【传统 CV 算法】图像局部特征点检测算法

目 录

第一章 概述

1. 图像特征概述

研究图像特征检测已经有一段时间了，图像特征检测的方法很多，又加上各种算法的变形，所以难以在短时间内全面的了解，只是对主流的特征检测算法的原理进行了学习。总体来说，图像特征可以包括颜色特征、纹理特等、形状特征以及局部特征点等。其中局部特征具有很好的稳定性，不容易受外界环境的干扰，本篇文章也是对这方面知识的一个总结。

2. 局部特征点

图像特征提取是图像分析与图像识别的前提，它是将高维的图像数据进行简化表达最有效的方式，从一幅图像的 M×N×3 的数据矩阵中，我们看不出任何信息，所以我们必须根据这些数据提取出图像中的关键信息，一些基本元件以及它们的关系。

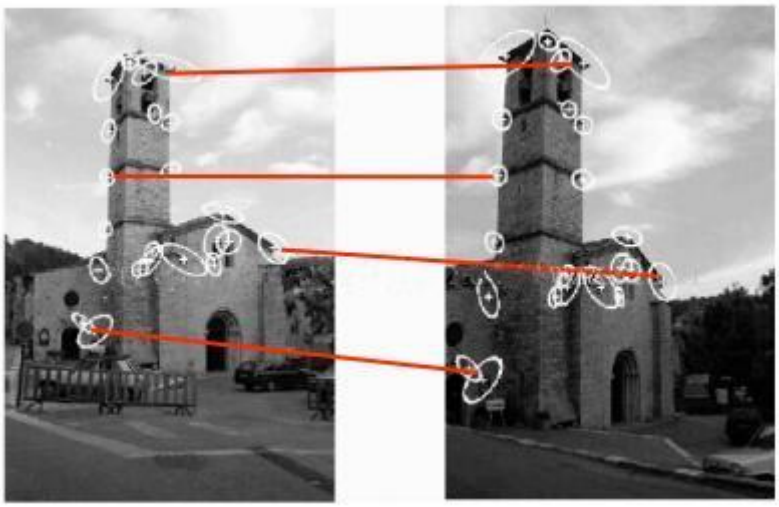
局部特征点是图像特征的局部表达，它只能反映图像上具有的局部特殊性，所以它只适合于对图像进行匹配，检索等应用。对于图像理解则不太适合。而后者更关心一些全局特征，如颜色分布，纹理特征，主要物体的形状等。全局特征容易受到环境的干扰，光照，旋转，噪声等不利因素都会影响全局特征。相比而言，局部特征点，往往对应着图像中的一些线条交叉，明暗变化的结构中，受到的干扰也少。

而斑点与角点是两类局部特征点。斑点通常是指与周围有着颜色和灰度差别的区域，如草原上的一棵树或一栋房子。它是一个区域，所以它比角点的抗噪能力要强，稳定性要好。而角点则是图像中一边物体的拐角或者线条之间的交叉部分。

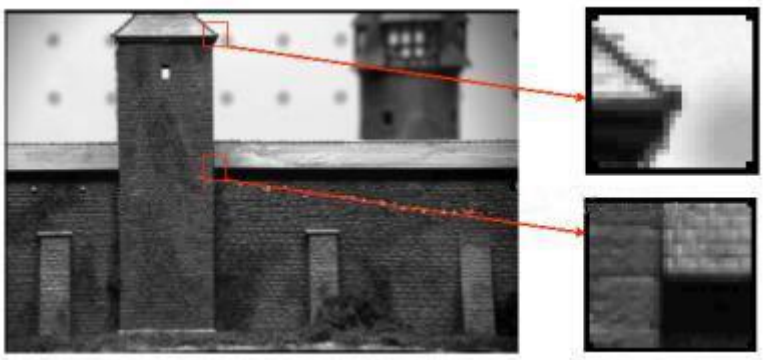
第二章 角点检测

1. 初识角点

在现实世界中，角点对应于物体的拐角，道路的十字路口、丁字路口等。下面有两幅不同视角的图像，通过找出对应的角点进行匹配。



再看下图所示，放大图像的两处角点区域：



我们可以直观的概括下角点所具有的特征：

>轮廓之间的交点；

>对于同一场景，即使视角发生变化，通常具备稳定性质的特征；

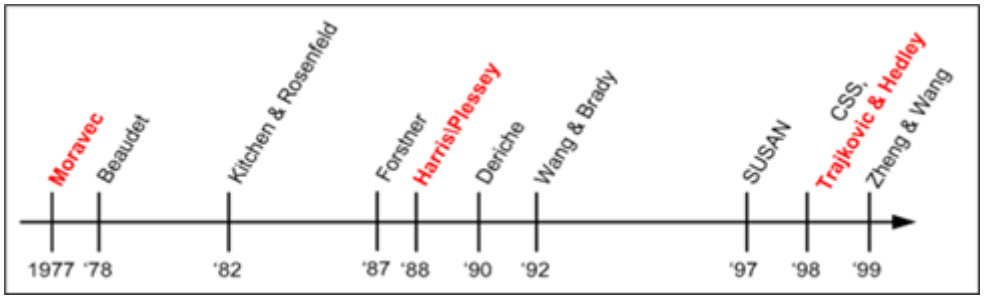
>该点附近区域的像素点无论在梯度方向上还是其梯度幅值上有着较大变化；

从图像分析的角度来定义角点可以有以下两种定义：

1. 角点可以是两个边缘的角点；
2. 角点是邻域内具有两个主方向的特征点；

前者往往需要对图像边缘进行编码，这在很大程度上依赖于图像的分割与边缘提取，具有相当大的难度和计算量，且一旦待检测目标局部发生变化，很可能导致操作的失败。早期主要有 Rosenfeld 和 Freeman 等人的方法，后期有 CSS 等方法。

基于图像灰度的方法通过计算点的曲率及梯度来检测角点，避免了第一类方法存在的缺陷，此类方法主要有 Moravec 算子、Forstner 算子、Harris 算子、SUSAN 算子等。



总体来说，对于角点检测算法而言，基本思想是使用一个固定窗口在图像上进行任意方向上的滑动，比较滑动前与滑动后两种情况，窗口中的像素灰度变化程度，如果存在任意方向上的滑动，都有着较大灰度变化，那么我们可以认为该窗口中存在角点。

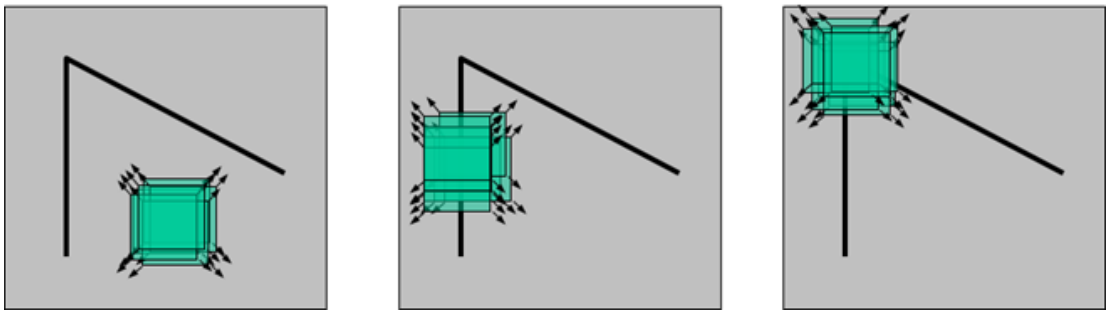
本文主要介绍的角点方法包括 Harris 角点，FAST 角点，FAST-ER 角点，SUSAN 角点等，其中 SUSAN 角点被放在“【传统 CV 算法】图像边缘检测算法”中介绍。另外，Moravec 算子和 Forstner 算子本文不作介绍，想要了解的可以前往以下博主处：

https://blog.csdn.net/gdut2015go/article/details/46669907

2. Harris 角点

2.1 基本原理

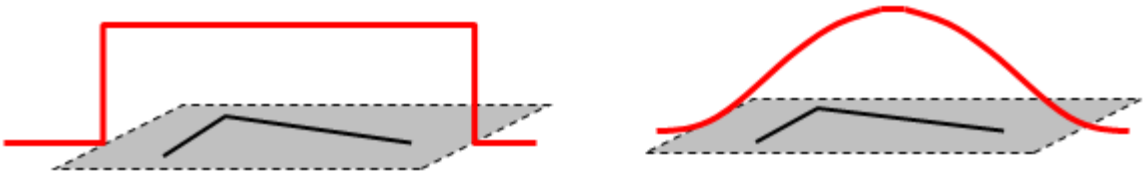
人眼对角点的识别通常是在一个局部的小区域或小窗口完成的。如果在各个方向上移动这个特征的小窗口，窗口内区域的灰度发生了较大的变化，那么就认为在窗口内遇到了角点。如果这个特定的窗口在图像各个方向上移动时，窗口内图像的灰度没有发生变化，那么窗口内就不存在角点；如果窗口在某一个方向移动时，窗口内图像的灰度发生了较大的变化，而在另一些方向上没有发生变化，那么，窗口内的图像可能就是一条直线的线段。



对于图像 I(x,y)，当在点(x,y)处平移(Δx,Δy)后的自相似性，可以通过自相关函数给出：

$$c(x,y;\Delta x,\Delta y)=\sum_{(u,v)\in W(x,y)} w(u,v)(I(u,v)-I(u+\Delta x,v+\Delta y))^2$$

其中，W(x,y)是以点 (x,y)为中心的窗口， w(u,v)为加权函数，它既可是常数，也可以是高斯加权函数。我们来理解一下这个自相关函数：[u,v]是窗口的偏移量；(x,y)是窗口内所对应的像素坐标位置，窗口有多大，就有多少个位置；w(x,y)是窗口函数，最简单情形就是窗口内的所有像素所对应的 w 权重系数均为 1，但有时候，我们会将 w(x,y)函数设定为以窗口中心为原点的二元正态分布。如果窗口中心点是角点时，移动前与移动后，该点的灰度变化应该最为剧烈，所以该点权重系数可以设定大些，表示窗口移动时，该点在灰度变化贡献较大；而离窗口中心(角点)较远的点，这些点的灰度变化几近平缓，这些点的权重系数，可以设定小点，以示该点对灰度变化贡献较小，那么我们自然想到使用二元高斯函数来表示窗口函数，这里仅是个人理解，大家可以参考下。



根据泰勒展开，对图像 I(x,y)在平移 (Δx,Δy)后进行一阶近似：

$$I(u+\Delta x,v+\Delta y)=I(u,v)+I_x(u,v)\Delta x+I_y(u,v)\Delta y+O(\Delta x^2,\Delta y^2)\approx I(u,v)+I_x(u,v)\Delta x+I_y(u,v)\Delta y$$

其中， Ix,Iy 是图像 I(x,y)的偏导数，这样的话，自相关函数则可以简化为：

$$c(x,y;\Delta x,\Delta y)\approx\sum_w(I_x(u,v)\Delta x+I_y(u,v)\Delta y)^2=[\Delta x,\Delta y]M(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

具体的简化过程如下：

$$\begin{aligned}&\sum[I(x+u,y+v)-I(x,y)]^2\\ &\approx\sum[I(x,y)+uI_x+vI_y-I(x,y)]^2\\ &=\sum u^2I_x^2+2uvI_xI_y+v^2I_y^2\\ &=\sum\begin{bmatrix}u & v\end{bmatrix}\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}\\ &=\begin{bmatrix}u & v\end{bmatrix}\left(\sum\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}\right)\begin{bmatrix}u\\ v\end{bmatrix}\end{aligned}$$

其中，

$$M(x,y)=\sum_w\begin{bmatrix}I_x(x,y)^2 & I_x(x,y)I_y(x,y)\\ I_x(x,y)I_y(x,y) & I_y(x,y)^2\end{bmatrix}=\begin{bmatrix}\sum_w I_x(x,y)^2 & \sum_w I_x(x,y)I_y(x,y)\\ \sum_w I_x(x,y)I_y(x,y) & \sum_w I_y(x,y)^2\end{bmatrix}=\begin{bmatrix}A & C\\ C & B\end{bmatrix}$$

也就是说图像 I(x,y)在点(x,y)处平移 (Δx,Δy)后的自相关函数可以近似为二项函数：

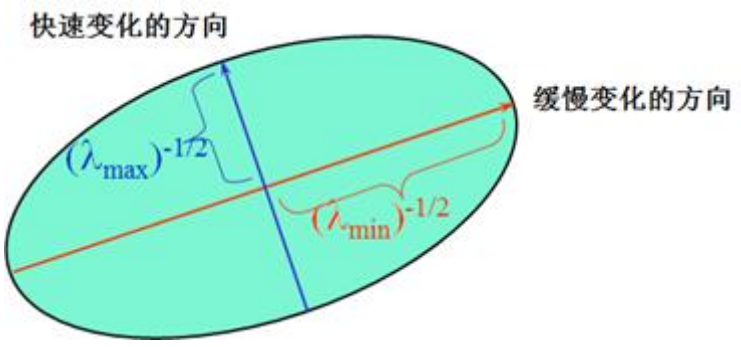
$$c(x,y;\Delta x,\Delta y)\approx A\Delta x^2+2C\Delta x\Delta y+B\Delta y^2$$

其中，

$$A=\sum_w I_x^2,B=\sum_w I_y^2,C=\sum_w I_xI_y$$

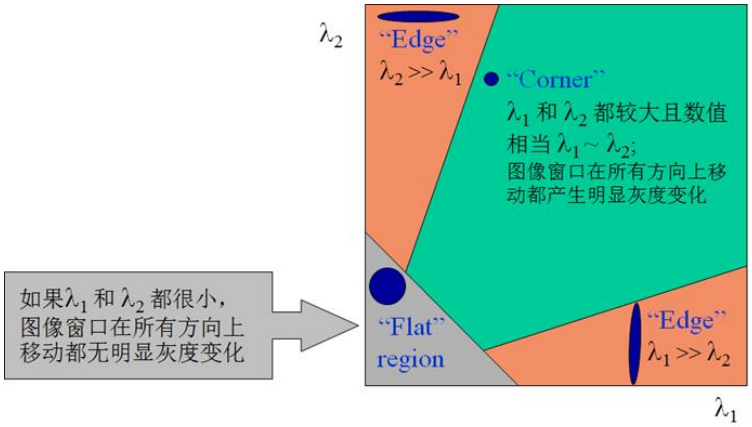
二次项函数本质上就是一个椭圆函数。椭圆的扁率和尺寸是由 M(x,y)的特征值 λ1、λ2 决定的，椭贺的方向是由 M(x,y)的特征矢量决定的，如下图所示，椭圆方程为：

$$[\Delta x,\Delta y]M(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}=1$$



椭圆函数特征值与图像中的角点、直线（边缘）和平面之间的关系如下图所示。共可分为三种情况：

- 图像中的直线。一个特征值大，另一个特征值小，λ1≫λ2 或 λ2≫λ1。自相关函数值在某一方向上大，在其他方向上小。
- 图像中的平面。两个特征值都小，且近似相等；自相关函数数值在各个方向上都小。
- 图像中的角点。两个特征值都大，且近似相等，自相关函数在所有方向都增大。



根据二次项函数特征值的计算公式，我们可以求 M(x,y)矩阵的特征值。但是 Harris 给出的角点差别方法并不需要计算具体的特征值，而是计算一个角点响应值 R 来判断角点。R 的计算公式为：

$$R = detM - \alpha(traceM)^2$$

式中，detM 为矩阵 $M = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$ 的行列式；traceM 为矩阵 M 的直迹；α 为经常常数，取值范围为 0.04~0.06。事实上，特征是隐含在 detM 和 traceM 中的，因为，

$$detM = \lambda_1\lambda_2 = AC - B^2$$

$$traceM = \lambda_2 + \lambda_2 = A + C$$

2.2 Harris 角点算法实现

根据上述讨论，可以将 Harris 图像角点检测算法归纳如下，共分以下五步：

1. 计算图像 I(x,y)在 X 和 Y 两个方向的梯度 Ix、Iy。

$$I_x = \frac{\partial I}{\partial x} = I \otimes (-1\ 0\ 1),\ I_y = \frac{\partial I}{\partial x} = I \otimes (-1\ 0\ 1)^T$$

2. 计算图像两个方向梯度的乘积。

$$I_x^2 = I_x \cdot I_y,\ I_y^2 = I_y \cdot I_y,\ I_{xy} = I_x \cdot I_y$$

3. 使用高斯函数对I_x^2、I_y^2和I_{xy}进行高斯加权（取 σ=1），生成矩阵 M 的元素 A、B 和 C。

$$A = g(I_x^2) = I_x^2 \otimes w,\ C = g(I_y^2) = I_y^2 \otimes w,\ B = g(I_{x,y}) = I_{xy} \otimes w$$

4. 计算每个像素的 Harris 响应值 R，并对小于某一阈值 t 的 R 置为零。

$$R = \{R : detM - \alpha(traceM)^2 < t\}$$

5. 在 3×3 或 5×5 的邻域内进行非最大值抑制，局部最大值点即为图像中的角点。

2.3 Harris 角点的性质

1. 参数 α 对角点检测的影响

假设已经得到了矩阵 M 的特征值 λ1≥λ2≥0，令 λ2=kλ1,0≤k≤1。由特征值与矩阵 M 的直迹和行列式的关系可得：

$$detM = \prod_i \lambda_i \quad traceM = \sum_i \lambda_i$$

从而可以得到角点的响应

$$R = \lambda_2\lambda_2 = \alpha(\lambda_2 + \lambda_2)^2 = \lambda^2(k - \alpha(1+k)^2)$$

假设 R≥0，则有：

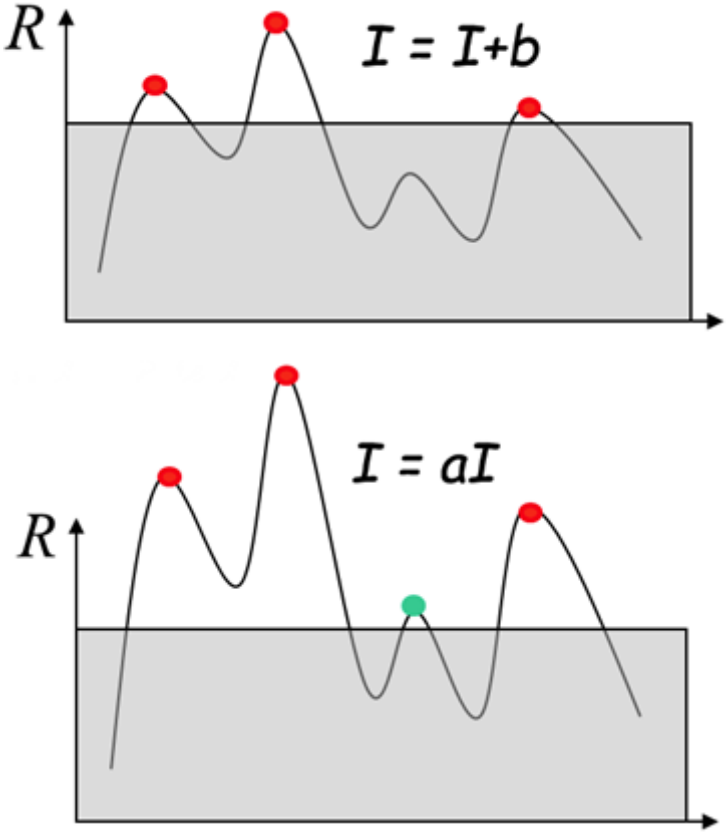
$$0 \le \alpha \le \frac{k}{(1+k)^2} \le 0.25$$

对于较小的 k 值，$R \approx \lambda^2(k-\alpha)$,α<k。

由此，可以得出这样的结论：增大 α 的值，将减小角点响应值 R，降低角点检测的灵性，减少被检测角点的数量；减小 α 值，将增大角点响应值 R，增加角点检测的灵敏性，增加被检测角点的数量。

