文章编号 1004-924X(2011)07-1686-09

加速的 Fast Hessian 多尺度斑点特征检测

韩 冰[1*],王永明[2],孙继银[1]
(1. 第二炮兵工程学院,陕西 西安 710025;2. 北京信息高技术研究所,北京 100085)

摘要:针对目前效率最高的斑点检测算法—Fast Hessian 算法的运行速度还无法满足目标识别与跟踪等对图像实时性要求较高的技术应用的问题,提出了加速的 Fast Hessian 多尺度斑点特征检测算法来进一步提升 Fast Hessian 的运行速度。该算法从减少算法过程中滤波运算量的角度入手,有选择地计算每个 Octave 首末两个尺度层中的采样点值;与原算法在这两层中求取所有采样点值的方法进行比较,提出的算法明显降低了滤波运算量,从而缩短了斑点检测过程的耗时。实验结果表明,加速的 Fast Hessian 算法在斑点检测结果上与原算法一致,而运行速度比原算法提升了近 40%,因此更加适合实时应用。

关 键 词:加速的 Fast Hessian 算法;多尺度斑点特征;尺度空间;初次定位;二次定位
中图分类号:TP391.4 **文献标识码**:A **doi**:10.3788/OPE.20111907.1686

Accelerated Fast Hessian multi-scale blob feature detection

HAN Bing[1*], WANG Yong-ming[2], SUN Ji-yin[1]

(1. *The Second Artillery Engineering College, Xi'an* 710025, *China*;
2. *Beijing Information High Technology Institute, Beijing* 100085, *China*)
* *Corresponding author, E-mail: icytg@126.com*

Abstract: As the origional Fast Hessian which is the most efficient blob feature detection algorithm can not meet requirements of those images in real-time applications to the target recognition, target tracking and so on, an accelerated Fast Hessian multi-scale blob feature detection algorithm is proposed to upgrade the detecting speed of the Fast Hessian. The basic idea of the proposed algorithm is to decrease the number of filter operations and to calculate selectively the values of sample points in the first and last scales for each Octave. Compared to the original one which calculates all the sample point values in these scales, the number of filter operations are distinctly decreased and the consuming time of detecting processing is also reduced. The experiments indicate that the accelerated Fast Hessian algorithm and the original one have the same detection results, but the implementation speed of the accelerated Fast Hessian is upgraded nearly 40% of the original one. It concludes that the accelerated algorithm is much more fit for real-time applications.

收稿日期:2010-09-22;**修订日期**:2010-11-18.
基金项目:第二炮兵科技创新基金资助项目

Key words: accelerated Fast Hessian algorithm; multi-scale blob feature; scale space; first location; second location

1 引　言

图像中的局部特征是近几年图像处理领域的研究热点，因为其具备很强的抵抗几何畸变、亮度变化的能力，因此常被用于目标识别、目标跟踪、景象匹配和机器人技术中。多尺度斑点是局部特征中的一个重要特例，可定义为图像中与周围有着颜色和灰度差别的区域。与单纯的角点特征相比，它的稳定性好，抗噪能力较强[1]，目前，典型的多尺度斑点特征提取算法有 LoG(Laplacian of Gaussian)[2-3]、DoH[2-3](Determinant of Hessian)、DoG[4-7](Difference of Gaussian)和 Fast Hessian[8-10] 等。其中效率最高的当属 Herbert Bay[8-9] 于 2006 年正式提出的 Fast Hessian 算法。仿照 DoG 近似 LoG 的思想，Fast Hessian 是对 DoH 的近似和简化，它先利用方框滤波去近似 Hessian 矩阵中的 3 个方向的二阶高斯导数，然后滤波时再利用积分图像[11-12] 快速求出卷积和，不仅大大提升了算法速度，而且保持了较高的特征点重复率，因此 Fast Hessian 是一种真正意义上高效快速的斑点提取算法。

虽然 Fast Hessian 在所有斑点提取算法中效率最高，但是面对目标识别与跟踪等对图像实时性处理要求苛刻的技术应用，其运行速度也无法满足所有要求。以目标跟踪为例，目标跟踪通常都是在实时采集一系列视频图像序列的同时对这些图像中的目标进行检测和锁定的。一般情况下，只有帧速率(frame/s)达到 5 以上，才能基本保证跟踪过程的实时性，达到 8 以上，才能保证采集视频的连贯性，即要求对每一帧图像进行目标检测和锁定的过程必须在 200 ms 甚至更短的时间之内完成。对于 Fast Hessian 算法而言，其在 CPU 为 Pentium IV，主频 2.80 GHz 的 PC 机上对大小 320×256 的单幅图像进行斑点特征检测的耗时是 60 ms(详见本文 4.2 节)，比规定时间要求的 1/4 还要多。如果再加上后续的斑点描述、斑点匹配、仿射矩阵求取和目标定位等过程的耗时，单幅图像目标锁定的时间肯定会超过 200 ms，除非进一步提升硬件性能，否则根本无法保证目标跟踪的实时性。

