特征描述统计类方法存在复杂度高和时效性差等缺点；另外，受机械振动、离焦模糊等因素的影响，上述算法会使拼接后的图像产生几何畸变，进而干扰后续图像缺陷的定位和测量结果。

针对上述问题，本文提出基于区域蛙跳搜索（RFL）的轮廓最优匹配算法，通过寻找图像轮廓线索感知相似度（CATS）和均方误差距离（MSED）实现图像拼接。该方法简化匹配策略并减小计算量，具有较高的拼接精度和稳定性，以保证拼接的质量和速度，可用于特征稀少的图像，以及工业显微图像等机器视觉领域的图像拼接。

2 图像轮廓曲线距离

针对连续采集且有重叠区域的前后两张图像，引入两种指标衡量其轮廓曲线的相似距离，用于图像的精确匹配。其中，CATS 是基于轮廓曲线形状的距离，而 MSED 是基于轮廓曲线点的距离，两者分别用于计算图像轮廓曲线间的相似度和曲线离散距离。

2.1 轮廓线索感知相似度量

给定空间阈值 ε 和分别来自两条轮廓曲线（T_i 和 T_j）的数据点 $p_{i,l}=(x_{i,l},y_{i,l})$，$p_{j,k}=(x_{j,k},y_{j,k})$，$l$ 为该点在此轮廓的索引顺序，定义 $p_{i,l}$ 和 $p_{j,k}$ 的空间衰减函数[6]为

$$f_\varepsilon(p_{i,l},p_{j,k})=\begin{cases}0, & \text{if } D_{\text{dist}}(p_{i,l},p_{j,k})>\varepsilon\\ 1-\dfrac{D_{\text{dist}}(p_{i,l},p_{j,k})}{\varepsilon}, & \text{otherwise}\end{cases},\tag{1}$$

式中：$D_{\text{dist}}(\cdot,\cdot)$表示两个数据点之间的欧几里德距离。空间衰减函数的取值范围为 0 到 1，两个数据点越近，则值越大；如果两个数据点的位置完全相同，则值为 1；如果两个点之间的距离大于 ε，则值为 0。图 1 为两个轮廓线 T_1 和 T_2 的轮廓线索感知相似性，图中实心点表示 T_1 和 T_2 的数据点，轮廓线上下标出的数字表示其在曲线中的序列顺序。

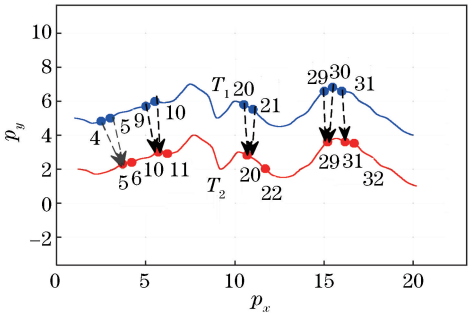

图 1 T_1 到 T_2 的轮廓线索感知相似性

Fig. 1 Clue-aware trajectory similarity from T_1 to T_2

在空间衰减函数中，参数 ε 可以包容数据点的空间偏差和移位。空间衰减函数执行连续的空间量化（即从 0 到 1），反映了两个数据点之间的接近程度，如果将其值设置为 10，则考虑两种情况：1）距离为 1 的两个点；2）距离为 9 的两个点。如上所述，对于两个数据点，如果它们两者之间的位置越接近，那么它们之间的线索感知相似性就越强，因为两者很可能位于彼此的附近区域。因此，在这种情况下，情况 1 揭示了比情况 2 相似性更强的线索。而在最长公共子序列（LCSS）和编辑距离（EDR）中使用的离散空间量化（即 0 或 1），不区分这两种情况，因为其距离都小于 10。

根据数据点的空间和序列信息，可以计算关于参考轮廓数据点的分数。对于数据点，有多种方法可以评估该点和参考轮廓之间的线索，常用方法是

通过容忍一些序列顺序上的偏移，将该数据点映射到参考轮廓上最近的点，即将数据点与参考轮廓中线索性最强的点对齐。为此，定义数据点相对于参考轮廓曲线的线索评分，给定数据点 $p_{i,\ell}\in T_i$ 和参考曲线 T_j，将数据点 $p_{i,\ell}$ 相对于曲线 T_j 的线索用于从空间和序列维度上识别针对 $p_{i,\ell}$ 在曲线 T_j 上的最佳映射点。

在线索评分的定义中，将序列参数 τ 用于检索其发生在特定序列间隔内的轮廓的数据点，使用 CATS 处理序列偏移。由于要映射的数据点受序列参数的约束，线索评分对序列敏感，同样，线索评分也对位置敏感。但其仍然能够容忍空间和序列阈值指定的空间和时间偏差。根据数据点定义的线索评分，将两条轨迹之间的线索感知相似性定义如下：给定空间阈值 ε 和序列阈值 τ，从 T_i 到 T_j 的线索感知轨迹相似性为

$$C_{\mathrm{CATS},\varepsilon,\tau}(T_i,T_j)=\frac{1}{|T_i|}\times\sum_{p_{i,\ell}\in T_i}S_{\mathrm{score},\varepsilon,\tau}(p_{i,\ell},T_j)。\tag{2}$$

