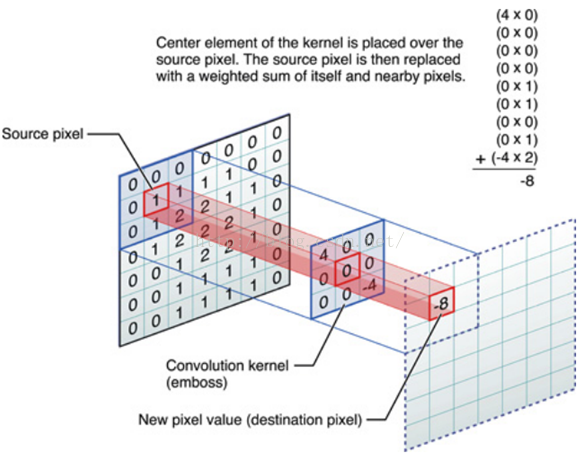
# 图像卷积的基本原理和常见的滤波算法

## 图像卷积滤波的基本原理



对图像和滤波矩阵进行逐个元素相乘再求和的操作就相当于将一个二维的函数移动到另一个二维函数的所有位置，这个操作就叫卷积或者协相关，如上图中间为3\*3的卷积核，通过与图像左上角逐个像素再求和，所得值对应新图像中心位置像素。卷积和协相关的差别是，卷积需要先对滤波矩阵进行180的翻转。

滤波步骤：

1. 对原始图像的边缘进行某种方式的填充（一般为0填充）。
2. 将掩膜划过整幅图像，计算图像中每个像素点的滤波结果。

卷积步骤：

1. 180度翻转卷积核。
2. 不做边界填充，直接对图像进行相应位置乘积和。

从以上步骤可以看出，如果卷积核不是中心对称的，那么卷积和滤波操作将会得到完全不一样的结果。另外，**卷积操作会改变图像大小！**

## 常见的滤波

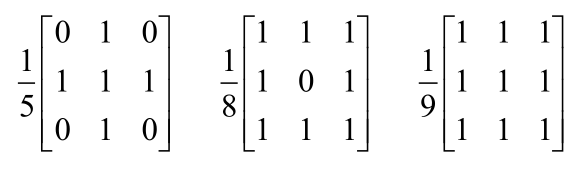
* + 1. **几种噪声的介绍**

滤波主要是为了去噪声，几种噪声简介：<https://www.jianshu.com/p/67f909f3d0ce>

* + 1. **均值滤波**

均值滤波是典型的线性滤波算法，它是指在图像上对目标像素给一个模板，该模板包括了其周围的临近像素，再用模板中的全体像素的平均值来代替原来像素值。平均整个窗口范围内的像素值，即它不能很好地保护图像细节，在图像去噪的同时也破坏了图像的细节部分，从而使图像变得模糊，不能很好地去除噪声点。

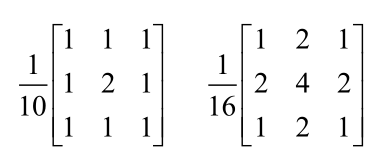
常见3\*3卷积核：



* + 1. **加权平均值滤波**

是对递推平均滤波法的改进，即不同时刻的数据加以不同的权通常是，越接近现时刻的数据，权取得越大。给予新采样值的权系数越大，则灵敏度越高。

常见3\*3卷积核：

****

* + 1. **中值滤波**

将每一[像素](https://baike.baidu.com/item/%E5%83%8F%E7%B4%A0/95084" \t "_blank)点的[灰度值](https://baike.baidu.com/item/%E7%81%B0%E5%BA%A6%E5%80%BC/10259111)设置为该点某邻域窗口内的所有像素点灰度值的[中值](https://baike.baidu.com/item/%E4%B8%AD%E5%80%BC/9661467)，对椒盐噪声有效。方法：

1. 确定窗口及位置(含有奇数个像素)
2. 窗口内像素按灰度大小排序
3. 取中间值代替原窗口中心像素值
   * 1. **形态学滤波**
        1. **形态学滤波的基本操作**

**数学形态学**：形态学被定义为一种分析空间结构的理论，其目的在于分析目标的形状和结构。

**形态学滤波**：由数学形态学的基本预算构成的滤波器叫做形态学滤波器。有选择的抑制图像的结构，那些结构可以是噪声，也可以是不相关的结构目标。

基本运算包含：膨胀、腐蚀、开运算和闭运算。

* + - 1. **膨胀与腐蚀**

按数学方面来说，膨胀或者腐蚀操作就是将图像（或图像的一部分区域，我们称之为A）与核（我们称之为B）进行卷积。

核可以是任何的形状和大小，它拥有一个单独定义出来的参考点，我们称其为锚点（anchorpoint）。多数情况下，核是一个小的中间带有参考点和实心正方形或者圆盘，其实，我们可以把核视为模板或者掩码。

* **膨胀**

膨胀就是求局部最大值的操作，核B与图形卷积，即计算核B覆盖的区域的像素点的最大值，并把这个最大值赋值给参考点指定的像素。这样就会使图像中的高亮区域逐渐增长。

膨胀就有扩大图像的作用，通过膨胀可以让图像裂缝得到填补，如一面破镜子的照片，可以通过膨胀处理恢复成完好的样子。

图1 - 膨胀效果图

* **腐蚀**

腐蚀就是膨胀的反运算，用来求局部最小值的操作。

腐蚀可以收缩图像，消除物体边界点，可以把小于结构元素的物体（如毛刺、小凸起）去除，选取不同大小的结构元素，就可以在原图像中去掉不同大小的物体；腐蚀的最简单的应用是从图中消除不相关的细节

图2 - 腐蚀效果图

总体来说，腐蚀和膨胀是对白色部分（高亮部分）而言的，不是黑色部分。 膨胀就是图像中的高亮部分进行膨胀，“邻域扩张”，效果图拥有比原图更大的高亮区域。腐蚀就是原图中高亮部分被腐蚀，“邻域被蚕食”，效果图拥有比原图更小的高亮区域。

* + - 1. **开运算与闭运算**

**开运算**：先腐蚀后膨胀 （ ），使图像的轮廓变得光滑，断开狭窄的间断和消除细的突出物。

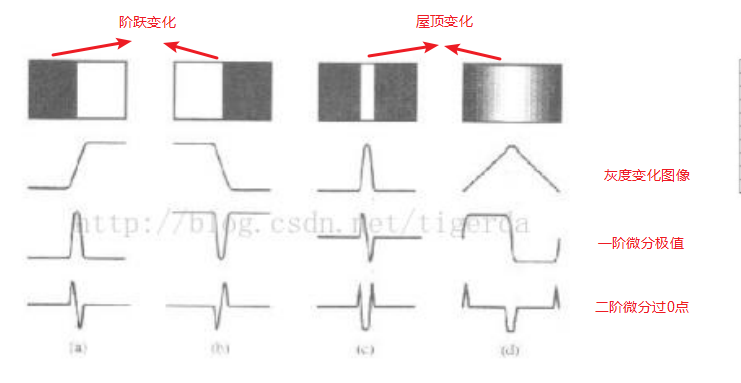
**闭运算**：先膨胀后腐蚀（），同样可使图像的轮廓变得光滑，但与开运算操作相反，它能消除狭窄的尖端和长细的鸿沟，消除小的孔洞，并填补轮廓线中的裂痕。

