SURF

第一章 简述

SURF（SpeedUp Robust Features）——加速稳健特征算法，在 2006 年由 Bay.H 和 Van Gool.H 共同提出（“SURF: Speeded Up Robust Features”），SIFT 是尺度不变特征变换 SIFT 的加速版。一般来说，标准的 SURF 算子比 SIFT 算子快好几倍，并且在多幅图像上具有更好的稳定性。SURF 最大的特征在于采用了 Harr 特征以及积分图的概念，这大大加快了程序的运行时间, 可以应用于物体识别以及三维重建中。

与 SIFT 一样，SURF 的特征也是局部不变特征。

SURF 在执行效率上有两大制胜法宝——一个是积分图在 Hessian（黑塞矩阵）上的使用，一个是降维的特征描述子的使用。

在 SIFT 中，Lowe 在构建尺度空间时使用 DoG 对 LoG 进行近似，则使用盒子滤波器（box filter）对 DoH 进行近似。

第二章 斑点（关键点）检测

2.1 Hessian 矩阵

我们知道：SIFT 算法建立一幅图像的金字塔，在每一层进行高斯滤波并求取图像差（DOG）进行特征点的提取，而 SURF 则用的是 Hessian Matrix 进行特征点的提取，所以 Hessian 矩阵是 SURF 算法的核心。

在 SIFT 算法中，我们首先使用 LoG 算子对图像进行卷积获得中的极值点作为该尺度下的斑点，之后再在尺度空间获得的极值点作为最终的斑点（关键点）。在一个一维信号中，让它和高斯二阶导数进行卷积，也就是拉普拉斯变换，那么在信号的边缘处就会出现过零点，如下图所示：

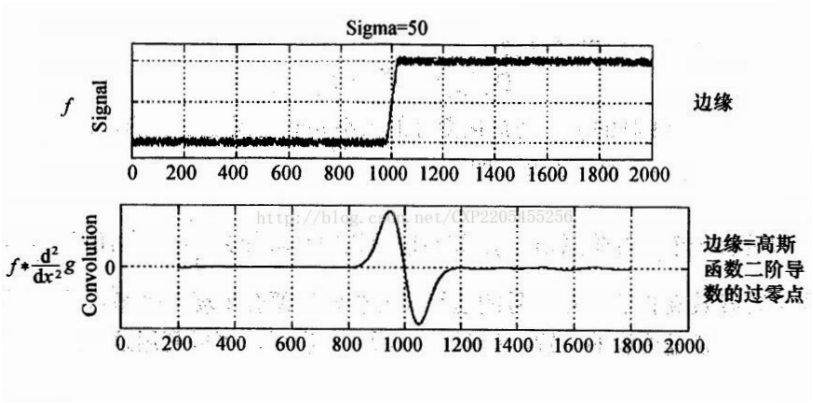

图 2-1 一维信号与 LoG 算子的卷积

总结 SIFT 算法时提到过，高斯拉普拉斯 LoG 算子的响应值就是在衡量图像与 LoG 算子的相似性，如下图是一个图像的高斯拉普拉斯变换的三维图和灰度图显示，在图像中该尺度下的斑点尺寸与高斯拉普拉斯函数的形状趋于一致时，图像的拉普拉斯响应抵达最大。其后使用 DoG 算子对 LoG 算子进行近似，构建 DoG 金子塔，获得相应尺度上的斑点图。

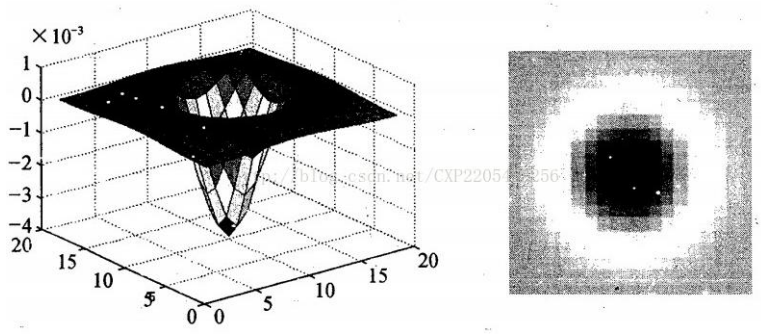

图 2-2 图像与 LoG 算子卷积图（三维与二维）

而 SURF 算法则使用 Hessian 矩阵进行单一尺度下斑点检测。具体是用 Hessian 矩阵行列式的最大值标记斑状结构（blob-like structure）的位置。同时行列式也作为尺度选择的依据。这里是参考了 Lindeberg 的做法（《Feature detection with automatic scale selection》）。

假设图像的函数为 $I(x,y)$，Hessian 矩阵 H 是由函数偏导数组成的。图像中某个像素点的 Hessian 矩阵定义为

$$H\left(I(x,y)\right)=\begin{bmatrix}\dfrac{\partial^2 I}{\partial x^2} & \dfrac{\partial^2 I}{\partial x\partial y}\\ \dfrac{\partial^2 I}{\partial x\partial y} & \dfrac{\partial^2 I}{\partial y^2}\end{bmatrix} \tag{2.1}$$

