

文章编号: 2095-4980(2022)05-0479-08

基于一致性敏感哈希的高精确度图像匹配算法

蒋子贤

(南京工业大学 计算机科学与技术学院, 江苏 南京 211816)

摘 要: 针对基于特征点的图像匹配方式在复杂纹理场景中匹配效果不理想的问题, 提出一种将加速稳健特征算法(SURF)与一致性敏感哈希匹配结合的图像匹配算法(CSH)。使用 SURF 算法对图像进行特征点提取, 再以特征点为圆心构建特征区域, 最后对特征区域使用 CSH 进行匹配, 从而实现高精度匹配。为了进一步加快算法运行速度, 对现有的 SURF 算法进行修改, 在提取 SURF 特征点时去除了对于特征点方向的计算。仿真实验证明, 算法较一般的特征算法在复杂纹理图像匹配中效果更佳, 且较 CSH 算法效率提升了 10%~15%。

关键词: 图像匹配; 纹理匹配; 加速稳健特征算法; 一致性敏感哈希匹配算法

中图分类号: TP751.1

文献标志码: A

doi: 10.11805/TKYDA2020230

High precision image matching algorithm based on consistency sensitive Hashing

JIANG Zixian

(Department of Computer Science and Technology, Nanjing Tech University, Nanjing Jiangsu 211816, China)

Abstract: The feature point-based matching methods suffer from unsatisfactory matching effect when employed in complex texture scenes. Therefore, a block based image matching method, which combines Speeded Up Robust Features(SURF) algorithm and Coherency-Sensitive Hashing(CSH) is proposed. This method is divided into three stages: firstly, using the SURF algorithm to extract feature points from the image; secondly, determining the feature areas with the SURF feature points as the center; thirdly, the feature areas are matched by CSH. In order to further speed up the method, SURF algorithm is modified to remove the calculation of the direction of feature points when extracting feature points. Experimental results show that the new method is better than the general image matching methods based on feature point in complex texture image matching. The efficiency of the proposed method has been increased by 10% to 15% compared with that of the CSH algorithm.

Keywords: image matching; texture matching; Speeded Up Robust Features; Coherency-Sensitive Hashing

图像匹配技术指的是把 2 幅图像进行三维空间上的配准, 并在 2 幅图像之间找寻共有的子图像。图像匹配技术应用十分广泛, 例如: 指纹识别、辅助医疗影像分析协助诊断、流水线智能监控等。对于强纹理图像匹配一直是图像匹配研究中的难点。BARNES C 等提出了基于图像块的匹配方法 Patch-Match^[1], 且它被证明在图像纹理匹配方面非常优秀。Patch-Match 通过发挥图像的连贯性的特性, 以图像中的块为基础, 向其周围的块传递高质量的匹配。Patch-Match 随机选定块作为原始的匹配种子, 缺点是这样做会产生许多质量较差的匹配, 从而增加算法执行时的迭代次数。局部敏感哈希(Locality-Sensitivity Hashing, LSH)^[2]是一种专门用来处理大量高维数据最近邻查找的算法, LSH 通过对高维数据进行哈希运算处理, 运用特殊的哈希函数使得相似度很高的数据以较高的概率落入同一个桶中, 在进行相似数据查找时, 仅对特定桶中的数据进行线性查找即可。CSH^[3]算法将二者合并, 用 LSH 中的哈希算法替代了 Patch-Match 中的随机搜索, 这大大提高了匹配中的针对性和匹配的 efficiency。换言之, 就是让匹配在图像平面中距离接近的或外观相似的块中进行。这使得它的匹配速度要快于 Patch-Match 4 到

收稿日期: 2020-05-24; 修回日期: 2020-08-17

基金项目: 江苏省“六大人才高峰”基金资助项目(2012-WLW-023)

5 倍。同时 CSH 被证明在纹理匹配精准度方面要强于 Patch-Match 很多。

为了将 CSH 在纹理匹配方面的优势运用到图像特征匹配当中, 本文提出了一种将 SURF 特征与 CSH 匹配算法结合的一种新的图像匹配算法, 在 CSH 进行运行之前, 对图像进行 SURF 特征点的提取, 并以特征点为圆心构建特征区域, 再将这些特征区域作为块放入 CSH 中进行匹配, 最后输出匹配结果。