为了使 Fast Hessian 能够更好地适应实时应用的要求，尽量缩短斑点检测过程的耗时，本文从减少尺度空间滤波过程运算量的方向入手，重新对 Fast Hessian 算法的步骤进行设计，提出了一种加速的 Fast Hessian 多尺度图像斑点特征检测算法。与原算法相比，本文算法不仅保证了检测结果的一致性，而且在尺度空间的建立过程中减少了大量的滤波运算，提升了算法的速度。对不同大小的图像进行测试的结果表明，本文算法的执行速度比原算法提升了近 40%。

2 原 Fast Hessian 检测斑点

原 Fast Hessian 算法分为计算积分图像、构建尺度空间、斑点初始定位和斑点插值精确定位 4 个步骤，对于以上步骤的详细描述请见文献[8-10]。这里仅对原 Fast Hessian 算法中与卷积过程和结果相关的步骤进行分析。

2.1 斑点检测依据

Fast Hessian 的斑点检测依据如公式(1)～(3)所示。

$$Det(H)=D_{xx}D_{yy}-\omega^2 D_{xy}D_{xy}, \tag{1}$$

$$Sig=\begin{cases}1, D_{xx}+D_{yy}>0\\ -1, D_{xx}+D_{yy}<0\end{cases}, \tag{2}$$

$$D=\begin{cases}0, & Det(H)<0\\ Sig\cdot Det(H), & Det(H)>0\end{cases}. \tag{3}$$

公式(1)中：$Det(H)$ 表示近似 Hessian 矩阵的行列式值；D_{xx}、D_{yy} 和 D_{xy} 是用方框滤波模板对高斯函数 3 个二阶导数 L_{xx}、L_{yy} 和 L_{xy} 的分别近似，如图 1 所示，目的是为了能够利用积分图像使原本复杂的卷积运算转变成几步简单的加减运算，以提升速度；ω 是为了平衡原值与近似值之间关系所添加的加权值，一般等于 0.9；为了能够在不同尺寸模板滤波结果之间进行比较，D_{xx}、D_{yy}、D_{xy} 的值求出之后，还要将它们各自除以模板面积做归一化处理。

公式(2)中 Sig 表示 Laplacian 算子 $D_{xx}+D_{yy}$ 的符号，主要用于区分斑点的亮暗程度，以便在以后的斑点匹配过程中提升速度。如果 Sig 等

于1，则斑点为暗斑，否则为亮斑。

公式(3)将 D 定义为 Fast Hessian 的斑点检测依据，最后作为采样点值存入尺度空间。由于在斑点检测的过程中，会选取极大值点作为斑点，因此凡是 $Det(H)<0$ 的点均被直接赋0值，$Det(H)>0$ 的点会存储 Sig 和 $Det(H)$ 的乘积，这样采样点的亮暗程度就可以在尺度空间中得到标识了。

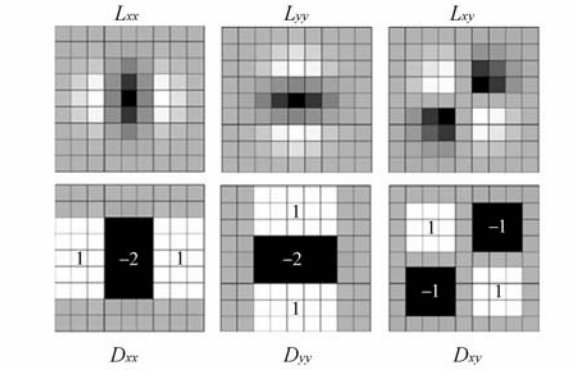

图1 由方框滤波 D_{xx}、D_{yy} 和 D_{xy} 近似的高斯二阶导数 L_{xx}、L_{yy} 和 L_{xy}

Fig. 1 Gaussian second derivatives L_{xx}, L_{yy} and L_{xy} approximated by box filters D_{xx}, D_{yy} and D_{xy}

2.2 构建尺度空间

为了能够检测出不同尺度的斑点特征，Fast Hessian 首先将尺度空间划分为若干个 Octave，然后又将每个 Octave 划分为若干个尺度层，即 Scale，不同的 Scale 对应着不同的斑点尺度。根据文献[10]的分析，当尺度>10之后，斑点特征在相应尺度层的检测数量会急剧下滑。因此，Fast Hessian 所建立的尺度空间大小一般都为4×4(即4个 Octave，每个 Octave 含4个 Scale)，这是因为4×4的尺度空间尺度是1.2～26，能够检测出图像中大部分的斑点特征。