从而每一个像素点都可以求出一个 Hessian 矩阵。

Hessian 矩阵判别式为：

$$\det(H)=\frac{\partial^2 I}{\partial x^2}\frac{\partial^2 I}{\partial y^2}-\left(\frac{\partial^2 I}{\partial x\partial y}\right)^2 \tag{2.2}$$

判别式的值是 H 矩阵的特征值，可以利用判定结果的符号将所有点分类，根据判别式取值正负，从来判别该点是或不是极点的值。

在 x_0 点上，Hessian 矩阵是正定的，且各分量的一阶偏导数为 0，则 x_0 为极小值点。在 x_0 点上，Hessian 矩阵是负定的，且各分量的一阶偏导数为 0，则 x_0 为极大值点。

对于某个局部区域，若 Hessian 矩阵是半正定的，则这个区域是凸的（反之依然成立）；若负定，则这个区域是凹的(反之依然成立)。而对于正定和负定来说，Hessian 矩阵的行列式总是大于等于 0 的。反过来就是说：某个点若是极大值/极小值，Hessian 矩阵的行列式必然要大于等于 0，而该点 Hessian 矩阵的行列式大于等于 0，则这个点不一定是极大值/极小值(还要判断一阶导数)。所以后面还要进行极大值抑制.

2.2 盒子滤波器

与 SIFT 算法相同，为了保证获得关键点尺度的不变性，也要在不同尺度上获取斑点（极值点），因此先定义尺度空间图像为：

$$L(x,y;\sigma)=I(x,y)*G(x,y;\sigma) \tag{2.3}$$

尺度空间图像的 Hessian 矩阵定义如下：

$$H(x,y;\sigma)=\begin{bmatrix} L_{xx}(x,y;\sigma) & L_{xy}(x,y;\sigma) \\ L_{xx}(x,y;\sigma) & L_{yy}(x,y;\sigma) \end{bmatrix} \tag{2.4}$$

其中

$$\begin{aligned} L_{xx}(x,y;\sigma)&=\frac{\partial^2}{\partial x^2}\left(L(x,y;\sigma)\right) \\ &=\frac{\partial^2}{\partial x^2}\left(I(x,y)*G(x,y;\sigma)\right) \\ &=\left[\frac{\partial^2}{\partial x^2}G(x,y;\sigma)\right]*I(x,y) \end{aligned} \tag{2.5}$$

因此式(2.4)中的$L_{xx}(x,y;\sigma)$可以通过使用高斯二阶微分算子$\left[\frac{\partial^2}{\partial x^2}G(x,y;\sigma)\right]$对图像做积分获得，同理式(2.4)中的$L_{xy}(x,y;\sigma)$和$L_{yy}(x,y;\sigma)$也可以使用高斯二阶微分算子$\left[\frac{\partial^2}{\partial x\partial y}G(x,y;\sigma)\right]$和$\left[\frac{\partial^2}{\partial y^2}G(x,y;\sigma)\right]$对图像做卷积获得。他们的图像如下图所示：

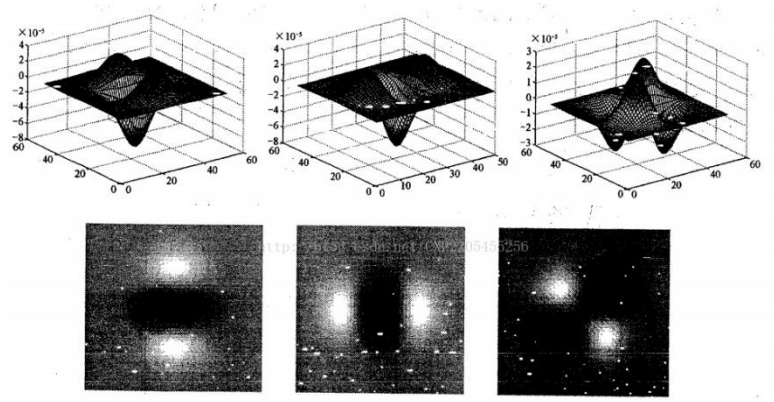

图 2-3 高斯二阶微分算子图像

但是利用 Hessian 行列式进行图像斑点检测时，有一个缺点。由于二阶高斯微分被离散化和裁剪的原因，导致了图像在旋转奇数倍的时，即转换到模板的对角线方向时，特征点检测的重复性降低（也就是说，原来特征点的地方，可能检测不到特征点了）。而此时，特征点检测的重现率最高。但这一小小的不足不影响我们使用 Hessian 矩阵进行特征点的检测。