图 1 显示了根据线索评分进行的数据映射关系，其中有两个轮廓曲线 T_1 和 T_2，设 $\varepsilon=5$，$\tau=3$。例如，轮廓 T_1 上的数据点 $p_{1,9}$，关于轮廓 T_2 的数线索在 $\ell-\tau$ 和 $\ell+\tau$ 之间的序列间隔内，寻找与 T_2 的最佳映射数据点。$f_5(p_{1,9},p_{2,10})$ 是所有其他点中的最大值，所以 $S_{\mathrm{score},5,3}(p_{1,9},T_2)=f_5(p_{1,9},p_{2,10})$，找到最佳映射。根据数据点的线索进行评分匹配，图 1 中箭头表示从 T_1 的每个数据点到 T_2 的数据点的映射关系，最终可知从 T_i 到 T_j 的线索感知轨迹相似性 $C_{\mathrm{CATS},\varepsilon,\tau}(T_i,T_j)$。CATS 的算法流程如图 2 所示。

2.2 MSED

图 3 表示轮廓曲线中所有数据点及其相应的轨迹。从图 3 可以看出，T_1 和 T_2 的数据点在序列或空间维度上并不完全对齐，即使这两个轮廓表示从图像同一位置提取的轮廓线，这种不完全对齐造成的偏差称之为轮廓的序列偏移或空间偏差。由于数据丢失或在形成轮廓时采用的边缘提取策略的差异，可能出现没有数据点存在的持续序列。

在连续曲线中，给定轮廓曲线的函数 $f_1(x)$、$f_2(x)$，定义 $d(f_1,f_2)=\int_{c_0}^{c_1}|f_1(x)-f_2(x)|\mathrm{d}x$ 为两个函数之间的距离，$[c_1,c_2]$ 为函数的定义域，给定阈值 ε，如果 $d(f_1,f_2)\leqslant\varepsilon$，则认为 $f_1(x)$、$f_2(x)$ 相似，否则不相似。当计算轮廓离散曲线距离时，进行如下插补方法。

Algorithm1: CATS:Clue-Aware Trajectory Similarity Algorithm

Input: Trajectories: T_i, T_j; Thresholds: ε, τ

Output: CATS value: $C_{\mathrm{CATS}_{\varepsilon,\tau}}(T_i,T_j)$

$T_{\mathrm{TotalScore}}\leftarrow 0$;
foreach $p_{i,k}\in T_i$ **do**
  foreach $p_{j,\ell}\in T_j$ **do**
    $T_{\mathrm{ClueScore}}\leftarrow 0$;
    If $|t_{i,k}-t_{j,\ell}|\leq\tau$, **then**
      $T_{\mathrm{ClueScore}}\leftarrow\max[T_{\mathrm{ClueScore}},f_\varepsilon(p_{i,k},p_{j,\ell})]$;
  $T_{\mathrm{TotalScore}}\leftarrow T_{\mathrm{TotalScore}}+T_{\mathrm{ClueScore}}$;
  return $T_{\mathrm{ClueScore}}/\mathrm{length}(T_i)$.

图 2 线索感知相似度的算法流程

Fig. 2 Algorithm flowchart of clue-aware trajectory similarity

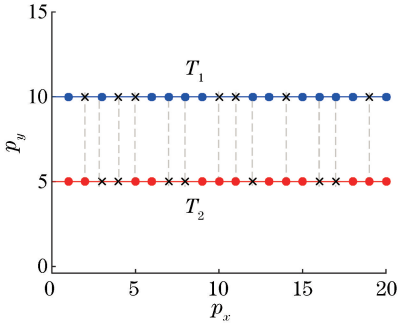

图 3 轮廓曲线 T_1 和 T_2 的插补

Fig. 3 Interpolation of contour curves T_1 and T_2

1）从 n 张图像中分别提取图像的边缘 $T=\{T_1,T_2,\cdots,T_n\}$，第 i 张图像的边缘曲线为 $T_i=\{p_{i,1},p_{i,2},\cdots,p_{i,n}\}$，数据点 $p_{i,\ell}$ 表示边缘曲线 T_i 的像素坐标；T_i 中的数据点在水平方向并不是连续的，可能存在缺失，如图 3 所示，我们称为待补充数据 $B_i=\{b_{i,1},b_{i,2},\cdots,b_{i,m}\}$，分别统计 B_i 中属于水平序列上连续的空缺数据段，设共有 s 个连续的空缺数据段，每段上连续空缺数据数量为 q；