为了给尺度空间内的各采样点赋予 D 值，每个尺度层都要按照公式(1)～(3)使用相应尺寸的方框滤波模板和积分图像进行滤波。总体来说，模板尺寸是按照 Octave 和 Scale 的不断增大而逐渐增大的，其与各尺度层之间的具体对应关系如图2所示。其中9×9的滤波模板作为 Fast Hessian 尺度空间的初始层，所对应的尺度为1.2，以此为基准就可求取其它尺寸的模板所对应的尺度大小。这样等滤波过程结束后，每个尺度层的每一个采样点都获取了相应的 D 值，如图3所示。图3中的 O 表示 Octave 的索引，S 表示 Scale 的索引，Sample 是每个 Octave 内滤波时使用的采样间隔，其大小为 2^O。

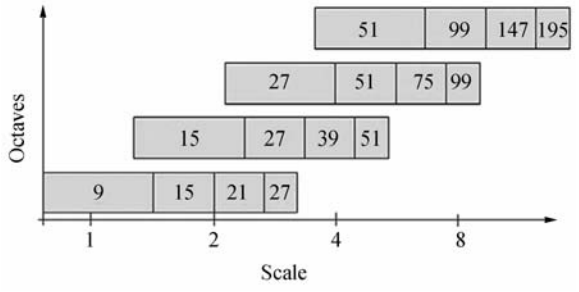

图2 各尺度层对应的方框滤波模板尺寸

Fig. 2 Box filter mask size for each scale

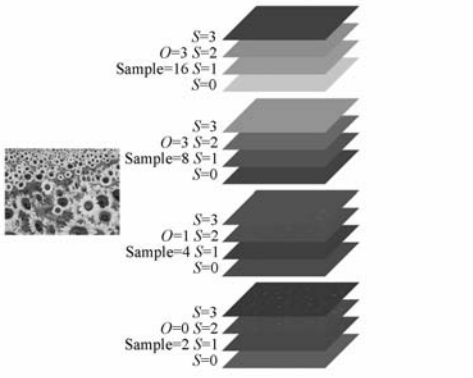

图3 4×4尺度空间的构建结果

Fig. 3 Building result for scale space of 4×4

2.3 多尺度斑点检测

尺度空间构建完毕后，就进入了斑点检测的步骤。Fast Hessian 首先进行一次斑点粗定位，然后再进行一次斑点精确定位。粗定位采用三维非极值抑制法，即对每个 Octave 内第2个 Scale 至倒数第2个 Scale 平面中的各采样点进行遍历，取各点处 D 的绝对值进行比较，这是因为 D 的本身包含了 Laplacian 算子的符号用来标识亮暗斑点，因此要先取绝对值后再进行比较。如果 $|D|$ 的值大于一个事先设定的阈值并大于其3×3×3邻域内的其他26个点值，那么该点就被定为局部极值点，即粗斑点，如图4所示。

由于三维非极值抑制法是在离散空间内进行的，因此所检测到的局部极值点并非真正意义上的极值点，所以粗斑点确定后还要进一步采用插值的方法来精确定位斑点，该步骤的详细过程可

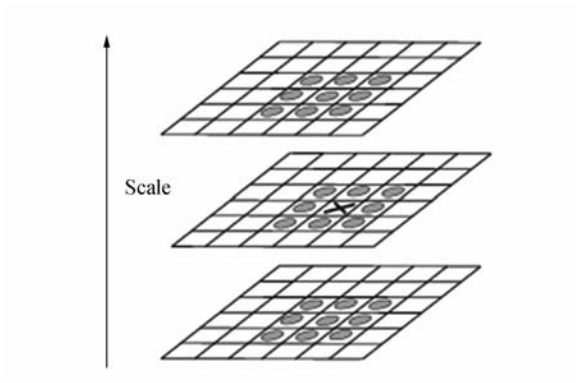

图 4　斑点搜索与定位[4]

Fig. 4　Search and location for blob features[4]

见文献[9]。

2.4　分析

以上是 Fast Hessian 检测多尺度斑点的核心步骤。其中从 Fast Hessian 算法斑点粗定位的过程可以看出，由于所采用的三维非极值抑制法只在每个 Octave 的中间两层进行遍历，因此所有粗斑点都是在每个 Octave 的中间两层中选取出来的，而每个 Octave 中首末两层的采样点值只作为比较使用，并不参与斑点定位。如果一开始在构建尺度空间时，就不对每个 Octave 的首末两层进行滤波，而只对中间两层进行滤波，这样就能使整个过程中公式(1)～(3)的求取数量减少一半。