在这里，与 SIFT 类似，Bay.H 等人也进行了对模板的近似处理，如下图所示，使用盒子滤波器近似高斯二阶微分算子：

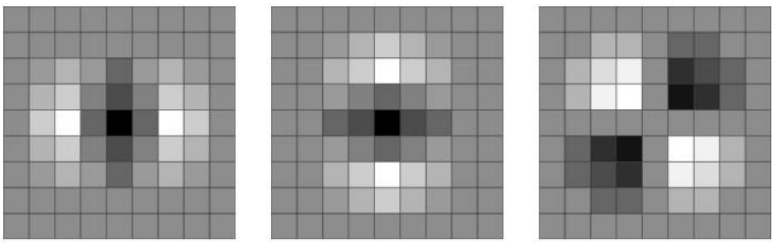

图 2-4 高斯二阶微分算子模板

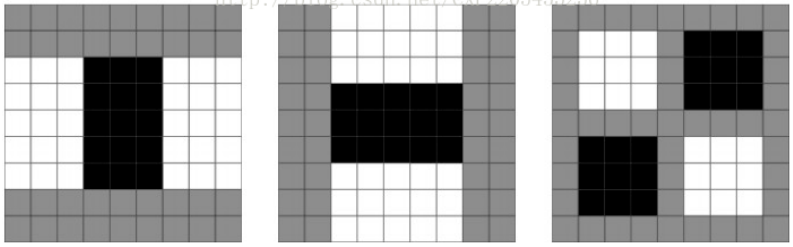

图 2-5 与图 2-4 对应的盒子滤波器模板

采用盒子滤波器与图像卷积的结果记为D_{xx}、D_{yy}和D_{xy}。

2.3 Hessian 矩阵行列式的近似

当我们用$\sigma=1.2$的高斯二阶微分算子进行滤波时，模板尺寸为9×9最为最小的尺度空间值对图像进行滤波和斑点检测，Hession 矩阵的行列式可做如下简化：

$$\begin{aligned}\det(H)&=L_{xx}L_{yy}-L_{xy}L_{xy}\\&=D_{xx}\frac{L_{xx}}{D_{xx}}D_{yy}\frac{L_{yy}}{D_{yy}}-D_{xy}\frac{L_{xy}}{D_{xy}}D_{xy}\frac{L_{xy}}{D_{xy}}\\&=D_{xx}D_{yy}\left(\frac{L_{xx}}{D_{xx}}\frac{L_{yy}}{D_{yy}}\right)-D_{xy}D_{xy}\left(\frac{L_{xy}}{D_{xy}}\frac{L_{xy}}{D_{xy}}\right)\\&=A\left(\frac{L_{xx}}{D_{xx}}\frac{L_{yy}}{D_{yy}}\right)-B\left(\frac{L_{xy}}{D_{xy}}\frac{L_{xy}}{D_{xy}}\right)\\&=\left(A-B\left(\frac{L_{xy}}{D_{xy}}\frac{L_{xy}}{D_{xy}}\right)\left(\frac{D_{xx}}{L_{xx}}\frac{D_{yy}}{L_{yy}}\right)\right)\left(\frac{L_{xx}}{D_{xx}}\frac{L_{yy}}{D_{yy}}\right)\\&=(A-BYZ)C=\left(A-\omega B\right)C\end{aligned}\tag{2.6}$$

$$Y=\frac{\left|L_{xy}(1.2)\right|_F\left|D_{xx}(9)\right|_F}{\left|L_{xx}(1.2)\right|_F\left|D_{xy}(9)\right|_F}=0.912\cong0.9\tag{2.7}$$

$$Z=\frac{\left|L_{xy}(1.2)\right|_F\left|D_{yy}(9)\right|_F}{\left|L_{yy}(1.2)\right|_F\left|D_{xy}(9)\right|_F}=0.912\cong0.9\tag{2.8}$$

其中$\left|M\right|_F$为 Frobenius 范数常数，由于使用了盒子滤波器近似高斯二阶微分算子，因此$L_{xx}\approx D_{xx}$、$L_{yy}\approx D_{yy}$，$C\approx1$不影响极值点的比较，所以最终简化式如下，这也是 SURF 论文里面 Hession 响应值计算公式的来源：

$$\det(H_{approx})=D_{xx}D_{yy}-(0.9D_{xy})^2=D_{xx}D_{yy}-\omega D_{xy}^2\tag{2.9}$$

另外，响应值还要根据滤波器大小进行归一化处理，以保证任意大小滤波器的 F 范数是统一的。0.9^2 是滤波器响应的相关权重 ω，是为了平衡 Hessian 行列式的表示式。这是为了保持高斯核与近似高斯核的一致性。理论上来说对于不同的 σ 的值和对应尺寸的模板尺寸，ω 值是不同的，但为了简化起见，可以认为它是同一个常数。使用近似的 Hessian 矩阵行列式来表示图像中某一点 x 处的斑点响应值，遍历图像中所有的像元点，便形成了在某一尺度下斑点检测的响应图像。使用不同的模板尺寸，便形成了多尺度斑点响应的金字塔图像，利用这一金字塔图像，就可以进行斑点响应极值点的搜索。

这个简化的推导中使用了 D_{xx}、D_{yy} 和 D_{xy}，很多参考资料中并没有介绍它们是什么，沿用 2.2 节的描述，我个人觉得它们应该是通过盒子滤波器卷积获得的值，不同的尺度参数 σ 表示盒子滤波器的大小不同，这个还有待查阅原论文。如果我猜想的没错，则式(2.6)、(2.9)即为能够使用盒子滤波器近似高斯二阶微分算子的原因。