先开后闭可有效去除噪声。

# 边缘检测

## 边缘检测的基本原理

边缘一般是指图像在某一局部强度剧烈变化的区域，强度变化一般有两种情况：



边缘检测基本原理：

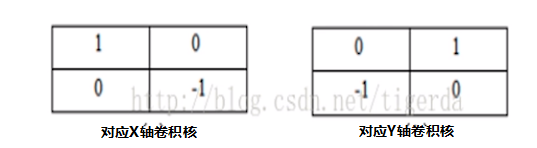
既然边缘是灰度变化最剧烈的位置，最直观的想法就是求微分。

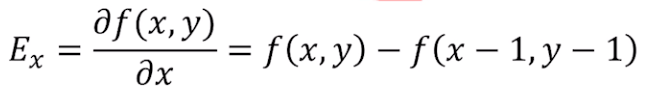
对于阶跃变化：一阶微分的峰值为边缘点，二阶微分的零点为边缘点。

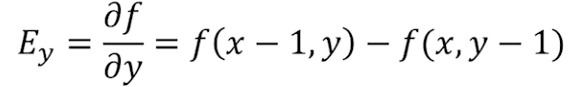
对于屋顶变化：一阶微分的零点为边缘点，二阶微分的峰值为边缘点。

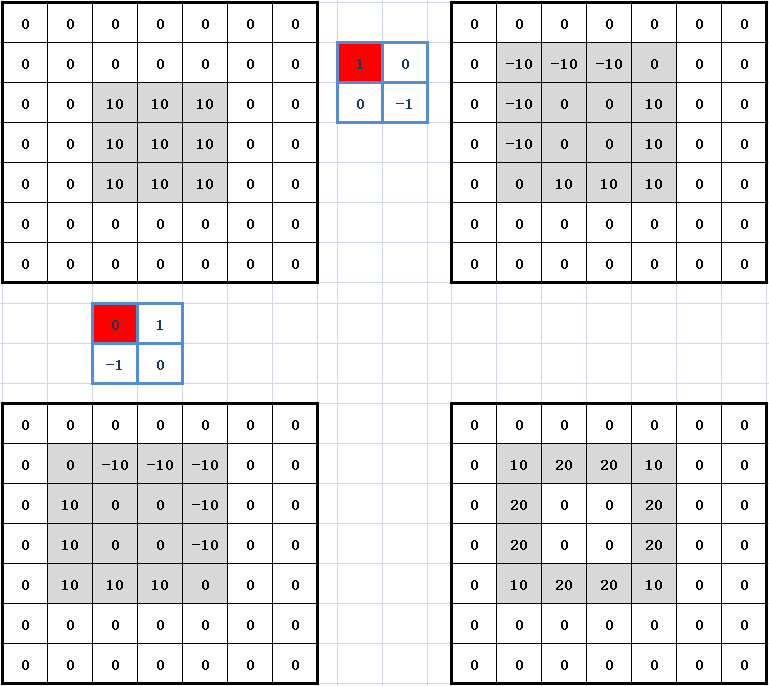
## Robert算子

Robert算子卷积核如下图：



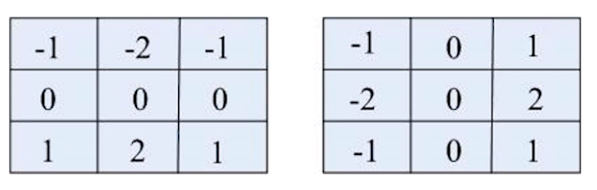
X轴微分：

Y轴微分：



## Sobel算子

Sobel算子卷积核：



## Laplace算子

一个二维图像函数的拉普拉斯变换是各向同性的二阶导数，定义为：

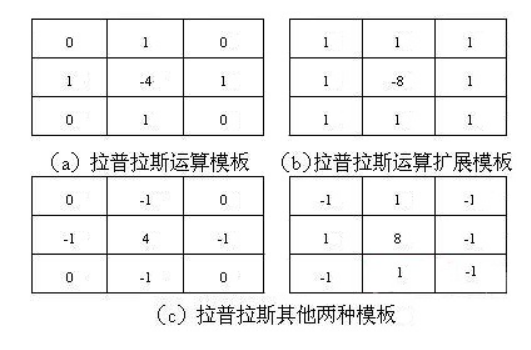


为了更适合于数字图像处理，将该方程表示为离散形式：



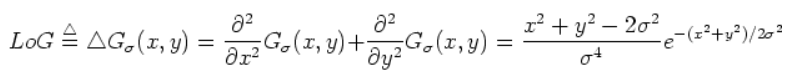
可以理解为图像中上下左右四个点减去中心点的4倍。

Laplace模板：



## LoG算子

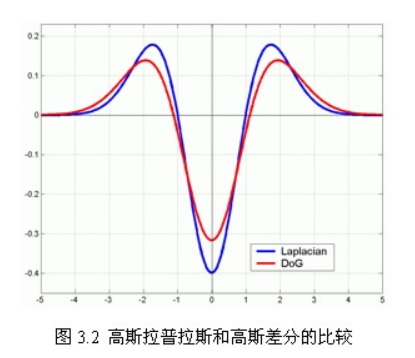
首先对原始图像进行高斯滤波，然后再求laplace计算，LOG核函数定义为：



Laplacian of Gaussian计算可以利用高斯差分来近似，其中差分是由两个高斯滤波与不同变量的卷积结果逼近求得：



该函数逼近是因为发现高斯二阶导和原高斯函数对sigma参数求导之后函数模型的关系如上述的公式，然后约等于右边的导数表达式（严格的话需要加上极限的）！模型逼近如下：好处是可以提高算法的效率减少计算量。



从两个平滑算子的差分得出的是二阶边缘检测，反直观。近似计算可能如下图所示。

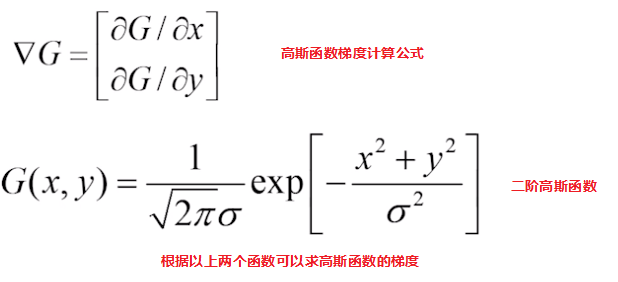
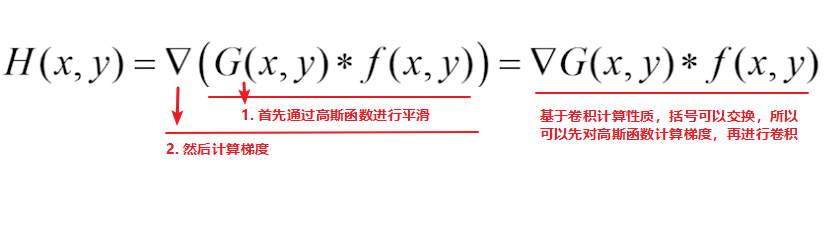
注意最后计算的模板算子的权重和应该保证为1，不是1的可以进行归一化！确保在平塘区不会检测到边沿