另一方面，为了保证斑点检测结果的正确性，各 Octave 中首末两层的滤波过程可以放在斑点定位的步骤中有选择地进行，即待每个 Octave 中间两层的滤波过程结束后，先在这两层的 3×3 空间筛选一遍极值点，这样就可以剔除掉一大部分的非斑点采样点，得到包含粗特征斑点的小数量候选粗斑点。针对这些候选粗斑点，再在 Octave 的首末两层中选择需要与候选粗斑点做比较的采样点并按公式(1)～(3)进行运算，这样首末两层所进行的运算量也能比在原算法时减少一大部分。总体来说，采用以上方法来建立尺度空间和定位斑点，能比原算法过程的滤波运算量小很多，因此算法速度定会得到提升。基于以上思想，本文提出了一种加速的 Fast Hessian 多尺度斑点特征检测算法。

3　加速的 Fast Hessian 检测斑点

基于尽可能减少尺度空间滤波过程运算量的思想，本文重新设计了 Fast Hessian 的算法步骤，提出了一种加速的 Fast Hessian 多尺度斑点特征检测算法。该算法共分为以下 5 个步骤：求取积分图像、建立尺度空间、斑点初次定位、斑点二次定位和插值精确定位。其中因为计算积分图像的步骤与原 Fast Hessian 算法相同，不再赘述，具体的求取过程请参见文献[12]。这里仅详细介绍加速的 Fast Hessian 算法的其余几个关键步骤。

3.1　构建尺度空间

加速 Fast Hessian 的检测依据与原算法相同，都是采用公式(1)～(3)对尺度层进行滤波。沿用原 Fast Hessian 算法的思想，加速 Fast Hessian 也将尺度空间的大小定为 4×4，并将其中所有像素点的值都初始化为 0。但是在构建时，加速 Fast Hessian 只在每个 Octave 的中间两个尺度层上执行公式(1)～(3)所示的滤波运算，以求取这些尺度层上各采样点处的 D 值。这样，在尺度空间构建完毕后，每个 Octave 中只有 $S=1$ 和 $S=2$ 的尺度层有数据，而 $S=0$ 和 $S=3$ 的平面上像素值依旧还保持为 0，如图 5 所示。

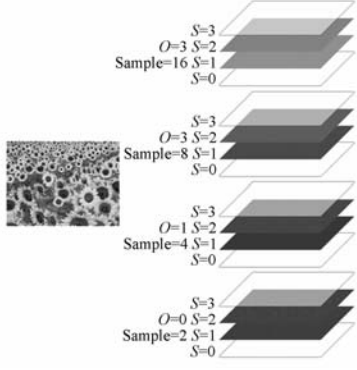

图 5　加速 Fast Hessian 算法构建尺度空间

Fig. 5　Builiding scale space based on accelerated Fast Hessian

单独对照尺度空间构建的步骤，加速的 Fast Hessian 比原算法所执行的滤波运算要减少一半，因此在该步骤的执行速度会有 50% 的提升。

这里需要说明的是，为了保证检测结果的正确性，每个 Octave 首末两个尺度层也是要做滤波运算的，但是不在本步骤中执行，而是将其移至斑点二次定位的步骤中执行。这样做的根本目的就是想尽量避免跟原算法一样为这两层的所有采样点都滤波，详细过程见 3.3 节。

3.2 斑点初次定位

尺度空间构建完毕后，就可以对其中已存入 D 值的采样点进行第一次筛选，本文将其称为斑点特征的初次定位。这里需设定一个阈值，凡是 $|D|$ 小于该阈值的采样点均被舍去。整个过程以 Octave 为单位进行筛选，且 Octave 和 Scale 的索引 O 和 S 都是从 0 开始，至 3 结束。每个 Octave 中均执行以下两个步骤：

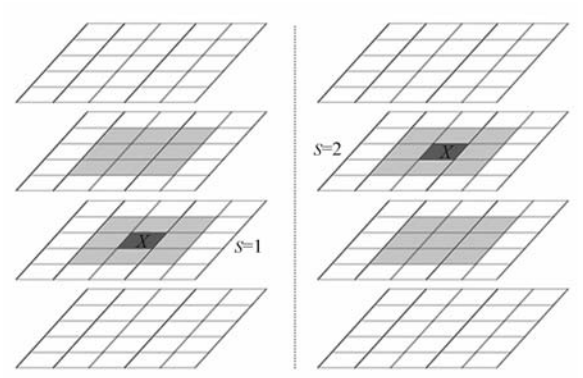

图 6　斑点初次定位

Fig. 6　First location of blobs

Step 1. 在 $S=1$ 的尺度层上，以各采样点为中心，将其位置上的 $|D|$ 先与其周围 3×3 邻域的采样点进行比较，然后再与 $S=2$ 尺度层上与之对应的 3×3 邻域的采样点进行比较，如图 6 左侧所示。如果该点处的 $|D|$ 值大于阈值且大于其余 17 个点的值，则保留该点作为候选粗斑点，否则剔除该点。