1 相关工作

1.1 Patch-Match

Patch-Match^[4]的核心是对一幅图像平面区域内的所有块进行重复搜索, 以此来得到另一幅图像平面中与之最相似的块。Patch-Match 其特点在于极大地发挥了图像连贯性的作用。Patch-Match 在一对图像匹配中多次运行, 对于用户输入的 2 幅图像: 图像 A 和图像 B, Patch-Match 随机地将图像 A 中的块分配给图像 B 中的块。在这些分配当中, 大部分的分配产生的匹配质量较低, 只有少部分分配会产生高质量的匹配。然后, Patch-Match 将好的匹配传递到图像平面内的周围块中。为了避免被困在局部最小值中, 它还为每个块执行许多随机的块分配, 在每个阶段之后保持最佳匹配。该算法通常在少量迭代后收敛。

1.2 用于最近邻搜索的局部敏感哈希

局部敏感哈希 (LSH) 的概念最早是 Indyk 和 Motwani^[5-6]提出。之后冯小康等又对其进行了改进^[7]。原始数据空间中的 2 个相邻点, 在经过 LSH 函数族的映射后, 这 2 个点在新的数据空间中相邻的概率很大。LSH 在高维数据中用于最近邻搜索的第一次运用是在高维二元 Hamming 空间的工作^[8]中。本文对于 LSH 算法的使用与 Datar 等提出的 LSH 最近邻搜索方法一致^[9]。

LSH 算法的核心是函数族 H 和最近邻搜索算法。其中最近邻搜索算法分为 2 个阶段: 索引和搜索。其中索引阶段, 利用函数族 H 中的哈希函数创建索引, 这使得相似的点会有很高的概率落入相同的桶中。 M 个这样的哈希函数被串联起来作为代码, 代码的作用在于放大相距较远的点之间碰撞概率和相距较近的点之间碰撞概率之差。这样的代码通过在所有数据集上进行评估来创建单个哈希表。在搜索阶段, 将搜索点哈希到表格桶中, 从中选择最近的驻留数据集点。为了降低落入空桶的概率, 使用多个 (L) 随机码来创建 L 个哈希表, 使其在搜索阶段被顺序搜索。

1.3 SURF 特征算法

SURF^[10]是一种鲁棒性较高的局部特征描述子, 且具有尺度、旋转、平移不变性, 使用 SURF 产生出来的图像特征向量性能稳定, 且对视角的变化、噪声、目标遮掩都能保持较好的稳定性, SURF 算法由 Bay H 在 2006 年提出^[11]。

2 一致性敏感哈希

本文将 CSH 算法布局为近似密集最近块的搜索算法, 先使用 LSH 将相似的块映射到相同的桶中, 再使用 Patch-Match 对同一桶中的块进行相似性的匹配。

2.1 索引

索引使用 Datar 等提出的 LSH 方法^[9], 使用形式为 $h_{a,b}(\mathbf{v}) = \frac{a\mathbf{v}+b}{r}$ 的 LSH 函数的函数族 h , 其中 b 是在 $[0,r]$ 内均匀分布的随机数, 其中 r 是预定义的整数, $\mathbf{v}$ 是一个 d 维度矢量。该函数族的这种随机函数对向量 $\mathbf{v}$ (即对块) 的作用可以通过 3 个阶段来描述:

- 1) 取一条由向量 $\mathbf{a}$ 定义的随机线, 将其等量划分为多个宽度为 r 的桶, 并为该式子的分子添加一个大小为 $b \in [0,r]$ 的随机偏移量;
- 2) 将向量 $\mathbf{v}$ 投影到线上;
- 3) 为其分配一个哈希值, 作为其落入的桶的索引。换句话说, b 的作用是保证相似的块能以高概率落入同一个桶中。