2. Harris 角点检测算子对亮度和对比度的变化不敏感

这是因为在进行 Harris 角点检测时，使用了微分算子对图像进行微分运算，而微分运算对图像密度的拉升或收缩和对亮度的抬高或下降不敏感。换言之，对亮度和对比度的仿射变换并不改变 Harris 响应的极值点出现的位置，但是，由于阈值的选择，可能会影响角点检测的数量。

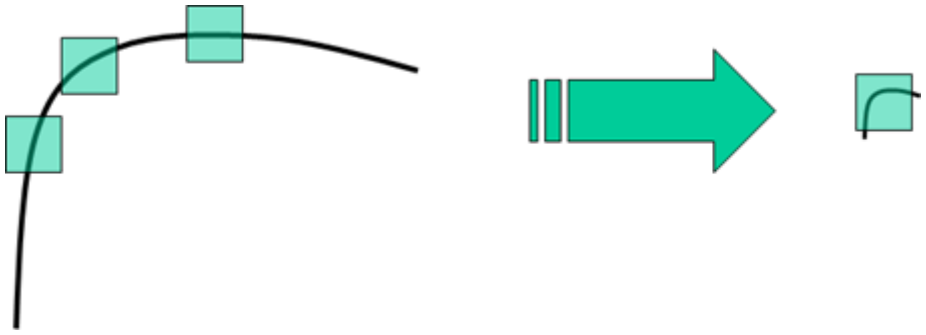


3. Harris 角点检测算子具有旋转不变性

Harris 角点检测算子使用的是角点附近的区域灰度二阶矩矩阵。而二阶矩矩阵可以表示成一个椭圆，椭圆的长短轴正是二阶矩矩阵特征值平方根的倒数。当特征椭圆转动时，特征值并不发生变化，所以判断角点响应值 R 也不发生变化，由此说明 Harris 角点检测算子具有旋转不变性。

4. Harris 角点检测算子不具有尺度不变性

如下图所示，当右图被缩小时，在检测窗口尺寸不变的前提下，在窗口内所包含图像的内容是完全不同的。左侧的图像可能被检测为边缘或曲线，而右侧的图像则可能被检测为一个角点。



2.4 *多尺度 Harris 角点

虽然 Harris 角点检测算子具有部分图像灰度变化的不变性和旋转不变性，但它不具有尺度不变性。但是尺度不变性对图像特征来说至关重要。人们在使用肉眼识别物体时，不管物体远近，尺寸的变化都能认识物体，这是因为人的眼睛在辨识物体时具有较强的尺度不变性。下面将 Harris 角点检测算子与高斯尺度空间表示相结合，使用 Harris 角点检测算子具有尺度的不变性。

仿照 Harris 角点检测中二阶矩的表示方法，使用 M=μ(x,σI,σD)为尺度自适应的二阶矩：

$$\boldsymbol{M} = \mu(x, \sigma_I, \sigma_D) = \sigma_D^2 g(\sigma_I) \otimes \begin{bmatrix} L_x^2(x, \sigma_D) & L_x L_y(x, \sigma_D) \\ L_x L_y(x, \sigma_D) & L_y^2(x, \sigma_D) \end{bmatrix}$$

其中，$g(\sigma_I)$表示尺度为$sigma_I$的高斯卷积核，x 表示图像的位置。与高斯测度空间类似，使用 L(x)表示经过高斯平滑后的图像，符号⊗表示卷积，$L_x(x, \sigma_D)$和$L_y(x, \sigma_D)$表示对图像使用高斯 $g(\sigma_D)$函数进行平滑后，在 x 或 y 方向取其微分的结果，即 $L_x=\partial_x L$ 和 $L_y=\partial_y L$。通常将σ_I称为积分尺度，它是决定 Harris 角点当前尺度的变量，σ_D为微分尺度或局部尺度，它是决定角点附近微分值变化的变量。显然，积分尺度σ_I应该大于微分尺度σ_D。

2.5 *多尺度 Harris 角点实现

首先，检测算法从预先定义的一组尺度中进行积分尺度搜索，这一组尺度定义为

$$\sigma_1 \ldots \sigma_n = \sigma_0 \ldots k^n \sigma_0$$

一般情况下使用 k=1.4。为了减少搜索的复杂性，对于微分尺度σ_D的选择，我们采用在积分尺度的基础上，乘以一个比例常数，即$\sigma_D=s\sigma_I$，一般取 s=0.7。这样，通常使用积分和微分的尺度，便可以生成 $\mu(x, \sigma_I, \sigma_D)$，再利用 Harris 角点判断准则，对角点进行搜索，具体可以分两步进行。

1. 与 Harris 角点搜索类似，对于给定的尺度空间值σ_D，进行如下角点响应值计算和判断：

$$cornerness = det(\mu(x, \sigma_n) - \alpha trace^2(\mu(x, \sigma_n))) > threshold_H$$

2. 对于满足 1 中条件的点，在点的 8 邻域内进行角点响应最大值搜索（即非最大值抑制）出在 8 邻域内角点响应最大值的点。对于每个尺度 $\sigma_n(1,2,\ldots,n)$都进行如上搜索。

由于位置空间的候选点并不一定在尺度空间上也能成为候选点，所以，我们还要在尺度空间上进行搜索，找到该点的所谓特征尺度值。搜索特征尺度值也分两步。

1. 对于位置空间搜索到的每个候选点，进行拉普拉斯响应计算，并满足其绝对值大于给定的阈值条件：

$$F(x, \sigma_n) = \sigma_n^2 |L_{xx}(x, \sigma_n) + L_{yy}(x, \sigma_n)| \geq threshold_L$$

2. 与邻近的两个尺度空间的拉普拉斯响应值进行比较，使其满足：

$$F(x,\sigma_n) > F(x,\sigma_l), \quad l \in \{n-1.n+1\}$$

满足上述条件的尺度值就是该点的特征尺度值。这样，我们就找到了在位置空间和尺度空间都满足条件的 Harris 角点。

2.6 *更多的讨论

在上面描述的 Harris 角点具有光照不变性、旋转不变性、尺度不变性，但是严格意义上来说并不具备仿射不变性。Harris-Affine 是一种新颖的检测仿射不变特征点的方法，可以处理明显的仿射变换，包括大尺度变化和明显的视角变化。Harris-Affine 主要是依据了以下三个思路：

1. 特征点周围的二阶矩的计算对区域进行的归一化，具有仿射不变性；
2. 通过在尺度空间上归一化微分的局部极大值求解来精化对应尺度；
3. 自适应仿射 Harris 检测器能够精确定位牲点；

这篇文章不对 Harris-Affine 作进一步的描述，有时间会对这一算法做专门的分析，有兴趣的可以参考原论文：Scale & Affine Invariant Interest Point Detectors.

2.7 *Shi-Tomasi 算法

Shi-Tomasi 算法是 Harris 算法的改进。Harris 算法最原始的定义是将矩阵 M 的行列式值与 M 的迹相减，再将差值同预先给定的阈值进行比较。后来 Shi 和 Tomasi 提出改进的方法，若两个特征值中较小的一个大于最小阈值，则会得到强角点。

具体来说，对自相关矩阵 M 进行特征值分析，产生两个特征值 (λ_0, λ_1) 和两个特征方向向量。因为较大的不确定度取决于较小的特征值，也就是 $a_0^{-1/2}$ ，所以通过寻找最小特征值的最大值来寻找好的特征点也就解释的通了。

Shi 和 Tomasi 的方法比较充分，并且在很多情况下可以得到比使用 Harris 算法更好的结果。

3. FAST 角点

3.1 FAST 算法原理

Edward Rosten 和 Tom Drummond 在 2006 年发表的“Machine learning for high-speed corner detection[2]”文章中提出了一种 FAST 特征，并在 2010 年对这篇论文作了小幅度的修改后重新发表[3]。FAST 的全称为 Features From Accelerated Segment Test。Rosten 等人将 FAST 角点定义为：若某像素点与其周围领域内足够多的像素点处于不同的区域，则该像素点可能为角点。也就是某些属性与众不同，考虑灰度图像，即若该点的灰度值比其周围领域内足够多的像素点的灰度值大或者小，则该点可能为角点。

3.2 FAST 算法步骤

1. 从图片中选取一个像素 P，下面我们将判断它是否是一个特征点。我们首先把它的亮度值设为 I_p。
2. 设定一个合适的阈值 t。
3. 考虑以该像素点为中心的一个半径等于 3 像素的离散化的 Bresenham 圆，这个圆的边界上有 16 个像素（如图 1 所示）。

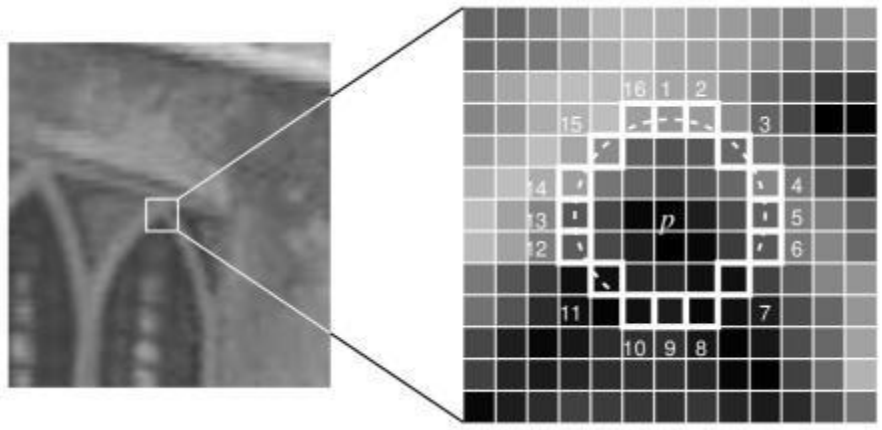


图 1 FAST 特征点示意图

4. 现在，如果在这个大小为 16 个像素的圆上有 n 个连续的像素点，它们的像素值要么都比 I_p+t 大，要么都比 I_p-t 小，那么它就是一个角点。(如图 1 中的白色虚线所示)。n 的值可以设置为 12 或者 9，实验证明选择 9 可能会有更好的效果。

上面的算法中，对于图像中的每一个点，我们都要去遍历其邻域圆上的 16 个点的像素，效率较低。我们下面提出了一种高效的测试（high-speed test）来快速排除一大部分非角点的像素。该方法仅仅检查在位置 1，9，5 和 13 四个位置的像素，首先检测位置 1 和位置 9，如果它们都比阈值暗或比阈值亮，再检测位置 5 和位置 13。如果 P 是一个角点，那么上述四个像素点中至少有 3 个应该必须都大于 I_p+t 或者小于 I_p-t，因为若是一个角点，超过四分之三圆的部分应该满足判断条件。如果不满足，那么 p 不可能是一个角点。对于所有点做

上面这一部分初步的检测后，符合条件的将成为候选的角点，我们再对候选的角点，做完整的测试，即检测圆上的所有点。

上面的算法效率实际上是很高的，但是有点一些缺点：

1. 当我们设置 n<12 时就不能使用快速算法来过滤非角点的点；
2. 检测出来的角点不是最优的，这是因为它的效率取决于问题的排序与角点的分布；
3. 对于角点分析的结果被丢弃了；
4. 多个特征点容易挤在一起。

3.3 使用机器学习做一个角点分类器

1. 首先选取你进行角点提取的应用场景下很多张的测试图片。
2. 运行 FAST 角点检测算法来获取测试图片集上的所有角点特征。
3. 对于每个角点，我们把它邻域圆上的 16 个点存储下来保存在一个 vector 内，处理所有步骤 2 中得到的角点，并把它们存储在 P 中。
4. 对于图像上的点 p,它周围邻域圆上位置为 x,$x \in \{1 \ldots 16\}$的点表示为 $p \rightarrow x$，可以用下面的判断公式将该点 $p \rightarrow x$ 分为 3 类：

$$S_{p \rightarrow x} = \begin{cases} d, & I_{p \rightarrow x} \leq I_p - t & (darker) \\ s, & I_p - t \leq I_{p \rightarrow x} < I_p + t & (similar) \\ b, & I_p + t \leq I_{p \rightarrow x} & (brighter) \end{cases}$$

5. 设 P 为训练图像集中所有像素点的集合，我们任意 16 个位置中的一个位置 x，可以把集合 P 分为三个部分 P_d,P_s 和 P_b，其中 P_d 的定义如下，P_s 和 Pb 的定义与其类似

$$P_b = \{p \in P : S_{p \rightarrow x} = b\}$$

换句话说，对于任意给定的位置 x，它都可以把所有图像中的点分为三类，第一类 P_d 包括了所有位置 x 处的像素在阈值 t 下暗于中心像素，第二类 P_s 包括了所有位置 x 处的像素在阈值 t 下近似于中心像素，P_b 包括了所有位置 x 处的像素在阈值 t 下亮于中心像素。

6. 定义一个新的布尔变量 K_p，如果 p 是一个角点，那些 K_p 为真，否则为假。
7. 使用 ID3 算法（决策树分类器）来查询每一个子集。
8. 递归计算所有的子集直到 K_p 的熵为 0;
9. 被创建的决策树就用于其他图片的 FAST 检测。

3.4 非极大值抑制

从邻近的位置选取了多个特征点是另一个问题，我们可以使用 Non-Maximal Suppression 来解决。

1. 为每一个检测到的特征点计算它的响应大小（score function）V。这里 V 定义为点 p 和它周围 16 个像素点的绝对偏差的和。
2. 考虑两个相邻的特征点，并比较它们的 V 值。
3. V 值较低的点将会被删除。

3.5 总结

FAST 算法比其他已知的角点检测算法要快很多倍，但是当图片中的噪点较多时，它的健壮性并不好，而且算法的效果还依赖于一个阈值 t。而且 FAST 不产生多尺度特征而且 FAST 特征点没有方向信息，这样就会失去旋转不变性。

4. *FAST-ER 角点检测子

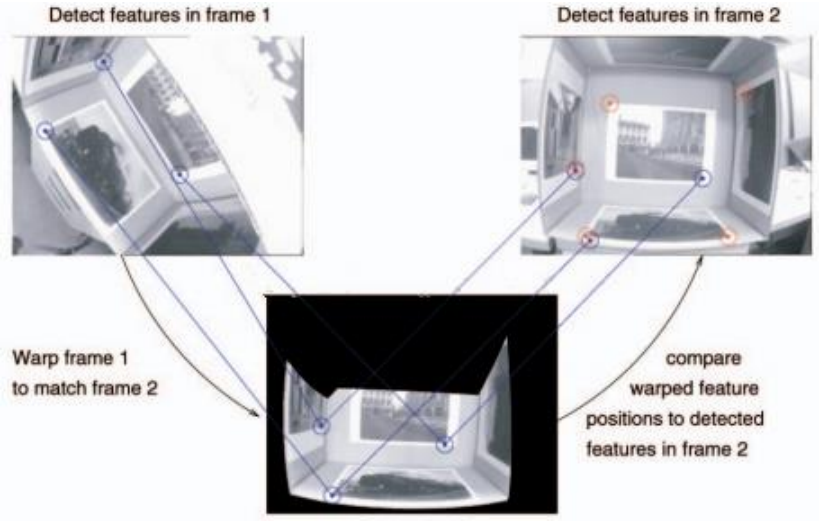
4.1 FAST-ER 算法原理

FAST-ER 是 FAST 算法原作者在 2010 年提出的，它在原来算法里提高特征点检测的重复度，重复意味着第一张图片内的检测的点，也可以在第二张图片上的相应位置被检测出来，重复度可以由如下式子定义：

$$R = \frac{N_{\text{repeated}}}{N_{\text{useful}}}$$

这里的$N_{repeated}$指第一张图片内的检测点有多少能在第二张被检测到，而N_{useful}定义为有用的特征点数。这里计算的一组图像序列的总的重复度，所以$N_{repeated}$和N_{useful}是图像序列中图像对的和。

在衡量检测的特征点是否能在第二张图像上被检测到，这也是一个大问题，通常的方法先将第一张图像内的点进行变形，使其能匹配第二张，然后再将变换的特征点位置同第二张对应位置比较，看是否被再次检测到，这里也允许一定误差，一般会在 3*3 的窗口去寻找（也就是说大概只允许一个像素的误差）。



然后实际上，由于视点变化等形变较大因素造成的形变，很有可以使角点检测偏移到不同位置，所以这个邻近区域搜索，找到匹配的特征点是很难确定的，如果搜索范围大太，一方面容易把不匹配的点检测到，另一方面计算量也增大。而如果搜索范围太小，则容易把本来匹配点给忽视了，从面减少重复率。

由于一些形变较大因素造成的形变，很难通过简单且固定的模板将所以的角点检测出来，而原来的 FAST 算法其决策树的结构是固定的三层树，并不能最优的实现区分角点（实现最优的重复率）。FAST-ER 就是针对这样的问题而提出的，其主要是通过模拟退火（也有通过最速下降法的）优化原先决策树的结构，从而提高重复率。

4.2 引入角点检测的不变性

原先一个像素点及其附近的点送往决策树进行比较时，只需要比较两个位置的点，如果这个点被检测出是角点，但在其区域发生一定旋转、变形或强度反转（白变黑，黑变白）之后，再次重新判定，很有可能被认为是非角点。

在这种情况下，为了使角点检测具有旋转、反射、强度倒转等不变性，最简单的办法就是将所以变化后的结果都计算，即不同的变换建立不同树，只要有一个树能检测出角点，即是角点。不过这样的话，计算量太大，为了减少计算复杂率，每次树被评估时，一般只需要应用 16 种变换：四个旋转方向变化（各相差 90 度），并结合反射（对左右对称及上下对称）同强度倒转，共 16 个变换操作。如果一个点能被 6 种变换中任一种的决策树视为角点，那么这个点就是角点。

由此以来我们建立了 16 棵对应不同变换的决策树。

4.3 决策树的结构优化

对于 FAST 算法来说，原来的三层决策树太过简单，不能达到最好得重复度，而重复度是关于决策树结构的非凸函数，这涉及到非凸函数的优化问题，这里许多方法，而 FAST-ER 则是通过模拟退火方法来优化决策树的。

1. 退火温度

首先考虑建立树的总成本如下：

$$k = \left(1 + \left(\frac{w_r}{r}\right)^2\right)\left(1 + \frac{1}{N}\sum_{i=1}^{N}\left(\frac{d_i}{w_n}\right)^2\right)\left(1 + \left(\frac{s}{w_s}\right)^2\right)$$

这里的 r 定义为重复率，d_i定义为第 i 帧检测的角点数，而 N 定义为总共的帧数，s 是决策树的大小（树共有多少个节点），w_s，w_n，w_r指影响因子。注意，为了提高计算效率，重复率以一个固定的阈值 t 和每帧固定数量的特征点来计算。

因为每次迭代过程，我们都需要对原树进行随机调整（启发式随机寻找最优），这些调整可以通过波尔兹曼接受准则来实施，其在第 I 次迭代中接受调整概率为 P：

$$P = e^{\frac{k_{I-1}-k_I}{T}}$$

这里的是接受标准之后的成本，而 T 是退火温度：

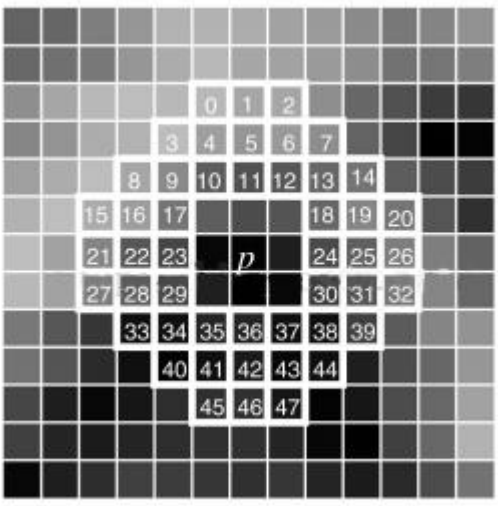
$$T = \beta e^{-\alpha \frac{I}{I_{\max}}}$$