Step 2. 在 $S=2$ 的尺度层上，同样也以采样点为中心，将其位置上的 $|D|$ 先与其周围 3×3 邻域的采样点进行比较，然后再与 $S=1$ 尺度层上与之对应的 3×3 邻域的采样点进行比较，如图 6 右侧所示。如果该点处的 $|D|$ 值大于阈值且大于其余 17 个点的值，则同样也保留该点作为候选粗斑点，否则剔除该点。

这样通过第一轮的筛选，即可用尺度空间构建步骤提供的所有已知信息，获得小数目的候选粗斑点。原 Fast Hessian 在该步骤时，对中间两层的每个采样点都进行了 26 次比较，而加速的 Fast Hessian 算法只进行了 17 次比较，缺少了 9 次比较，这里将把这 9 次比较移至下面斑点二次定位的步骤进行，目的就是只针对各候选粗斑点为其需要用到的采样点求取 D 值，而不是把每个 Octave 首末两层内的所有采样点的 D 值都求取出来，这样首末两层的滤波运算量就比原算法大大减少了，详细过程见 3.3 节。

3.3 斑点二次定位

斑点二次定位的过程，也就是将第一轮筛选后得到的候选粗斑点与其相对应的首或末尺度层内的 3×3 邻域点值进行比较的过程。由于已构建完毕的尺度空间内首末两个尺度层的所有采样点值至今还为 0，因此需要在斑点二次定位的过程中，对首末两尺度层中用于比较的采样点按照公式(1)～(3)做滤波运算。

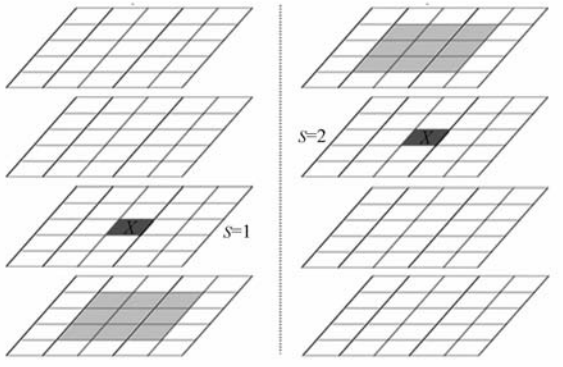

图 7　斑点二次定位

Fig. 7　Second location of blobs

这里会遇到一种特殊情况，即初次定位得到的两个候选粗斑点位于同一尺度层且坐标相邻，那么两者在首层或末层上所对应的 3×3 邻域内必然会出现重叠的采样点。如果这种相邻的候选粗斑点很多，则需要重复求取采样点 D 值的数目也会增多，那么扩展到所有候选斑点都相邻的极端情况，如果还针对每一个候选粗斑点都求取 9 个 D 值，那么提出的加速算法根本就起不到任何优化效果，甚至还不如原算法。

解决该问题，只需对首末两个尺度层内各采样点存储的 D 值稍加改进即可。使其既能完成比较的任务，同时也能具备标识功能。改进的方法如公式(4)所示。

$$D^* = |D| + 1 , \tag{4}$$

式中：D^* 表示首末两层采样点处存储的检测依据；D 的定义如公式(3)所示，这里用到的是它的绝对值，由于首末两层采样点不参与定位，因此 Laplician 的符号在此不用起到表示亮暗的作用；加 1 是为了与首末两层中的 0 值进行区分，因为公式(2)有可能使 D 值为 0，加 1 后能够说明该采样点已经被计算过，不用再重复计算，这样就解决

了以上问题。

斑点二次定位的过程，需要根据候选粗斑点所在的尺度层分两种情况考虑。

Situation 1. 如果候选粗斑点位于 $S=1$ 的尺度层内，那么让该点处的 $|D|$ 与 $S=0$ 尺度层中正对的 3×3 个采样点的 D^*-1 值进行比较，如图 7 的左侧所示。这 9 个采样点 D^* 值是在比较前逐一求出的，求取时须先判断各点处的值是否为 0，为 0 就执行公式(4)，大于 0 说明该点的 D^* 值已被求出，不用再求。这样，如果候选粗斑点的 $|D|$ 值均大于上述 9 个点的 D^*-1 值，就保留该候选斑点的信息，否则就剔除。

Situation 2：如果候选粗斑点位于 $S=2$ 的尺度层内，那么就让它的 $|D|$ 与 $S=3$ 尺度层中正对的 3×3 个采样点的 D^*-1 值进行比较，如 7 图右侧所示。这 9 个采样点 D^* 值的求取过程与 Situation 1 相同。同样，如果候选粗斑点的 $|D|$ 值均大于上述 9 个点的 D^*-1 值，就保留该候选斑点的信息，否则就剔除。