2）当 $s=0$ 时，直接执行步骤 6），当 $s\neq0$ 时重复执行步骤 3）～5）；

3）设第 s 个连续空缺数据段在 T_i 中的位置为 $\{b_{i,j},b_{i,j+1},\cdots,b_{i,j+q-1}\}$；

4）求 $\Delta p=\dfrac{x_{j+q}-x_{j-1}}{q+1}$ 和 $x_{j+k}=\sum_{k=0}^{q-1}[x_{j-1}+(k+1)\times\Delta p]$；

5）$s=s-1$，返回步骤 2）；

6）对于不同图像的轮廓曲线，T_i 经过步骤 2）～5）后可以得到插补后的轮廓 $\hat{T}_i=\{\hat{p}_{i,1},\hat{p}_{i,2},\cdots,\hat{p}_{i,n}\}$。

设定轮廓曲线相似观测窗口的尺度为 L，则边缘曲线 T_i 和 T_{i+1} 位于观测窗口区域的区间分别为：$[|\hat{T}_i|-L,|\hat{T}_i|]$，$[1,L]$，在相似观测窗口中边缘曲线 T_i 和 T_{i+1} 的 MSED 为

$$M_{\mathrm{MSED}}(T_i,T_{i+1},L)=\frac{1}{L}\sum_{l=0}^{L-1}\left|p_{i,|\hat{T}_i|-L+l}-p_{i+1,l+1}\right|^2。\tag{3}$$

设轮廓匹配阈值为 ε，若 $M_{\mathrm{MSED}}(T_i,T_{i+1})<\varepsilon$，则轮廓 T_i 与 T_{i+1} 在观测相似度尺度窗口 L 中匹配，即边缘 T_i 和 T_{i+1} 重叠区域拼接成功，否则匹配失败。MSED 的算法流程如图 4 所示。

Algorithm2:MSED:Mean Square Error Distance

Input: Trajectories: T_i, T_j; Window size: L

Output: MSED value: $M_{\mathrm{MSED}}(T_i,T_j)$

Count s, q;

Initialize B_i;

while $s\neq 0$ **do**

$\Delta p\leftarrow\frac{x_{j+q}-x_{j-1}}{q+1}$;

for k **from** 0 **to** $q-1$

$x_{j+k}\leftarrow\sum_{k=0}^{q-1}[x_{j-1}+(k+1)\times\Delta p]$;

Update s, q;

Update B_i;

$\hat{T}_i\leftarrow\{T_i,B_i\}$, $\hat{T}_j\leftarrow\{T_j,B_j\}$;

return $M_{\mathrm{MSED}}(T_i,T_j,L)$;

图 4 均方误差距离的算法流程

Fig. 4 Algorithm flowchart of mean square error distance

3 区域蛙跳搜索算法

3.1 算法原理

区域蛙跳算法模拟了青蛙群体(解)在池塘(解空间)中跳跃觅食的行为。利用蛙鸣分贝越大，对同伴吸引力越强这一现象，通过青蛙个体在决策域范围内寻找鸣叫分贝最高的邻居来实现寻优的目的。

在觅食过程中，每只青蛙作为群体之间交流的载体，发出特定频率的鸣叫，以此方式与其他青蛙实现信息传递和思想交流。借鉴萤火虫优化算法[7]中群体交流的思想，对其进行改进及简化。蛙鸣声音分贝大小取决于其决策区域内 $0<r\leqslant r_s^i$ 猎物密度，并与其所在位置的目标函数相关，分贝越大的青蛙表示其所在的位置越好，即有较优的目标值。青蛙个体在决策域范围内的邻居集合中寻找最优邻居，在邻居集合 N_{d}^i 中，鸣叫分贝越大的邻居 N_{obj}^i 具有越强的吸引力，能够吸引青蛙向这个方向跳跃，每次蛙跳方向的选择都会随着决策域邻居的不同而改变。此外，决策域的大小也会受到邻居蛙鸣分贝的影响，目标猎物越近，青蛙的决策半径越小，以便寻找更精准的邻居；如果邻居蛙鸣分贝较弱，决策半径保持不变。

群体中每只蛙都包含其所属区域的目标函数，利用其周围不同邻居传递的信息，进行蛙跳搜索，每次搜索和跳跃迭代行为结束后进行一次信息交流，该过程一直重复演进，并持续到收敛条件为止。