这里的 Imax 是最大的迭代次数。

2. 扩大的判定位置

通过随机调整决策树，从而找到最优的重复率，而这个调整涉及到两个方面，一个是树节点选择分类的 x 位置，另一个是整个树的结构。

每个树的叶子确定分类（如果为 1，则是角点，而为 0，则不是），而每个节点都有一个对应的判断位置 x，FAST 是圆环上的其中一点，而在引入退火方法，我们需要给这个位置引入一个相对于中心点的随机偏移，进一步扩大判断位置的选择范围。随机偏移如下定义（共有 48 个选择位置）：



3. 决策树的调整

通过模拟退火的方式来随机调整树，首先选择树中的一个节点（不能是根结点），然后调整它（调整概率 P，上面定义过了，随着迭代次数，其越来越趋于稳定），如果：

A）如果挑选的点是叶节点，那么其有相等的概率做如下：

Ø 用一个深度为 1 随机子树节点代替，将其下继续分成三个叶结点

Ø 翻转叶子的分类（由角点变为非角点，或相反），如果叶子的类被严格限制就不进行此操作（如 s 子树的叶节点被限制为非角点，因为已经有两个位置检测为同中心相似，这样的点肯定是非角点，可以保证在阈值 t 增加时，其角点数将通常会减少）

B）如果节点，那么其也有相等概率做如下操作：

Ø 将偏移位置用新的随机偏移位置[0,47]代替，并更新下面的叶结点，前面 B）有说明。

Ø 将节点用随机叶节点代替（也要满足以上限制条件）

Ø 移除节点随机选择的分支，并用该节点下另一随机选择的分支拷贝到其位置，比如，b 分支可以由 s 分支代替。

4. 迭代过程的终止

通过对决策树的这些随机调整，树最后很可能不是原来 FAST 的简单三层树了。树调整后，再对结果应用 FAST-9，进一步确定角点，计算新的重复率，如果重复率比原来的重复率高，那么我们就将当前结果视为我们目前最优树，继续应用上面决策树调整，最终在达到一定程度重复率或迭代次数时，停止迭代过程。这个优化也可以通过不同的多个随机种子来运行。我们包含了这些调整可以让 FAST-ER 更容易地学习到相似的结构。

4.4 其作用和总结

因为每个迭代过程中，都需要对重新应用新的决策树进行检测，而且 16 个变换中每一个都需要对应一个候选树，所以这样的检测算法并不十分有效，因此，从效率上考虑，上述的算法一般用于产生训练数据，之后获得较为精确的角点检测结果，我们就可以通过原来的 FAST 算法来产生单个树。

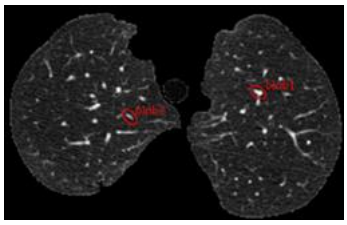
第三章 斑点检测

1.初识斑点

斑点通常是指与周围有着颜色和灰度差别的区域。在实际地图中，往往存在着大量这样的斑点，如一颗树是一个斑点，一块草地是一个斑点，一栋房子也可以是一个斑点。由于斑点代表的是一个区域，相比单纯的角点，它的稳定性要好，抗噪声能力要强，所以它在图像配准上扮演了很重要的角色。

同时有时图像中的斑点也是我们关心的区域，比如在医学与生物领域，我们需要从一些X光照片或细胞显微照片中提取一些具有特殊意义的斑点的位置或数量。

比如下图中天空的飞机、向日葵的花盘、X线断层图像中的两个斑点。



在视觉领域，斑点检测的主要思路都是检测出图像中比它周围像素灰度值大或比周围灰度值小的区域。一般有两种方法来实现这一目标：

1 基于求导的微分方法，这类的方法称为微分检测器；
2 基于局部极值的分水岭算法。

这里我们重点介绍第一种方法，主要用来检测 LOG 斑点。

2.LOG 斑点检测

2.1 斑点检测基本原理

利用高斯拉普通拉斯（Laplace of Gaussian,LOG）算子检测图像斑点是一种十分常用的方法，对于二维高斯函数：

$$G(x,y;\sigma)=\frac{1}{2\pi\sigma^2}exp(-\frac{x^2+y^2}{2\sigma^2})$$

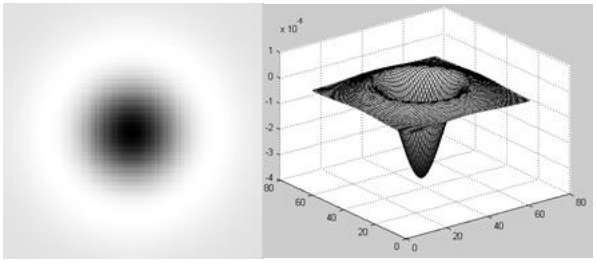
它的拉普拉斯变换为:

$$\nabla^2 g=\frac{\partial^2 g}{\partial x^2}+\frac{\partial^2 g}{\partial y^2}$$

规范化的高斯拉普变换为：

$$\nabla^2_{norm}=\sigma^2\nabla^2 g=\sigma^2(\frac{\partial^2 g}{\partial x^2}+\frac{\partial^2 g}{\partial y^2})=-\frac{1}{2\pi\sigma^2}[1-\frac{x^2+y^2}{\sigma^2}]\cdot exp(-\frac{x^2+y^2}{2\sigma^2})$$

规范化算法子在二维图像上显示是一个圆对称函数，如下图所示。我们可以用这个算子来检测图像中的斑点，并且可以通过改变σ的值，可以检测不同尺寸的二维斑点。



2.2 LOG 原理详解

其实从更直观的角度去解释为什么LOG算子可以检测图像中的斑点是：

图像与某一个二维函数进行卷积运算实际就是求取图像与这一函数的相似性。同理，图像与高斯拉普拉斯函数的卷积实际就是求取图像与高斯拉普拉斯函数的相似性。当图像中的斑点尺寸与高斯拉普拉斯函数的形状趋近一致时，图像的拉普拉斯响应达到最大。

从概率的角度解释为：假设原图像是一个与位置有关的随机变量X的密度函数，而LOG为随机变量Y的密度函数，则随机变量X+Y的密度分布函数即为两个函数的卷积形式。如果想让X+Y能取到最大值，则X与Y能保持步调一致最好，即X上升时，Y也上升，X最大时，Y也最大。

那么LOG算子是怎么被构想出来的呢？

事实上我们知道Laplace可以用来检测图像中的局部极值点，但是对噪声敏感，所以在我们对图像进行Laplace卷积之前，我们用一个高斯低通滤波对图像进行卷积，目标是去除图像中的噪声点。这一过程 可以描述为:

先对图像f(x,y)用方差为σ的高斯核进行高斯滤波，去除图像中的噪点。

$$L(x,y;\sigma) = f(x,y) * G(x,y;\sigma)$$

然后对图像的拉普拉斯图像则为：

$$\nabla^2 = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$$

而实际上有下面等式：

$$\nabla^2[G(x,y) * f(x,y)] = \nabla^2[G(x,y)] * f(x,y)$$

所以，我们可以先求高斯核的拉普拉斯算子，再对图像进行卷积。也就是一开始描述的步骤。

2.3 多尺度检测

我们注意到当σ尺度一定时，只能检测对应半径的斑点，那么检测的是多大半径的斑点呢，我们可以通过对规范化的二维拉普拉斯高斯算子求导：

规范化的高斯拉普拉斯函数为：

$$\nabla^2_{norm} = -\frac{1}{2\pi\sigma^2}[1 - \frac{x^2+y^2}{\sigma^2}] \cdot exp(-\frac{x^2+y^2}{2\sigma^2})$$

求∇^2_{norm}的极点值等价于求取下式：

$$\frac{\partial(\nabla^2_{norm})}{\partial\sigma} = 0$$

得到：

$$(x^2+y^2-2\sigma^2) \cdot exp(-\frac{(x^2+y^2)}{2\sigma^2})$$

$$r^2 - 2\sigma^2 = 0$$

对于图像中的斑点，在尺度σ=r/√2时，高斯拉普拉斯响应值达到最大。同理，如果图像中的圆形斑点黑白反向，那么，它的高斯拉普拉斯响应值在σ=r/√2时达到最小。将高斯拉普拉斯响应达到峰值时的尺度σ值，称为特征尺度。

那么在多尺度的情况下，同时在空间和尺度上达到最大值（或最小值）的点就是我们所期望的斑点。对于二维图像I（x,y），计算图像在不同尺度下的离散拉普拉斯响应值，然后检查位置空间中的每个点；如果该点的拉普拉斯响应值都大小于或小于其他26个立方空间领域(9+8+9)的值，那么该点就是被检测到的图像斑点。

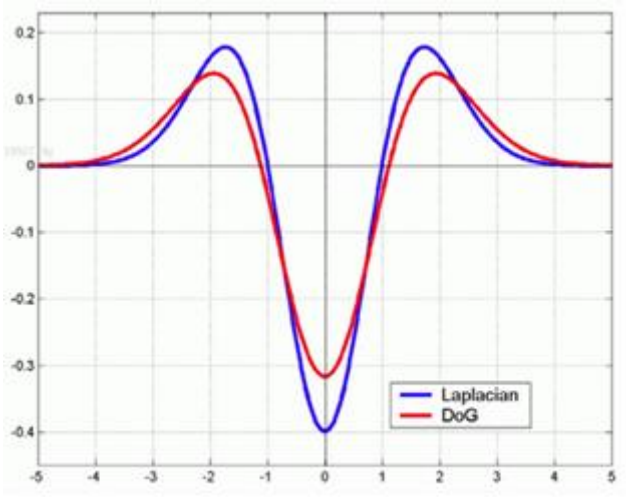
2.4 *LOG 的近似——DOG

一个与 LOG 滤波核近似的是高斯差分 DOG 滤波核，它的定义为：

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))*I(x,y)=L(x,y,k\sigma)-L(x,y,\sigma)$$

其中 k 为两个相邻尺度间的比例因子。

DOG 可以看作为 LOG 的一个近似，但是它比 LOG 的效率更高。



前面介绍的微分算子在近圆的斑点检测方面效果很好，但是这些检测算子被限定于只能检测圆形斑点，而且不能估计斑点的方向，因为 LOG 算子等都是中心对称的。如果我们定义一种二维高斯核的变形，记它在 X 方向与 Y 方向上具有不同的方差，则这种算子可以用来检测带有方向的斑点。

$$G(x,y) = \mathcal{A} \cdot exp(-[(ax^2 + 2bxy + cy^2)])$$

$$a = \frac{cos^2\theta}{2\sigma_x^2} + \frac{sin^2\theta}{2\sigma_y^2}, b = -\frac{sin2\theta}{2\sigma_x^2} + \frac{sin2\theta}{4\sigma_y^2}, c = \frac{sin^2\theta}{2\sigma_x^2} + \frac{cos^2\theta}{2\sigma_y^2}$$

其中 A 是规一性因子。

2.5 *DOH 斑点检测

除了利用高斯拉普拉斯算子检测的方法（LOG）外,斑点检测还有另外一种思路，就是利用像素点 Hessian 矩阵（二阶微分）及其行列式值的方法——DOH。

LoG 的方法已经在前文里作了详细的描述。因为二维高斯函数的拉普拉斯核很像一个斑点，所以可以利用卷积来求出图像中的斑点状的结构。

DoH 方法就是利用图像点二阶微分 Hessian 矩阵：

$$H(L)=[L_{xx}L_{xy}L_{xy}L_{yy}]$$

以及它的行列式的值 DoH(Determinant of Hessian):

$$det=\sigma_4(L_{xx}(x,y,\sigma)L_{yy}(x,y,\sigma)-L_{2xy}(x,y,\sigma))$$

Hessian 矩阵行列式的值，同样也反映了图像局部的结构信息。与 LoG 相比，DoH 对图像中的细长结构的斑点有较好的抑制作用。

无论是 LoG 还是 DoH，它们对图像中的斑点进行检测，其步骤都可以分为以下两步：

1）使用不同的 σ 生成 $(\frac{\partial^2 g}{\partial x^2}+\frac{\partial^2 g}{\partial y^2})$或$\frac{\partial^2 g}{\partial x^2}$, $\frac{\partial^2 g}{\partial y^2}$, $\frac{\partial^2 g}{\partial x \partial y}$模板，并对图像进行卷积运算；

2）在图像的位置空间与尺度空间中搜索 LoG 与 DoH 响应的峰值。

3.SIFT 斑点检测

3.1 SIFT 算法综述

SIFT（尺度不变特征变换，Scale - Invariant Feature Transform）是在计算机视觉领域中检测和描述图像中局部特征的算法，该算法于 1999 年被 David Lowe 提出，并于 2004 年进行了补充和完善。该算法应用很广，如目标识别，自动导航，图像拼接，三维建模，手势识别，视频跟踪等。不幸的是，该算法已经在美国申请了专利，专利拥有者为 Lowe 所在的加拿大不列颠哥伦比亚大学，因此我们不能随意使用它。

SIFT 算法所检测到的特征是局部的，而且该特征对于图像的尺度和旋转能够保持不变性。同时，这些特征对于亮度变化具有很强的鲁棒性，对于噪声和视角的微小变化也能保持一定的稳定性。SIFT 特征还具有很强的可区分性，它们很容易被提取出来，并且即使在低概率的不匹配情况下也能够正确的识别出目标来。因此鲁棒性和可区分性是 SIFT 算法最主要的特点。

SIFT 算法分为 4 个阶段：

1、**尺度空间极值检测**：该阶段是在图像的全部尺度和全部位置上进行搜索，并通过应用高斯差分函数可以有效地识别出尺度不变性和旋转不变性的潜在特征点来；

2、**特征点的定位**：在每个候选特征点上，一个精细的模型被拟合出来用于确定特性点的位置和尺度。而特征点的最后选取依赖的是它们的稳定程度；

3、**方向角度的确定**：基于图像的局部梯度方向，为每个特性点分配一个或多个方向角度。所有后续的操作都是相对于所确定下来的特征点的角度、尺度和位置的基础上进行的，因此特征点具有这些角度、尺度和位置的不变性；

4、**特征点的描述符**：在所选定的尺度空间内，测量特征点邻域区域的局部图像梯度，将这些梯度转换成一种允许局部较大程度的形状变形和亮度变化的描述符形式。

下面就详细阐述 SIFT 算法的这 4 个阶段。

3.2 Step1 尺度空间极值检测

特征点检测的第一步是能够识别出目标的位置和尺度，对于同一个目标在不同的视角下这些位置和尺度可以被重复的分配。并且这些检测到的位置是不随图像尺度的变化而改变的，因为它们是通过搜索所有尺度上的稳定特征得到的，所应用的工具就是被称为尺度空间的连续尺度函数。

真实世界的物体只有在一定尺度上才有意义，例如我们能够看到放在桌子上的水杯，但对于整个银河系，这个水杯是不存在的。物体的这种多尺度的本质在自然界中是普遍存在的。尺度空间就是试图在数字图像领域复制这个概念。又比如，对于某幅图像，我们是想看到叶

子还是想看到整棵树，如果是树，那么我们就应该有意识的去除图像的细节部分（如叶子、细枝等）。在去除细节部分的过程中，我们一定要确保不能引进新的错误的细节。因此在创建尺度空间的过程中，我们应该对原始图像逐渐的做模糊平滑处理。进行该操作的唯一方法是高斯模糊处理，因为已经被证实，高斯函数是唯一可能的尺度空间核。

图像的尺度空间用 $L(x, y, \sigma)$函数表示，它是由一个变尺度的高斯函数 $G(x, y, \sigma)$与图像 $I(x, y)$通过卷积产生，即

$$L(x,y,\sigma) = G(x,y,\sigma) \otimes I(x,y)$$

其中，⊗表示在 x 和 y 两个方向上进行卷积操作，而 $G(x,y,\sigma)$为

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

σ 是尺度空间因子，它决定着图像模糊平滑处理的程度。在大尺度下（σ 值大）表现的是图像的概貌信息，在小尺度下（σ 值小）表现的是图像的细节信息。因此大尺度对应着低分辨率，小尺度对应着高分辨率。(x, y)则表示在 σ 尺度下的图像像素坐标。

需要说明的是，公式 1 中的图像 $I(x, y)$是具有无限分辨率的图像，也就是说它的尺度 σ=0，即 $I(x, y) = L(x, y, 0)$。因此公式 1 得到的尺度空间图像 $L(x, y, \sigma)$是由尺度为 0 的图像 $L(x, y, 0)$生成的。很显然，尺度为 0，即无限分辨率的图像是无法获得的，Lowe 就是把初始图像的尺度设定为 0.5。那么由 $L(x, y, \sigma_1)$得到 $L(x, y, \sigma_2)$，即由尺度为 σ_1 的图像生成尺度为 σ_2 的图像的公式为：

$$L(x,y,\sigma_2) = G\left(x, y, \sqrt{\sigma_2^2 - \sigma_1^2}\right) \otimes L(x,y,\sigma_1), \qquad \sigma_2 \geq \sigma_1$$

其中，

$$G\left(x, y, \sqrt{\sigma_2^2 - \sigma_1^2}\right) = \frac{1}{2\pi(\sigma_2^2 - \sigma_1^2)} e^{-\frac{x^2+y^2}{2(\sigma_2^2-\sigma_1^2)}}$$

由于尺度为 0 的图像无法得到，因此在实际应用中要想得到任意尺度下的图像，一定是利用公式 3 生成的，即由一个已知尺度（该尺度不为 0）的图像生成另一个尺度的图像，并且一定是小尺度的图像生成大尺度的图像。

利用 LoG（高斯拉普拉斯方法，Laplacian of Gaussian），即图像的二阶导数，能够在不同的尺度下检测到图像的斑点特征，从而可以确定图像的特征点。但 LoG 的效率不高。因此 SIFT 算法进行了改进，通过对两个相邻高斯尺度空间的图像相减，得到一个 DoG（高斯差分，Difference of Gaussians）的响应值图像 D(x, y, σ)来近似 LoG：

$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) \otimes I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$

其中，k 为两个相邻尺度空间倍数的常数。

可以证明 DoG 是对 LoG 的近似表示，并且用 DoG 代替 LoG 并不影响对图像斑点位置的检测。而且用 DoG 近似 LoG 可以实现下列好处：第一是 LoG 需要使用两个方向的高斯二阶微分卷积核，而 DoG 直接使用高斯卷积核，省去了卷积核生成的运算量；第二是 DoG 保留了个高斯尺度空间的图像，因此在生成某一空间尺度的特征时，可以直接使用公式 1（或公式 3）产生的尺度空间图像，而无需重新再次生成该尺度的图像；第三是 DoG 具有与 LoG 相同的性质，即稳定性好、抗干扰能力强。

为了在连续的尺度下检测图像的特征点，需要建立 DoG 金字塔，而 DoG 金字塔的建立又离不开高斯金字塔的建立，如下图所示，左侧为高斯金字塔，右侧为 DoG 金字塔：

