基于积分图的 LATCH 算法改进①

陈方飞[1], 冯 瑞[2]

[1](复旦大学 计算机科学技术学院, 上海 201210)
[2](上海视频技术与系统工程研究中心, 上海 201210)

摘 要: LATCH(基于机器学习的三像素块描述子)在原有局部二值描述子的基础上通过将二像素块比较改为三像素块提升了局部二值描述子的准确率, 但提高准确率的同时带来了更大的时耗, 在研究 LATCH 以及其他二值描述子的基础上, 借鉴积分图在目标检测中的应用, 将积分图的思想应用在改进 LATCH 描述子中, 减少了 LATCH 描述子中各像素块内的重复计算量. 实验证明, 改进算法的描述子计算时间较原算法缩减了 30%–40%, 而配准精度与原算法保持相近.

关键词: 局部二值描述子; LATCH; 像素块; 积分图

Improved LATCH Based on Integral Image

CHEN Fang-Fei[1], FENG Rui[2]

[1](School of Computer Science, Fudan University, Shanghai 201210, China)
[2](Shanghai Engineering Research Center for Video Technology and System, Shanghai 201210, China)

Abstract: LATCH (Learned Arrangements of Three Patch Codes) improves the accuracy of local binary descriptor by comparing three pixel blocks rather than tow pixel blocks. However, the improvement of accuracy brings a larger time consuming. Based on the study of LATCH and other local binary descriptors, using integral graph theory in improved LATCH descriptor, it reduces repeated calculation of each pixel blocks in LATCH descriptors. According to the experiment results, the computation time of the improved algorithm is reduced by 30%–40% compared with the original algorithms, while the accuracy of the improved algorithm is similar to that of the original algorithm.

Key words: local binary descriptor; LATCH; patch; integral image

图像局部视觉特征信息的有效表示在许多计算机视觉应用中都是至关重要的, 如图像索引, 需要从查询库中进行大量局部特征提取与计算而定位与查询项匹配度最高的图像. 因此计算机视觉领域研究中有很大部分关注于如何确定局部信息描述子或如何提升已有描述子的精度与效率. 经过多年的研究与发展, 图像局部信息描述子形成了两大主流研究方向: 基于分布的描述子与二值描述子.

基于分布的描述子主要描述特征表征量(如梯度, 梯度方向等)的图像分布信息, 其中突出的描述子有 HOG[1], SIFT[2]描述子, SIFT 描述子通过计算特征局部区域梯度方向分布表征特征点局部视觉信息. 基于分布的描述子在特征描述的准确度有显著效果, 但随着精度不断增加, 计算时耗与内存开销也在不断增加.

近年来, 由于计算机视觉应用中图像数据库容量急剧上升, 图像分辨率增大. 对特征描述子的计算时耗需求触动了二值描述子的逐渐发展. 二值特征描述子比较像素块之间的值得到一个二进制串, 二进制串可以直接通过计算哈明码距离进行匹配, 极大提高了运算速度. 其中突出的描述子有 BRIEF[3], ORB[4]以及 FREAK[5].

传统二值描述子提高速度的同时带来了匹配准确

① 基金项目:国家科技支撑计划(2013BAH09F01);上海市科委科技创新行动计划(14511106900);基于 BIM+GIS 城市大数据计算平台的智慧临港应用示范(ZN2016020103)

收稿时间:2016-08-08;收到修改稿时间:2016-09-18 [doi:10.15888/j.cnki.csa.005716]

度的下降, 2015 年 Gil Levi 与 Tal Hassner 提出了 LATCH[6]算法, 并且以加入 OpenCV 函数库中, 在传统二值描述子的基础上, 针对匹配准确度, 从以下两方面进行了改进:

① 三像素块比较代替两像素比较

② 像素块位置通过机器学习选出最优位置组合

LATCH 算法在准确度上提升了, 但三像素块带来更大的计算量. 通过深层研究发现, LATCH 中主要时间消耗在每次对每像素块进行遍历计算中.

参照文献[7]中, 通过积分图消除重复运算的方法, 本文对 LATCH 算法进行如下改进

① 修改像素块比较公式, 适应积分图计算

② 应用积分图计算局部特征描述子

实验证明, 改进算法的描述子计算时间较原算法缩减了 30%～40%, 而配准精度与原算法保持相近.