2.2 搜索

在上一阶段, 本文算法构建了一组哈希表 L (其中子表命名为 $T_i, i=1,2,\dots,L$)。如果对于图像 A 中的每个块直接使用 LSH 搜索方案, 直接简单地将 B 中与之相似的块哈希到相同的桶中。这样不但没有利用块的已知空间顺

序,还会使得各个块被阻隔开(块之间的联系被忽视)。本文算法通过以新颖的方式结合图像的外观和将图像的连贯性作为线索来创建多个候选块来解决上述问题。

2.2.1 候选块的创建

候选块的创建方式归结为3种,如图1所示。

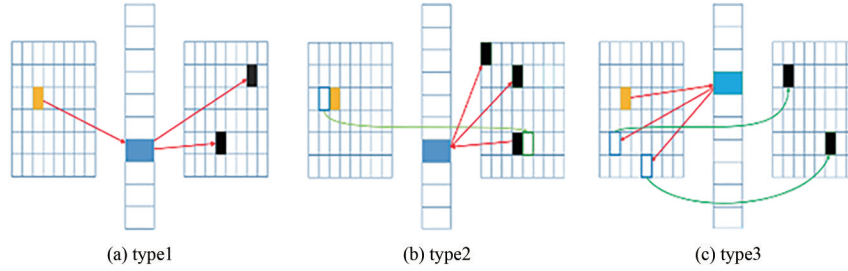


Fig.1 Candidate types for a patch
图1 候选块类型

令 g_i 表示用于创建哈希表 T_i 的哈希函数。对图像 A 的块进行筛选时,哈希函数 g 将被表示为 g_A ; 同理 g_B 表示图像 B 中块的哈希函数。

$\text{Left}(p), \text{Right}(p), \text{Top}(p), \text{Bottom}(p)$ 分别表示块 p 的左、右、上、下的邻接块。 a, a_1, a_2 是图像 A 中的块, b, b_1, b_2 是图像 B 中的块, 对于选择候选匹配块的方法分别对应为图 1 中的 3 种类型。

- 1) 如果 $g_A(a) = g_B(b)$, 那么 b 为 a 的候选块; 如果 b_1 是 a 的候选块, 且 $g_B(b_1) = g_B(b_2)$, 那么 b_2 就是 a 的候选块;
- 2) 如果 b 是 $\text{Left}(a)$ 的候选块, 那么 $\text{Right}(b)$ 是 a 的候选块(这里对 Left/Right , Bottom/Top , Top/Bottom 同样适用);
- 3) 如果 b 是 a_1 的候选块且有 $g_A(a_1) = g_B(a_2)$, 那么 b 是 a_2 的候选块;

上述方法对应式(1):

$$\begin{cases} \text{Cand}(a)=b, & \text{if } g_A(a)=g_B(b) \\ \text{Cand}(a)=b_2, & \text{if } \text{Cand}(a)=b_1 \text{ and } g_B(b_1)=g_B(b_2) \\ \text{Cand}(a)=\text{Right}(b), & \text{if } \text{Cand}(\text{Left}(a))=b \\ \text{Cand}(a_2)=b, & \text{if } \text{Cand}(a_1)=b \text{ and } g_A(a_1)=g_A(a_2) \end{cases} \quad (1)$$

2.2.2 候选块的近似排序

在确定了一个块的匹配候选块集后,需要找寻与之相距最近的一个候选块,而此过程所需的时间在整个算法中最长,为了加快算法效率,本文算法在此阶段当中采用了近似的方法,这样做对整体精确度的影响可忽略不计,并且将大大减少算法在此阶段的耗时,这一步中采取了和 Hel-Or 等一样的方法来使用沃尔什-阿达玛内核,并在图案匹配的筛选方案中使用了 Hel-Or Y 等的算法^[12]。

2.3 CSH 算法具体步骤

第1步:建立索引(图像 A 和图像 B 中所有块的索引)