经斑点二次定位过程筛选后，部分不符合要求的候选粗斑点被剔除，保留下来的点称为粗斑点，其应与 2.3 节中经三维非极值抑制法所得到粗斑点的结果一致。

3.4　插值精确定位

计算出所有粗斑点后，也需要进行插值精确定位的步骤。该步骤与文献[9]的描述基本一致。但由于在斑点二次定位时，需给首末两个尺度层内的采样点添加是否已被计算的标识，各采样点值存储的都是 D 取绝对值加 1 后的结果。因此，插值过程如果要调用首末两个尺度层上的采样点，需要将其减 1 后再使用。

以上就是本文提出的加速的 Fast Hessian 多尺度斑点特征检测算法的执行步骤。可以看出，其检测结果与原算法保持了一致，因为并未对斑点的检测依据做任何更改。另一方面，加速的 Fast Hessian 在每个 Octave 首末两个尺度层的滤波运算量比原算法大大减少了，因此速度会有大幅提升，相应的验证过程见第 4 节。

4　实验与分析

本文在 CPU 为 Intel Pentium IV，主频2.80 GHz，内存 1 G 的 PC 机环境下，利用 Visual Studio 2008 的 C# 语言分别对本文第 2 部分介绍的原 Fast Hessian 算法和第 3 部分提出的加速的 Fast Hessian 算法进行了编程实现。并对它们的斑点检测结果和时间花费进行了测试比较，实验结果与分析如下。

4.1　检测结果对比

本文将原 Fast Hessian 算法和加速的 Fast

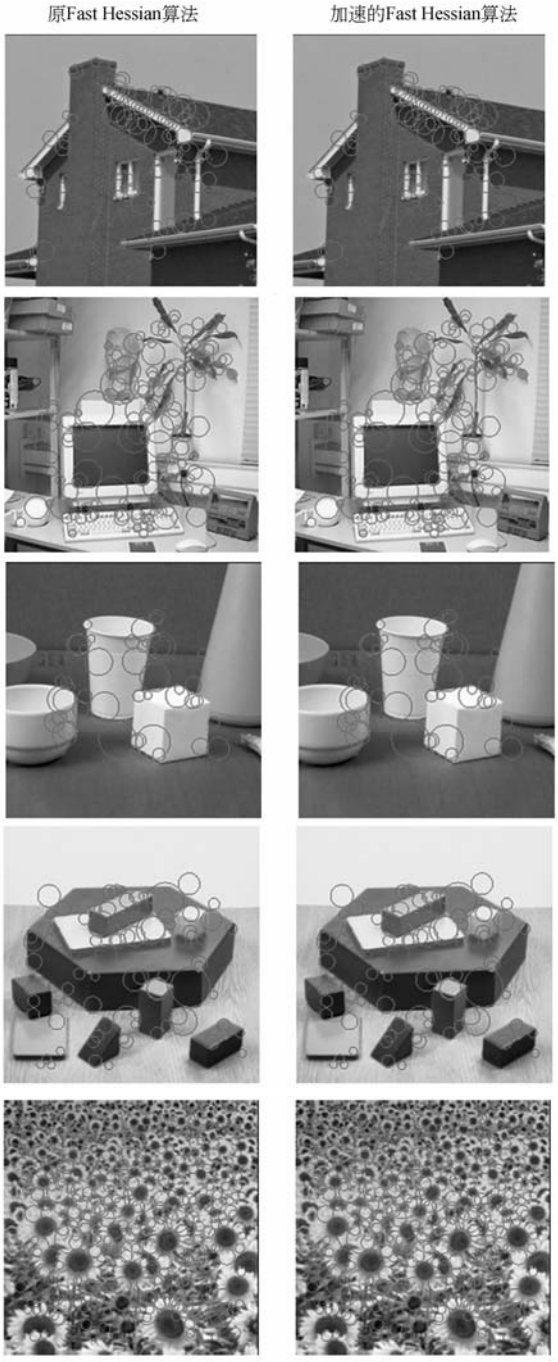

图 8　两种 Fast Hessian 算法检测斑点的结果对比

Fig. 8　Comparison of detection results by two kinds of Fast Hessian algorithms

Hessian 算法的检测阈值都设置为 50，然后对大量的图像进行了斑点检测结果的对比测试，结果表明加速的 Fast Hessian 算法和原算法所检测到的斑点图像坐标和特征尺度完全一致。由于篇幅限制，这里仅列出 5 幅 256×256 图像的斑点检测结果，如图 8 所示。

从图 8 可以看出，两种 Fast Hessian 算法的检测结果是一模一样的。因此可以证明，虽然加速的 Fast Hessian 算法的执行过程中减少了滤波运算量，但是并未对斑点检测的结果造成任何影响，从而也证明了加速的 Fast Hessian 算法的正确性。