1 LATCH算法

LATCH 前二值特征描述子比较以特征点 K 为中心检测窗口 W 内特定像素对比较的结果. 设检测窗口 W 内 T 对有序样本坐标集如式(1)所示:

$$S=\{s_t\}_{t=1...T}=\{[p_{t,1},p_{t,2}]\}_{t=1...T}$$
$$p_{t,1}=(x_{t,1},y_{t,1}) \quad p_{t,1}=(x_{t,1},y_{t,1}) \tag{1}$$

通过如下公式(2)可以求出每个像素对比较的二值结果:

$$f(W,S_t)=\begin{cases}0 & \text{if } W(p_{t,1})>W(p_{t,2})\\ 1 & \text{otherwise}\end{cases} \tag{2}$$

比较窗口内所有像素对, 形成该特征点处二进制串, 可以描述该关键点特征. 由于是单像素间的比较, 容易受噪声影响. LATCH 在此基础上提出了三像素块之间的比较计算得到比特串, 增强抗噪能力.

1.1 三像素块比较

单像素间的比较, 容易受噪声影响, 部分研究通高斯平滑消除噪声影响, 但消除噪声也将造成关键点处信息丢失. Gil Levi 与 Tal Hassner 提出通过三像素块的比较解决噪声的影响并提升匹配精度.

在每个检测窗口 W 内定义了 T 组三像素块, 窗口内像素块集如式(3):

$$S=\{s_t\}_{t=1...T}=\{[p_{t,a},p_{t,1},p_{t,2}]\}_{t=1...T} \tag{3}$$

每个像素块定义为 像素块, 上述表达式中每个像素块坐标 $k\times k$ 为像素块中心点坐标. 通过式 4 评估锚点像素块$p_{t,a},p_{t,1},p_{t,2}$为像素块中心点坐标. 通过式 4 评估锚点像素块$p_{t,a}$与两个对比像素块$p_{t,1}$与$p_{t,1}$之间的相似度作为一个该特征点的一个比位.

$$f(W,S_t)=\begin{cases}0 & \text{if } ||P_{t,a}-P_{t,1}||_F^2>||P_{t,a}-P_{t,1}||_F^2\\ 1 & \text{otherwise}\end{cases} \tag{4}$$

1.2 基于学习的像素块组合

每个检测窗口中三像素块位置组合数规模庞大, 一个 49×49 检测窗口, 假定像素块大小为 7×7, 三像素块位置组合数规模在亿的数量级. 而特征描述计算过程中用到的三像素块组合是有限的, 如何在规模庞大的数据组合中选出有限的组合, 使匹配效果达到最优是一个重要问题. Gil Levi 与 Tal Hassner 提出了基于机器学习确定较优组合的方法.

1.2.1 学习数据集

学习数据集是建立在文献[8]所提供据集, 数据集包括: Liberty, Notre Dame 以及 Yosemite 三个独立数据集, 每个数据集包括约 20 万个 局部图像窗体, 窗体是从不同角度拍摄, 并在文本文件中标注各个局部图像窗体是否相同(整个图像是对场景的 3D 拍摄场景, 通过 Harris 提取算法提取出不同场景点的局部窗口, 同一场景点的不同拍摄角度具有相同索引号, 即为相同). LATCH 中使用了其中 20 万个局部窗体块, 其中一半窗体为相同的, 另一半为不同.

1.2.2 学习算法

LATCH 学习算法的基本思想是通过对大量组合进行匹配结果比较, 筛选出较优组合. 首先通过随机算法产生 5 万 6 千个三像素块组合. 以每个像素块组合匹配所有局部窗体结果, 每个组合方案根据其匹配结果与标注结果比较可以产生一个 大小的比特串(其中 1 表示匹配结果与标注结果一致, 0 为不一致). 评估每个组合方案好坏可以通过计算每个比特串的和, 即公式(5):

$$b_i=\begin{cases}0 & \text{the same result with bench} \quad (i=1...n)\\ 1 & \text{otherwise}\end{cases}$$
$$Sum=\sum_{i=1}^{n}b_i \qquad n=500000 \tag{5}$$

和值越高, 效果越好.