1) 计算 M 沃尔什-阿达玛内核中 A 和 B 中每个块的映射: $\{WH_j\}_{j=1}^M$ 此处采用了文献[13]中的格雷码内核(GCK)技术。

2) 创建一组哈希表 L 。 $\{T_i\}_{i=1}^L$ 其中 T_i 的组成如下:

a) 定义代码: $g_i(p) = h_1(p), h_2(p), \dots, h_M(p)$ 是 M 维形式为 $\{h_j\}_{j=1}^M$ 的向量的串联,具体展开:

$$h_j(p) = \frac{WH_j p + b_j}{r} \quad (2)$$

式中: $\{WH_j\}_{j=1}^M$ 为用格雷码内核技术提取沃尔什-阿达玛内核生成的 M 维度投影向量; p 为特征向量; b_j 为随机偏移量,范围在 $[0, r)$, r 为预设阈值。

b) 然后将图像 A 和图像 B 中所有的块储存在 $T_i[g_i(p)]$ 中。

第2步:搜索

- 1) 任意初始化最佳候选映射的近似最近邻域(ANNF);
- 2) 对所有哈希表执行以下操作:
对图像 A 中所有的块 a :
1) 使用表 T_i 和当前映射的 ANNF 创建一组最近候选块集 P_B (候选块的创建方法如 2.2.1 所示);
2) 如果 b 是 P_B 中与 a 最相似的块, 且满足 $\text{dist}(a, b) < \text{dist}(a, \text{ANNF}(a))$ 时, 则更新 $\text{ANNF}(a) = b$ 并返回 ANNF 的值。

3 SURF 算法的修改以及匹配区域的选定

3.1 特征区域选定

以特征点为圆心自适应地构造出半径为 r 的圆形特征区域。具体步骤:

- 1) 先对图像进行 SURF 特征点的定位, 并记录每个点的尺度值;
- 2) 将每个特征点的尺度值作为各点构造圆形区域的半径;
- 3) 将每个圆形区域四周使用“补零”的方法映射成外接正方形。

此处本文为何会选用 SURF 特征点, 主要原因在于其提取速度较快^[14], 且对图像边缘部分的描述较好。比起尺度不变特征转换(Scale Invariant Feature Transform, SIFT)^[15]特征算法, SURF 算法在保留了 SIFT 算子优良性能的基础上, 在提取速度上明显要快于 SIFT, 因此选择了前者。

3.2 SURF 算法的修改

SURF 算法在提取特征点时, 对于特征点方向的计算会花费大量的时间, 而 CSH 是基于像素块的立体匹配算法, 仅需要图像中特征点的位置作为图像块选取的引导, 所以本文算法在提取特征点时只计算特征点的位置, 省略对于特征点方向的计算, 修改后的特征点提取步骤:

- 1) 使用高斯核函数计算图像亮度分量中各点的概率密度 $P(X)$, 具体步骤:

$$P(X) = \sum_w \frac{C}{h_s^2 h_r} k_1 \left(\left\| \frac{X - X_j}{h_s} \right\| \right) k_2 \left(\left\| \frac{c - c_j}{h_r} \right\| \right) \quad (3)$$

式中: W 为以点为中心的概率密度计算窗口; X_j 为一个窗口中心为 X 的 W 的其中的一点; c_j 为其亮度; C 为一个常数; h_s 为图像空间的带宽; h_r 为亮度空间的带宽, 其中 k_1 和 k_2 表示为:

$$k_1 = g(\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, k_2 = g_t(\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-t^2/2\sigma^2} \quad (4)$$

- 2) 计算每个点 $X(x, y)$ 的概率密度二阶导数, 计算方法如下:

$$\begin{cases} D_{xx}(X, \sigma) = \frac{d^2 D(X)}{dx^2} = \sum_w \frac{C}{h_s^2 h_r} \frac{d^2}{dx^2} \left(k_1 \left(\left\| \frac{X - X_j}{h_s} \right\| \right) k_2 \left(\left\| \frac{c - c_j}{h_r} \right\| \right) \right) \\ D_{yy}(X, \sigma) = \frac{d^2 D(X)}{dy^2} = \sum_w \frac{C}{h_s^2 h_r} \frac{d^2}{dy^2} \left(k_1 \left(\left\| \frac{X - X_j}{h_s} \right\| \right) k_2 \left(\left\| \frac{c - c_j}{h_r} \right\| \right) \right) \\ D_{xy}(X, \sigma) = \frac{d^2 D(X)}{dx \cdot dy} = \sum_w \frac{C}{h_s^2 h_r} \frac{d^2}{dx \cdot dy} \left(k_1 \left(\left\| \frac{X - X_j}{h_s} \right\| \right) k_2 \left(\left\| \frac{c - c_j}{h_r} \right\| \right) \right) \end{cases} \quad (5)$$