4.2 时间花费对比

首先对原 Fast Hessian 算法和加速的 Fast Hessian 算法分别定义时间花费函数。假设积分图像求取过程消耗的时间为 T_1，尺度空间构建的时间为 T_2，斑点初步定位的时间为 T_3，插值精确定位的时间 T_4，那么原 Fast Hessian 算法检测斑点特征的总耗时 T_{d_1} 的计算方法如公式(5)所示。对于加速的 Fast Hessian 算法，由于要比原算法多一步斑点二次定位的过程，因此假设该步骤耗时为 T_5，那么加速的 Fast Hessian 算法检测斑点特征的总耗时 T_{d_2} 的求取方法如公式(6)所示。

$$T_{d_1}=T_1+T_2+T_3+T_4, \tag{5}$$

$$T_{d_2}=T_1+T_2+T_3+T_5+T_4, \tag{6}$$

斑点检测时间的长短与输入图像的大小有关。图像越大，时间花费就越长。因此按图像尺寸对所实现的原 Fast Hessian 算法和加速的 Fast Hessian 算法的斑点检测时间进行以下对比测试。测试用图选择 http://www.robots.ox.au.uk/~vgg 提供的 Graffiti 图像，如图 9 所示。将该图像大小调整为 320×256，480×384，640×512 和 800×640 后进行测试。

根据公式(5)和(6)，表 1 给出了两种算法各步骤的执行时间对比。

从表 1 的数据可以看出，随着图像大小的不断增大，算法各步骤的时间花费也随之增长。但是无论图像尺寸如何变化，文中提出的加速的 Fast Hessian 算法的总执行时间均比原 Fast Hessian 算法要少。对于积分图像的求取时间

图 9 测试花费时间用到的 Graffiti 图像，其大小被调整为 320×256，480×384，640×512 和 800×640

Fig. 9 Image Graffiti for testing cost time, which size was adjusted to 320×256, 480×384, 640×512 and 800×640

T_1，由于原算法与快速算法一致，因此两者在该项上的时间花费基本相同；对于尺度空间构建的时间 T_2，由于加速的 Fast Hessian 算法在每个 Octave 中间两个尺度层上进行滤波运算，因此理论上加速的算法的 T_2 值应为原算法的 1/2；对于斑点初步定位的时间 T_3，原 Fast Hessian 是将各采样点与其三维邻域中 26 个点进行比较，而加速的 Fast Hessian 是与其邻域中 17 个点进行比较，因此后者的 T_3 值应略小于原算法的 T_3 值；对于二次定位的时间 T_5，由于该步骤只在加速的 Fast Hessian 算法中出现，因此原算法中无该步骤的执行时间；对于插值精确定位的时间 T_4，加速的 Fast Hessian 算法在调用首末两个 Scale 中的采样点值时，需要减 1 后使用，比原算法多了几步减法运算，因此加速算法的 T_4 值应略大于原算法。以上分析均在表 1 获取的实验数据中得到了验证。可以看出，对于算法的总执行时间 T_{d_1} 和 T_{d_2}，无论图像尺寸是多少，加速的 Fast Hessian 算法的时间花费均明显少于原算法。最后对 Fast Hessian 算法的速度提升幅度进行了测试，表 1 中最后 1 行数据显示，对于 4 幅不同大小的图像，加速的 Fast Hessian 的速度提升幅度在 36.0%～38.1%之间，接近 40%。因此，可以得出结论，加速的 Fast Hessian 算法的时间花费确实比原算法大大减少，其算法整体速度的提升幅度接近 40%。

表 1　两种算法的时间花费对比

Tab. 1　Consuming time comparison for two kinds of algorithms

花费时间/ms	原 Fast Hessian 算法				加速的 Fast Hessian 算法			
	320×256	480×384	640×512	800×640	320×256	480×384	640×512	800×640
积分图像 T_1	3.208	7.032	12.554	21.837	3.197	7.086	13.433	20.651
尺度空间 T_2	50.641	125.092	234.308	379.976	24.132	61.125	116.283	187.896
初步定位 T_3	3.276	7.993	14.077	21.192	3.258	7.718	13.420	20.605
二次定位 T_5	—	—	—	—	4.776	10.685	21.329	28.845
插值定位 T_4	1.978	4.380	6.770	8.817	2.046	4.482	6.899	9.144
总时间 T_{d_1} 和 T_{d_2}	59.103	144.497	267.709	431.822	37.409	91.096	171.364	267.141
速度提升/%	—	—	—	—	36.7	37.0	36.0	38.1

5　结　论

为了使 Fast Hessian 算法能够更好地适应目标识别、目标跟踪等实时性应用的要求，本文提出了加速的 Fast Hessian 多尺度斑点特征检测算法。该算法从减少算法过程中的滤波运算量的角度入手，有选择地计算每个 Octave 首末两尺度层中的采样点值。与原算法将这两层中所有采样点值都求取出来的做法相比，其滤波运算量得到有效降低。实验证明，加速的 Fast Hessian 的斑点检测结果与原算法一致，但其运行速度比原算法提升了 40%。

参考文献：

[1] 王永明，王贵锦. 图像局部不变性特征与描述[M]. 北京：国防工业出版社，2010.
WANG Y M, WANG G J. *Image Local Invariant Features and Descriptors*[M]. Beijing: National Defense Industry Press, 2010. (in Chinese)

[2] LINDEBERG T. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention[J]. *International Journal of Computer Vision*, 1993, 11(3): 283-318.