2.4 积分图

积分图像的概念是由 Viola 和 Jones 提出的。积分图像是输入的灰度图像经过一种像素间的累加运算得到种新的图像媒介。对于一幅灰度的图像，积分图像中的任意一点 (x, y) 的值是指从图像的左上角到这个点的所构成的矩形区域内所有的点的灰度值之和。其数学公式如下：

$$ii(i, j) = \sum_{i' \le i, j' \le j} I(i', j') \tag{2.10}$$

图 2-6 即为积分图的图像表示，图中蓝色像素的积分图像的灰度值就是黄色框内的所有像素的灰度值之和。其他像素同样的计算方法。

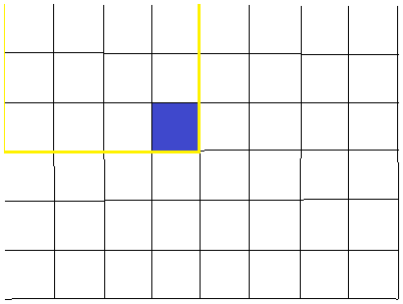

图 2-6 积分图

有了积分图，当我们想要计算图片一个区域的积分，就只需计算这个区域的四个顶点在积分图像里的值，便可以通过 2 步加法和 2 步减法计算得出，其数学公式如下：

$$\sum W = ii(\mathrm{i}_4, \mathrm{j}_4) - ii(\mathrm{i}_2, \mathrm{j}_2) - ii(\mathrm{i}_3, \mathrm{j}_3) + ii(\mathrm{i}_1, \mathrm{j}_1) \tag{2.11}$$

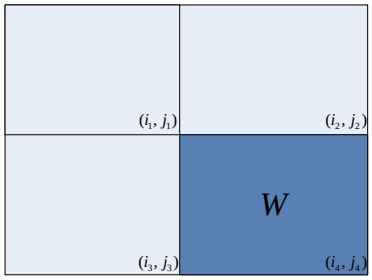

图 2-7 计算 W 区域灰度值和

而积分图可以仅通过遍历图像一次就可以获得，在 SURF 算法中，通过积分图就可获得经过盒子滤波器卷积后的值，因此大大提高了计算 Hessian 矩阵行列式的速度。

2.5 尺度金字塔构造

相比于 SIFT 算法的高斯金字塔构造过程，SRUF 算法速度有所提高。在 SIFT 算法中，每一组（octave）的图像大小是不一样的，下一组是上一组图像的降采样（1/4 大小）；在每一组里面的几幅图像中，他们的大小是一样的，不同的是他们采用的高斯函数的尺度参数 σ 不同。而且在模糊的过程中，他们的高斯模板大小总是不变的（？），只是尺度参数 σ 改变。对于 SURF 算法，图像的大小总是不变的，改变的只是高斯模糊模板的尺寸，当然，尺度 σ 也是在改变的。 如下图所示，分别是 SIFT 和 SURF 构造尺度金字塔的过程：

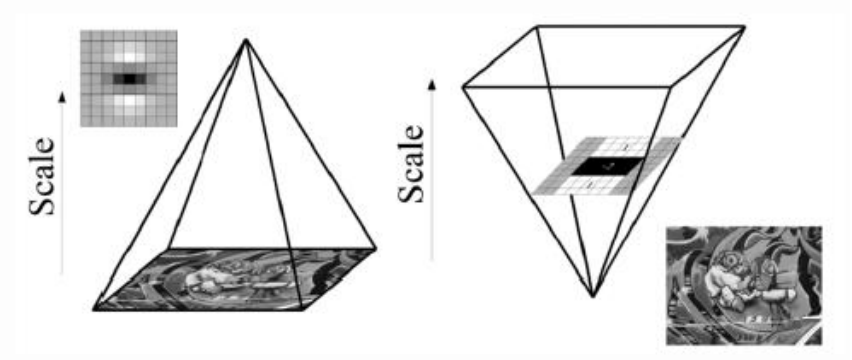

图 2-8 SIFT 和 SURF 尺度金字塔

之所以这么做的目的考虑的主要目的还是效率问题（这样可以利用积分图有关的快速计算，用不同 size 的 Mask 进行卷积运算，复杂度是一样的，仅仅是三个加减法而已）。而且，由于没有对图像进行降采样，所以不存在混叠现象。

SURF 中采用9×9尺寸的滤波器作为起始滤波器，之后的滤波器尺寸可由以下公式计算得出：

$$FilterSize = 3\times\left(2^{octave+1}\times(\text{interval}+1)+1\right) \tag{2.12}$$

octave、interval 在公式中都是从 0 开始，也就是当第 0 组第 0 层时，在公式中 octave = 0， interval = 0。

滤波器响应长度 l（小写 L）、滤波器尺寸 L、组索引 o、层索引 s、尺度σ之间的关系如下：

$$\begin{aligned} l &= 2^{o+1}(s+1)+1 \\ L &= 3\times l = 3\times\left(2^{o+1}(s+1)+1\right) \\ \sigma &= 1.2\times\frac{L}{9} = 1.2\times\frac{l}{3} \end{aligned} \tag{2.13}$$