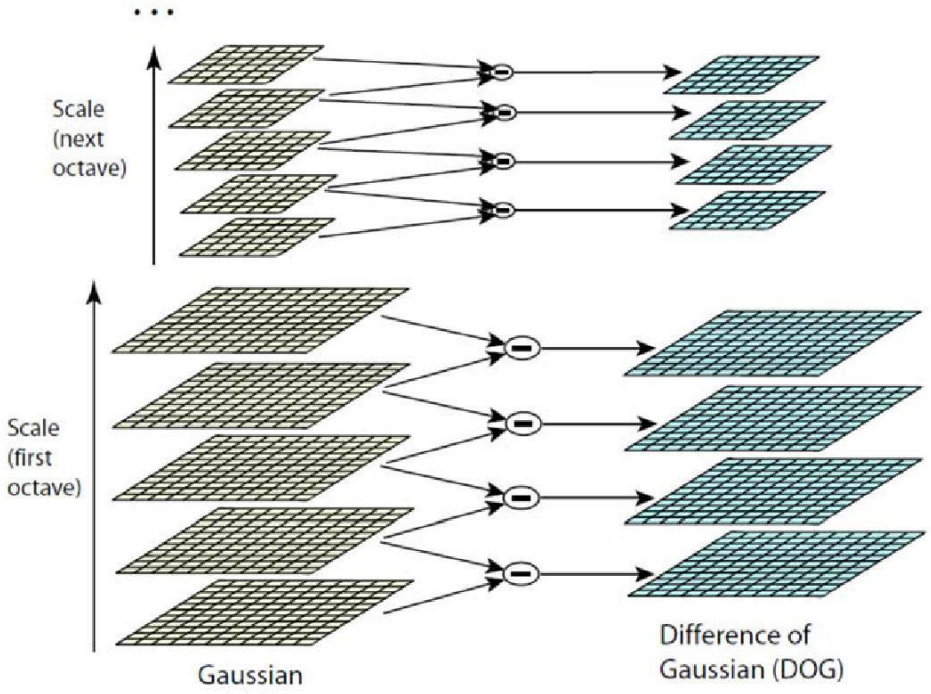


图 1 高斯金字塔和 DoG 金字塔

高斯金字塔共分 O 组（Octave），每组又分 S 层（Layer）。组内各层图像的分辨率是相同的，即长和宽相同，但尺度逐渐增加，即越往塔顶图像越模糊。而下一组的图像是由上一组图像按照隔点降采样得到的，即图像的长和宽分别减半。高斯金字塔的组数 O 是由输入图像的分辨率得到的，因为要进行隔点降采样，所以在执行降采样生成高斯金字塔时，一直到不能降采样为止，但图像太小又毫无意义，因此具体的公式为：

$$O = [\log_2 \min(X, Y) - 2]$$

其中，X 和 Y 分别为输入图像的长和宽，[ ]表示向下取整。

金字塔的层数 S 为：

$$S=s+3$$

Lowe 建议 s 为 3。需要注意的是，除了公式 7 中的第一个字母是大写的 S 外，后面出现的都是小写的 s。

高斯金字塔的创建是这样的：设输入图像的尺度为 0.5，由该图像得到高斯金字塔的第 0 组的第 0 层图像，它的尺度为 σ_0，我们称 σ_0 为基准层尺度，再由第 0 层得到第 1 层，它的尺度为 $k\sigma_0$，第 2 层的尺度为 $k^2\sigma_0$，以此类推。这里的 k 为：

$$k = 2^{\frac{1}{s}}$$

我们以 s=3 为例，第 0 组的 6（s+3=6）幅图像的尺度分别为：

$$\sigma_0，k\sigma_0，k^2\sigma_0，k^3\sigma_0，k^4\sigma_0，k^5\sigma_0$$

写成更一般的公式为：

$$\sigma = k^r\sigma_0 \qquad r\in[0,\ldots,s+2]$$

第 0 组构建完成后，再构建第 1 组。第 1 组的第 0 层图像是由第 0 组的倒数第 3 层图像经过隔点采样得到的。由公式 10 可以得到，第 0 组的倒数第 3 层图像的尺度为 $k^s\sigma_0$，k 的值代入公式 8，得到了该层图像的尺度正好为 $2\sigma_0$，因此第 1 组的第 0 层图像的尺度仍然是 $2\sigma_0$。但由于第 1 组图像是由第 0 组图像经隔点降采样得到的，因此相对于第 1 组图像的分辨率来说，第 0 层图像的尺度为 σ_0，即尺度为 $2\sigma_0$ 是相对于输入图像的分辨率来说的，而尺度为 σ_0 是相对于该组图像的分辨率来说的。这也就是为什么我们称 σ_0 为基准层尺度的原因（它是每组图像的基准层尺度）。第 1 组其他层图像的生成与第 0 组的相同。因此可以看出，第 1 组各层图像的尺度相对于该组分辨率来说仍然满足公式 10。这样做的好处就是编程的效率会提高，并且也保证了高斯金字塔尺度空间的连续性。而之所以会出现这样的结果，是因为在参数选择上同时满足公式 7、公式 8 以及对上一组倒数第 3 层图像降采样这三个条件的原因。

那么第 1 组各层图像相对于输入图像来说，它们的尺度为：

$$\sigma = 2k^r\sigma_0 \qquad r\in[0,\ldots,s+2]$$

该公式与之前公式相比较可以看出，第 1 组各层图像的尺度比第 0 组相对应层图像的尺度大了一倍。高斯金字塔的其他组的构建以此类推，不再赘述。下面给出相对于输入图像的各层图像的尺度公式：

$$\sigma(o,r) = 2^o k^r\sigma_0 \qquad o\in[0,\ldots,O\text{-}1],\quad r\in[0,\ldots,s+2]$$

其中，o 表示组的坐标，r 表示层的坐标，σ0 为基准层尺度。k 用公式 8 代入，得：

$$\sigma(o,r) = \sigma_0 2^{o+\frac{r}{s}} \qquad o \in [0,\ldots,O-1],\ r \in [0,\ldots,s+2]$$

在高斯金字塔中，第 0 组第 0 层的图像是输入图像经高斯模糊后的结果，模糊后的图像的高频部分必然会减少，因此为了最大程度的保留原图的信息量，Lowe 建议在创建尺度空间前首先对输入图像的长宽扩展一倍，这样就形成了高斯金字塔的第 - 1 组。设输入图像的尺度为 0.5，那么相对于输入图像，分辨率扩大一倍后的尺度应为 1，由该图像依次进行高斯平滑处理得到第 - 1 组的各个层的尺度图像，方法与其他组的一样。由于增加了第 - 1 组，因此公式重新写为：

$$\sigma(o,r) = \sigma_0 2^{o+\frac{r}{s}} \qquad o \in [-1,0,\ldots,O-1],\ r \in [0,\ldots,s+2]$$

DoG 金字塔是由高斯金字塔得到的，即高斯金字塔组内相邻两层图像相减得到 DoG 金字塔。如高斯金字塔的第 0 组的第 0 层和第 1 层相减得到 DoG 金字塔的第 0 组的第 0 层图像，高斯金字塔的第 0 组的第 1 层和第 2 层相减得到 DoG 金字塔的第 0 组的第 1 层图像，以此类推。需要注意的是，高斯金字塔的组内相邻两层相减，而两组间的各层是不能相减的。因此高斯金字塔每组有 $s+3$ 层图像，而 DoG 金字塔每组则有 $s+2$ 层图像。

极值点的搜索是在 DoG 金字塔内进行的，这些极值点就是候选的特征点。

在搜索之前，我们需要在 DoG 金字塔内剔除那些像素值过小的点，因为这些像素具有较低的对比度，它们肯定不是稳定的特征点。

极值点的搜索不仅需要在它所在尺度空间图像的邻域内进行，还需要在它的相邻尺度空间图像内进行，如图 2 所示。

每个像素在它的尺度图像中一共有 8 个相邻点，而在它的下一个相邻尺度图像和上一个相邻尺度图像还各有 9 个相邻点（图 2 中绿色标注的像素），也就是说，该点是在 3×3×3 的立方体内被包围着，因此该点在 DoG 金字塔内一共有 26 个相邻点需要比较，来判断其是否为极大值或极小值。这里所说的相邻尺度图像指的是在同一个组内，因此在 DoG 金字塔内，每一个组的第 0 层和最后一层各只有一个相邻尺度图像，所以在搜索极值

点时无需在这两层尺度图像内进行，从而使极值点的搜索就只在每组的中间 s 层尺度图像内进行。

搜索的过程是这样的：从每组的第 1 层开始，以第 1 层为当前层，对第 1 层的 DoG 图像中的每个点取一个 3×3×3 的立方体，立方体上下层分别为第 0 层和第 2 层。这样，搜索得到的极值点既有位置坐标（该点所在图像的空间坐标），又有尺度空间坐标（该点所在层的尺度）。当第 1 层搜索完成后，再以第 2 层为当前层，其过程与第 1 层的搜索类似，以此类推。

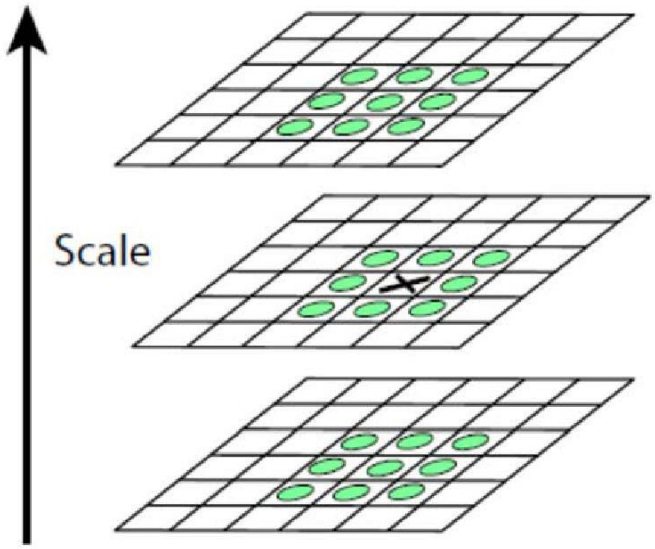


图 2 DoG 中极值点的搜索

3.3 Step2 特征点的定位

通过上一步，我们得到了极值点，但这些极值点还仅仅是候选的特征点，因为它们还存在一些不确定的因素。首先是极值点的搜索是在离散空间内进行的，并且这些离散空间还是经过不断的降采样得到的。如果把采样点拟合成曲面后我们会发现，原先的极值点并不是真正的极值点，也就是离散空间的极值点并不是连续空间的极值点。在这里，我们是需要精确定位特征点的位置和尺度的，也就是要达到亚像素精度，因此必须进行拟合处理。

我们使用泰勒级数展开式作为拟合函数。如上所述，极值点是一个三维矢量，即它包括极值点所在的尺度，以及它的尺度图像坐标，即 $\boldsymbol{X}=(x,y,\sigma)^T$，因此我们需要三维函数的泰勒级数展开式，设我们在 $\boldsymbol{X_0}=(x_0,y_0,\sigma_0)^T$ 处进行泰勒级数展开，则它的矩阵形式为：

$$f\left(\begin{bmatrix}x\\y\\\sigma\end{bmatrix}\right)\approx f\left(\begin{bmatrix}x_0\\y_0\\\sigma_0\end{bmatrix}\right)+\begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \sigma}\end{bmatrix}\left(\begin{bmatrix}x\\y\\\sigma\end{bmatrix}-\begin{bmatrix}x_0\\y_0\\\sigma_0\end{bmatrix}\right)+$$

$$\frac{1}{2}([x \quad y \quad \sigma]-[x_0 \quad y_0 \quad \sigma_0])\begin{bmatrix}\frac{\partial^2 f}{\partial x\partial x} & \frac{\partial^2 f}{\partial x\partial y} & \frac{\partial^2 f}{\partial x\partial \sigma}\\ \frac{\partial^2 f}{\partial x\partial y} & \frac{\partial^2 f}{\partial y\partial y} & \frac{\partial^2 f}{\partial y\partial \sigma}\\ \frac{\partial^2 f}{\partial x\partial \sigma} & \frac{\partial^2 f}{\partial y\partial \sigma} & \frac{\partial^2 f}{\partial \sigma\partial \sigma}\end{bmatrix}\left(\begin{bmatrix}x\\ y\\ \sigma\end{bmatrix}-\begin{bmatrix}x_0\\ y_0\\ \sigma_0\end{bmatrix}\right)$$

这是舍去高阶项的形式，而它的矢量表示形式为

$$f(X)=f(X_0)+\frac{\partial f^T}{\partial X}(X-X_0)+\frac{1}{2}(X-X_0)^T\frac{\partial^2 f}{\partial X^2}(X-X_0)$$

在这里 $\mathbf{X_0}$ 表示离散空间下的插值中心（在离散空间内也就是采样点）坐标，$\mathbf{X}$ 表示拟合后连续空间下的插值点坐标，设$\hat{\mathbf{X}}$= **X- X**$_0$，则$\hat{\mathbf{X}}$表示相对于插值中心，插值后的偏移量。因此上式经过变量变换后，又可写成

$$f(\hat{X})=f(X_0)+\frac{\partial f^T}{\partial X}\hat{X}+\frac{1}{2}\hat{X}^T\frac{\partial^2 f}{\partial X^2}\hat{X}$$

对上式求导，得：

$$\frac{\partial f(\hat{X})}{\partial \hat{X}}=\frac{\partial f^T}{\partial X}+\frac{1}{2}\left(\frac{\partial^2 f}{\partial X^2}+\frac{\partial^2 f^T}{\partial X^2}\right)\hat{X}=\frac{\partial f^T}{\partial X}+\frac{\partial^2 f}{\partial X^2}\hat{X}$$

让原始式的导数为 0，即上式=0，就可得到极值点下的相对于插值中心 $\boldsymbol{X_0}$ 的偏移量：

$$\hat{X}=-\frac{\partial^2 f^{-1}}{\partial X^2}\frac{\partial f}{\partial X}$$

把公式 19 得到的极值点带入公式 17 中，就得到了该极值点下的极值：

$$\begin{aligned}f(\hat{X}) &= f(X_0)+\frac{\partial f^T}{\partial X}\hat{X}+\frac{1}{2}\left(-\frac{\partial^2 f^{-1}}{\partial X^2}\frac{\partial f}{\partial X}\right)^T\frac{\partial^2 f}{\partial X^2}\left(-\frac{\partial^2 f^{-1}}{\partial X^2}\frac{\partial f}{\partial X}\right)\\ &= f(X_0)+\frac{\partial f^T}{\partial X}\hat{X}+\frac{1}{2}\frac{\partial f^T}{\partial X}\frac{\partial^2 f^{-T}}{\partial X^2}\frac{\partial^2 f}{\partial X^2}\frac{\partial^2 f^{-1}}{\partial X^2}\frac{\partial f}{\partial X}\\ &= f(X_0)+\frac{\partial f^T}{\partial X}\hat{X}+\frac{1}{2}\frac{\partial f^T}{\partial X}\frac{\partial^2 f^{-1}}{\partial X^2}\frac{\partial f}{\partial X}\\ &= f(X_0)+\frac{\partial f^T}{\partial X}\hat{X}+\frac{1}{2}\frac{\partial f^T}{\partial X}(-\hat{X})\\ &= f(X_0)+\frac{1}{2}\frac{\partial f^T}{\partial X}\hat{X}\end{aligned}$$

对于公式 19 所求得的偏移量如果大于 0.5（只要 x、y 和 σ 任意一个量大于 0.5），则表明插值点已偏移到了它的临近的插值中心，所以必须改变当前的位置，使其为它所偏移到的插值中心处，然后在新的位置上重新进行泰勒级数插值拟合，直到偏移量小于 0.5 为止（x、y 和 σ 都小于 0.5），这是一个迭代的工程。当然，为了避免无限次的迭代，我们还需要设置一个最大迭代次数，在达到了迭代次数但仍然没有满足偏移量小于 0.5 的情况下，该极值点就要被剔除掉。另外，如果由上式得到的极值 f($\hat{\mathbf{X}}$)过小，即| f($\hat{\mathbf{X}}$)|<0.03 时（假设图像的灰度值在 0～1.0 之间），则这样的点易受到噪声的干扰而变得不稳定，所以这些点也应该剔除。而在 opencv 中，使用的是下列公式来判断其是否为不稳定的极值：

$$|f(\hat{X})|<\frac{T}{s}$$

其中 T 为经验阈值，系统默认初始化为 0.04。

极值点的求取是在 DoG 尺度图像内进行的，DoG 图像的一个特点就是对图像边缘有很强的响应。一旦特征点落在图像的边缘上，这些点就是不稳定的点。这是因为一方面图像边缘上的点是很难定位的，具有定位的歧义性；另一方面这样的点很容易受到噪声的干扰而变得不稳定。因此我们一定要把这些点找到并剔除掉。它的方法与 Harris 角点检测算法相似，即一个平坦的 DoG 响应峰值往往在横跨边缘的地方有较大的主曲率，而在垂直边缘的方向上有较小的主曲率，主曲率可以通过 2×2 的 Hessian 矩阵 $\boldsymbol{H}$ 求出：

$$\boldsymbol{H}(x,y)=\begin{bmatrix}D_{xx}(x,y) & D_{xy}(x,y)\\ D_{xy}(x,y) & D_{yy}(x,y)\end{bmatrix}$$

其中 $D_{xx}(x, y)$、$D_{yy}(x, y)$和 $D_{xy}(x, y)$分别表示对 DoG 图像中的像素在 x 轴方向和 y 轴方向上求二阶偏导和二阶混合偏导。在这里，我们不需要求具体的矩阵 $\boldsymbol{H}$ 的两个特征值——α 和 β，而只要知道两个特征值的比例就可以知道该像素点的主曲率。

矩阵 $\boldsymbol{H}$ 的直迹和行列式分别为：

$$\mathrm{Tr}(\boldsymbol{H})=D_{xx}+D_{yy}=\alpha+\beta$$

$$\mathrm{Det}(\boldsymbol{H})=D_{xx}D_{yy}-(D_{xy})^2=\alpha\beta$$

我们首先剔除掉那些行列式为负数的点，即 $\mathrm{Det}(\boldsymbol{H})<0$，因为如果像素的曲率有不同的符号，则该点肯定不是特征点。设 $\alpha>\beta$，并且 $\alpha=\gamma\beta$，其中 $\gamma>1$，则

$$\frac{\mathrm{Tr}(\boldsymbol{H})^2}{\mathrm{Det}(\boldsymbol{H})}=\frac{(\alpha+\beta)^2}{\alpha\beta}=\frac{(\gamma\beta+\beta)^2}{\gamma\beta^2}=\frac{(\gamma+1)^2}{\gamma}$$

上式的结果只与两个特征值的比例有关，而与具体的特征值无关。我们知道，当某个像素的 $\boldsymbol{H}$ 矩阵的两个特征值相差越大，即 γ 很大，则该像素越有可能是边缘。对于公式 25，当两个特征值相等时，等式的值最小，随着 γ 的增加，等式的值也增加。所以，要想检查主曲率的比值是否小于某一阈值 γ，只要检查下式是否成立即可：

$$\frac{\mathrm{Tr}(\boldsymbol{H})}{\mathrm{Det}(\boldsymbol{H})}<\frac{(\gamma+1)^2}{\gamma}$$

对于不满足上式的极值点就不是特征点，因此应该把它们剔除掉。Lowe 给出 γ 为 10。在上面的运算中，需要用到有限差分法求偏导，在这里我们给出具体的公式。为方便起见，我们以图像为例只给出二元函数的实例。与二元函数类似，三元函数的偏导可以很容易的得到。

设 $f(i,j)$是 y 轴为 i、x 轴为 j 的图像像素值，则在(i,j)点处的一阶、二阶及二阶混合偏导为：

$$\frac{\partial f}{\partial x}=\frac{f(i,j+1)-f(i,j-1)}{2h},\quad \frac{\partial f}{\partial y}=\frac{f(i+1,j)-f(i-1,j)}{2h}$$

$$\frac{\partial^2 f}{\partial x^2}=\frac{f(i,j+1)+f(i,j-1)-2f(i,j)}{h^2},\quad \frac{\partial^2 f}{\partial y^2}=\frac{f(i+1,j)+f(i-1,j)-2f(i,j)}{h^2}$$

$$\frac{\partial^2 f}{\partial x\,\partial y}=\frac{f(i-1,j-1)+f(i+1,j+1)-f(i-1,j+1)-f(i+1,j-1)}{4h^2}$$