[3] LINDEBERG T. Scale-space theory: a basic tool for analysing structures at different scales[J]. *Journal of Applied Statistics*, 1994, 21(2): 224-270.

[4] LOWE D G. Distinctive image features from scale-invariant keypoints[J]. *International Journal of Computer Vision*, 2004, 60(2): 91-110.

[5] 纪华，吴元昊，孙宏海，等. 结合全局信息的 SIFT 特征匹配算法[J]. 光学 精密工程，2009, 17(2): 439-444.
JI H, WU Y H, SUN H H, *et al*.. SIFT feature matching algorithm with global information[J]. *Opt. Precision Eng.*, 2009, 17(2): 439-444. (in Chinese)

[6] 杨晓敏，吴炜，卿粼波，等. 图像特征点提取及匹配技术[J]. 光学 精密工程，2009, 17(9): 2276-2282.
YANG X M, WU W, QING L B, *et al*.. Image feature extraction and matching technology[J]. *Opt. Precision Eng.*, 2009, 17(9): 2276-2282. (in Chinese)

[7] 丁雪梅，王维雅，黄向东. 基于差分和特征不变量的运动目标检测与跟踪[J]. 光学 精密工程，2007, 15(4): 570-576.
DING X M, WANG W Y, HUANG X D. New method for detecting and tracking of moving target based on difference and invariant[J]. *Opt. Precision Eng.*, 2007, 15(4): 570-576. (in Chinese)

[8] BAY H, TUYTELAARS T, VAN GOOL L, *et al*.. Surf: speeded up robust features[C]. *Proceeding of European Conference on Computer Vision*, 2006: 404-417.

[9] BAY H. *From wide-baseline point and line correspondences to 3D*[D]. Switzerland: ETH Zurich, 2006.

[10] BAY H, ESS A, TUYTELAARS T, *et al*.. Speeded-up robust features (SURF)[J]. *Com-*

puter Vision and Image Understanding, 2008, 110(3):346-359.

[11] CROW F. Summed-area tables for texture mapping[C]. *Proceeding of SIGGRAPH*, 1984, 18(3):207-212.

[12] VIOLA P, JONES M. Robust real-time face detection[J]. *International Journal of Computer Vision*, 2004, 57(2):137-154.

作者简介：

韩　冰(1982－)，女，山东兖州人，博士研究生，2004年、2007年于第二炮兵工程学院分别获得学士、硕士学位，主要从事数字图像处理、目标自动识别与跟踪方面的研究。E-mail: icytg@126.com

孙继银(1952－)，男，山东单县人，教授，博士生导师，1976年于国防科技大学获得学士学位，主要从事数字图像处理和多媒体技术方面的研究。E-mail: sjy44340@163.com

导师简介：

王永明(1958－)，男，浙江富阳人，研究员，博士生导师，1996年于清华大学获得博士学位，主要从事数字图像处理、景象匹配、目标识别等方面的研究。E-mail: wym58@hotmail.com

●下期预告

非正交二维 MEMS 倾斜镜的研制

庄须叶[1]，汪为民[1]，陶逢刚[1,2]，姚　军[1]，高福华[3]

(1. 中国科学院 光电技术研究所微细加工光学技术国家重点实验室，四川 成都 610209；
2. 中国科学院 研究生院，北京 100039；
3. 四川大学 物理学院高能量密度物理与技术教育部重点实验室，四川 成都 610064)

为提高在非正交方向上的光转换效率，克服传统二维倾斜镜空间适应性差的缺点，设计并加工了一种转轴非正交的二维 MEMS 倾斜镜。倾斜镜上电极非对称地固定在基底上，通过控制上、下电极的加电方式实现倾斜镜在两个非正交轴上的偏转变形。采用三层膜的结构设计，消除了上电极应力变形对镜面平整度的影响。通过在上电极加工微小突起，减小上下电极的重叠面积，避免了倾斜镜的吸合失效，且上、下电极仅在其边缘处重叠，确保了静电力的有效利用。研制的倾斜镜驱动电压低，在 3.5 V 的电压下可实现绕水平 X 轴 0.16°、绕倾斜 Y 轴 0.03°的偏转，Y 轴与 X 轴成 145.37°。倾斜镜结构简单，可实现绕两个非正交转轴的偏转，空间适应性好，且有效避免了静电吸合对镜子的损坏。