- 3) 构造 Hessian 矩阵^[16]

这一步是 SURF 算法的核心部分, 通过对 Hessian 矩阵行列式的计算来得出特征点。以图中点 x 为例已知尺度为 σ , 则 Hessian 矩阵 $H(x, \sigma)$ 定义如下:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (6)$$

式中 $L_{xx}(x, \sigma)$ 是高斯二阶偏导数 $\frac{\partial^2}{\partial x^2} g(\sigma)$ 与图像在点 x 处的卷积。 $L_{yy}(x, \sigma)$ 和 $L_{xy}(x, \sigma)$ 与之同理。 $H(x, \sigma)$ 的判别式为:

$$\det(H) = \frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 \quad (7)$$

4) 构建尺度空间

SURF 的初始滤波器大小为 9×9 ，每层滤波器的大小遵循式(8)来调整：

$$L = 3 \times (2^{octave} \times layer + 1) \tag{8}$$

式中： $octave$ 为分析组数； $layer$ 为每组分析的层数。

5) 使用滤波器进行特征点定位。算法流程如图 2 所示。

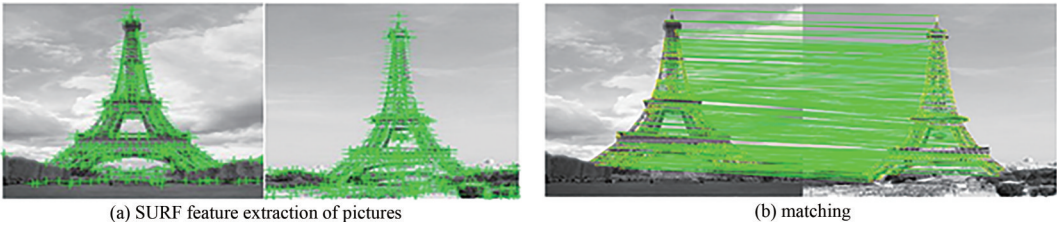


Fig.2 Algorithm flow
图 2 算法流程

4 实验

本文实验环境：处理器为 Intel Core i5-4460 CPU 3.20 GHz，内存 8 GB，Windows 10 64 位操作系统，程序编写语言为 Matlab。在 CSH 对全局特征区域进行匹配时，将算法的迭代次数设置为 8 次， $k=2$ (k 代表每个块返回的最近匹配个数)，最后输出 2 幅图的匹配映射图。实验部分选取了纹理复杂度较高的建筑物进行匹配实验，并引入 SIFT^[17]、OFRB(Oriented Fast and Rotated Brief)^[18]、BRISK^[19] 分别对图像进行匹配，作为实验的参照组，实验结果如图 3 和图 4 所示。