## Ganny算子

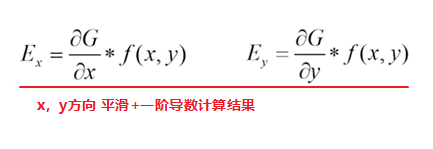
Ganny算子计算步骤：

1. 彩色图像转换为灰度图像
2. 对图像进行平滑与微分

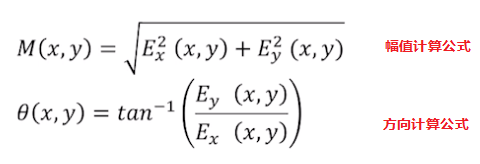
首先使用高斯函数对原始图像进行平滑，然后再计算一阶导数。



1. 计算图像梯度，根据梯度计算图像边缘幅值与角度（用到了微分边缘检测算子来计算梯度幅值方向）

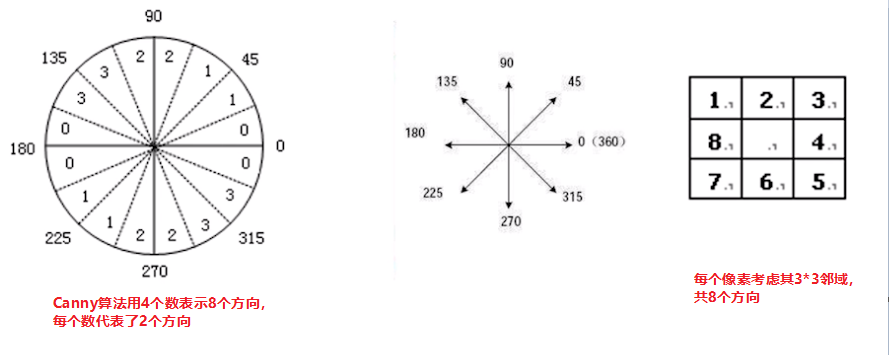


进一步计算出梯度的幅值和方向:



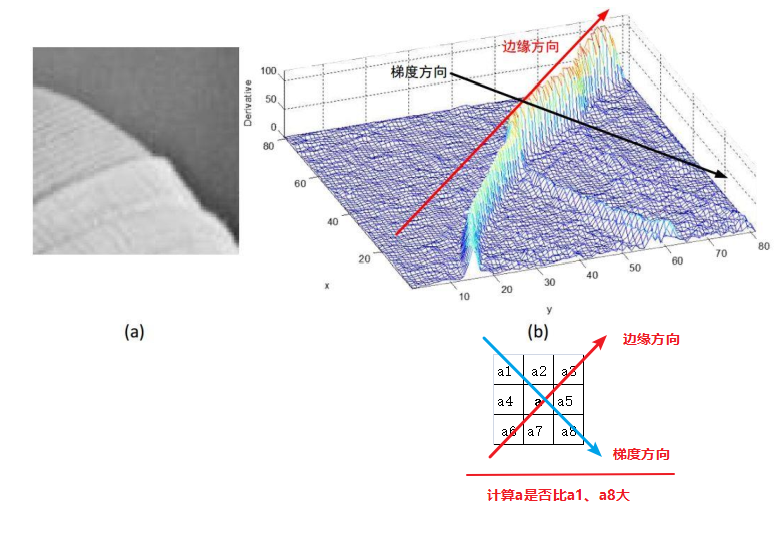
方向离散化：

任何一个像素考虑其3\*3邻域，则共有8个方向；Canny算子也将方向离散成8个方向，分别用0,1,2,3来表示，其中每个数对应2个不同的方向，所以用数来表示的话实际上只有4个。



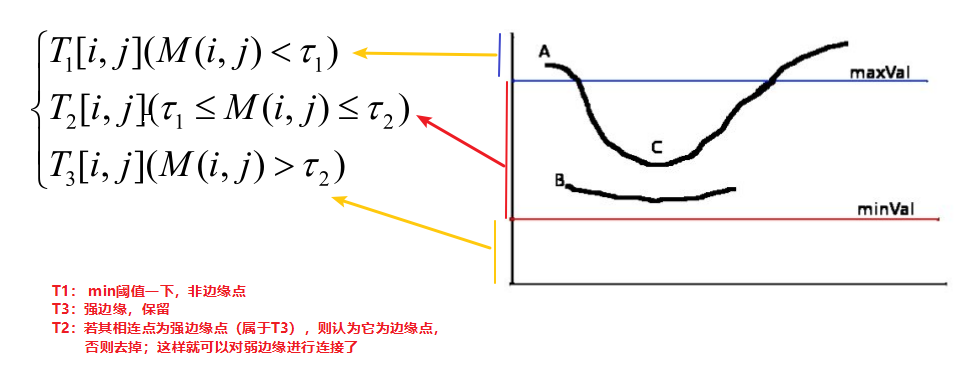
1. 梯度幅值进行非极大值抑制（边缘细化）

细化梯度幅值图像中的屋脊带，只保留幅值局部变化最大的点; 使用一个3\*3邻域作用于幅值阵列的所有点。在每一点上, 邻域的中心像素与沿梯度方向的两个梯度幅值的插值结果进行较，仅保留极大值点。



1. 双阈值边缘连接处理

Canny算法应用双阈值，即一个高阈值和一个低阈值来区分边缘像素。如果边缘像素点梯度值大于高阈值，则被认为是强边缘点。如果边缘梯度值小于高阈值，大于低阈值，则标记为弱边缘点。小于低阈值的点则被抑制掉。



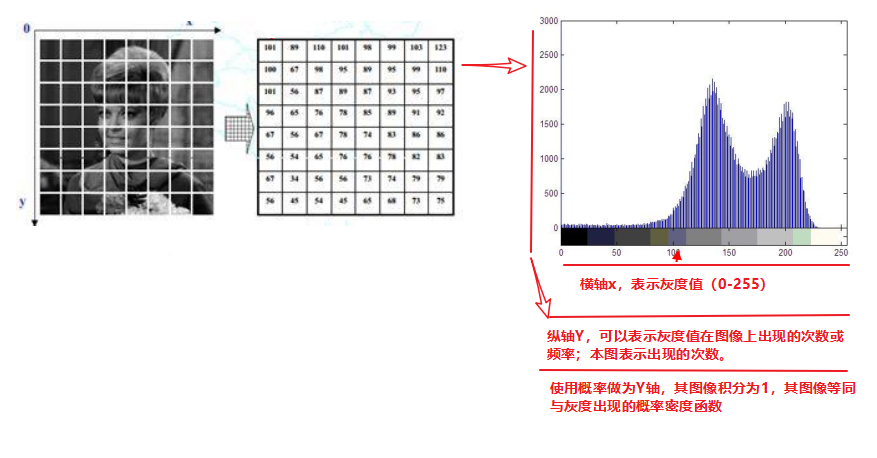
## 各个边缘检测的对比

|  |  |
| --- | --- |
| **算子** | **优缺点比较** |
| Roberts | 对具有头桥的低噪声的图像处理效果较好，但利用Roberts算子提取的边缘结果是边缘比较粗的，因此边缘定位不是很准确。 |
| Sobel | 对灰度渐变和噪声比较多的图像处理效果比较好，Sobel算子对边缘定位比较准确 |
| Laplacian | 对图像中的阶跃性边缘点定位准确，对噪声敏感，丢失一部分边缘的方向信息，造成一些不连续的检测边缘。 |
| LoG | LoG算子经常出现双边缘像素边界，而且检测方法对噪声比较敏感，所以比较少用其检测边缘，而是用来判断边缘像素是否位于图像的明区还是暗区。 |
| Canny | 边缘可自动连接，且此方法不容易受噪声的干扰，能够检测到真正的弱边缘。在edge函数中，最有效的边缘检测方法时Canny方法。该方法的有点在于使用两种不同的阈值分别检测强边缘和弱边缘，并且晋档弱边缘与强边缘相连时，才将弱边缘包含在输出图像中。因此这种方法不容易被噪声填充，更容易检测出弱边缘。 |