由于在图像中，相邻像素之间的间隔都是 1，所以这里的 h = 1。

3.4 Step3 方向角度的确定

经过上面两个步骤，一幅图像的特征点就可以完全找到，而且这些特征点是具有尺度不变性。但为了实现旋转不变性，还需要为特征点分配一个方向角度，也就是需要根据检测到的特征点所在的高斯尺度图像的局部结构求得一个方向基准。该高斯尺度图像的尺度 σ 是已知的，并且该尺度是相对于高斯金字塔所在组的基准层的尺度，也就是公式 10 所表示的尺度。而所谓局部结构指的是在高斯尺度图像中以特征点为中心，以 r 为半径的区域内计算所有像素梯度的幅角和幅值，半径 r 为：

$$r = 3 \times 1.5\sigma$$

其中 σ 就是上面提到的相对于所在组的基准层的高斯尺度图像的尺度。

像素梯度的幅值和幅角的计算公式为：

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1), L(x,y-1))^2}$$

$$\theta(x,y) = \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

因为在以 r 为半径的区域内的像素梯度幅值对圆心处的特征点的贡献是不同的，因此还需要对幅值进行加权处理，这里采用的是高斯加权，该高斯函数的方差 σ_m 为：

$$\sigma_m = 1.5\sigma$$

在完成特征点邻域范围内的梯度计算后，还要应用梯度方向直方图来统计邻域内像素的梯度方向所对应的幅值大小。具体的做法是，把 360°分为 36 个柱，则每 10°为一个柱，即 0°～9°为第 1 柱，10°～19°为第 2 柱，以此类推。在以 r 为半径的区域内，把那些梯度方向在 0°～9°范围内的像素找出来，把它们的加权后的梯度幅值相加在一起，作为第 1 柱的柱高；求第 2 柱以及其他柱的高度的方法相同，不再赘述。为了防止某个梯度方向角度因受到噪声的干扰而突变，我们还需要对梯度方向直方图进行平滑处理。Opencv2.4.9 所使用的平滑公式为：

$$H(i) = \frac{h(i-2) + h(i+2)}{16} + \frac{4 \times (h(i-1) + h(i+1))}{16} + \frac{6 \times h(i)}{16}, \qquad i = 0, \ldots, 15$$

其中 h 和 H 分别表示平滑前和平滑后的直方图。由于角度是循环的，即 0°=360°，如果出现 $h(j)$，j 超出了(0,…,15)的范围，那么可以通过圆周循环的方法找到它所对应的、在 0°～360°之间的值，如 $h(-1) = h(15)$。

这样，直方图的主峰值，即最高的那个柱体所代表的方向就是该特征点处邻域范围内图像梯度的主方向，也就是该特征点的主方向。由于柱体所代表的角度只是一个范围，如第 1 柱的角度为 0°～9°，因此还需要对离散的梯度方向直方图进行插值拟合处理，以得到更精确的方向角度值。例如我们已经得到了第 i 柱所代表的方向为特征点的主方向，则拟合公式为：

$$B = i + \frac{H(i-1) - H(i+1)}{2 \times (H(i-1) + H(i+1) - 2 \times H(i))}, \qquad i = 0, \ldots, 15$$

$$\theta = 360 - 10 \times B$$

其中，H 为由公式 34 得到的直方图，角度 θ 的单位是度。同样的，公式 35 和公式 36 也存在着公式 34 所遇到的角度问题，处理的方法同样还是利用角度的圆周循环。

每个特征点除了必须分配一个主方向外，还可能有一个或更多个辅方向，增加辅方向的目的是为了增强图像匹配的鲁棒性。辅方向的定义是，当存在另一个柱体高度大于主方向柱体高度的 80%时，则该柱体所代表的方向角度就是该特征点的辅方向。

在第 2 步中，我们实现了用两个信息量来表示一个特征点，即位置和尺度。那么经过上面的计算，我们对特征点的表示形式又增加了一个信息量——方向，即 $\boldsymbol{K}(x, y, \sigma, \theta)$。如果某个特征点还有一个辅方向，则这个特征点就要用两个值来表示——$\boldsymbol{K}(x, y, \sigma, \theta_1)$和 $\boldsymbol{K}(x, y, \sigma, \theta_2)$，其中 θ_1 表示主方向，θ_2 表示辅方向，而其他的变量——x, y, σ 不变。

3.5 Step4 特征点描述符生成

通过上面三个步骤的操作，每个特征点被分配了坐标位置、尺度和方向。在图像局部区域内，这些参数可以重复的用来描述局部二维坐标系统，因为这些参数具有不变性。下面就来计算局部图像区域的描述符，描述符既具有可区分性，又具有对某些变量的不变性，如光亮或三维视角。

描述符是与特征点所在的尺度有关的，所以描述特征点是需要在该特征点所在的高斯尺度图像上进行的。在高斯尺度图像上，以特征点为中心，将其附近邻域划分为 $d \times d$ 个子区域（Lowe 取 $d = 4$）。每个子区域都是一个正方形，正方形的边长为 3σ，也就是说正方形的边长有 3σ 个像素点（这里当然要对 3σ 取整）。σ 为相对于特征点所在的高斯金字塔的组的基准层图像的尺度，即公式 10 所表示的尺度。考虑到实际编程的需要，特征点邻域范围的边长应为 $3\sigma(d+1)$，因此特征点邻域区域一共应有 $3\sigma(d+1) \times 3\sigma(d+1)$个像素点。

为了保证特征点具有旋转不变性，还需要以特征点为中心，将上面确定下来的特征点邻域区域旋转 θ（θ 就是该特征点的方向）。由于是对正方形进行旋转，为了使旋转后的区域包括整个正方形，应该以从正方形的中心到它的边的最长距离为半径，也就是正方形对角线长度的一半，即：

$$r = \frac{3\sigma(d+1)\sqrt{2}}{2}$$

所以上述的特征点邻域区域实际应该有$(2r+1) \times (2r+1)$个像素点。由于进行了旋转，则这些采样点的新坐标为：

$$\begin{bmatrix} x` \\ y` \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad x, y \in [-r, r]$$

其中$[x`, y`]^T$ 为旋转后的像素新坐标，$[x, y]^T$ 为旋转前的坐标。

这时，我们需要计算旋转以后特征点邻域范围内像素的梯度幅值和梯度幅角。这里的梯度幅值还需要根据其对中心特征点贡献的大小进行加权处理，加权函数仍然采用高斯函数，

它的方差的平方为 $d^2/2$。在实际应用中，我们是先以特征点为圆心，以 r 为半径，计算该圆内所有像素的梯度幅角和高斯加权后的梯度幅值，然后再根据上式得到这些幅值和幅角所对应的像素在旋转以后新的坐标位置。

在计算特征点描述符的时候，我们不需要精确知道邻域内所有像素的梯度幅值和幅角，我们只需要根据直方图知道其统计值即可。这里的直方图是三维直方图，如图 3 所示。

图 3 中的三维直方图为一个立方体，立方体的底就是特征点邻域区域。如前面所述，该区域被分为 4×4 个子区域，即 16 个子区域，邻域内的像素根据其坐标位置，把它们归属于这 16 个子区域中的一个。立方体的三维直方图的高为邻域像素幅角的大小。我们把 360 度的幅角范围进行 8 等分，每一个等份为 45 度。则再根据邻域像素梯度幅角的大小，把它们归属于这 8 等份中的一份。这样三维直方图就建立了起来，即以特征点为中心的邻域像素根据其坐标位置，以及它的幅角的大小被划归为某个小正方体（如图 4 中的 C 点，在这里，可以通过归一化处理，得到边长都为单位长度的正方体的）内，该直方图一共有 4×4×8=128 个这样的正方体。而这个三维直方图的值则是正方体内所有邻域像素的高斯加权后的梯度幅值之和，所以一共有 128 个值。我们把这 128 个数写成一个 128 维的矢量，该矢量就是该特征点的特征矢量，所有特征点的矢量构成了最终的输入图像的 SIFT 描述符。

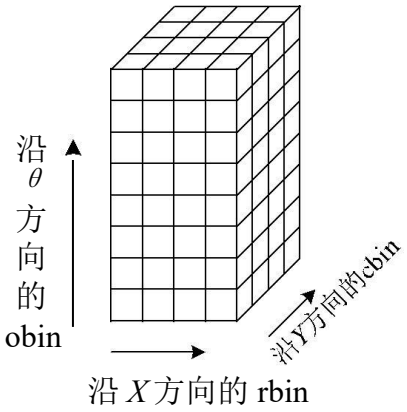


图 3 描述符的三维直方图

显然，正方体的中心应该代表着该正方体。但落入正方体内的邻域像素不可能都在中心，因此我们应该对上面提到的梯度幅值做进一步处理，根据它对中心点位置的贡献大小进行加权处理，即在正方体内，根据像素点相对于正方体中心的距离，对梯度幅值做加权处理。所以三维直方图的值，即正方体的值共需要下面 4 个步骤完成：

1、计算落入该正方体内的邻域像素的梯度幅值 A；

2、根据该像素相对于特征点的距离，对 A 进行高斯加权处理，得到 B；

3、根据该像素相对于它所在的正方体的中心的贡献大小，再对 B 进行加权处理，得到 C；

4、对所有落入该正方体内的像素做上述处理，再进行求和运算 ΣC，得到 D。

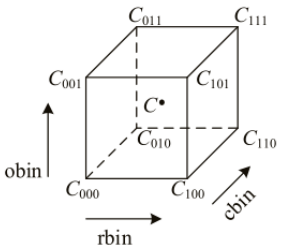


图 4　三维直方图中的正方体

由于计算相对于正方体中心点的贡献大小略显繁琐，因此在实际应用中，我们需要经过坐标平移，把中心点平移到正方体的顶点上，这样只要计算正方体内的点对正方体的 8 个顶点的贡献大小即可。根据三线性插值法，对某个顶点的贡献值是以该顶点和正方体内的点为对角线的两个顶点，所构成的立方体的体积。如图 4 中，C_{000} 的坐标为$(0,0,0)$，C_{100} 的坐标为$(1,0,0)$，以此类推，C 的坐标为(r,c,θ)，（这里的 r,c,θ 值的大小肯定是在 0 和 1 之间），则 C 点与 8 个顶点所构成的立方体的体积，也就是对 8 个顶点的贡献分别为：

C_{000}：$r\times c\times\theta$，

C_{100}：$(1-r)\times c\times\theta$

C_{010}：$r\times(1-c)\times\theta$

C_{001}：$r\times c\times(1-\theta)$

C_{110}：$(1-r)\times(1-c)\times\theta$

C_{101}：$(1-r)\times c\times(1-\theta)$

C_{011}：$r\times(1-c)\times(1-\theta)$

C_{111}：$(1-r)\times(1-c)\times(1-\theta)$

经过上面的三维直方图的计算，最终我们得到了该特征点的特征矢量 $P=\{p_1,p_2,\ldots,p_{128}\}$。为了去除光照变化的影响，需要对特征矢量进行归一化处理，即

$$q_i=\frac{p_i}{\sqrt{\sum_{j=1}^{128}p_j}},\qquad i=1,2,\ldots,128$$

则 $Q=\{q_1,q_2,\ldots,q_{128}\}$为归一化后的特征矢量。尽管通过归一化处理可以消除对光照变化的影响，但由于照相机饱和以及三维物体表面的不同数量不同角度的光照变化所引起的非线性光照变化仍然存在，它能够影响到一些梯度的相对幅值，但不太会影响梯度幅角。为了消除这部分的影响，我们还需要设一个 $t=0.2$ 的阈值，保留 Q 中小于 0.2 的元素，而把 Q 中大于 0.2 的元素用 0.2 替代。最后再对 Q 进行一次归一化处理，以提高特征点的可区分性。

4.SURF 斑点检测

4.1 SURF 算法综述

SURF (Speeded Up Robust Features)是一种具有鲁棒性的局部特征检测算法，它首先由 Herbert Bay 等人于 2006 年提出，并在 2008 年进行了完善。其实该算法是 Herbert Bay 在博士期间的研究内容，并作为博士毕业论文的一部分发表。

SURF 算法的部分灵感来自于 SIFT 算法，但正如它的名字一样，该算法除了具有重复性高的检测器和可区分性好的描述符特点外，还具有很强的鲁棒性以及更高的运算速度，如 Bay 所述，SURF 至少比 SIFT 快 3 倍以上，综合性能要优于 SIFT 算法。与 SIFT 算法一样，SURF 算法也在美国申请了专利。

之所以 SURF 算法有如此优异的表现，尤其是在效率上，是因为该算法一方面在保证正确性的前提下进行了适当的简化和近似，另一方面它多次运用积分图像（integral image）的概念。

在讲解 SURF 算法之前，我们先来大致介绍一下积分图像。积分图像很早就被应用在计算机图形学中，但直到 2001 年才由 Viola 和 Jones 应用到计算机视觉领域中。积分图像 IΣ(x, y)的大小尺寸与原图像 I(x, y)的大小尺寸相等，而积分图像在(x, y)处的值等于原图像中横坐标小于等于 x 并且纵坐标也小于等于 y 的所有像素灰度值之和，也就是在原图像中，从其左上角到(x, y)处所构成的矩形区域内所有像素灰度值之和，即：

$$I_{\Sigma}(x,y)=\sum_{i=0}^{i\le x}\sum_{j=0}^{j\le 0}I(x,y)$$

事实上，积分图像的计算十分简单，只需要对原图像进行一次扫描，就可以得到一幅完整的积分图像，它的计算公式有几种，下列公式是其中的一种：

$$I_{\Sigma}(x,y)=I(x,y)+I_{\Sigma}(x-1,y)+I_{\Sigma}(x,y-1)-I_{\Sigma}(x-1,y-1)$$

其中，$I_{\Sigma}(x-1,y)$、$I_{\Sigma}(x,y-1)$和 $I_{\Sigma}(x-1,y-1)$都是在计算 $I_{\Sigma}(x,y)$之前得到的值。

利用积分图像可以计算原图像中任意矩形内像素灰度值之和。如图 1 所示，某图像 $I(x,y)$中有四个点，它们的坐标分别为 A=(x_0,y_0)，B=(x_1,y_0)，C=(x_0,y_1)和 D=(x_1,y_1)。由这 4 个点组成了矩阵 W，该 W 内的像素灰度之和为

$$\sum_{\substack{x_0<x\le x_1\\ y_0<y\le y_1}}I(x,y)=I_{\Sigma}(\mathrm{D})+I_{\Sigma}(\mathrm{A})-I_{\Sigma}(\mathrm{B})-I_{\Sigma}(\mathrm{C})$$

其中，I_{Σ} 为 I 的积分图像。

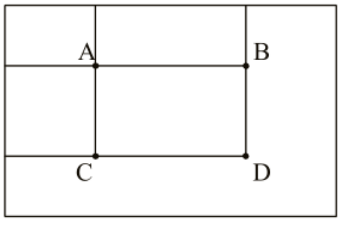


图 1　积分图像求矩阵内灰度值之和

由公式 3 可知，一旦图像的积分图像确定下来，图像中任意矩阵区域内的灰度值之和就可以很容易的得到。更重要的是，无论矩阵面积多大，所需要的运算量都是相同的。因此当算法中需要大量重复的计算不同矩阵区域内的灰度值之和时，应用积分图像就可以大大地提高效率。SURF 算法正是很好的利用了这个性质，以近乎恒定的时间完成了不同尺寸大小的盒状滤波器（box filter）的快速卷积运算。

积分图像就介绍到这里，下面进入主题，重点讲解 SURF 算法。

SURF 算法包括下面几个阶段：

第一部分：特征点检测

1、基于 Hessian 矩阵的特征点检测
2、尺度空间表示
3、特征点定位

第二部分：特征点描述

1、方向角度的分配
2、基于 Haar 小波的特征点描述符

4.2 Step1 特征点检测

1. 基于 Hessian 矩阵的特征点检测

关于特征点，还没有一个统一的定义，但 Bay 认为，特征点是那些在两个不同方向上局部梯度有着剧烈变化的小的图像区域。因此角点、斑点和 T 型连接都可以被认为是特征点。目前检测特征点最好的方法是基于 Harris 矩阵的方法和基于 Hessian 矩阵的方法，Harris 矩阵能够检测出角点类特征点，Hessian 矩阵能够检测出斑点类特征点。SURF 算法应用的是 Hessian 矩阵，这是因为该矩阵在运算速度和特征点检测的准确率上都具有一定的优势。 Hessian 矩阵检测特征点的方法是计算图像所有像素的 Hessian 矩阵的行列式，极值点处就是图像特征点所在的位置。由于在特征点检测的过程中采取了一系列加速运算量的方法，因此 SURF 算法中的特征点检测方法也称为 Fast-Hessian 方法。

无论是 Harris 方法还是 Hessian 方法，都无法实现尺度的不变性。LoG（高斯拉普拉斯方法，Laplacian of Gaussian）是最好的能够保证尺度不变性的方法。Mikolajczyk 和 Schmid 把 Harris 和 LoG 相结合，提出了 Harris-Laplace 方法，从而实现了 Harris 方法的尺度不变性。仿照 Harris-Laplace 方法，SURF 算法把 Hessian 和 LoG 结合，提出了 Hessian-Laplace 方法，也同样保证了用 Hessian 方法所检测到的特征点的尺度不变性。

给定图像 I 中的某点 $\mathbf{x}=(x,y)$，在该点 $\mathbf{x}$ 处，尺度为 σ 的 Hessian 矩阵 $\boldsymbol{H}(\mathbf{x},\sigma)$定义为：

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

其中，$L_{xx}(\mathbf{x},\sigma)$是高斯二阶微分$\frac{\sigma^2 g(\sigma)}{\sigma x^2}$在点 $\mathbf{x}=(x,y)$处与图像 I 的卷积，$L_{xy}(\mathbf{x},\sigma)$和 $L_{yy}(\mathbf{x},\sigma)$具有相似的含义。这些微分就是 LoG。

Bay 指出，高斯函数虽然是最佳的尺度空间分析工具，但由于在实际应用时总是要对高斯函数进行离散化和剪裁处理，从而损失了一些特性（如重复性）。这一因素为我们用其他工具代替高斯函数对尺度空间的分析提供了可能，因为既然高斯函数会带来误差，那么其他

工具所带来的误差也可以被忽略，只要误差不大就可以。况且，SIFT 利用 DoG 近似 LoG 的成功经验也进一步验证了高斯函数的可替代性。

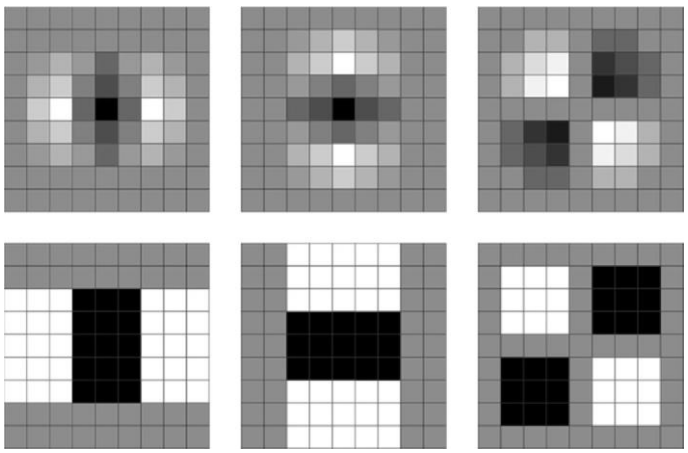


图 2 LoG 的近似

如图 2 所示，第一行图像就是经过离散化，并被裁减为 9×9 方格，σ = 1.2 的沿 x 方向、 y 方向和 xy 方向的高斯二阶微分算子，即 L_{xx} 模板、L_{yy} 模板、L_{xy} 模板。这些微分算子可以用加权后的 9×9 盒状滤波器——D_{xx} 模板、D_{yy} 模板、D_{xy} 模板替代，即图 2 中的第二行图像。因此尺寸大小为 9×9 的盒状滤波器对应于尺度 σ 为 1.2 的高斯二阶微分。在 SURF 算法中，σ 为 1.2 代表着最小的尺度，即最大的空间分辨率。在盒状滤波器中，白色部分的权值为 1，灰色部分的权值为 0，D_{xx} 模板和 D_{yy} 模板的黑色部分的权值为-2，D_{xy} 模板的黑色部分的权值为-1，我们把黑色部分和白色部分统称为突起部分（**lobe**），则灰色部分对应于平坦部分。经过实验证明，盒状滤波器的性能近似或更好于经过离散化和剪裁过的高斯函数。更重要的是，如果利用积分图像，盒状滤波器的运算完全不取决于它的尺寸大小。