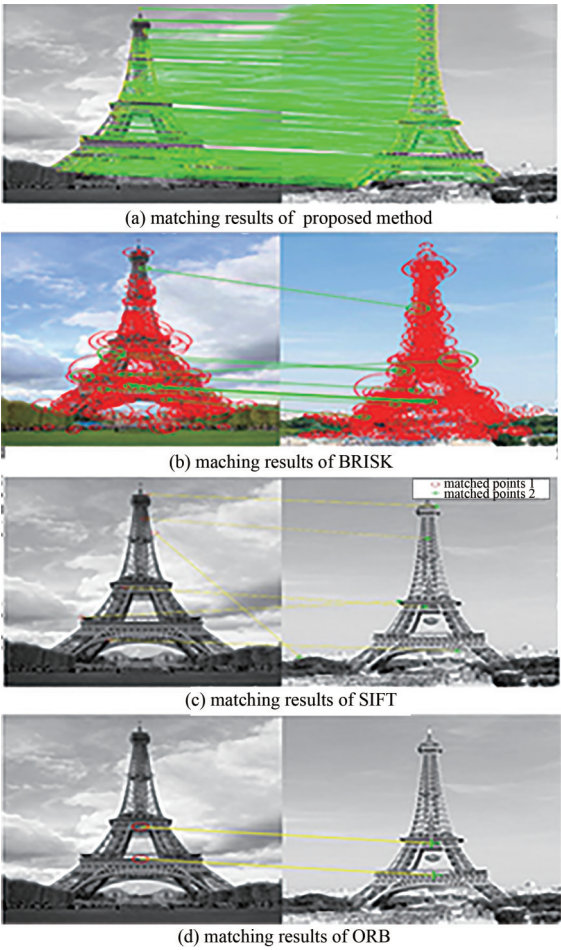


Fig.3 Experiment A
图 3 实验 A

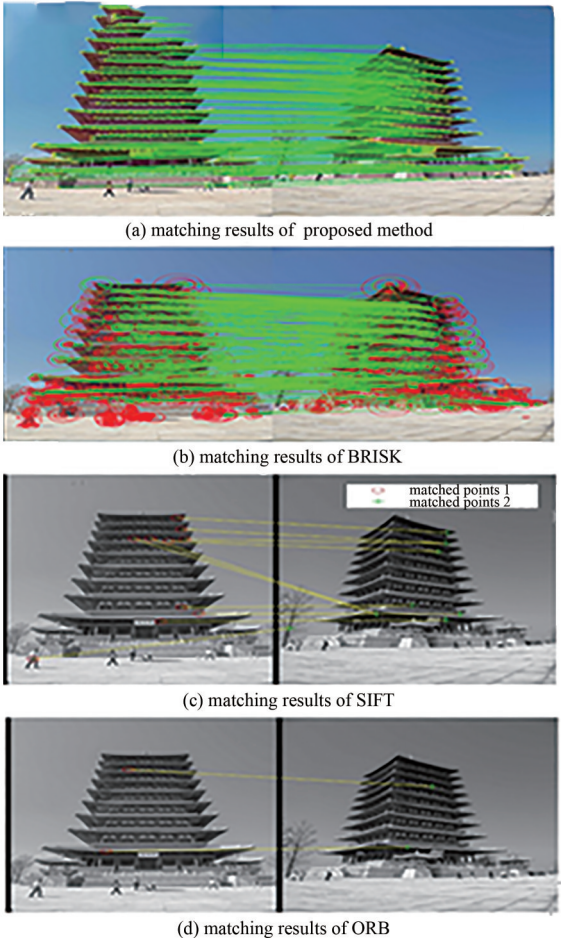


Fig.4 Experiment B
图 4 实验 B

通过实验 A、B 可以看出对于具有复杂纹理的建筑物图像进行匹配时，SIFT 和定向快速旋转(Oriented FAST and Rotated BRIEF, ORB)匹配算法匹配成功的点数稀少，匹配效果不佳，BRISK 算法虽在实验 B 中匹配数量有所提升，但匹配点的数量并未超过本文的算法，并且本文算法较其具有更高的稳定性，这归功于 CSH 算法在纹理匹配方面的优势。表 1 中的数据证明了本文算法在提供良好匹配效果的基础上，保持了与特征点匹配算法一样的实时性。在低分辨率的图像匹配中由于图像较小，本文的方法过滤掉的非特征区域有限，所以在运行时间上与 SIFT,ORB 和 BRISK 相差无几，但随着图像分辨率的提高，本文方法可以过滤掉的非特征区域不断增加，大大减少了需要进行匹配运算的区域，缩短了算法的运行时间，使得本文的算法较其他 3 种算法具有更高的实时性。