通常想要获取不同尺度的斑点，必须建立图像的尺度空间金字塔。一般的方法是通过不同的高斯函数，对图像进行平滑滤波，然后重采样图像以获得更高一层的金字塔图像。SIFT 特征检测算法中就是通过相邻两层图像金字塔相减得到 DoG 图像，然后再在 DoG 图像上进行斑点和边缘检测工作的。由于采用了盒子滤波和积分图，所以，我们并不需要像 SIFT 算法那样去直接建立图像金字塔，而是采用不断增大盒子滤波模板的尺寸的间接方法。通过不同尺寸盒子滤波模板与积分图像求取 Hessian 矩阵行列式的响应图像。然后在响应图像上采用 3D 非最大值抑制，求取各种不同尺度的斑点。

如前文所述，我们使用的模板对图像进行滤波，其结果作为最初始的尺度空间层（此时，尺度值为$\sigma=1.2$，近似的高斯微分），后续的层将通过逐步放大滤波模板尺寸，以及放大后的模板不断与图像进行滤波得到。由于采用盒子滤波和积分图像，滤波过程并不随着滤波模板尺寸的增加而使运算工作量增加。

与 SIFT 算法类似，我们需要将尺度空间划分为若干组（Octaves）。一个组代表了逐步放大的滤波模板对同一输入图像进行滤波的一系列响应图。每个组又

由若干固定的层组成。由于积分图像离散化的原因，两个层之间的最小尺度变化量是由高斯二阶微分滤波器在微分方向上对正负斑点的响应长度 l 决定的，它是盒子滤波器模板尺寸的$1/3$。对于9×9的盒子滤波器模板，$l_0=3$，它的下一层的响应长度至少应该在该层l_0的基础上增加2个像素，以保证一边增加一个像素，即$l=5$。这样模板的尺寸就为15×15。以此类推，我们可以得到一个尺寸增大模板序列，它们的尺寸分别为：9×9、15×15、21×21、27×27，黑色、白色区域的长度增加偶数个像素，以保证一个中心像素的存在。因此可以推出式(2.13)的结果。

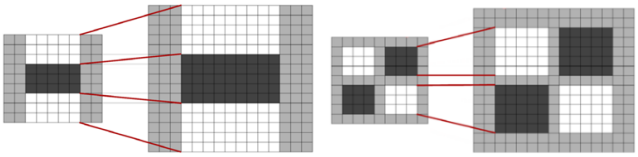

图 2-9 模板大小随层数的变化

采用类似的方法来处理其他几组的模板序列。我们已经知道了第一组滤波器尺寸增量为 6（该组每层之间的滤波器尺寸增加量），获得其他组滤波器模板大小的方法是将滤波器尺寸增加量翻倍（6，12，24，38），但是每一组的第一层滤波器模板大小的获取方法不太一样，针对第二组第一层，其滤波器尺寸相对于第一组第一层滤波器模板大小增加 6，仍采用第一组的滤波器尺寸增量，获得了第二组的首层后再根据该组滤波器增量，计算第二组其他层的滤波器尺寸，对第二组，滤波器尺寸增量为 12，因此第二组第二层的滤波器模板大小为 27。第三组第一层获得方法与第二组第一层类似，使用第二组滤波器的尺寸增量，相对于第二组第一层增加 12。获得了第三组的首层后再根据第三组的滤波器尺寸增量 24 计算第三组其他层滤波器尺寸。后面的其他组方法相同。这种方法也对应了通过式(2.12)获得的滤波器尺寸。金字塔中的滤波器尺寸如下图所示：

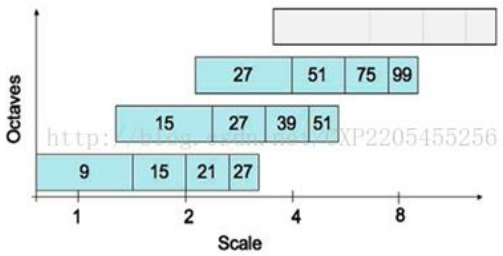

图 2-10 SURF 金字塔中盒子滤波器的尺寸变化

在通常尺度分析情况下，随着尺度的增大，被检测到的斑点数量迅速衰减。所以一般进行 3-4 组就可以了，与此同时，为了减少运算量，提高计算的速度，可以考虑在滤波时，将采样间隔设为 2。

对于尺寸为 L 的模板，当用它与积分图运算来近似二维高斯核的滤波时，获取近似对应的二维高斯核的尺度参数至关重要，尤其是在后面计算描述子时，用于计算邻域的半径时，这个就可以通过使用式(2.13)中的第三个式子计算获得。Bay 建议将尺度空间分为四组，每组中包括四层。为了保持尺度空间的连续性，SURF 算法尺度空间相邻组中有部分层重叠，同时每组中的盒子滤波器的尺寸都是逐渐增大的。这一点 在非极大抑制过程中可以体现。