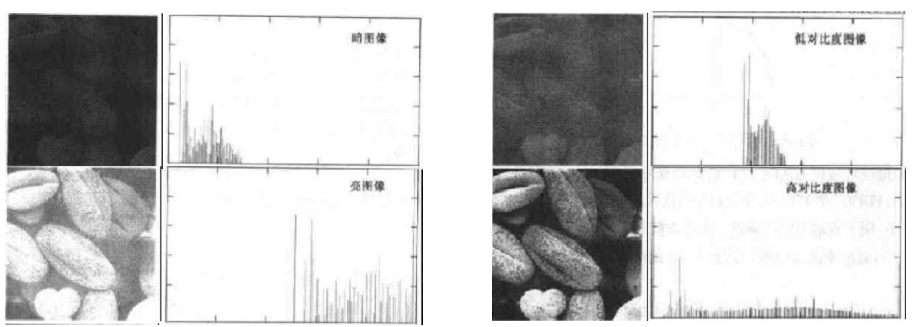
# 图像分割

## 直方图的基本概念

一幅图像由不同灰度值的像素组成，图像中灰度的分布情况是该图像的一个重要特征。图像的灰度直方图就描述了图像中灰度分布情况，能够很直观的展示出图像中各个灰度级所占的多少。图像的灰度直方图是灰度级的函数，描述的是图像中具有该灰度级的像素的个数：其中，横坐标是灰度级，纵坐标是该灰度级出现的频率。



通过灰度直方图看到的图像照明效果：



暗图像：灰度值集中在横轴左侧； 亮度像：灰度值集中在横轴右侧

低对比度：集中在一个区域； 高对比度：均匀分布

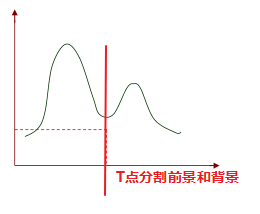
## 灰度阈值分割

前提假设：图像中目标区域和背景区域之间或者不同目标区域之间，存在不同的灰度或平均灰度。

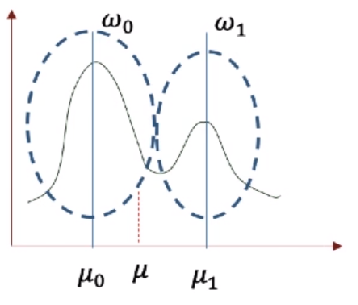
凡是灰度值包含于z的像素都变成某一灰度值，其他的变成另一个灰度值，则该图像就以z为界被分成两个区域，如果=1和=0，分割后的图像为二值图像。

## 大津算法

大津法是一种著名的全局阈值分割方法，其假设前景与背景的的灰度或灰度值不同，则灰度直方图可能为下图：



那么我们需要用程序来确定T点的位置，现在记T为前景与背景的分割阈值，前景点数占图像比例为w0，平均灰度为u0；背景点数占图像比例为w1，平均灰度为u1，图像的总平均灰度为u，前景和背景图象的方差，如下图：



则有：





上述两式联立可得：



如何求得方差g最大：遍历u=（1~254，0和255可不求），求g最大时，u的值既是T。

大津算法的局限性：

* 1. 噪声影响大
  2. 灰度渐变时，分割效果不佳
  3. 仅可对单一目标进行分割，或者感兴趣的目标都属于同一灰度范围，若需探测目标灰度范围分布较大，则必将有一部分目标探测丢失。

## 区域生成法

从种子点开始，按照一定准则(如相邻像素灰度相似性)向周围扩散，将邻域相似像素加入区域中。

区域生长实现步骤：

1. 对图像顺序扫描!找到第1个还没有归属的像素, 设该像素为(x 0 , y 0 );

2. 以(x 0 , y 0 )为中心, 考虑(x 0 , y 0 )的8邻域像素(x, y)，如果(x, y)满足生

长准则, 将(x, y)与 (x 0 , y 0 )合并, 同时将(x, y)压入堆栈;

3. 从堆栈中取出一个像素, 把它当作(x 0 , y 0 )返回到步骤2;

4. 当堆栈为空时，返回到步骤1;

5. 重复步骤1 - 4直到图像中的每个点都有归属时。生长结束。

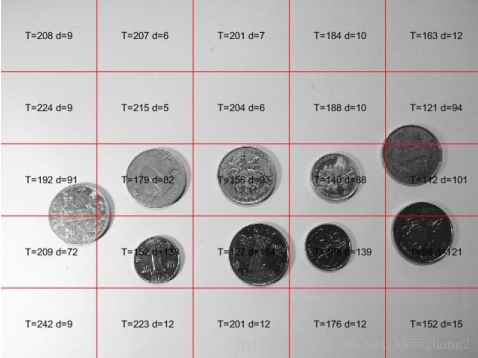
区域生长有广度优先和深度优先两种方式

## 局部阈值法

可以考虑一个问题，如果图像中有多个目标，大津算法是否还可行？如下图图像：



由于大津算法仅对全局图像去一个阈值点，所以无法准确分割多个目标的情况，所以这里有了局部阈值法。其思想就是把图像分割成若干个区域，再进行阈值分割，如下图：



对图像进行5\*5的区域分割，再对每个区域分别进行阈值分割，注意某些区域中可能没有目标，所以计算时可通过计算前景和背景的均值，判断两者距离是否过近，若过近则不需要进行分割了。

局部分割的优点在于可以进行多目标分割，缺点在于基于局部阈值分割出的目标连结性较差，包含噪声。

添加对比图

## 分水岭算法分割

图像中的目标物体是连接在一起的，则分割起来会更困难，分水岭分割算法经常用于处理这类问题。

由于噪声点或其他干扰因素，分水岭算法常存在过度分割的现象，这是因为很多很小的局部极值点的存在。为了解决这样的问题，可以选择一个区域作为种子区域。

## 分割算法的对比

区域生长法和大津算法都是针对单目标而言的，而局部阈值分割法和分水岭算法是针对多目标的分割算法。

|  |  |
| --- | --- |
| **算子** | **优缺点比较** |
| 大津算法 | 大津法的优点在于可以快速有效的找到类间分割阈值，但其缺点也很明显，就是只能针对单一目标分割，或者感兴趣的目标都属于同一灰度范围，若需探测目标灰度范围分布较大，则必将有一部分目标探测丢失 |
| 局部阈值分割 | 局部分割的优点在于可以进行多目标分割，缺点在于基于局部阈值分割出的目标连结性较差，包含噪声。 |
| 区域生长法 |  |
| 分水岭算法 |  |