3.2 算法流程

蛙跳算法主要包括蛙群的初始分布、鸣叫分贝的更新、青蛙的跳跃和决策域的更新四个阶段。

初始化青蛙群体分布。参数初始化，设定基蛙种子数 s，在第 i 只青蛙周围生成 s 个邻居蛙形成初始蛙群，初始跳跃步长 S_{step0}，最小步长 S_{stepmin}，青蛙听觉感知分贝 $D_{\min}$，迭代次数 t，最大迭代次数 $I_{\mathrm{iter\_max}}$；初次蛙跳，初始跳跃位置 $x_0(t)$，青蛙的初始鸣叫分贝 D_0^i，即第 i 只青蛙在 t 次迭代时的位置 $x_i(t)$ 对应的目标函数值 $D_0^i=J[x_i(t)]$。

决策域半径及邻居集合。设置决策域半径 r_0^i，感知范围 $0<r\leqslant r_s^i$，决策邻居集合 N_s^i 和位置 X_s^i 为

$$[r\lambda(-1)^m,\ r(1-\lambda)(-1)^n]\&[r(-1)^m,\ r(-1)^n],\ \lambda,\ m,\ n=0,1。\tag{4}$$

鸣叫分贝更新。更新邻居集合 N_s^i 内每只青蛙的鸣叫分贝 D_s^i，其中 $D_c^i=J[x_c^i(t)]$，$0<c\leqslant s$。

最优邻居选择和信息交流。最优邻居 N_{obj}^i 将邻居集合内的青蛙个体按照分贝(目标函数)值降序排列，找到全局最优解 D_{obj}^i；通过与邻居集合之间的分贝传递进行信息交流，得到蛙跳激活值

$$F_{\mathrm{flag}}=\begin{cases}1,\ \text{if}\ (D_{\mathrm{obj}}^i>D_0^i)\cap(D_{\mathrm{obj}}^i>D_{\min})\\0,\ \text{otherwise}\end{cases}。\tag{5}$$

蛙跳策略。青蛙是否跳跃到新的解空间取决于蛙跳激活值。当 $F_{\mathrm{flag}}=1$，表示最优邻居蛙 N_{obj}^i 位置具有更强的吸引力，且在听觉范围 $D_{\min}$ 内，以步长 $S_{\mathrm{step},t}=r_{\mathrm{obj}}^i(t)$ 发生跳跃；当 $F_{\mathrm{flag}}=0$ 时，则不发生跳跃，缩小决策半径继续寻找新邻居。

更新当前位置分贝：

$$D_0^i=J[x_i(t+1)]=F_{\mathrm{flag}}\cdot D_{\mathrm{obj}}^i+(1-F_{\mathrm{flag}})\cdot D_0^i。\tag{6}$$

决策域更新：

$$r_{\mathrm{d}}^i(t+1)=F_{\mathrm{flag}}\cdot r_{\mathrm{d}}^i(t)+(1-F_{\mathrm{flag}})\cdot\frac{r_{\mathrm{d}}^i(t)}{2}。\tag{7}$$

判断收敛条件。当蛙跳步长 S_{step} 和迭代次数 t 满足算法收敛条件时停止计算，否则重新设置邻居集合。

4 轮廓曲线拼接算法

4.1 拼接算法流程

基于区域蛙跳搜索的轮廓最优匹配算法（RFL-CSA），寻找图像中最优的匹配轮廓线，并依据此变换作为图像匹配的最终结果，实现图像拼接。由于工业显微系统下的图像通常为灰度图像，且金属材质表面可用特征稀少，为避免陷入 SIFT 和加速稳健特征（SURF）算法中特征点匹配失败的困境，RFL-CSA 算法将零件的边缘作为可用信息，提取图像的轮廓曲线，用于图像匹配。其中，提取轮廓曲线的方法包括高斯中值滤波、形态学操作、canny 边缘检测等步骤，从连续采集的图像中分别提取轮廓。在初始重叠区间内，计算其轮廓曲线的相似度函数，在此解空间内利用区域蛙跳算法寻优，求解相似度最高的轮廓匹配，使轮廓曲线完全吻合，达到精准的图像拼接效果。基于区域蛙跳搜索的图像拼接算法流程图如图 5 所示。