但单纯这种选择选择结过可能导致相关性强的几个组合方案同时被选择, 为避免该情况需要对每个待选择组合方案与所有已选中组合方案进行相关性计算, 只有相关性小于一定阈值才可被选中, 相关性的计算

可以通过计算各比特串间汉明码确定.

2 LATCH算法改进

LATCH 以像素块取代像素点的计算, 提升了准确度, 而由于需要对每个像素块遍历计算, 增加了运算量. 本文在研究 LATCH 算法与积分图算法的基础上, 在保持整体不变的情况下, 通过改进 LATCH 算法计算公式以适应积分图运算. 减少了计算量, 同时也保证了准确度.

2.1 积分图算法

积分图最早是由 Paul Viola[9]等人提出的, 并被应用到实时的目标检测框架中. 是能够计算矩形块特征的快速有效方法. 积分图中每个像素位置(*x*,*y*)的值不同于简单灰度, 或彩色图存储的是像素值, 而是从原点到(*x*,*y*)包含区域所有像素点的和, 其表达式如公式(6):

$$ii(x,y)=\sum_{x^*\le x,y^*\le y} i(x^*,y^*) \tag{6}$$

其中$ii(x,y)$为积分图像值, $i(x^*,y^*)$为原图像值.

任意矩形区域 像素和如图 1 所示.

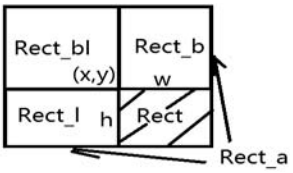

图 1 积分原理图

传统计算$Rect$区域内像素和$\sum_{Rect}$通过遍历区域求和, 运算效率低下. 而在积分图的基础上可以快速有效计算, $Rect$区域内与相邻区域像素和存在如公式(7)所示关系:

$$\sum_{Rect}=\sum_{Rect\_a}+\sum_{Rect\_bl}-\sum_{Rect\_b}-\sum_{Rect\_l} \tag{7}$$

所以, 利用积分图计算矩形区域特征如公式(8)所示:

$$\begin{aligned}\sum_{Rect}&=ii(x+w,y+h)+ii(x,y)\\&\quad-ii(x+w,y)-ii(x,y+h)\end{aligned} \tag{8}$$

根据公式, 如下代码代码段 1 是积分图计算伪码:

Generate Integral Image

```
1 : procedure GenerateIntegralImage(img, out)
2 :   for all row in img.rows do
3 :     for all col in img.cols do
4 :       out[row][col] =
5 :       out[row − 1][col] + out[row][col − 1]
6 :        − out[row − 1][col − 1] + out[row][col]
7 :         end for
8 :   end for
9 : end procedure
```

2.2 基于积分图的 LATCH 算法改进

由 3.1 可知, 积分图在处理矩形区域和特征计算能够非常有效, 而 LATCH 算法中超过 95%计算量在每像素块之间的计算中. 如果能将积分图算法引用到 LATCH 中像素块计算中, 可以提高 LATCH 算法计算效率.

由式(4)LATCH 像素块比较计算式可知, 像素块计算式对像素块对中每个对应像素进行求差的绝对值再求和. 因此不利于积分图计算的开展. 而如果只求差, 不求绝对值, 再求和, 如式(9):

$$f(W,S_t)=\begin{cases}0 & \text{if } \sum(P_{t,a}-P_{t,1})>\sum(P_{t,a}-P_{t,2})\\1 & \text{otherwise}\end{cases} \tag{9}$$

对上式进行变换, 可得到公式(10):

$$f(W,S_t)=\begin{cases}0 & \text{if } \sum P_{t,a}-\sum P_{t,1}>\sum P_{t,a}-\sum P_{t,2}\\1 & \text{otherwise}\end{cases} \tag{10}$$

由上式可知以及积分图算法原理可知, 该计算式可以通过积分图算法提升计算效率. 如下代码段 2 是改进 LATCH 伪代码.