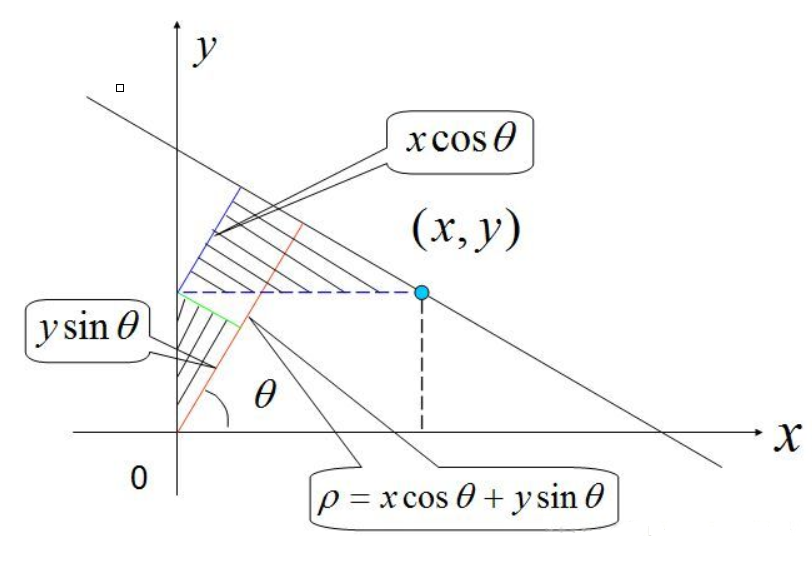
# 直角检测（hough变换）

## Hough基本原理

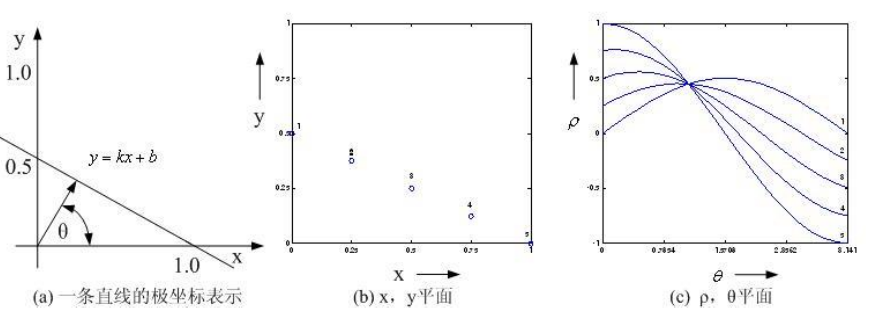
1、对于直角坐标系中的任意一点A(x0,y0)，经过点A的直线满足Y0=k\*X0+b.(k是斜率，b是截距)

2、那么在X-Y平面过点A(x0,y0)的直线簇可以用Y0=k\*X0+b表示，但对于垂直于X轴的直线斜率是无穷大的则无法表示。因此将直角坐标系转换到极坐标系就能解决该特殊情况。

3、在极坐标系中表示直线的方程为ρ=xCosθ+ySinθ（ρ为原点到直线的距离）,如图所示：

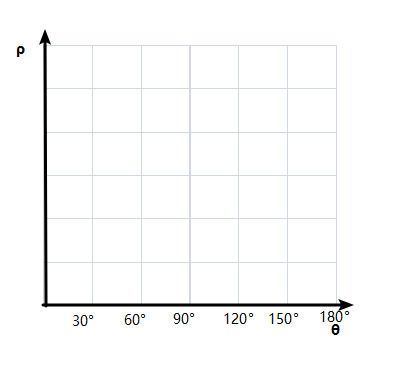


同一条直线上的多个点，在极坐标系下必相交于一点



## Hough变换计算步骤

1. 将（**ρ，θ**）空间量化成许多小格，如下图：



注意：实际中，其θ的间隔会比较小。

1. 根据x-y平面每一个直线点代入θ的量化值，算出各个ρ，将对应格计数累加。

（x-y为边缘检测后的图像中的边缘点，根据公式，已知θ，x，y求ρ）

1. 当全部点变换后，对小格进行检验。设置累计阈值T，计数器大于T的小格对应于共线点，其可以用作直线拟合参数。小于T的反映非共线点，丢弃不用

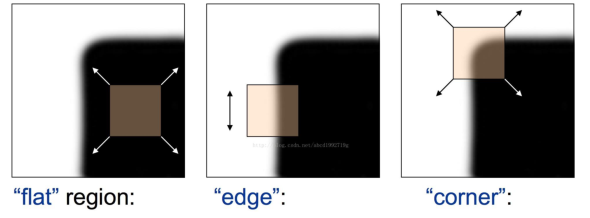
# Harris角点检测

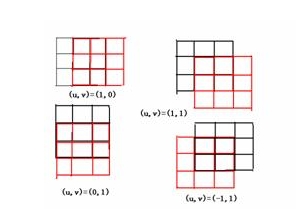
作业4-简述Harris算子对角点的定义，进行角点检测的基本原理，并说明引入角点响应函数的意义。

## Moravec角点检测的基本原理

 Moravec角点检测算子的思想其实特别简单，在图像上取一个W\*W的“滑动窗口”，不断的移动这个窗口并检测窗口中的像素变化情况E。在灰度变化平缓区域，窗口内像素灰度积分（灰度累加值）近似保持不变。像素变化情况E可简单分为以下三种：

* 平坦区域（flat）：灰度积分近似不变；
* 在一条边上（edge）：沿这条边滑动时E变化不大，而在沿垂直于这条边的方向滑动窗口时，E的变化会很大；
* 在角点处（corner）：任意方向均剧烈变化；