在 SURF 算法的尺度空间中，每一组中任意一层包括 D_{xx}、D_{yy} 和 D_{xy} 三种盒子滤波器。对一幅输入图像进行滤波后通过 Hessian 行列式(2.9)计算公式可以得到对应尺度坐标下的 Hessian 行列式的值，所有 Hessian 行列式值构成一幅 Hessian 行列式图像。如图 2-11 所示。

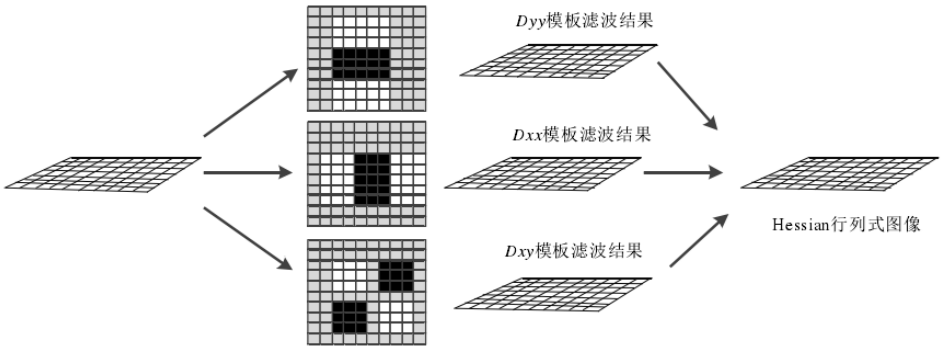

图 2-11 一幅 Hessian 行列式图像的产生过程

一幅灰度图像经过前文获得的尺度空间中不同尺寸的盒子滤波器的滤波处理后，可以生成多幅 Hessian 行列式图像，从而构成了图像金字塔。

2.6 关键点精确定位

2.6.1 非极大抑制

与 SIFT 对 DoG 金字塔的处理方式相同在每一组中选取相邻的三层 Hessian 行列式图像，对于中间层的每一个 Hessian 行列式值都可以做为待比较的点，在

空间中选取该点周围的 26 个点进行比较大小，若该点大于其他 26 个点，则该点为特征点。从上诉过程可以知道，当尺度空间每组由四层构成时，非极大值抑制只会在中间两层进行，相邻的组之间不进行比较。如图 2-12 所示。

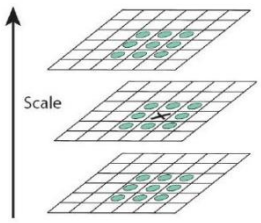

图 2-12 非极大抑制过程图

第一组的滤波器模板大小分别为（9、15、21、27），通过非极大抑制可以获得滤波器模板大小为 15 和 21 层的斑点（关键点），对于第二组，滤波器模板大小分别为（15、27、39、51），可以获得关键点层的滤波器模板大小为 27、39，同理第三组可以获得关键点层的滤波器模板大小为 51、75。这样从前三组获得的关键点的层的滤波器模板的大小为 15、21、39、51、75，体现了获取关键点尺度的连续性。

2.6.2 设定 Hessian 矩阵行列式的阈值

低于 Hessian 行列式阈值的点不能作为最终的特征点。在实际选择阈值时，根据实际应用中对特征点数量和精确度的要求改变阈值。阈值越大，得到的特征点的鲁棒性越好。在处理场景简单的图像时，其阈值可以适当的调低。在复杂的图像中，图像经旋转或者模糊后特征点变化的数量较大，测试需要适当提高阈值。

2.6.3 准确定位关键点

由于我们通过前文描述的方法，获得的关键点是在离散空间中的关键点，因此也需要通过拟合准确定位关键点的位置。这里与 SIFT 算法处理方式相同，每个特征点包含三个信息 $H(x,y;\sigma)$，即位置与尺度。使用子像元插值的方法，利用泰勒展开式，可以得到如下表达式：

$$H(\boldsymbol{x}) = H + \frac{\partial H^T}{\partial x}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\frac{\partial^2 H}{\partial x^2}\boldsymbol{x} \tag{2.14}$$

求导并令其为 0，可得

$$\tilde{\boldsymbol{x}} = -\frac{\partial H}{\partial \boldsymbol{x}}^{-1}\frac{\partial H}{\partial \boldsymbol{x}} \tag{2.15}$$

求得 $\tilde{\boldsymbol{x}} = \left(\triangle x, \triangle y, \triangle z\right)^{\mathrm{T}}$，即在三个方向的偏移量