New LATCH

```
1 : procedure NewLatchDescriptor(img, keypoints)
2 :   GenerateIntegralImage(img, out)
3 :   for all keypoint in keypoints do
4 :     for i = 0 → patches_size − 1 do
5 :       suma = out[patch[i]_a]
6 :       sumb = out[patch[i]_b]
7 :       sumc = out[patch[i]_c]
8 :       desc[i] = suma − sumb
9 :                 > suma − sumb?0 : 1
8 :   end for
9 : end procedure
```

2.3 区域分布特异化

由式(9)可知, 改进的 LATCH 算法中像素块比较公式, 所体现的是整个像素块特征, 不能在像素块区域内体现区域分布特征. 而特征描述中, 区域分布同样是表示特征的非常重要信息. 因此为了在像素块内同时体现区域信息, 对像素块内不同像素带赋予不同权值. 其像素带分布如图 2 所示, 根据不同像素带, 确定不同权值如公式(11)所示:

$$w_k=1-\frac{K}{R} \tag{11}$$

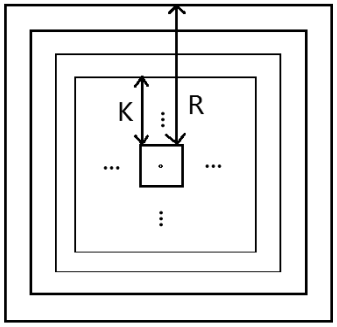

图 2　像素带分布

3　实验结果

本文实验平台为 Windows 7, 处理器型号是 Intel Core i5-3470,内存大小为 4GB, 程序采用 C++语言编程, 通过调用 OpenCV 3.0 对图像进行基本操作. 实验数据集如 2.2.1 描述数据集, 该数据集共有约 60 万个图像窗口, 使用其中 20 万个图像窗口进行学习训练得到三像素块位置, 剩余 40 万作为验证实验数据集.

3.1 实验评估指标

实验采用文献[10]中局部描述子方法对实验结果进行评估, 即基于正匹配率对负匹配率的 ROC 值得评估方法. 其中:

正匹配率(True Positive Rate): 指正确匹配的样本数量与可能匹配的样本比值,如式(12):

$$p_{correct} = \frac{\#\ correct\ matches}{\#\ possible\ matches} \tag{12}$$

负匹配率(False Positive Rate): 指在描述子数据集中被错误地匹配的概率, 如式(13):

$$p_{false} = \frac{\#\ false\ matches}{(\#\ database\ points)(\#\ query\ image\ points)} \tag{13}$$

%95 错误率(%95 Error Rate): 正匹配率为 95%时的负匹配率,如式(14)所示:

$$p_{95\%err} = p_{false},\quad p_{correct} = 95\% \tag{14}$$

3.2 参数分析

本文所述描述子主要受两个参数影响:每个特征点描述子字节长度, 以及像素块大小, 下面将对两个影响因子分别分析.

1) 描述子长度分析

描述子长度是指每个特征形成的描述特征信息的二进制位数, 本文分别对长度为 8 字节、16 子节、32 字节以及 64 字节进行试验, 图 3 是各个字节长度 ROC 曲线. 由图 3 可知, 随着字节长度增大, 效果越好, 64 字节长度虽然比 32 字节整体效果好, 但相差没有特别明显, 而且 64 字节长度将大大提升描述子计算及匹配时间, 因此本文采用 32 字节作为推荐描述子长度.

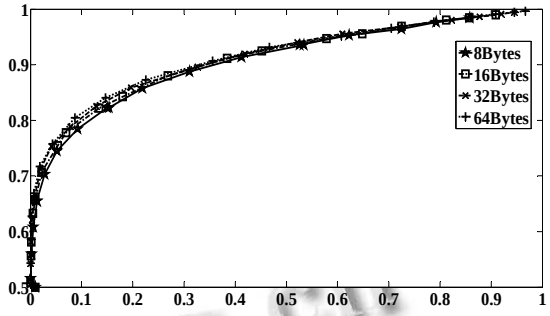

图 3　描述子长度对比结果

2) 像素块大小分析

影响描述子表示特征信息能力的另一个因子是像素块大小. 本文分别以 5×5、7×7、9×9、11×11 像素块大小进行比较试验, 图 4 是各个尺寸 ROC 曲线, 由图可知, 随着尺寸增大, 描述子表示特征信息效果越好, 但当增大一定尺寸后, 效果开始变差, 这是由于像素块尺寸过大, 将导致描述子各位间区分度下降. 从而使描述子整体准确度降低.