下面我们就给出利用积分图像求 D_{xx}、D_{yy}、和 D_{xy} 方法。首先利用公式 1 把输入图像转换为积分图像，然后应用盒状滤波器逐一对积分图像中的像素进行处理。从图 2 可以看出，盒状滤波器的灰色部分的权值为 0，因此该部分不参与计算，仅仅起到填充模板大小的作用。而 D_{xx} 模板和 D_{yy} 模板都各有两个白色部分和一个黑色部分，因此它们的盒状滤波器共有三个突起部分，而 D_{xy} 模板有两个白色部分和两个黑色部分，因此它的盒状滤波器共有四个突起部分。那么利用盒状滤波器对图像进行滤波处理所得到的响应值的一般公式为：

$$D = \sum_{n=1}^{N} \frac{w_n}{S_n} (I_\Sigma(A_n) + I_\Sigma(D_n) - I_\Sigma(B_n) - I_\Sigma(C_n))$$

其中，N 表示突起部分的总和，对于 D_{xx} 模板和 D_{yy} 模板来说，N=3，对于 D_{xy} 模板来说，N=4；S_n 表示第 n 个突起部分的面积，如对于 9×9 的 D_{xx} 模板和 D_{yy} 模板来说，突起部分的面积都是 15（即像素的数量），而对于 9×9 的 D_{xy} 模板来说，突起部分的面积都是 9，除以 S_n 的作用是对模板进行归一化处理；w_n 表示第 n 个突起部分的权值；而后面的括号部分就是公式 3，求模板的每个突起部分对应于图像中四个点 A、B、C、D 所组成的矩阵区域的灰度之和。

我们在前面提到，Hessian 矩阵的行列式的极值处即为特征点，而用盒状滤波器（D_{xx}、D_{yy}、D_{xy}）近似替代高斯二阶微分算子（L_{xx}、L_{yy}、L_{xy}）得到 Hessian 矩阵的行列式不是简单 $D_{xx}D_{yy}$-D_{xy}^2，而是需要加上一定的权值，即：

$$\det(\boldsymbol{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2$$

其中，w 为权值，它的作用是用以平衡因近似所带来的偏差。w 的值为：

$$w = \frac{|L_{xy}(1.2)|_F|D_{yy}(9)|_F}{|L_{yy}(1.2)|_F|D_{xy}(9)|_F} = 0.912 \ldots \cong 0.9$$

其中，$|x|_F$ 表示 Frobenius 范数。尽管该权值 w 是在 σ 为 1.2 和盒状滤波器的大小为 9×9 的情况下得到，但它也适用于其他的情况。这是因为我们还要对盒状滤波器进行归一化处理（公式 5 中的除以 S_n），而且把 w 保持为常数所引入的误差对最终的结果影响不大。

用盒状滤波器（D_{xx}、D_{yy}、D_{xy}）近似替代高斯二阶微分算子（L_{xx}、L_{yy}、L_{xy}）得到 Hessian 矩阵的迹的公式为：

$$\mathrm{tr}(\boldsymbol{H}_{approx}) = D_{xx} + D_{yy}$$

2. 尺度空间表示

要想实现特征点的尺度不变性，就需要在尽可能多的尺度图像上检测特征点。如何建立连续多幅的尺度图像，就成了不变性特征检测的关键。目前比较好的方法是建立尺度图像金字塔，如 SIFT 那样。方法是用高斯核对输入图像进行迭代的卷积，并重复进行降采样处理，以减小它的尺寸，也就是金字塔中的图像由底向顶尺度逐渐增加，而图像尺寸大小则逐渐缩小。但该方法效率不高，因为金字塔中各层图像的创建完全依赖于它的前一层图像。而在 SURF 中，尺度图像的建立是依靠盒状滤波器模板，而不是高斯核，因此它采用了不改变输入图像的尺寸大小，而仅仅改变盒状滤波器模板大小的方式，即多幅尺度逐渐增加的尺度图像的尺寸大小完全一致，我把它称为图像堆。我们在前面介绍过，盒状滤波器的滤波处理可以使用积分图像，因此无论盒状滤波器模板是大是小，运算速度都是一样的。所以采用图像堆可以大大提高运算效率。

图像堆也是被分为若干组（octave），每组又由若干层组成，每一组的各层图像都是由输入图像经过不同尺寸大小的盒状滤波器的滤波处理得到，图像的尺度 s 等于该盒状滤波器的尺度 σ。每一组内图像的尺度的变化范围大约是 2 倍的关系。在 SURF 中，最小的尺度图像（即第 1 组第 1 层图像）是由 9×9 的盒状滤波器（即图 2）得到，它的尺度 $s=1.2$，对应于高斯函数 $\sigma = 1.2$，我们把它称为基准滤波器。图像堆中的其他层图像所需的盒状滤波器都是该基准滤波器通过扩展得到，而它们的模板尺寸大小与其所对应的尺度和基准滤波器模板的比率相同，即

$$s_{approx} = L \times \frac{s_0}{L_0} = L \times \frac{1.2}{9}$$

其中，$L0$ 和 $s0$ 分别表示基准滤波器模板的尺寸和其所对应图像的尺度，L 表示当前层滤波器模板的尺寸。

为了扩展大尺寸的盒状滤波器模板，还需要考虑下面两个因素：1、首先模板尺寸的增加受限于模板突起部分的长度，即突起部分短边的长度必须是模板大小的三分之一；2、为了使扩展后的中心不变，模板的四周必须同时扩展，并且突起部分至少扩展 2 个像素长。

由于 D_{xx} 和 D_{yy} 模板有一边包括全部 3 个突起部分，因此 D_{xy} 模板的扩展以 D_{xx} 和 D_{yy} 模板为准。突起部分的短边的长度决定着长边扩展的长度。在 D_{xx} 和 D_{yy} 模板扩展时，如果三个突起部分的短边都扩展 2 个像素长，则模板扩展 6 个像素长，所以 6 个像素长是

最小的扩展步长，也就是说两个连续模板的长度最小差 6 个像素。

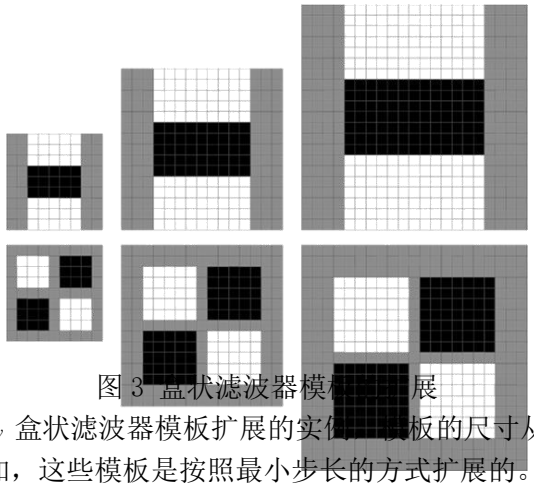


图 3 盒状滤波器模板扩展

图 3 为 D_{yy} 和 D_{xy} 盒状滤波器模板扩展的实例，模板的尺寸从 9×9 扩展为 15×15 和 21×21。由前面分析可知，这些模板是按照最小步长的方式扩展的。由公式 9 可知，15×15 模板对应的尺度为 2.0，21×21 模板对应的尺度为 2.8。

下面介绍图像堆中各组中各层图像所使用的盒状滤波器模板的变化规律。在第一组中，各层的模板大小分别为 9×9、15×15、21×21、27×27，也就是按照最小步长 6 的方式扩展；第二组中模板扩展的步长翻倍，为 12，模板的大小为 15×15、27×27、39×39、51×51；第三组的步长再翻倍，为 24，则模板大小为 27×27、51×51、75×75、99×99；第四组的步长为 48，模板大小为 51×51、99×99、147×147、195×195；其他组以此类推。一般来说图像堆分为 3 至 4 组，每组有 4 层。下面给出每层模块的尺寸大小及其所对应的尺度。

	第一层		第二层		第三层		第四层	
	尺寸	尺度 s	尺寸	尺度 s	尺寸	尺度 s	尺寸	尺度 s
第一组	9	1.2	15	2.0	21	2.8	27	3.6
第二组	15	2.0	27	3.6	39	5.2	51	6.8
第三组	27	3.6	51	6.8	75	10.0	99	13.2
第四组	51	6.8	99	13.2	147	19.6	195	26.0

我们在前面提到过，每一组图像的尺度的变化范围大约是 2 倍的关系，我们来验证一下是否满足上述条件。以第一组为例，因为在后面我们还要在 3×3×3 的范围内进行非最大值抑制，所以不是在第一层图像而是在第二层图像找特征点，并且还要进行插值运算，因此最精细的尺度应该在第一层和第二层图像之间，即$(1.2+2.0)/2=1.6$，同理最粗糙的尺度为$(3.6+2.8)/2=3.2$。它们之间的变化范围正好为 2 倍的关系。其他组的计算方法相同，尺度的变化范围要略大于 2 倍的关系，但不会超过 2.3 倍。所以满足上述条件。

从上面的表可以看出，各个层之间尺度有重叠的现象，其目的是为了覆盖所有可能的尺度。另外随着尺度的增大，被检测到的特征点的数量会锐减，因此为了降低运算量，我们可以在大尺度的图像内减少检测的次数。通常的做法是采用隔点采样的方法，即在第一组图像内，我们对所有像素都进行采样从而计算它们的滤波响应值，也就是采样间隔为 $1=2^0$；在第二组图像内，我们每隔一个像素采样一次，采样间隔为 $2=2^1$，并且只对采样像素进行滤波处理；同理第三组的采样间隔为 $4=2^2$，第四组的采样间隔为 $8=2^3$。

下面我们用公式来表述一下图像所在组和层与其所对应的滤波器模板尺寸的关系：

$$L = 3 \times [2^{o+1} \times (l+1) + 1] \qquad o = 0,1,2,3 \qquad l = 0,1,2,3$$

其中，L 为模板尺寸，o 表示组索引，l 表示组内的层索引，索引值都是从 0 开始。从该公式可以看出，方括号内的部分其实是该模板突起部分短边的长度。

这里需要说明的是，公式 10 是通过 Bay 的原著总结出来的公式，但 opencv2.4.9 中没有应用该公式，而是使用的下列公式；

$$L = (9 + 6 \times l) \times 2^o \qquad o = 0,1,2,3 \qquad l = 0,1,2,3$$

3. 特征点定位

在上一步，我们通过建立图像堆，并在每一层图像上应用不同尺寸大小的盒状滤波器模板得到了滤波响应值，然后通过公式 6 得到了所有像素的 Hessian 矩阵的行列式，下面我们就基于这些行列式值找出特征点，并精确确定它们的位置。它包括三个步骤——阈值、非最大值抑制和插值。

我们首先取阈值，去掉那些行列式值低的像素，仅保留那些最强的响应值。很明显，阈值选择得大，所检测到的特征点就多，反之阈值小，特征点就少

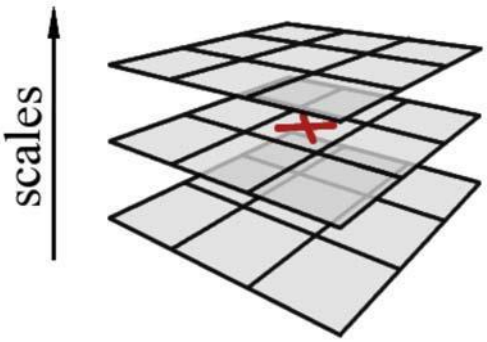


图 4 非最大值抑制所在的 3×3×3 区域

然后就是进行非最大值抑制，即在邻域范围内去掉那些不是最大值的像素。为了满足尺度不变性，在进行非最大值抑制时不仅需要与被检测像素所在的尺度图像的邻域像素相比较，还需要与相邻尺度的图像像素比较，也就是要像如图 4 所示的那样在 3×3×3 的区域内进行比较，同一层内有 8 个邻域像素，下层和上层各有 9 个邻域像素。

非最大值抑制是在图像堆的组内进行，也就是同组的相邻层之间进行比较。由于每组的第一层图像和最后一层图像各只有一个相邻层，因此这两层不能进行非最大值抑制比较。以每组有 4 层图像为例，这样就只有中间两层可以进行比较，我们把可以进行非最大值抑制比较的层称为中间层。如果图像堆有 4 组，那么图像堆一共有 16（4×4）层图像，而中间层则只有 8（2×4）层。

最后是应用插值法确定特征点位置。由于离散化，前面所检测到的特征点的位置往往并不是真正的位置，因此我们还要应用插值法找到亚像素级精度的特征点位置。特征点的位置不仅应该包括所在层图像的坐标位置，还应该包括尺度（滤波器模板的尺寸也可以，因为它们之间可以通过公式 9 换算），因此称为尺度坐标——$\boldsymbol{X} = (x, y, s)^T$。

SURF 算法与 SIFT 算法一样，应用的是泰勒级数展开式来进行插值计算。设 $B(\boldsymbol{X})$为在 $\boldsymbol{X}$ 处的特征点响应值，即 Hessian 矩阵的行列式，则它的泰勒级数（只展开到二项式）为：

$$B(\boldsymbol{X}) = B + \left(\frac{\partial B}{\partial \boldsymbol{X}}\right)^T \boldsymbol{X} + \frac{1}{2}\boldsymbol{X}^T \frac{\partial^2 B}{\partial \boldsymbol{X}^2}\boldsymbol{X}$$

相对于点 $\boldsymbol{X}$ 的偏移量为:

$$\hat{\boldsymbol{x}} = -\left(\frac{\partial^2 B}{\partial \boldsymbol{X}^2}\right)^{-1} \frac{\partial B}{\partial \boldsymbol{X}}$$

其中，

$$\frac{\partial^2 B}{\partial \boldsymbol{X}^2} = \begin{bmatrix} d_{xx} & d_{xy} & d_{xs} \\ d_{xy} & d_{yy} & d_{ys} \\ d_{xs} & d_{ys} & d_{ss} \end{bmatrix}$$

$$\frac{\partial B}{\partial \boldsymbol{X}} = \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix}$$

上述两公式中等号右侧分别表示对 Hessian 矩阵行列式值图像的二阶偏导和一阶导数

4.3 Step2 特征点描述

1. 方向角度的分配

为了实现旋转不变性，就必须为每一个特征点分配一个复现性好的方向角度。为此，我们首先建立一个以特征点为中心，半径为 6s 的圆形邻域，我们称为 6s 圆邻域，并对该圆以 s 为采样间隔进行采样，其中 s 为特征点所在的尺度图像的尺度，则结果如图 5 所示。图中中心的位置为特征点，其他的点为采样像素。

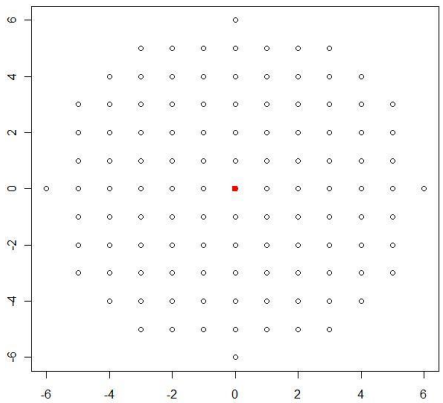


图 5　6s 圆邻域

然后对该 6s 圆邻域内的所有采样像素计算它们的 x 方向和 y 方向上的 Haar 小波响应值， Haar 小波响应的边长尺寸是 4s。Haar 小波是一种最简单的滤波器，用它可以检测出 x 方向和 y 方向的梯度。如图 6 所示，左侧为 x 方向的 Haar 小波响应，右侧为 y 方向。黑色部分的权值为 1，白色部分的权值为-1。用 Haar 小波的另一个好处是结合积分图像，不管 Haar 小波响应的尺寸多大，都只需 6 个运算即可得到 x 方向和 y 方向的梯度。

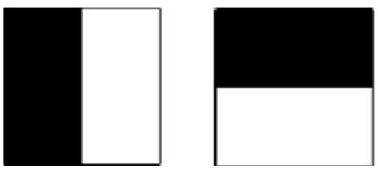


图 6 Haar 小波响应

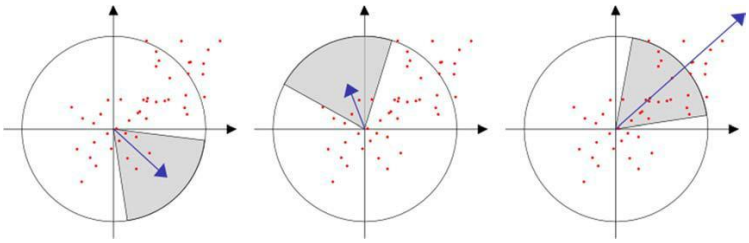


图 7 梯度坐标系内求方向

很明显，距离特征点越近，采样像素对特征点的影响越大，因此我们还需要对 x 方向和 y 方向的 Haar 小波响应值进行加权处理。常用的加权函数为高斯函数，它的方差设为 $2s$。

最后我们分别以加权后的 x 方向和 y 方向的 Haar 小波响应值为 x 轴和 y 轴建立一个梯度坐标系，所有的 $6s$ 圆邻域内的采样点都分布在该坐标系内。如图 7 所示，坐标系内的点为采样点，它的 x 轴坐标和 y 轴坐标分别对应于加权后的 x 方向和 y 方向的 Haar 小波响应值。为了求取特征点的方向，我们需要在该梯度坐标系内设计一个以原点为中心，张角为 60 度的扇形滑动窗口，如图 7 所示，要以一定的步长旋转这个滑动窗口，并对该滑动窗口内的所有点累计其 x 轴坐标和 y 轴坐标值，计算累计和的模和幅角。

$$m_w = \sum_k x + \sum_k y$$

$$\theta_w = arctan\left(\frac{\sum_k x}{\sum_k y}\right)$$

其中，m_w 和 θ_w 分别为 w 扇形滑动窗口内采样点的模和幅角，假设 w 窗口内共有 k 个点，则∑kX 和∑kY 分别表示这 k 个点的横坐标和纵坐标之和，前面已经介绍过，坐标之和也就是 Haar 小波响应值之和

这样旋转扇形滑动窗口，直至旋转一周为止，比较所有窗口下的模值，模值最大的那个窗口所对应的幅角就是该特征点的方向角度：

$$\theta = \theta_{w|max\{m_w\}}$$

要说明一点的是，在一些应用中，旋转不变性并不是必须的。这就像一个人如果保持正常的姿态，让他去观察一个固定的建筑物，可能会由于观察位置的变化建筑物的尺度和视角会变化，还可能由于天气的原因，亮度会变化，但建筑物的旋转角度变化不大，甚至不会变化。Bay 把这种不需要确定特征点方向的方法称为 U-SURF。实验表明 U-SURF 计算速度更快，并且对在正负 15 度范围内的旋转变化下，U-SURF 具备较强的鲁棒性。

2. 特征点描述符的生成

对特征点检测的要求是重复性要好，而对特征点描述符的要求是可区分性要好。

提取 SURF 描述符的第一步是构造一个以特征点为中心的正方形邻域，正方形的边长为 $20s$，s 仍然指的是尺度，我们称该邻域为 $20s$ 方邻域。当然为了实现旋转不变性，该 $20s$

方邻域需要校正为与特征点的方向一致，如图 8 所示。我们对 20*s* 方邻域进行采样间隔为 *s* 的等间隔采样，并把该邻域划分为 4×4=16 个子区域，则 20*s* 方邻域内共有 400 个采样像素，而每个子区域则有 25 个采样像素。

对每个子区域内的所有 25 个采样像素，仍然采用图 6 所示的 Haar 小波计算它们的 *x* 方向和 *y* 方向的梯度，在这里 Haar 小波响应的尺寸为 2s。我们把 *x* 方向和 *y* 方向的 Haar 小波响应值分别定义为 *dx* 和 *dy*。对于某个采样像素的 *dx* 和 *dy* 仍然需要根据该像素与中心特征点的距离进行高斯加权处理，高斯函数的方差为 3.3*s*。