通过以上的方法，即找到了每个特征点的位置。

第三章 关键点描述

3.1 确定关键点主方向

为了保证旋转不变性，首先要为特征点分配主方向。SIFT 特征点主方向分配是采用在特征点邻域内统计其梯度直方图，取直方图 bin 值最大的以及超过最大 bin 值 80%的那些方向作为特征点的主方向。而在 SURF 中，采用的是统计特征点圆形邻域内的 Harr 小波特征。以特征点为中心，6σ 为半径的圆域，统计所有像素点的 Harr 小波响应。算法上就是采用图 3-1 的模板对图像卷积运算，作为每个像素点水平与垂直方向 Harr 小波响应。Harr 小波模板的长度为 4σ。尺度 σ 根据当前模板的大小获得，即式(2.13)的第三个式子，这里重写该式。

$$\sigma = 1.2\times\frac{L}{9} = 1.2\times\frac{l}{3} = 1.2\times\frac{3\times\left(2^{o+1}\left(s+1\right)+1\right)}{3} \tag{3.1}$$

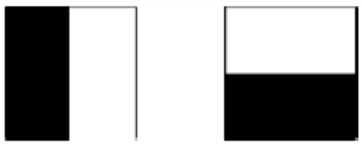

图 3-1 Harr 小波模板（黑色 1，白色-1）

圆域内的每个像素都计算出 Harr 小波水平响应与垂直响应值，同时乘以对应位置的高斯权重（$\sigma_{Gaussian} = 2.5\sigma$）。

为了求取主方向，采用一个张角为 60 度的扇形滑动窗口，根据式(3.2)和式(3.3)计算其区域内的 Harr 小波水平与垂直方向的响应 *dx*、*dy* 之和。

$$m_w = \sum_w dx + \sum_w dy \tag{3.2}$$

$$\theta_w = \arctan\left(\frac{\sum_w dx}{\sum_w dy}\right) \tag{3.3}$$

滑动扇形窗口，得到最大的响应区域对应的方向即为此特征点的主方向，如图 3-2 所示。

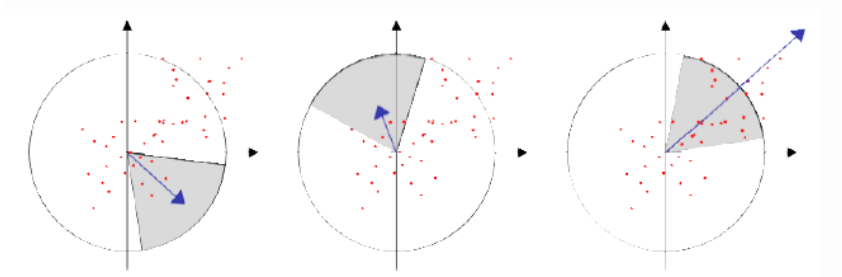

图 3-2 选取关键点主方向

3.2 关键点特征描述

生成特征点描述子与确定特征点方向有些类似，它需要计算图像的 Haar 小波响应。不过，与主方向的确定不同的是，这次我们不是使用一个圆形区域，而是在一个矩形区域来计算 Haar 小波响应。以特征点为中心，利用上一节讨论得到的主方向，沿主方向将 $20\sigma \times 20\sigma$ 的图像划分为 4×4 个子块，每个子块利用尺寸 2σ 的 Harr 模板进行响应值进行响应值计算，并统计每个子块中的 $\sum dx$ 、$\sum |dx|$ 、$\sum dy$ 、$\sum |dy|$ 然后对响应值进行统计，形成特征矢量。同时对 dx 、dy 进行高斯加权计算（$\sigma_{Gaussian} = 3.3\sigma$）。这样就有 $4\times4\times4=64$ 维的数据，如图 3-3 所示。

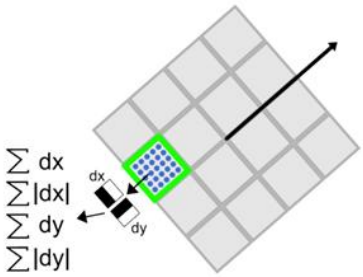

图 3-3 特征描述子表示

为了充分利用积分图像进行 Haar 小波的响应计算，我们并不直接旋转 Haar 小波模板求得其响应值，而是在积图像上先使用水平和垂直的 Haar 模板求得响应值 dy 和 dx，然后根据主方向旋转 dx 和 dy 与主方向保持一致，相应的计算公式如

下：

$$\begin{aligned} dx' &= w(-dx \times \sin(\theta) + dy \times \cos(\theta)) \\ dy' &= w(dx \times \cos(\theta) + dy \times \sin(\theta)) \end{aligned} \tag{3.4}$$

在 OpenSURF 的实现源码中采用的是另外一种方式，通过点旋转公式，把点旋转到主方向上并进行最近邻插值的对应点，公式如下：

$$\begin{aligned} x &= x_0 - j \times scale \times \sin(\theta) + i \times scale \times \cos(\theta) \\ y &= y_0 + j \times scale \times \cos(\theta) + i \times scale \times \sin(\theta) \end{aligned} \tag{3.5}$$