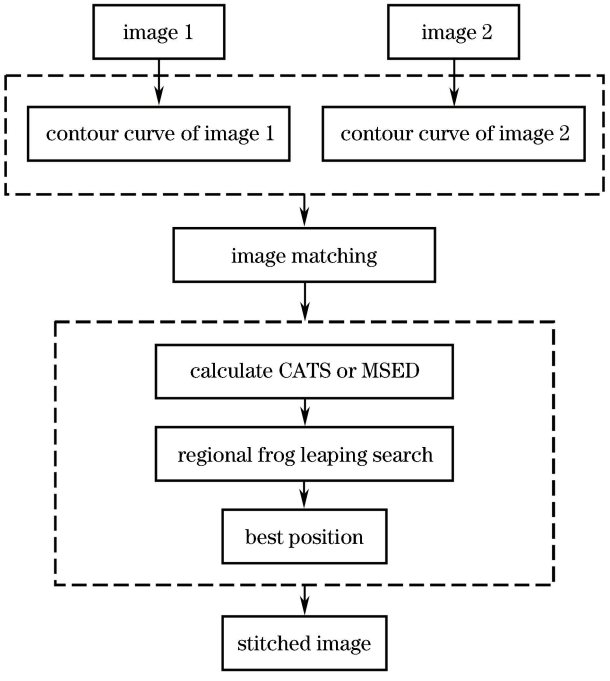

图 5 基于区域蛙跳的图像拼接算法流程图

Fig. 5 Flowchart of regional frog leaping based image stitching algorithm

两张连续采集图像 I_i 和 I_{i+1}，其两条轮廓曲线分别为 T_i 和 T_{i+1}，如图 6 所示。图 6(a)为初始位置，图 6(b)为调整后的位置。设此相邻的两条轮廓

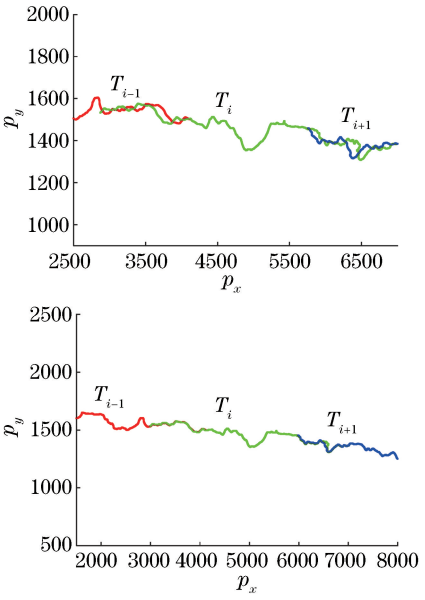

图 6 图像轮廓线的匹配。(a)初始位置；(b)调整后的位置

Fig. 6 Matching of image contour curves. (a) Initial position; (b) adjusted position

在首尾区域的重叠部分约占比例为 κ_0，首先进行初次蛙跳，T_{i+1} 中每个数据点 $p_{i,l}$ 两个分量的初次跳跃步长分别为 $\Delta H = p_{y_{i,\Delta W}} - p_{y_{i+1,1}}$，$\Delta W = \kappa_0 \cdot |T_i|$，即，

$$p_{i,l} = (p_{x_{i,l}} + \Delta W,\ p_{y_{i,l}} + \Delta H),\ p_{i,l} \in T_i。 \tag{8}$$

首先计算轮廓曲线 T_{i+1} 跳跃到初始位置 $x_0(t)$ 时与 T_i 的轮廓距离 $D_0^i = M_{\text{MSED}}(T_i, T_{i+1})$。设定邻居种子数 s，最大迭代次数 $I_{\text{iter\_max}}$，最小步长 S_{stepmin} 和感知分贝 $D_{\min}$，通过区域蛙跳算法寻找最优邻居。第 t 次迭代时，在 $x(t)$ 位置周围生成 s 个邻居，根据最优邻居和感知取值，采用蛙跳策略决定是否起跳，当 $F_{\text{flag}}=1$ 时确定蛙跳步长和方向，更新当前轮廓的位置 $x_i(t)$ 和此位置的轮廓匹配度，进入下一轮迭代；否则缩小决策半径，在较近的决策域寻找新的邻居，跳向轮廓相似度更高的位置。RFL-CSA 的算法流程如图 7 所示。

根据所得最优拼接线 T_1 生成两张掩码图像用于图像缝合，拼接后的重叠区域起点做竖直线 T_2，作为缝合线，将缝合线以及左侧像素值设置为 1，右侧像素值设置为 0，得到左侧掩码图，同理生成右侧掩码图。然后将两张连续采集的原始图像分别点乘其对应的掩码图，经过叠加产生最后的拼接图，如图 8 所示。图 8(a)为相邻且有重叠区域的三张原始图像，图 8(b)为经过算法产生的拼接图像。拼接后整张图像的像素值表示为

Algorithm3:RFL-CSA:Regional Frog Leaping based Contour Stitching Algorithm

Input: Trajectories: T_i, T_{i+1};Seed: s;
Thresholds: $D_{\min}$, $I_{\text{iter\_max}}$, r_0;
Output: Trajectories: T'_{i+1};Step: ΔW, ΔH