Moravec算子对四个方向进行加权求和来确定变化的大小，然和设定阈值，来确定到底是边还是角点。

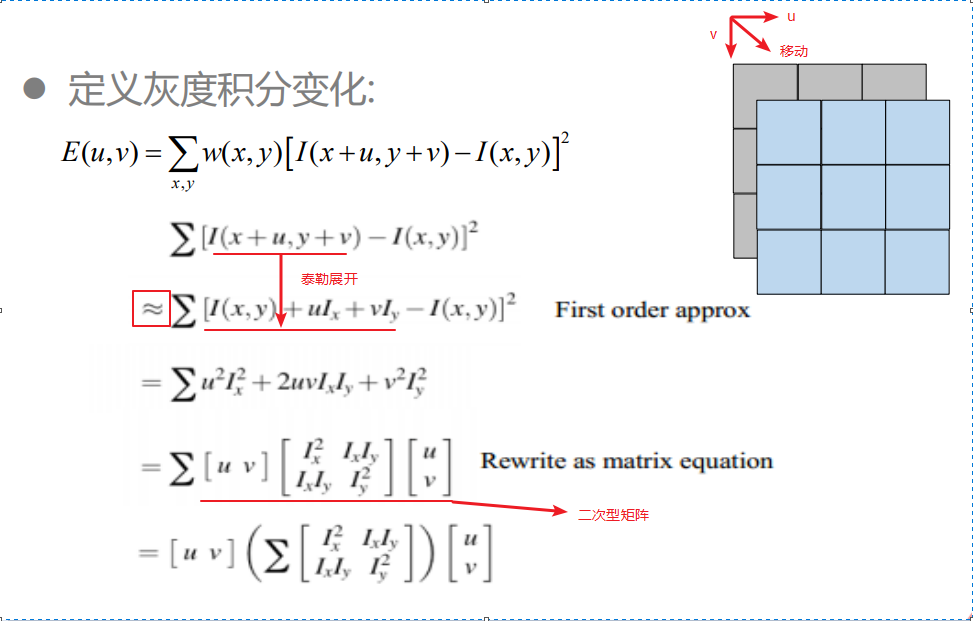
## Harris角点检测算子

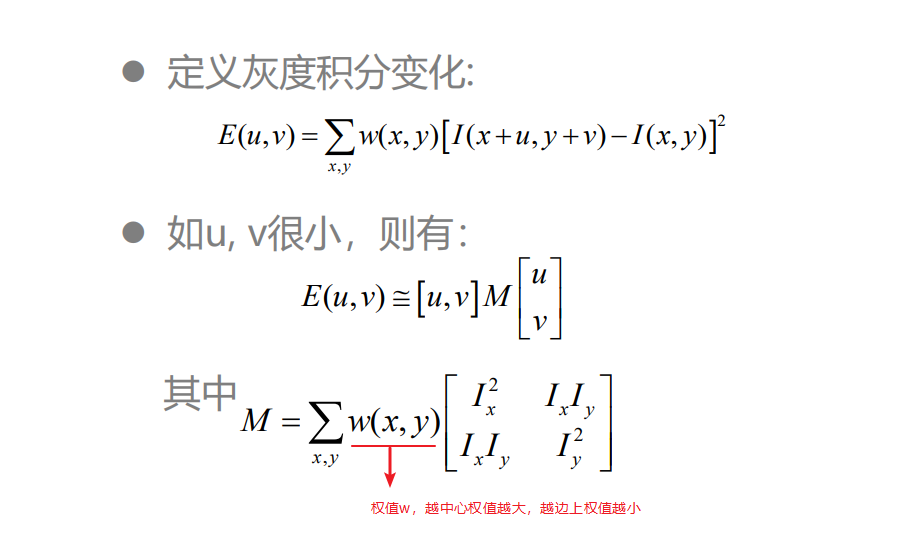
Harris角点检测算子实质上就是对Moravec算子的改良和优化。

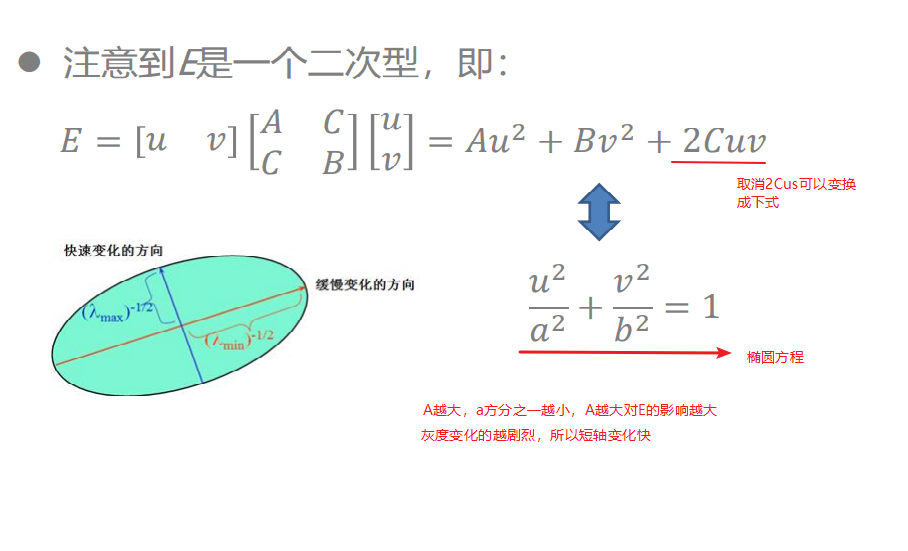
1. Moravec算子对方向的依赖性太强，在上文中我们可以看到，Moravec算子实际上只是移动了四个45度角的离散方向，真正优秀的检测算子应该能考虑到各个现象的移动变化情况

2.由于Moravec算子采用的是方形的windows，因此的E的响应比较容易受到干扰，Harris采用了一个较为平滑的窗口——高斯函数；

**3**.Moravec算子对边缘响应过于灵敏。为此，Harris提出了对E进行变形：



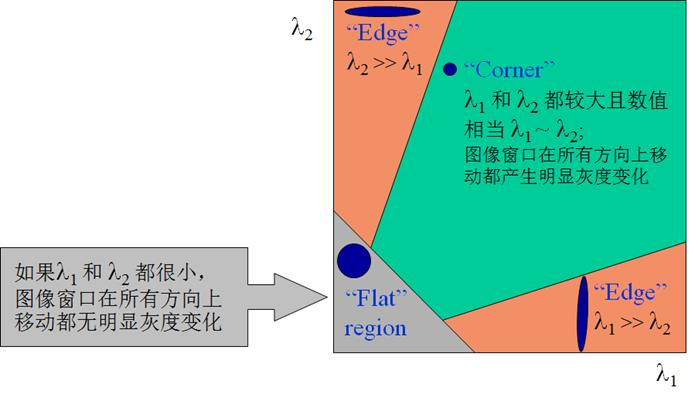




参看上图， 和 是椭圆的长短轴，则有：

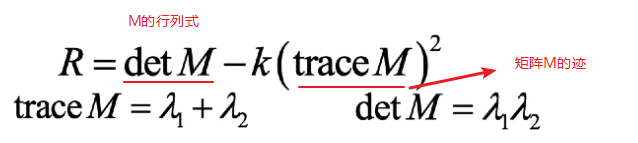
* 当 和 都比较小时，点（x,y）处于灰度平缓区域
* 当远大于，或者反之，点（x,y）为边界像素
* 当 和 都比较大是，且近似相等时，点（x,y）为角点。

如下图：



那如何定义和是小还是大呢？

使用角点函数：



* 当R接近于零时，处于灰度变化平缓区域；
* 当R<0时，点为边界像素；
* 当R>0时，点为角点。

其中k通常取0.04~0.045

Hough变换想法直接，易于实现，是其它角点提取算法的基础

# SIFT

## SIFT算法的特征

* 图像的局部特征，对旋转、尺度缩放、亮度发化保持不变，对视角发化、仿射发换、噪声也 保持一定程度的稳定性。
* 独特性好，信息量丰富，适用于海量特征库进行快速、准确的匹配。
* 多量性：即使是很少几个物体也可以产生大量的SIFT特征
* 高速性：改进的SIFT匹配算法甚至可以达到实时性
* 扩展性：可以很方便的不其他的特征向量进行联合。