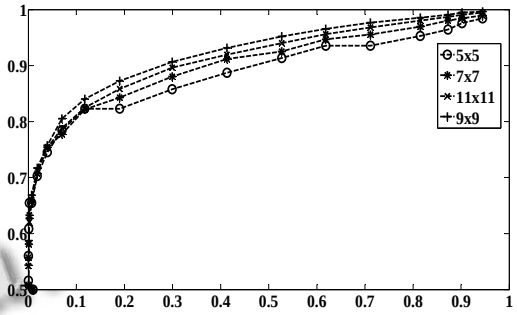

图 4　像素块尺寸对比结果

3.3 实验结果对比

本文主要针对 LATCH 算法在运行效率上的不足进行改进, 因此本文主要与 LATCH 算法实验结果进行比较. 定量比较改进算法与原算法在描述在计算效率以及表示特征信息准确度两方面效果.

(1) 运行效率

运行效率的比较首先通过 ORB 算法对 720P 图像特征点计算, 再计算原算法与改进算法对整副图像描述子计算耗时, 本文对 10 幅图像进行试验, 最终计算对每个特征点描述子计算平均耗时, 结果如表 1 所示, 由表可知, 改进算法相对原算法计算耗时极大减少.

表 1 运行效率对比

	特征点数(个)	平均耗时(ms)
改进算法	87659	0.478
原算法	87659	0.303

2) 准确度

为定量分析改进算法与原算法间的准确度差别, 本文在原算法与改进算法 ROC 曲线基础上, 分别计算 AUC 值, ACC 值以及 95Err 值进行比较. 下表 2, 是计算中对比结果, 由表可知, 改进算法在准确度上与原算法几乎一致.

表 2 准确度对比

	AUC	ACC	95%ERR
改进算法	0.92414	0.8584	59.2
原算法	0.91492	0.8496	59

4 结语

本文在研究 LATCH 算法的基础上, 提出了通过改进比较公式, 引用积分图算法, 减少算法在像素块内的计算时间, 实验表明本文在几乎不影响精确度的情况下, 极大地减少了运算时间, 提高运行效率. 本文主要有两个贡献点:(1)系统介绍了 LATCH 算法原理 (2)提出了基于积分图思想的 LATCH 算法改进, 改进了 LATCH 算法中像素块间比较公式以适应与 LATCH 算法, 并对改进算法与原算法进行定量分析.

参考文献

1 Dalal N, Triggs B. Histograms of oriented gradients for human detection. CVPR. San Diego, CA, USA. 2005, 1. 886–893.

2 Lowe DG. Distinctive image features from scale-invariant keypoints. IJCV, 2004, 60(2): 91–110.

3 Leutenegger S, Chli M, Siegwart RY. Brisk: Binary robust invariant scalable keypoints. ICCV, Barcelona. 2011. 2548–2555.

4 Yang X, Cheng KT. Ldb: An ultra-fast feature for scalable augmented reality on mobile devices. In Mixed and Augmented Reality (ISMAR). 2012. 49–57.

5 Alahi A, Ortiz R, Vandergheynst P. Freak: Fast retina keypoint. CVPR. Providence, USA. 2012. 510–517.

6 Levi G, Hassner T. LATCH: Learned arrangements of three patch codes. Winter Conference on Applications of Computer Vision (WACV). Lack Placid. 2016. 1–9.

7 黄文杰,陈斌.一种快速图像处理的积分图方法.计算机应用,2005,25(S1):266–269.

8 Brown M, Hua G, Winder S. Discriminative learning of local image descriptors. PAMI IEEE, 2011, 33(1): 43–57.

9 Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. CVPR. USA. 2001,1. 511–518.

10 Mikolajczyk K, Schmid C. A performance evaluation of local descriptors. PAMI IEEE, 2005, 27(10): 1615–1630.