$\Delta W=\kappa_0\cdot|T_i|$, $\Delta H=\Delta H_0$
$T_{i+1}\leftarrow T_{i+1}+(\Delta W,\Delta H)$;
$D_0^i=D_{\text{dist}}(T_i,T_{i+1})$
while $(r^i(t)>r_0)\wedge(t<I_{\text{iter\_max}})$ **do**
  Update N_s^i: $(r^i(t)\cdot(-1)^m,r^i(t)\cdot(-1)^n)$, $m,n=0,1$;
  foreach $N_c^i\in N_s^i$ **do**
    $T_{i+1}^c\leftarrow T_{i+1}+N_c^i$;
    Update overlap interval: $[|\hat{T}_i|-L,|\hat{T}_i|]$, $[1,L]$;
    $D_c^i=D_{\text{dist}}(T_i,T_{i+1}^c)$;
  $D_{\text{obj}}^i\leftarrow\max(D_c^i)$;
  if $(D_{\text{obj}}^i>D_0^i)\wedge(D_{\text{obj}}^i>D_{\min})$ **then**
    $T_i\leftarrow T_i+N_{\text{obj}}^i$;
    $\Delta W\leftarrow\Delta W+N_{\text{obj}}^i(p_x)$;
    $\Delta H\leftarrow\Delta H+N_{\text{obj}}^i(p_y)$;
    Update D_0^i;
  else
    $r^i(t+1)=r^i(t)/2$;
return T'_{i+1}.

图 7 RFL-CSA 图像拼接算法流程

Fig. 7 Flowchart of regional frog leaping based contour stitching algorithm

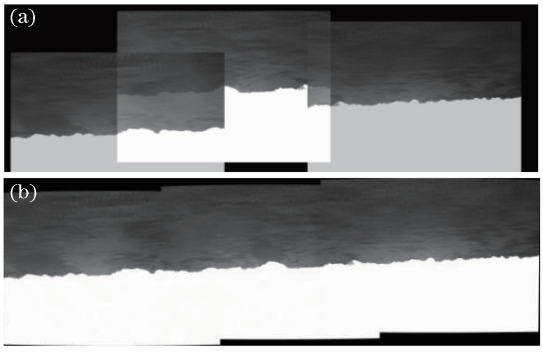

图 8 图像拼接结果。(a)原始图像;(b)拼接图像

Fig. 8 Results of image stitching. (a) Original images; (b) stitched image

$$P^*(x,y)=\begin{cases}P_1(x,y),x\leqslant\max T_2(y)\\P_2(x,y),x>\max T_2(y)\end{cases}。\quad(9)$$

4.2 轮廓线拼接算法的优化改进

1) 相似距离衡量指标

RFL-CSA 是一种基于区域蛙跳的图像拼接算法,其中的相似距离可以采用多种衡量指标:当 RFL-CSA 算法采用 CATS 作为相似距离的衡量指标时,称为 RFL-CSA 算法;当 RFL-CSA 算法采用 MSED 作为相似距离的衡量指标时,称为 RFL-MSED 算法。在计算轮廓曲线之间的相似距离时,分别采用基于点的距离 MSED 和基于形状的距离 CATS 两种方法作对比,即在 RFL-CSA 图像拼接算法的基础上分别形成了 RFL-MSED 和 RFL-CSA 两种拼接算法。另外,RFL-CATS8、RFL-MSED4_1、RFL-MSED4_2、RFL-MSED8 具体的搜索方式略有差异,其不同之处在于蛙跳搜索邻居集合的设置。其中,RFL-MSED4_1 和 RFL-MSED4_2 的初始子数 $s=4$,RFL-MSED8 和 RFL-CATS8 的初始子数 $s=8$。

2) 蛙跳搜索方式

在设置区域蛙跳算法决策域内的邻居集合中采用三种方式,寻找最优邻居。

在初始化青蛙群体分布时,基蛙种子数 s 默认为 4。当种子数 $s=4$ 时,设置两种搜索方式,决策邻居集合分别为 $N_{1,s=4}^i$ 和 $N_{2,s=4}^i$,即

$N_{1,s=4}^i$:$[r(-1)^m,r(-1)^n]$,$m,n=0,1$;

$N_{2,s=4}^i$:$[r\lambda(-1)^m,r(1-\lambda)(-1)^n]$,$\lambda,m,n=0,1$。

此外,当种子数 $s=8$ 时,集合为 $N_{s=8}^i$:$N_{s=8}^i=N_{1,s=4}^i\cup N_{2,s=4}^i$。

5 实验结果和讨论

本文实验硬件条件为 Intel(R) Core(TM) i5-6500 CPU @ 3.20 GHz,8 G RAM。

显微图像采集装置由目镜、物镜、摄像机和光源组成,整个显微系统决定了显微检测系统图像的分辨率。为了增加显微图像的可用特征和边缘信息,使用双光源的打光方式进行采集,上光源采用环形光,底光源采用平行光。采用 Carl Zeiss 显微物镜,摄像机采用 Basler 工业相机。实验使用的图像数据为动力电池分切刀具金属零件显微图像。

5.1 参数设置

最小决策半径 r_0 为 0.5,最大迭代次数 $I_{\text{iter\_max}}$ 均为 45,初始重叠比例 $\kappa_0=0.33$。采用 1200 张具有重叠区域的图像作为实验数据,对本文算法分别进行 600 次实验,其中包括了新出厂刀具、用后刀具、待返修刀具的显微图像,各取其平均结果,如图 7 所示。同时给出 RFL-CATS 和 RFL-MSED 算法的收敛曲线及搜索轨迹坐标,如图 9~10 所示。