表 1 算法运行时间
Table1 Running time of algorithms

image and its size	running time of algorithms/s			
	SIFT	ORB	BRISK	proposed method
Giraffe(400×300)	2.488	2.334	2.441	2.496
Tree(512×340)	1.254	1.129	1.243	1.131
Cattle(800×600)	7.164	6.211	6.488	6.155
House(1 024×768)	10.865	8.979	9.931	8.954
Beach(1 152×864)	11.667	10.389	11.556	9.811
Eiffel(1 280×720)	16.372	13.911	15.512	13.336

使用未改进的 CSH 算法和本文算法同时对 4 组图像进行匹配，并记录各自的匹配总数、有效匹配数(图像中主要匹配目标内的块匹配数量)、有效匹配率和算法运行时间，具体如表 2、表 3 所示。

表 2 CSH 实验表现
Table2 CSH performance

image and its size	total matches	number of target area matches	effective matching rate/%	running time/s
Cattle(800×600)	89	69	77.53	6.433
House(1 024×768)	120	87	72.50	9.472
Beach(1 152×864)	78	69	88.46	10.973
Eiffel(1 280×720)	132	75	56.81	15.574

表 3 SURF+CSH 实验表现
Table3 SURF+CSH performance

image and its size	total matches	number of target area matches	effective matching rate/(%)	running time/s
Cattle(800×600)	72	68	94.44	6.155 0
House(1 024×768)	88	81	92.04	8.954 0
Beach(1 152×864)	69	64	92.75	9.811 0
Eiffel(1 280×720)	85	72	84.71	13.336 6

本文算法在保证较高有效匹配数的基础上，大大减少了无关匹配的数量，从而保证了较高的有效匹配率和更高的运行效率。这主要是因为 SURF 有着对于图像强纹理区域的敏锐嗅觉，帮助 CSH 过滤掉了大部分背景区域，从而减少了许多不必要的匹配，提升实现算法效率，并且这个优势会随着图像分辨率的提升进一步扩大。

为了测试本文算法，对局部特征明显的小物件的匹配效果进行实验，如图 5 所示。

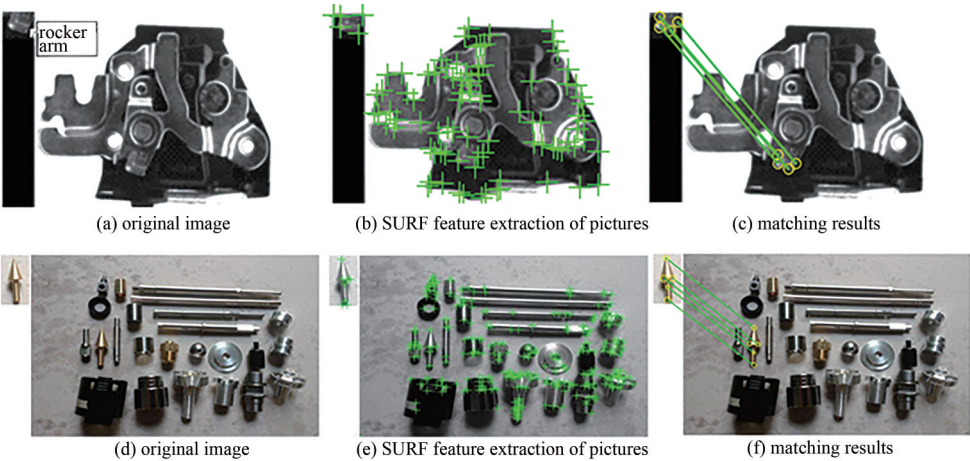


Fig.5 Industrial parts matching
图 5 工业零件匹配

- a) 实现对汽车门锁中摇臂的匹配；
- b) 实现零件的匹配。

以上对于工业零件的匹配实验证明，本文算法对于局部特征较为明显的小物件也有较好的匹配效果，这主要得益于SURF对于物体特征的精确定位，从而帮助CSH算法能够对关键的图像块进行匹配，以此完成较好的图像匹配。