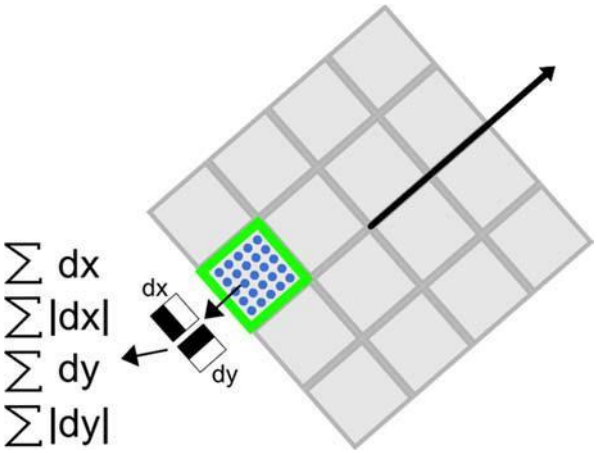


图 8 描述符表示

在每个子区域，我们需要累加所有 25 个采样像素的 *dx* 和 *dy*，这样形成了描述符的一部分。而为了把强度变化的极性信息也包括进描述符中，我们还需要对 *dx* 和 *dy* 的绝对值进行累加。这样每个子区域就可以用一个 4 维特征矢量 **v** 表示：

$$\mathbf{v} = [\ \Sigma dx,\ \Sigma dy,\ \Sigma|dx|,\ \Sigma|dy|\ ]$$

把所有 4×4 个子区域的 4 维特征矢量 **v** 组合在一起，就形成了一个 64 维特征矢量，即 SURF 描述符。

为了去除光照变化的影响，我们还需要对该描述符进行归一化处理。设 64 维特征矢量的 SURF 描述符为 $P=\{p_1,p_2,\ldots,p_{64}\}$，归一化公式为：

$$q_i = \frac{p_i}{\sqrt{\sum_{j=1}^{64} p_j^2}}, \qquad i = 1,2,\ldots,64$$

其中，$Q=\{q_1,q_2,\ldots,q_{64}\}$为归一化后的特征矢量。

Bay 在论文中指出，他们对各类小波特性进行了大量的实验，使用了 dx^2 和 dy^2，高阶小波，PCA，中值，均值等，并通过评估，得出了使用公式 20 那样的形式能够达到最佳的效果。而且他们还通过改变 20*s* 方邻域内的采样像素和子区域的数量，得出了 4×4 个子区域是最佳的划分方式。多于 4×4 个子区域的划分鲁棒性较差，并且增加了匹配时间；反之，匹配效果较差，但缩短了匹配时间。但 3×3 个子区域划分方式仍然是可以接受的，因为它要好过其他的描述符。

另外除了 64 维特征矢量的 SURF 描述符外，Bay 还提出了 128 维特征矢量的描述符方法。两者方法基本相同，区别在于根据 *dy* 是否小于 0，Σ*dx* 和 Σ|*dx*|分别被划分成两个部分，同理 Σ*dy* 和 Σ|*dy*|也是根据 *dx* 的正负号分别被划分成两个部分。这样每个子区域就用一个 8 维特征矢量表示，则描述符就增加到了 128 维。这么做虽然增加了匹配时间，但描述符的可区分性更强。

为了实现快速匹配，我们还可以利用特征点的拉普拉斯响应的正负号。SURF 算法检测的是斑点类的特征点，而特征点的拉普拉斯正负号分别代表着黑背景下的亮斑和白背景下的黑斑，如图 9 所示。在匹配的时候，只有拉普拉斯符合相同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搜索的时间。拉普拉斯符号就是 Hessian 矩阵的迹的符号，因此只要求出 Hessian 矩阵的迹的符号即可。这一步骤并不会增加运算量，因为在特征点检测时已对 Hessian 矩阵的迹进行了计算，我们只要把该值作为特征点的一个变量保存下来即可。

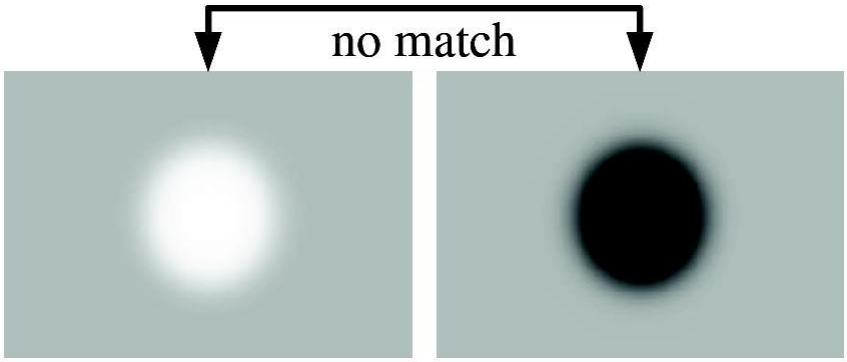


图 9 特征点的两种拉普拉斯响应

4.4 SURF 与 SIFT 的比较

SURF 算法在许多地方都借鉴了 SIFT 算法的成功经验，下面我们就比较一下两者：

1、SURF 算法的抗干扰能力要强于 SIFT 算法；

2、SURF 算法不像 SIFT 算法，它可以实现并行计算，这样更加快了运算速度；

3、SURF 算法的精度略逊于 SIFT 算法；

4、在亮度不变性和视角不变性方面，SURF 算法不如 SIFT 算法。

第四章 二进制字符串特征描述子

1.特征描述子

特征描述子（Feature Descriptors）指的是检测图像的局部特征（比如边缘、角点、轮廓等），然后据匹配目标的需要进行特征的组合、变换，以形成易于匹配、稳定性好的特征向量，从而把图像匹配问题转化为特征的匹配问题，进而将特征的匹配问题转化为特征空间向量的聚类问题。

2.BRIEF 描述子

2.1 BRIEF 的基本原理

我们已经知道 SIFT 特征采用了 128 维的特征描述子，由于描述子用的浮点数，所以它将会占用 512 bytes 的空间。类似地，对于 SURF 特征，常见的是 64 维的描述子，它也将占用 256bytes 的空间。如果一幅图像中有 1000 个特征点（不要惊讶，这是很正常的事），那么 SIFT 或 SURF 特征描述子将占用大量的内存空间，对于那些资源紧张的应用，尤其是嵌入式的应用，这样的特征描述子显然是不可行的。而且，越占有越大的空间，意味着越长的匹配时间。

但是实际上 SIFT 或 SURF 的特征描述子中，并不是所有维都在匹配中有着实质性的作用。我们可以用 PCA、LDA 等特征降维的方法来压缩特征描述子的维度。还有一些算法，例如 LSH，将 SIFT 的特征描述子转换为一个二值的码串，然后这个码串用汉明距离进行特征点之间的匹配。这种方法将大大提高特征之间的匹配，因为汉明距离的计算可以用异或操作然后计算二进制位数来实现，在现代计算机结构中很方便。

于是，BRIEF(Binary Robust Independent Elementary Features)应运而生。和传统的利用图像局部邻域的灰度直方图或梯度直方图提取特征的方式不同，BRIEF 是一种二进制编码的特征描述子，既降低了存储空间的需求，提升了特征描述子生成的速度，也减少了特征匹配时所需的时间。

值得注意的是，对于 BRIEF，它仅仅是一种特征描述符，它不提供提取特征点的方法。所以，如果你必须使一种特征点定位的方法，如 FAST、SIFT、SURF 等。这里，我们将使用 CenSurE 方法来提取关键点，对 BRIEF 来说，CenSurE 的表现比 SURF 特征点稍好一些。

2.2 BRIEF 算法步骤

它需要先平滑图像，然后在特征点周围选择一个 Patch，在这个 Patch 内通过一种选定的方法来挑选出来n_d个点对。然后对于每一个点对 (p,q)，我们来比较这两个点的亮度值，如果 I(p)>I(q)则这个点对生成了二值串中一个的值为 1，如果 I(p)<I(q)，则对应在二值串

中的值为-1，否则为 0。所有n_d个点对，都进行比较之间，我们就生成了一个n_d长的二进制串。

对于n_d的选择，我们可以设置为 128，256 或 512，这三种参数在 OpenCV 中都有提供，但是 OpenCV 中默认的参数是 256，这种情况下，非匹配点的汉明距离呈现均值为 128 比特征的高斯分布。一旦维数选定了，我们就可以用汉明距离来匹配这些描述子了。

2.3 BRIEF 采样方式

设我们在特征点的邻域块大小为 S×S 内选择n_d个点对 (p,q)，论文原作者 Calonder 在实验中测试了 5 种采样方法：

1）在图像块内平均采样；

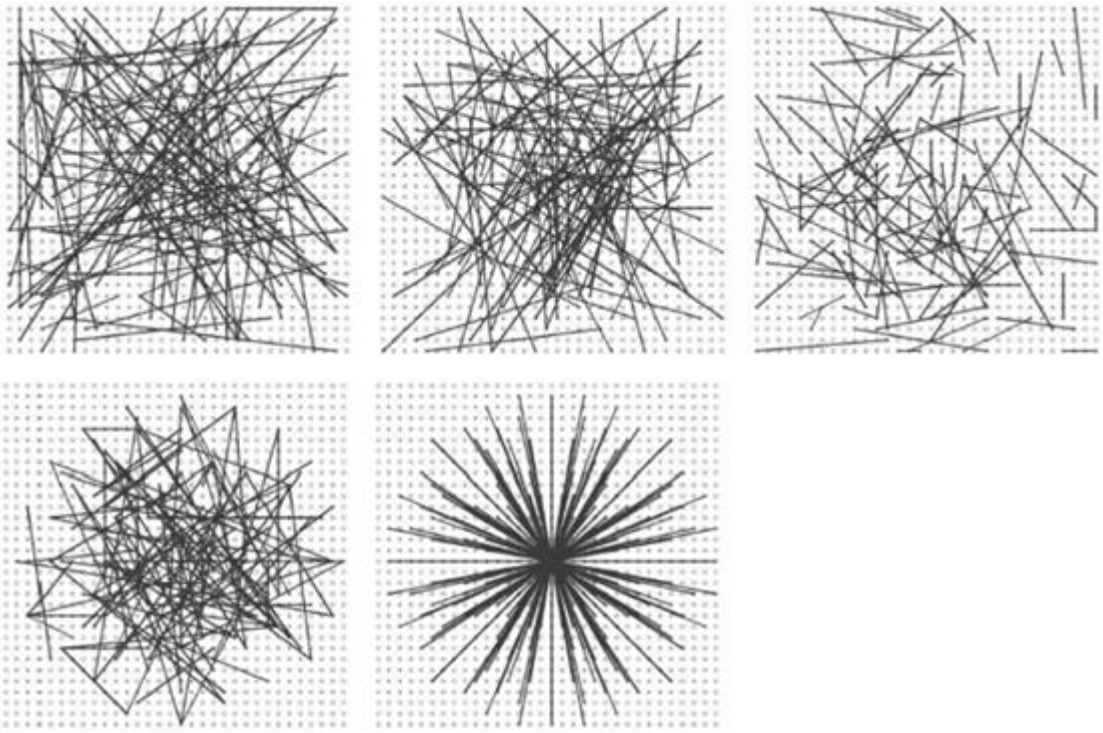
2）p 和 q 都符合 $(0,\frac{1}{25}S^2)$的高斯分布；

3）p 符合$(0,\frac{1}{25}S^2)$的高斯分布，而 q 符合 $(0, \frac{1}{100}S^2)$的高斯分布；

4）在空间量化极坐标下的离散位置随机采样

5）把 p 固定为(0,0)，q 在周围平均采样

下面是上面 5 种采样方法的结果示意图。



3.ORB 特征提取算法

3.1 ORB 概述

ORB 特征，从它的名字中可以看出它是对 FAST 特征点与 BREIF 特征描述子的一种结合与改进，这个算法是由 Ethan Rublee,Vincent Rabaud,Kurt Konolige 以及 Gary R.Bradski 在 2011 年一篇名为“ORB：An Efficient Alternative to SIFT or SURF”的文章中提出。就像文章题目所写一样,ORB 是除了 SIFT 与 SURF 外一个很好的选择,而且它有很高的效率，最重要的一点是它是免费的，SIFT 与 SURF 都是有专利的，你如果在商业软件中使用，需要购买许可。

3.2 ORB 算法原理

ORB 特征是将 FAST 特征点的检测方法与 BRIEF 特征描述子结合起来，并在它们原来的基础上做了改进与优化。

首先，它利用 FAST 特征点检测的方法来检测特征点，然后利用 Harris 角点的度量方法，从 FAST 特征点从挑选出 Harris 角点响应值最大的 N 个特征点。其中 Harris 角点的响应函数定义为：

$$R = detM - \alpha(traceM)^2$$

3.3 旋转不变性

我们知道 FAST 特征点是没有尺度不变性的，所以我们可以通过构建高斯金字塔，然后在每一层金字塔图像上检测角点，来实现尺度不变性。那么，对于局部不变性，我们还差一个问题没有解决，就是 FAST 特征点不具有方向，ORB 的论文中提出了一种利用灰度质心法来解决这个问题，灰度质心法假设角点的灰度与质心之间存在一个偏移，这个向量可以用于表示一个方向。对于任意一个特征点 p 来说，我们定义 p 的邻域像素的矩为：

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

其中 I(x,y)为点(x,y)处的灰度值。那么我们可以得到图像的质心为：

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$$

那么特征点与质心的夹角定义为 FAST 特征点的方向：

$$\theta = \arctan(m_{01}, m_{10})$$

为了提高方法的旋转不变性，需要确保 x 和 y 在半径为 r 的圆形区域内，即 $x,y \in [-r,r]$，r 等于邻域半径。

3.4 特征点的描述

ORB 选择了 BRIEF 作为特征描述方法，但是我们知道 BRIEF 是没有旋转不变性的，所以我们需要给 BRIEF 加上旋转不变性，把这种方法称为“Steer BREIF”。对于任何一个特征点来说，它的 BRIEF 描述子是一个长度为 n 的二值码串，这个二值串是由特征点周围 n 个点对（2n 个点）生成的，现在我们将这 2n 个点$(x_i,y_i),i=1,2,\cdots,2n$ 组成一个矩阵 S

$$S=\begin{pmatrix} x_1 & x_2 & \cdots & x_{2n} \\ y_1 & y_2 & \cdots & y_{2n} \end{pmatrix}$$

Calonder 建议为每个块的旋转和投影集合分别计算 BRIEF 描述子，但代价昂贵。ORB 中采用了一个更有效的方法：使用邻域方向 θ 和对应的旋转矩阵 R_θ，构建 S 的一个校正版本 S_θ。

$$S_\theta=R_\theta S$$

其中

$$R_\theta=\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

而 θ 即我们在 3.3 中为特征点求得的主方向。

实际上，我们可以把角度离散化，即把 360 度分为 12 份，每一份是 30 度，然后我们对这个 12 个角度分别求得一个 S_θ，这样我们就创建了一个查找表，对于每一个 θ，我们只需查表即可快速得到它的点对的集合 S_θ。

3.5 解决描述子的区分性

BRIEF 令人惊喜的特性之一是：对于 n 维的二值串的每个比特征位，所有特征点在该位上的值都满足一个均值接近于 0.5，而方差很大的高斯分布。方差越大，说明区分性越强，那么不同特征点的描述子就表现出来越大差异性，对匹配来说不容易误配。但是当我们把 BRIEF 沿着特征点的方向调整为 Steered BRIEF 时，均值就漂移到一个更加分散式的模式。可以理解为有方向性的角点关键点对二值串则展现了一个更加均衡的表现。而且论文中提到经过 PCA 对各个特征向量进行分析，得知 Steered BRIEF 的方差很小，判别性小，各个成分之间相关性较大。

为了减少 Steered BRIEF 方差的亏损，并减少二进制码串之间的相关性，ORB 使用了一种学习的方法来选择一个较小的点对集合。方法如下：

首先建立一个大约 300k 关键点的测试集，这些关键点来自于 PASCAL2006 集中的图像。对于这 300k 个关键点中的每一个特征点，考虑它的 31×31 的邻域，我们将在这个邻域内找一些点对。不同于 BRIEF 中要先对这个 Patch 内的点做平滑，再用以 Patch 中心为原点的高斯分布选择点对的方法。ORB 为了去除某些噪声点的干扰，选择了一个 5×5 大小的区域的平均灰度来代替原来一个单点的灰度，这里 5×5 区域内图像平均灰度的计算可以用积分图的方法。我们知道 31×31 的 Patch 里共有 N=(31−5+1)×(31−5+1)个这种子窗口，那么我们要 N 个子窗口中选择 2 个子窗口的话，共有C_N^2种方法。所以，对于 300k 中的每一个特征点，我们都可以从它的 31×31 大小的邻域中提取出一个很长的二进制串，长度为 M=C_N^2，表示为

$$binArray=[p_1,p_2,\cdots,p_M],p_i\in\{0,1\}$$

那么当 300k 个关键点全部进行上面的提取之后，我们就得到了一个 300k×M 的矩阵，矩阵中的每个元素值为 0 或 1。对该矩阵的每个列向量，也就是每个点对在 300k 个特征点上的测试结果，计算其均值。把所有的列向量按均值进行重新排序。排好后，组成了一个向量 T，T 的每一个元素都是一个列向量。

进行贪婪搜索：从 T 中把排在第一的那个列放到 R 中，T 中就没有这个点对了测试结果了。然后把 T 中的排下一个的列与 R 中的所有元素比较，计算它们的相关性，旭果相关超过了某一事先设定好的阈值，就扔了它，否则就把它放到 R 里面。重复上面的步骤，只到 R 中有 256 个列向量为止。如果把 T 全找完也，也没有找到 256 个，那么，可以把相关的阈值调高一些，再重试一遍。

这样，我们就得到了 256 个点对。上面这个过程我们称它为 rBRIEF。

4.BRISK 特征提取算法

4.1 BRISK 概述

BRISK 算法是 2011 年 ICCV 上《BRISK:Binary Robust Invariant Scalable Keypoints》文章中，提出来的一种特征提取算法，也是一种二进制的特征描述算子。它具有较好的旋转不变性、尺度不变性，较好的鲁棒性等。在图像配准应用中，速度比较：SIFT<SURF<BRISK<FREAK<ORB，在对有较大模糊的图像配准时，BRISK 算法在其中表现最为出色。

4.2 Step1 特征点检测

BRISK 算法主要利用 FAST9-16 进行特征点检测（为什么是主要？因为用到一次 FAST5-8）。要解决尺度不变性，就必须在尺度空间进行特征点检测，于是 BRISK 算法中构造了图像金字塔进行多尺度表达。

建立尺度空间

构造 n 个 octave 层（用 ci 表示）和 n 个 intra-octave 层（用 di 表示），文章中 n=4，i={0,1,...,n-1}。假设有图像 img，octave 层的产生：c0 层就是 img 原图像，c1 层是 c0 层的 2 倍下采样，c2 层是 c1 层的 2 倍下采样，以此类推。intra-octave 层的产生：d0 层是 img 的 1.5 倍下采样，d1 层是 d0 层的 2 倍下采样（即 img 的 2*1.5 倍下采样），d2 层是 d1 层的 2 倍下采样，以此类推。

则 ci、di 层与原图像的尺度关系用 t 表示为：$t(c_i) = 2^i$，$t(d_i) = 2^i \cdot 1.5$. ci、di 层与原图像大小关系为：

img	h(high)	w(width)
c0	h	w
d0	$\frac{2}{3}$h	$\frac{2}{3}$h
c1	$\frac{1}{2}$h	$\frac{1}{2}$h
d1	$\frac{1}{3}$h	$\frac{1}{3}$h
c2	$\frac{1}{4}$h	$\frac{1}{4}$h
d2	$\frac{1}{6}$h	$\frac{1}{6}$h
c3	$\frac{1}{8}$h	$\frac{1}{8}$h
d3	$\frac{1}{12}$h	$\frac{1}{12}$h