5.2 算法有效性测试

为了检验本文算法的有效性,分别采用不同邻

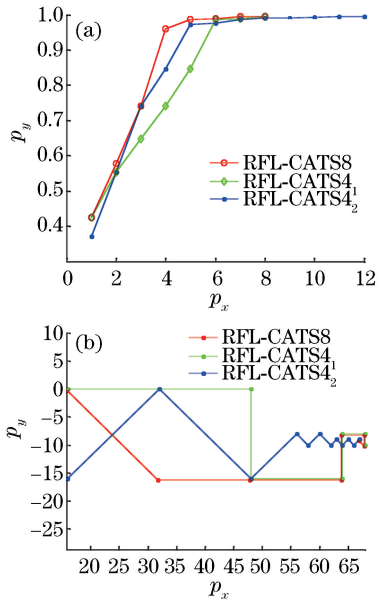

图 9 RFL-CATS 算法收敛曲线及搜索轨迹坐标。(a)收敛曲线;(b)搜索轨迹坐标

Fig. 9 Convergence curves and search trajectory coordinates of RFL-CATS algorithm. (a) Convergence curves; (b) search trajectory coordinates

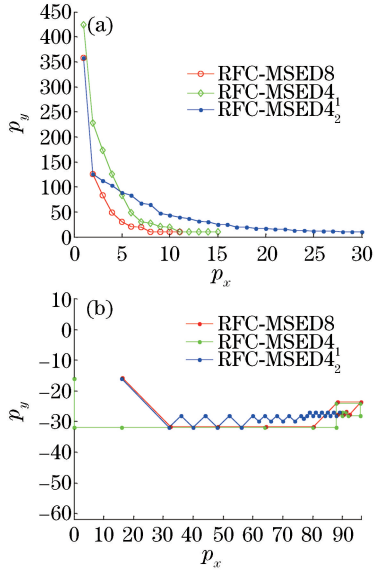

图 10 RFL-MSED 算法的收敛曲线及搜索轨迹坐标。(a)收敛曲线;(b)搜索轨迹坐标

Fig. 10 Convergence curves and search trajectory coordinates of RFL-MSED algorithm. (a) Convergence curves; (b) search trajectory coordinates

域种子 N_s^i 对 RFL-CATS 和 RFL-MSED 拼接算法进行测试。图 9(a)和(b)分别为 RFL-CATS 算法的收敛曲线和搜索轨迹;图 10(a)和(b)分别为 RFL-MSED 算法的离散距离曲线和轨迹图。对于 RFL-CATS 拼接算法,由图 9 可以看出,达到收敛时所需的迭代次数总体较少,迭代 10 次以内即可收敛,其中 RFL-CATS8 具有较高的求解精度和速度。对于 RFL-MSED 拼接算法,由图 10 可知,RFL-MSED8 迭代约 10 次即可达到最优值,平均求解误差比 RFL-MSED4_1 和 RFL-MSED4_2 减小 70%~85%。

为了进一步对比 RFL-CATS 算法与 RFL-MSED 算法在搜索精度、速度等优化性能上的差异,将 RFL-CATS 算法达到收敛时的 C_{CATS} 值转化为 M_{MSED} 值进行对比,距离指标 M_{MSED} 为离散曲线的绝对平方误差,曲线相似度指标 C_{CATS} 的取值为 0~1。RFL-CATS 算法收敛时的绝对平方误差 $M_{\mathrm{MSED}}=31.436$,RFL-MSED 算法达到稳定时 $M_{\mathrm{MSED}}=9.522$。结果表明,在相同的收敛条件下,RFL-MSED 算法具有较快的搜索速度和较小的求解误差,其寻优精度和稳定性高于 RFL-CATS 算法。由本研究所用的光学系统精度(0.345 μm/pixel)可知,RFL-MSED 算法的实际图像拼接误差为 3.29 μm。

5.3 算法的效率测试

为了进一步对本文算法的效率进行测试,对几种拼接算法进行对比测试实验。分别为 RFL-CATS 和 RFL-MSED 两个算法指定收敛精度,通过 600 次实验比较 M_{MSED} 与 C_{CATS} 在达到目标精度时的平均时耗和成功率(达到目标精度的实验次数/总实验次数)。

指定迭代次数上限为 50,即经过 50 次迭代均未能达到指定精度,认为算法匹配失败。数值实验结果如图 10 所示。由图 10 中数据可以明显看出,RFL-CATS 算法耗时次数远远少于 RFL-MSED 算法,但是正确匹配率略低,而 RFL-MSED 算法中 RFL-MSED8 算法只需要相对较少的迭代次数就能达到指定精度,表明 RFL-MSED8 算法不仅具有更好的寻优精度,还减少了迭代次数,加快了收敛速度。