## 尺度空间

人眼可自动调节尺度，完成对物体的检测和识别。比如例子比如Google地图，滑动鼠标轮可以改变观测地图的尺度，看到的地图绘制也不同；还有电影中的拉伸镜头等等

用机器视觉系统分析未知场景时，计算机并不知道图像中物体的尺度。我们需要同时考虑图像在多尺度下的描述，获知感兴趣物体的最佳尺度。另外如果不同的尺度下都有同样的关键点，那么在不同的尺度的输入图像下就都可以检测出来关键点匹配，也就是尺度不变性。图像的尺度空间表达就是图像在所有尺度下的描述。

尺度空间可以通过高斯凼数不原图像卷积，幵经过下采样，可建立原始图像的尺度空间模型。

* + 1. **金字塔模型**

金字塔是早期图像多尺度的表示形式。图像金字塔化一般包括两个步骤：使用低通滤波器平滑图像；对平滑图像进行降采样（通常是水平，竖直方向1/2），从而得到一系列尺寸缩小的图像。

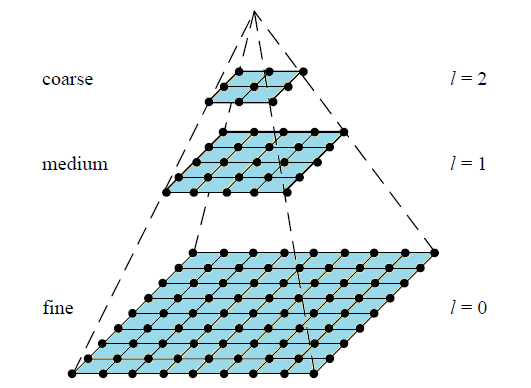


上图中（a）是对原始信号进行低通滤波，（b）是降采样得到的信号。

原图像为金子塔的第一层，每次降采样所得到的新图像为金字塔的下一层(每层一张图像)，每个金字塔共 n 层。金字塔的层数根据图像的原始大小和塔顶图像的大小共同决定，其计算公式如下：



其中 M ， N 为原图像的大小, t 为塔顶图像的最小维数的对数值。如，对于大小为512\*512的图像，当塔顶图像为4\*4时，n=7，当塔顶图像为2\*2时，n=8。

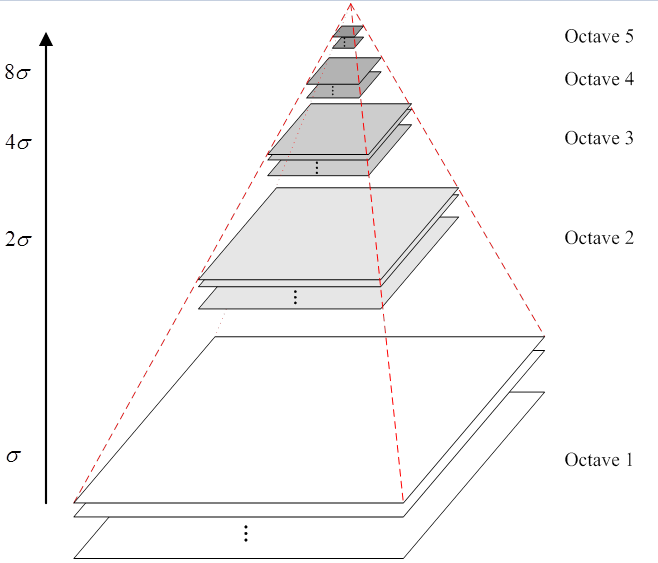


* + 1. **高斯金字塔构建（尺度空间表示）**

尺度空间在实现时使用高斯金字塔表示，高斯金字塔的构建分为两部分：

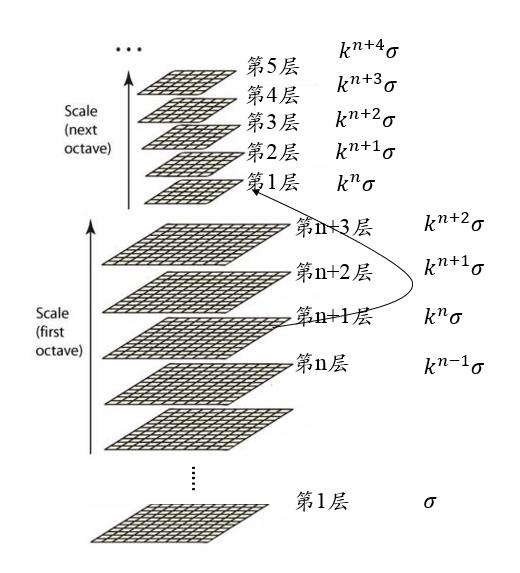
* 对图像做不同尺度的高斯模糊（后面详细介绍）；
* 对图像做降采样(隔点采样，水平，竖直方向1/2)。

高斯金字塔在多分辨率金字塔简单降采样基础上加了高斯滤波，也就是对金字塔每层图像用不同参数的σ做高斯模糊，使得每层金字塔有多张高斯模糊图像。金字塔每层多张图像合称为一组（Octave），每组有多张（也叫层Interval）图像。如下图：



另外，降采样时，金字塔上边一组图像的第一张图像（最底层的一张）是由前一组（金字塔下面一组）图像的倒数第三张隔点采样得到。

如下图：



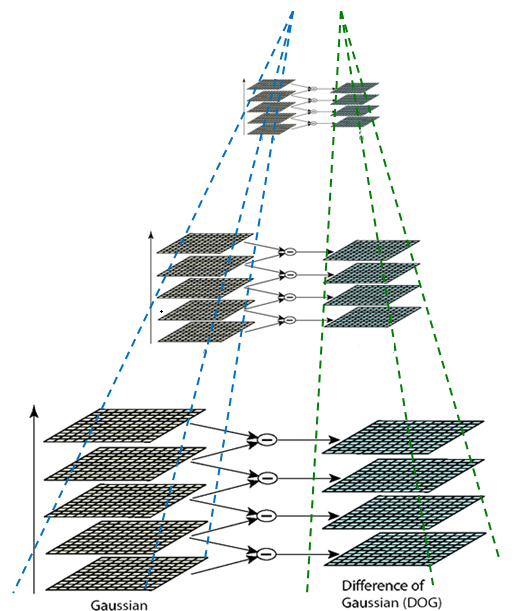
一个图像的尺度空间L(x,y,σ)，定义为一个发化尺度的高斯凼数G(x,y,σ)不原图像I(x,y)的卷积。 L(x,y,σ)=G(x,y,σ)\*I(x,y)，其中\*表示卷积运算。