5 结论

本文通过结合CSH和SURF特征算法构建了一种基于图像块的图像匹配算法，有效地将CSH算法在纹理匹配中的优势融入到图像特征匹配当中，实现了对于复杂纹理场景的高精度匹配。在提升了原有CSH算法的运行速率同时，实现了较高的有效匹配率。这也给基于CSH算法实现的图像修复和图像篡改检测等算法带来效率的提升，且本文算法较现有的一些特征点匹配算法在复杂纹理图像匹配中的匹配精确度更高，且在对高清图像的匹配当中速度较快。这使得本文算法可以更好地应用到工业零件装配检测、工业材质缺陷检测等方面。

本文算法还可以通过结合类似文献[20]的基于近似最近邻的图像去噪算法实现图像去噪后的匹配，从而进一步提高匹配精确度，这也是之后本文算法拓展研究的重点。

参考文献：

- [1] MORSE B, HOWARD J, COHEN S, et al. Patchmatch-based content completion of stereo image pairs[C]// Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission. Zurich, Switzerland: IEEE Computer Society, 2012:555-562.
- [2] DATAR M, IMMORLICA N, INDYK P, et al. Locality sensitive Hashing scheme based on p-stable distributions[C]// SoCG '04. New York, USA: ACM, 2004:9-11.
- [3] KORMAN S, AVIDAN S. Coherency sensitive Hashing[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016, 38(6):1099-1112.
- [4] BARNES C, SHECHTMAN E, FINKELSTEIN A. Patchmatch: a randomized correspondence algorithm for structural image editing[J]. In SIGGRAPH, ACM Transmission Graphics, 2009, 28(3):24.
- [5] INDYK P, MOTWANI R. Approximate nearest neighbors: towards removing the curse of dimensionality[J]. In Symposium on Theory of Computing, 1998(1):604-613.
- [6] PAULEVÉ Loïc, JÉGOU Hervé, AMSALEG L. Locality sensitive Hashing: a comparison of Hash function types and querying mechanisms[J]. Pattern Recognition Letters, 2010, 31(11):1348-1358.
- [7] 冯小康, 彭延国, 崔江涛, 等. 基于最优排序的局部敏感哈希索引[J]. 计算机学报, 2019, 43(5):930-947. (FENG Xiaokang, PENG Yanguo, CUI Jiangtao, et al. Locality sensitive Hashing index based on optimal linear order[J]. Chinese Journal of Computers, 2019, 43(5):930-947.)
- [8] GIONIS A, INDYK P, MOTWANI R. Similarity search in high dimensions via Hashing[C]// In International Conference on Very Large Data Bases. Edinburgh, Scotland: Proceedings of the 25th VLDB Conference, 1999:518-529.
- [9] DATAR M, IMMORLICA N, INDYK P, et al. Locality sensitive Hashing scheme based on p-stable distributions[C]// In Proceeding of Annual Symposium on Computational Geometry. New York, USA: ACM, 2004:253-262.
- [10] 胡晓彤, 任辉, 刘楠. 尺度与特征强度自适应的SURF特征点匹配算法[J]. 天津科技大学学报, 2019, 34(2):70-74, 80. (HU Xiaotong, REN Hui, LIU Nan. Adaptive SURF feature points matching algorithm based on scale and feature intensity[J]. Journal of Tianjin University of Science & Technology, 2019, 34(2):70-74, 80.)
- [11] BAY H, TUYTELAARS T, GOOL L V. SURF: speeded up robust features[J]. Computer Vision and Image Understanding, 2008(110):346-359.
- [12] HEL OR Y, HEL OR H. Real-time pattern matching using projection kernels[J]. IEEE Trans Pattern Anal Mach Intell, 2005, 27(9):1430-1445.
- [13] BEN-ARTZI G, HEL-OR H, HEL-OR Y. The gray-code filter kernels[J]. IEEE Trans Pattern Anal Mach Intell, 2007, 29(3):382-393.
- [14] 林志东, 杜滢钊. 特定条件下SIFT与SURF图像匹配算法的性能对比[J]. 厦门理工学院学报, 2019, 27(3):30-36. (LIN Zhidong, DU Yingzhao. Comparison of SIFT and SURF as image matching algorithms under certain conditions[J]. Journal of Xiamen University of Technology, 2019, 27(3):30-36.)
- [15] LOWE D G. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60(2):91-110.

(下转第512页)