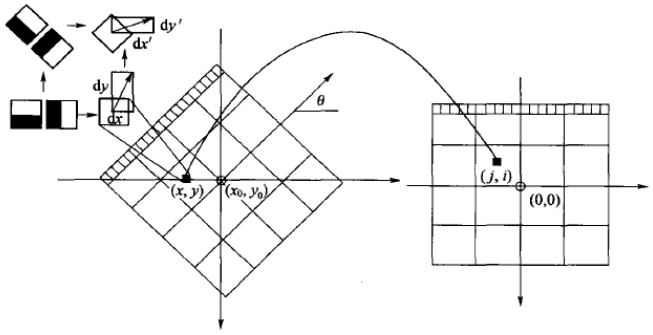

图 3-4 利用积分图像进行 Haar 小波响应计算示意图，左边为旋转后的图像，右边为旋转前的图像

SURF 描述子不仅具有尺度和旋转不变性，而且对光照的变化也具有不变性。小波响应本身就具有亮度不变性，而对比度的不变性则是通过将特征矢量进行归一化来实现。

图 3-5 为三种不同图像模式的子块得到的不同的结果。对于实际图像的描述子，我们可以认为它们是由这三种不同模式图像的描述子组合而成的。

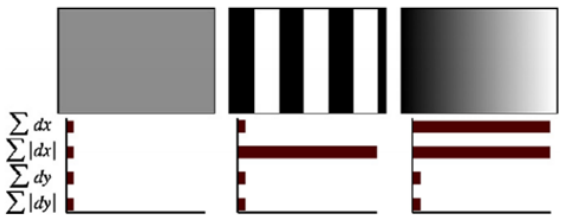

图 3-5 不同的图像密度模式得到的不同的描述子结果

第四章 关键点匹配

为了实现快速匹配，SURF 在特征矢量中增加了一个新的变量，即特征点的拉普拉斯响应正负号。在特征点检测时，将 Hessian 矩阵的迹的正负号记录下来，作为特征矢量中的一个变量。这样做并不增加运算量，因为特征点检测时已经对 Hessian 矩阵的迹进行了计算。在特征匹配时，这个变量可以有效地节省搜索的时间，因为只有两个具有相同正负号的特征点才有可能匹配，对于正负号不同的特征点就不进行相似性计算。

简单地说，我们可以根据特征点的响应值符号，将特征点分成两组，一组是具有拉普拉斯正响应的特征点，一组是具有拉普拉斯负响应的特征点，匹配时，只有符号相同组中的特征点才能进行相互匹配。显然，这样可以节省特征点匹配的时间。如图 4-1 所示。

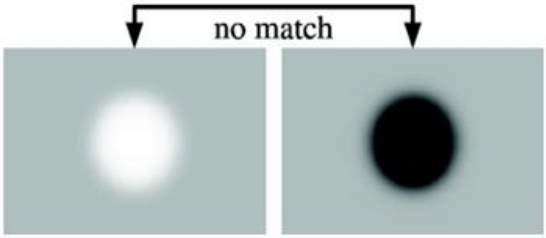

图 4-1 黑背景下的亮斑和白背景下的黑斑 因为它们的拉普拉斯响应正负号不同，不会对它们进行匹配

第五章 SURF 和 SIFT 对比总结

（1）在生成尺度空间方面，SIFT 算法利用的是差分高斯金字塔与不同层级的空间图像相互卷积生成。

SURF 算法采用的是不同尺度的 box filters 与原图像卷积

（2）在特征点检验时，SIFT 算子是先对图像进行非极大值抑制，再去除对比度较低的点。然后通过 Hessian 矩阵去除边缘的点

而 SURF 算法是先通过 Hessian 矩阵来检测候选特征点，然后再对非极大值的点进行抑制

（3）在特征向量的方向确定上，SIFT 算法是在正方形区域内统计梯度的幅值的直方图，找到最大梯度幅值所对应的方向。SIFT 算子确定的特征点可以有一个或一个以上方向，其中包括一个主方向与多个辅方向。

SURF 算法则是在圆形邻域内，检测各个扇形范围内水平、垂直方向上的 Haar 小波响应，找到模值最大的扇形指向，且该算法的方向只有一个。

（4）SIFT 算法生成描述子时，是将16×16的采样点划分为4×4的区域，从而计算每个分区种子点的幅值并确定其方向，共计$4\times4\times8=128$维。

SURF 算法在生成特征描述子时将$20\sigma\times20\sigma$的正方形分割成4×4的小方格，每个子区域 25 个采样点，计算小波 Haar 响应$v=\left[\sum dx,\sum|dx|,\sum dy,\sum|dy|\right]$，一共$4\times4\times4=64$维。

综上，SURF 算法在各个步骤上都简化了一些繁琐的工作，仅仅计算了特征点的一个主方向，生成的特征描述子也与前者相比降低了维数。

参考资料：

https://www.cnblogs.com/mvision/p/7992116.html；

https://www.cnblogs.com/gfgwxw/p/9415218.html；

https://blog.csdn.net/fengye2two/article/details/79135250；

https://blog.csdn.net/cxp2205455256/article/details/41311013；

https://wenku.baidu.com/view/cf0c164f2e3f5727a5e96238.html；

https://blog.csdn.net/songzitea/article/details/16986423；

https://blog.csdn.net/ecnu18918079120/article/details/78195792；

http://www.cnblogs.com/ronny/p/4045979.html；

https://www.cnblogs.com/ronny/p/4048213.html；