另外,由表 1 可知,对于本文提出的基于图像轮廓曲线的拼接算法:RFL-MSED 算法的正确匹配成功率均为 100%,RFL-CATS 拼接算法为 99.5%;而基于 SIFT 和 SURF[8-9] 拼接算法的图像成功率较低,SIFT-Stitch 的成功率不足 80%。结果表明,RFL-MSED 算法的正确匹配率更高,且具有相对较快的收敛速度,有效地解决了 SIFT 和 SURF 匹配算法中特征点个数不足、算法成功率低等问题,且效果较好。

表 1 拼接算法比较

Table 1 Comparison of different image stitching algorithms

Algorithm	Time /s	Accuracy /%	Precision /μm
RFL-CATS8	2.940	99.5	10.85
RFL-MSED4$_1$	6.720	100.0	3.34
RFL-MSED4$_2$	5.990	100.0	4.28
RFL-MSED8	3.167	100.0	3.21
SIFT-Stitch	25.900	78.0	—
SURF-Stitch	18.530	83.5	—

6 结 论

基于轮廓线离散距离和区域蛙跳寻优算法，建立了一种图像最优拼接线的搜索算法，用于机器视觉场景下显微图像的拼接。在图像轮廓线离散距离衡量准则、决策域邻居集合选择等方面进行优化，不仅提高了拼接速度，还提高了精准拼接精度，能够保证拼接目标的完整性、去除重叠区域，同时简化的匹配策略减小了算法的计算量，增强了算法的时效性。实验数据表明，区域蛙跳搜索轮廓匹配算法有效提高了求解效率和质量，具有较好的优化性能和实用性。

参 考 文 献

[1] Qiu G Q, Feng H Q, Jiang T Y, *et al*. Improved image mosaic algorithm based on Harris corner[J]. Computer Science, 2012, 39(11): 264-266, 297.
仇国庆, 冯汉青, 蒋天跃, 等. 一种改进的 Harris 角点图像拼接算法[J]. 计算机科学, 2012, 39(11): 264-266, 297.

[2] Xu J J. Fast image registration method based on Harris and SIFT algorithm [J]. Chinese Optics, 2015, 8(4): 574-581.
许佳佳. 结合 Harris 与 SIFT 算子的图像快速配准算法[J]. 中国光学, 2015, 8(4): 574-581.

[3] Xu X, Sun S Y, Sha Y J, *et al*. A method of infrared image mosaic based on improved RANSAC [J]. Laser & Optoelectronics Progress, 2014, 51(11): 111001.
徐鑫, 孙韶媛, 沙钰杰, 等. 一种基于改进 RANSAC 的红外图像拼接方法[J]. 激光与光电子学进展, 2014, 51(11): 111001.

[4] Huang Q D, Qiu Y H, Tian X P. Image automatic mosaic method based on feature points and optimization theory[J]. Acta Photonica Sinica, 2009, 38(8): 2139-2143.
黄琼丹, 邱跃洪, 田小平. 基于特征点及优化理论的图像自动拼接方法[J]. 光子学报, 2009, 38(8): 2139-2143.

[5] Feng J B, Su Z X, Liu X P. An similar-curve based automatic mosaic algroithm of panoramic image[J]. Chinese Journal of Computers, 2003, 26(11): 1604-1608.
封靖波, 苏志勋, 刘秀平. 一种基于相似曲线的全景图自动拼接算法[J]. 计算机学报, 2003, 26(11): 1604-1608.

[6] Hung C C, Peng W C, Lee W C. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes[J]. The VLDB Journal, 2015, 24(2): 169-192.

[7] Ouyang Z, Zhou Y Q. Self-adaptive step glowworm swarm optimization algorithm [J]. Journal of Computer Applications, 2011, 31(7): 1804-1807.
欧阳喆, 周永权. 自适应步长萤火虫优化算法[J]. 计算机应用, 2011, 31(7): 1804-1807.

[8] Li Z, Shen X M, Miao T J. Image mosaic based on contract threshold adaptive SIFT algorithm [J]. Infrared Technology, 2017, 39(10): 946-950.
李尊, 申小萌, 苗同军. 对比度阈值自适应的 SIFT 图像拼接算法[J]. 红外技术, 2017, 39(10): 946-950.

[9] Xia Y, Liu Z, Wang J R. Research on SURF registration algorithm in image mosaic[J]. Journal of Changchun University of Science and Technology (Natural Science Edition), 2017, 40(2): 98-101.
夏岩, 刘智, 王俊然. 图像拼接中 SURF 配准算法的研究[J]. 长春理工大学学报(自然科学版), 2017, 40(2): 98-101.