由于 n=4，所以一共可以得到 8 张图，octave 层之间尺度（缩放因子）是 2 倍关系，intra-octave 层之间尺度（缩放因子）也是 2 倍关系。

特征点检测

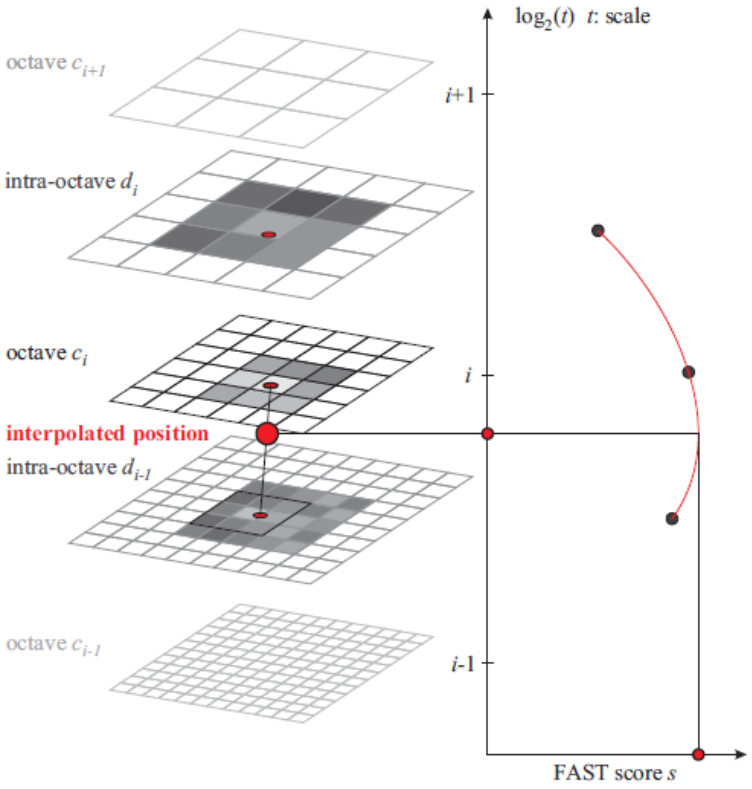
对上述 8 张图进行 FAST9-16 角点检测，得到具有角点信息的 8 张图，对原图像进行一次 FAST5-8 角点检测（当做 d(-1)层，虚拟层），总共会得到 9 幅有角点信息的图像。

非极大值抑制

对上述 9 幅图像，进行空间上的非极大值抑制（同 SIFT 算法的非极大值抑制）：特征点在位置空间（8 邻域点）和尺度空间（上下层 2x9 个点），共 26 个邻域点的 FAST 的得分值要最大，否则不能当做特征点；此时得到的极值点还比较粗糙，需要进一步精确定位。

亚像素插值

进过上面步骤，得到了图像特征点的位置和尺度，在极值点所在层及其上下层所对应的位置，对 FAST 得分值（共 3 个）进行二维二次函数插值（x、y 方向），得到真正意义上的得分极值点及其精确的坐标位置（作为特征点位置）；再对尺度方向进行一维插值，得到极值点所对应的尺度（作为特征点尺度）。

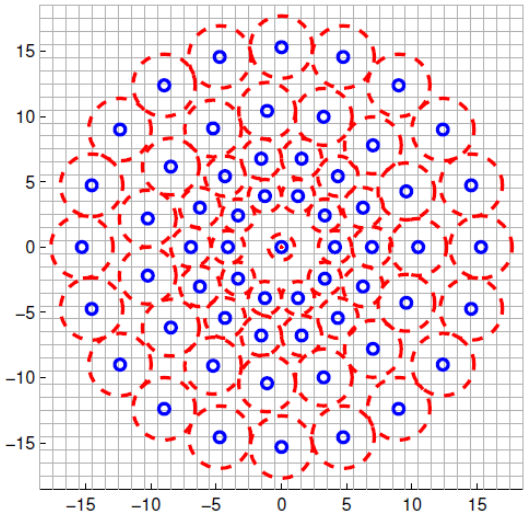


4.3 Step2 特征点描述

高斯滤波

现在，我们得到了特征点的位置和尺度（t）后，要对特征点赋予其描述符。均匀采样模式：以特征点为中心，构建不同半径的同心圆，在每个圆上获取一定数目的等间隔采样点（所有采样点包括特征点，一共 N 个），由于这种邻域采样模式会引起混叠效应，所以需要对同心圆上的采样点进行高斯滤波。

采样模式如下图，蓝圈表示；以采样点为中心，δ为方差进行高斯滤波，滤波半径大小与高斯方差的大小成正比，红圈表示。最终用到的 N 个采样点是经过高斯平滑后的采样点。下图是 t=1 时的。（文章中：N=60）



局部梯度计算

由于有 N 个采样点，则采样点两两组合成一对，共有 N(N-1)/2 钟组合方式，所有组合方式的集合称作采样点对，用集合$\mathcal{A}=\{(\mathbf{p}_i,\mathbf{p}_j)\in\mathbb{R}^2\times\mathbb{R}^2\mid i<N\wedge j<i\wedge i,j\in\mathbb{N}\}$表示，其中像素分别是$I(\mathbf{p}_i,\sigma_i)$、$I(\mathbf{p}_j,\sigma_j)$，δ 表示尺度。用$\mathbf{g}(\mathbf{p}_i,\mathbf{p}_j)$表示特征点局部梯度集合，则有：

$$\mathbf{g}(\mathbf{p}_i,\mathbf{p}_j)=(\mathbf{p}_j-\mathbf{p}_i)\cdot\frac{I(\mathbf{p}_j,\sigma_j)-I(\mathbf{p}_i,\sigma_i)}{\|\mathbf{p}_j-\mathbf{p}_i\|^2}$$

定义短距离点对子集、长距离点对子集（L 个）：

$$\mathcal{S} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| < \delta_{max}\} \subseteq \mathcal{A}$$
$$\mathcal{L} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| > \delta_{min}\} \subseteq \mathcal{A}.$$

其中，$\delta_{max} = 9.75t$，$\delta_{min} = 13.67t$，t 是特征点所在的尺度。

现在要利用上面得到的信息，来计算特征点的主方向（注意：此处只用到了长距离子集），如下：

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \cdot \sum_{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{L}} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j)$$

$$\alpha = \arctan2\,(g_y, g_x)$$

特征描述符

要解决旋转不变性，则需要对特征点周围的采样区域进行旋转到主方向，旋转后得到新的采样区域，采样模式同上。BRISK 描述子是二进制的特征，由采样点集合可得到 N(N-1)/2 对采样点对，就可以得到 N(N-1)/2 个距离的集合（包含长、短距离子集），考虑其中短距离子集中的 512 个短距离点对，进行二进制编码，判断方式如下：

$$b = \begin{cases} 1, & I(\mathbf{p}_j^\alpha, \sigma_j) > I(\mathbf{p}_i^\alpha, \sigma_i) \\ 0, & \text{otherwise} \end{cases}$$
$$\forall (\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) \in \mathcal{S}$$

其中，$I(\mathbf{p}_j^\alpha, \sigma_j)$带有上标，表示经过旋转 a 角度后的，新的采样点。由此可得到，512 Bit 的二进制编码，也就是 64 个字节（BRISK64）。

匹配方法

汉明距离进行比较，与其他二进制描述子的匹配方式一样。

5.FREAK 特征提取算法

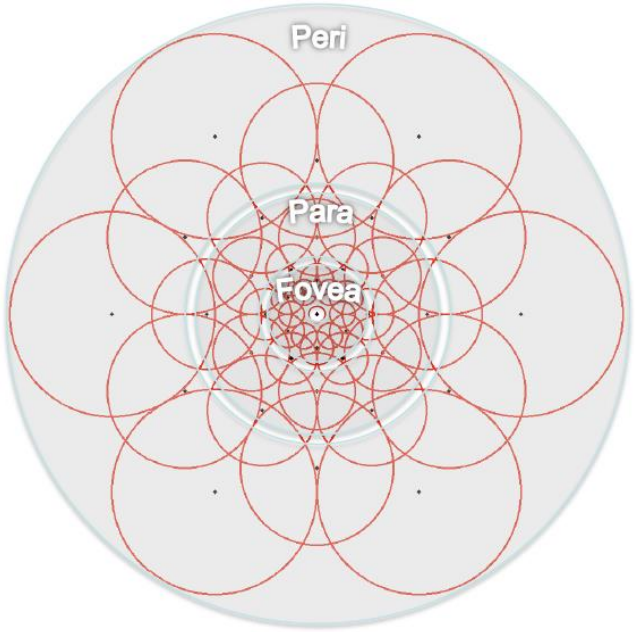
5.1 FREAK 概述

FREAK 算法是 2012 年 CVPR 上《FREAK: Fast Retina Keypoint》文章中，提出来的一种特征提取算法，也是一种二进制的特征描述算子。

它与 BRISK 算法非常相似，个人觉得就是在 BRISK 算法上的改进。FREAK 依然具有尺度不变性、旋转不变性、对噪声的鲁棒性等。

5.2 采样模式

在 BRISK 算法中，采样模式是均匀采样模式（在同一圆上等间隔的进行采样）；FREAK 算法中，采样模式发生了改变，它采取了更为接近于人眼视网膜接收图像信息的采样模型。图中展示了人眼视网膜拓扑结构，Fovea 区域主要是对高进度的图像信息进行处理，Para 主要是对低精度的图像信息进行处理。采样点为：6、6、6、6、6、6、6、1，那个 1 是特征点。



从图中可以看出，该结构是由很多大小不同并有重叠的圆构成，最中心的点是特征点，其它圆心是采样点，采样点离特征点的距离越远，采样点圆的半径越大，也表示该圆内的高斯函数半径越大。

5.3 特征描述

F 表示二进制描述子，Pa 是采样点对（与 BRISK 同理），N 表示期望的二进制编码长度。

$$F = \sum_{0 \le a < N} 2^a T(P_a)$$

$$T(P_a) = \begin{cases} 1 & \text{if } (I(P_a^{r_1}) - I(P_a^{r_2}) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$I(P_a^{r_1})$表示采样点对 Pa 中前一个采样点的像素值，同理，$I(P_a^{r_2})$表示后一个采样点的像素值。

当然得到特征点的二进制描述符后，也就算完成了特征提取。但是 FREAK 还提出，将得到的 N bit 二进制描述子进行筛选，希望得到更好的，更具有辨识度的描述子，也就是说要从中去粗取精。（也就是降维）

1、建立矩阵 D，D 的每一行是一个 FREAK 二进制描述符，即每一行有 N 个元素；在上图的采样模式中公有 43 个采样点，可以产生 N=43*(43-1)/2=903 个采样点对，也就是说矩阵 D 有 903 列；

2、对矩阵 D 的每一列计算其均值，由于 D 中元素都是 0/1 分布的，均值在 0.5 附近说明该列具有高的方差；

3、每一列都有一个均值，以均值最接近 0.5 的排在第一位，均值离 0.5 越远的排在越靠后，对列进行排序；

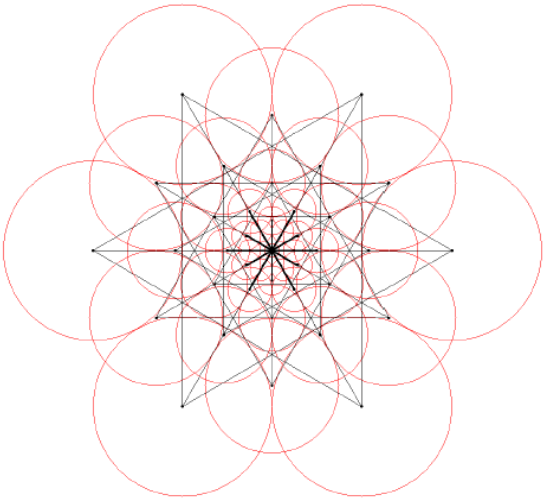
4、选取前 512 列作为最终的二进制描述符。(也可以是 256、128、64、32 等)

小结：最原始的二进制长度为 903，当然这包含了许多冗余或粗糙的信息，所以根据一定的方法取 N 个二进制长度，方法是建立矩阵 D。假如提取到 228 个特征点，那么矩阵 D 应该是 228 行*903 列，然后经过计算均值，将每个列重新排序，选取前 N 列，这个矩阵 D 就是 228*N 的矩阵了。当然这个 N 在文中写得是 512，也可以是 256、128、64、32 这些都是可以的。 最终 D 的每一行仍然是一个特征点的描述符，只是更短小精悍而已，即降低了维度。

由于 FREAK 描述符自身的圆形对称采样结构使其具有旋转不变性，采样的位置好半径随着尺度的变化使其具有尺度不变性，对每个采样点进行高斯模糊，也具有一定的抗噪性能，像素点的强度对比生成二进制描述子使其具有光照不变性。因此由上述产生的二进制描述子可以用来进行特征匹配。在匹配之前，再补充一下特征点的方向信息。

5.4 特征方向

由于特征点周围有 43 个采样点，可产生 43*(43-1)/2=903 个采样点对，FREAK 算法选取其中 45 个长的、对称的采样点对来提取特征点的方向，采样点对如下：



用 O 表示局部梯度信息，M 表示采样点对个数，G 表示采样点对集合，Po 表示采样点对的位置，则：

$$O = \frac{1}{M} \sum_{P_o \in G} (I(P_o^{r_1}) - I(P_o^{r_2})) \frac{P_o^{r_1} - P_o^{r_2}}{\|P_o^{r_1} - P_o^{r_2}\|}$$

同 BRISK 算法，可得到特征点的方向。

5.5 特征匹配

在特征描述中，我们得到了 512bit 的二进制描述符，该描述符的列是高方差——>低方差的排列，而高方差表征了模糊信息，低方差表征了细节信息，与人眼视网膜相似，人眼先处理的是模糊信息，再处理细节信息。因此，选取前 128bit 即 16bytes 进行匹配（异或），若两个待匹配的特征点前 16bytes 距离小于设定的阈值，则再用剩余的位信息进行匹配。这种方法可以剔除掉 90%的不相关匹配点。注意：这里的 16bytes 的选取是建立在并行处理技术（SIMD）上的，并行处理技术处理 16bytes 与处理 1bytes 的时间相同；也就是说，16bytes 并不是固定的，如果你的并行处理技术能处理 32bytes 与处理 1bytes 的时间相同的话，那么你也可以选取前 32bytes。

第五章 附录

1.相关论文

[1] Harris C. A combined corner and edge detector[C]// Proc. of Fourth Alvey Vision Conference. 1988:147--151.

关键词：Harris

下载地址：http://www.gwylab.com/download/Harris_1988.pdf

[2] Edward Rosten and Tom Drummond, “Machine learning for high speed corner detection” in 9th European Conference on Computer Vision, vol. 1, 2006, pp. 430–443.

关键词：FAST

下载地址：http://www.gwylab.com/download/FAST_2006.pdf

[3] Edward Rosten, Reid Porter, and Tom Drummond, “Faster and better: a machine learning approach to corner detection” in IEEE Trans. Pattern Analysis and Machine Intelligence, 2010, vol 32, pp. 105-119.

关键词：FAST-ER

下载地址：http://www.gwylab.com/download/FAST-ER_2010.pdf

[4] Marr D, Hildreth E. Theory of edge detection. Proc. of the Royal Society of London. Series B. Biological Sciences, 1980,207(1167): 187-217.

关键词：LoG, Edge Detection

下载地址：http://www.gwylab.com/download/LoG_1980.pdf

[5] Lowe D G. Object Recognition from Local Scale-Invariant Features[C]// iccv. IEEE Computer Society, 1999:1150.

关键词：SIFT, 1999

下载地址：http://www.gwylab.com/download/SIFT_1999.pdf

[6]Lowe D G. Distinctive Image Features from Scale-Invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(2):91-110.

关键词：SIFT, 2004

下载地址：http://www.gwylab.com/download/SIFT_2004.pdf

[7]Bay H, Tuytelaars T, Gool L V. SURF: speeded up robust features[C]// European Conference on Computer Vision. Springer-Verlag, 2006:404-417.

关键词：SURF

下载地址：http://www.gwylab.com/download/SURF_2006.pdf

[8]Calonder M, Lepetit V, Strecha C, et al. BRIEF: binary robust independent elementary features[C]// European Conference on Computer Vision. 2010:778-792.

关键词：BRIEF

下载地址：http://www.gwylab.com/download/BRIEF_2010.pdf

[9]Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF[C]// International Conference on Computer Vision. IEEE, 2012:2564-2571.

关键词：ORB

下载地址：http://www.gwylab.com/download/ORB_2012.pdf

[10] Leutenegger S, Chli M, Siegwart R Y. BRISK: Binary Robust invariant scalable keypoints[C]// International Conference on Computer Vision. IEEE Computer Society, 2011:2548-2555.

关键词：BRISK

下载地址：http://www.gwylab.com/download/BRISK_2011.pdf

[11]Vandergheynst P, Ortiz R, Alahi A. FREAK: Fast Retina Keypoint[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2012:510-517.

关键词：FREAK

下载地址：http://www.gwylab.com/download/FREAK_2012.pdf

2.相关代码

[代码 1]　Harris 角点检测的 C++实现代码：

https://github.com/RonnyYoung/ImageFeatures/blob/master/source/harris.cpp

[代码 2]　多尺度 Harris 角点检测 C++实现：

https://github.com/RonnyYoung/ImageFeatures/blob/master/source/harrisLaplace.cpp

[代码 3]　斑点检测 SIFT 创始人 Lowe 代码

http://www.cs.ubc.ca/~lowe/keypoints/

[代码 4]　斑点检测 SURF 实现代码

https://github.com/PetterS/surfmex

[代码合辑]　更多计算机视觉开源代码
https://www.cnblogs.com/ydxt/p/6240697.html

3.致谢及引用

声明：本资料全部整理自互联网，仅用作学习。现注明转载出处如下：

致谢：

第一章·概述

1. 图像局部特征点检测算法概述

网址：http://www.cnblogs.com/ronny/p/4260167.html

第二章·角点检测

2. Harris 角点检测子

网址：http://www.cnblogs.com/ronny/p/4009425.html

3. Fast 角点检测子

网址：http://www.cnblogs.com/ronny/p/4078710.html

4. FAST-ER 角点检测子

网址：https://blog.csdn.net/tostq/article/details/49335135

第三章·斑点检测

5.斑点检测之 LOG 算子

网址：http://www.cnblogs.com/ronny/p/3895883.html

6.斑点检测之 SIFT

网址：https://wenku.baidu.com/view/d7edd2464b73f242336c5ffa.html

7.斑点检测之 SURF

网址：https://wenku.baidu.com/view/2f1e4d8ef705cc1754270945.html

第四章·二进制字符串特征描述子

8. BRIEF 描述子

网址：http://www.cnblogs.com/ronny/p/4081362.html

9. ORB 特征提取算法

网址：https://www.cnblogs.com/ronny/p/4083537.html

10. BRISK 特征提取算法

网址：https://blog.csdn.net/hujingshuang/article/details/47045497

11. FREAK 特征提取算法

网址：https://blog.csdn.net/hujingshuang/article/details/47060677

更多推荐·利用 opencv 进行斑点检测的其他方法

12 OPENCV 源码分析——MSER 特征

网址：https://blog.csdn.net/zhaocj/article/details/40742191

13. OPENCV 源码分析——MSCR 特征

网址：https://blog.csdn.net/zhaocj/article/details/43191829

14. OPENCV 源码分析——SimpleBlobDetector

网址：https://blog.csdn.net/zhaocj/article/details/44886475

15. OPENCV 源码分析——DenseFeature

网址：https://blog.csdn.net/zhaocj/article/details/45198965

更多 OPENCV 的源码分析：
https://blog.csdn.net/zhaocj/article/category/2521441/1