https://img-my.csdn.net/uploads/201210/13/1350131032_7776.png

σ 是尺度空间因子，值越小表示图像被平滑的越少，相应的尺度也就越小。大尺度对应于图像的概貌特征，小尺度对应于图像的细节特征。

* + 1. **高斯差分金字塔（DoG）**

构建高斯金字塔之后，就是用金字塔相邻图像相减构造DoG金字塔。



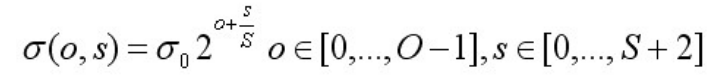
综上，构建尺度空间所需参数：

σ-尺度空间坐标

 O—组(octave)数

 S— 组内层数

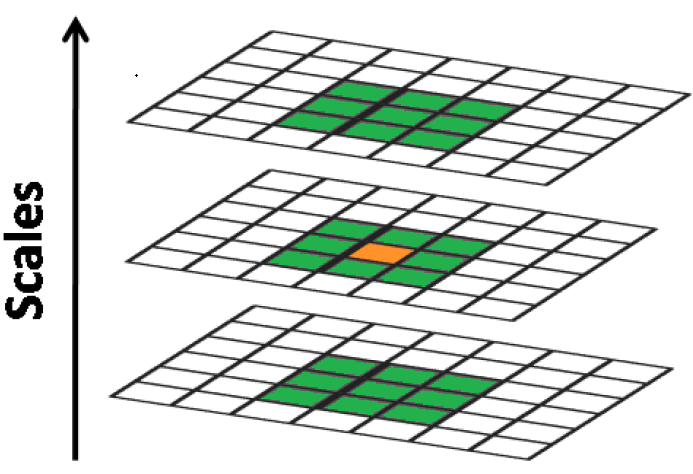
 在上述尺度空间中σ，O和S的关系如下：



 其中σ 0 是基准层尺度，o为组(octave)的索引，s为组内层的索引。关键点的尺度坐标就是按关键点所在的组和组内的层，利用上述公式计算而来。

* + 1. **关键点检测**
       1. **DoG局部极值点**

寻找DoG极值点时，每一个像素点和它所有的相邻点比较，当其大于（或小于）它的图像域和尺度域的所有相邻点时，即为极值点。如下图所示，比较的范围是个3×3的立方体：中间的检测点和它同尺度的8个相邻点，以及和上下相邻尺度对应的9×2个点——共26个点比较，以确保在尺度空间和二维图像空间都检测到极值点。

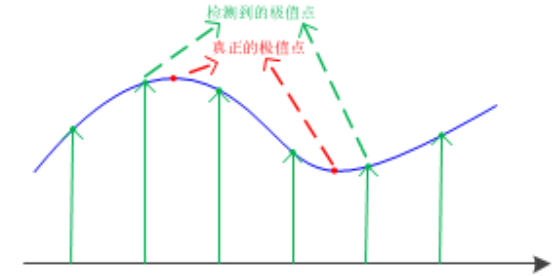


在一组中，搜索从每组的第二层开始，以第二层为当前层，第一层和第三层分别作为立方体的的上下层；搜索完成后再以第三层为当前层做同样的搜索。所以每层的点搜索两次。通常我们将组Octaves索引以-1开始，则在比较时牺牲了-1组的第0层和第N组的最高层。

为了在每组中检测 S 个尺度的极值点，则DOG金字塔每组需S+2层图像，而DOG金字塔由高斯金字塔相邻两层相减得到，则高斯金字塔每组需S+3层图像，实际计算时 S 在3到5之间。

* + - 1. **关键点精确定位**

以上极值点的搜索是在离散空间进行搜索的，由下图可以看到，在离散空间找到的极值点不一定是真正意义上的极值点。可以通过对尺度空间DoG函数进行曲线拟合寻找极值点来减小这种误差。



（待补充）

* + - 1. **删除**[**边缘效应**](https://www.baidu.com/s?wd=%E8%BE%B9%E7%BC%98%E6%95%88%E5%BA%94&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)

除了DoG响应较低的点，还有一些响应较强的点也不是稳定的特征点。DoG对图像中的边缘有较强的响应值，所以落在图像边缘的点也不是稳定的特征点。

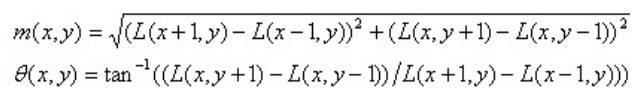
（待补充）

* + 1. **方向直方图**
       1. **梯度幅值**

在前文中，精确定位关键点后也找到改特征点的尺度值σ，根据这一尺度值，得到最接近这一尺度值的高斯图像：



使用有限差分，计算以关键点为中心，以3×1.5σ为半径的区域内图像梯度的幅角和幅值，公式如下：



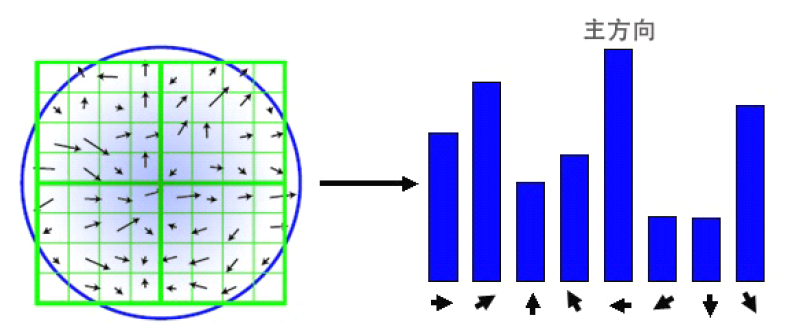
* + - 1. **方向直方图的构建**

在完成关键点邻域内高斯图像梯度计算后，使用直方图统计邻域内像素对应的梯度方向和幅值。

直方图可以看做是离散点的概率表示形式。此处方向直方图的核心是统计以关键点为原点，一定区域内的图像像素点对关键点方向生成所作的贡献。

梯度方向直方图的横轴是梯度方向角，纵轴是剃度方向角对应的梯度幅值累加值。梯度方向直方图将0°~360°的范围分为36个柱，每10°为一个柱。下图是从高斯图像上求取梯度，再由梯度得到梯度方向直方图的例图。

将梯度方向直方图中纵坐标最大的项代表的方向分配给当前关键点作为主方向，若在梯度直方图中存在一个相当于主峰值80%能量的峰值，则将这个方向认为是关键点的辅方向。辅方向的设计可以增强匹配的鲁棒性，Lowe指出，大概有15%的关键点具有辅方向，而恰恰是这15%的关键点对稳定匹配起到关键作用。

****

* + 1. **高斯模糊**

高斯核是唯一可以产生多尺度空间的核（《Scale-space theory: A basic tool for analysing structures at different scales》）。一个图像的尺度空间L(x,y,σ) ,定义为原始图像I(x,y)与一个可变尺度的2维高斯函数G(x,y,σ)卷积运算。

二维空间高斯函数：

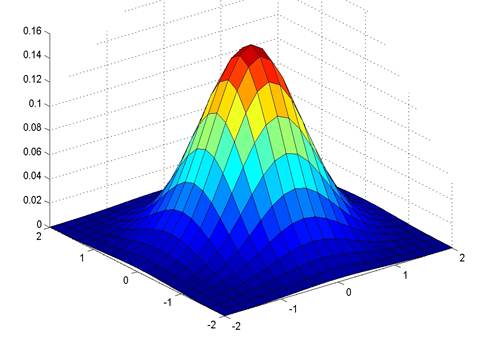
https://img-my.csdn.net/uploads/201210/13/1350131032_7776.png

其中， σ 是正态分布的标准差， σ 值越大，图像越模糊(平滑)。 r 为模糊半径，模糊半径是指模板元素到模板中心的距离

尺度空间：

https://img-my.csdn.net/uploads/201210/13/1350131056_3601.png

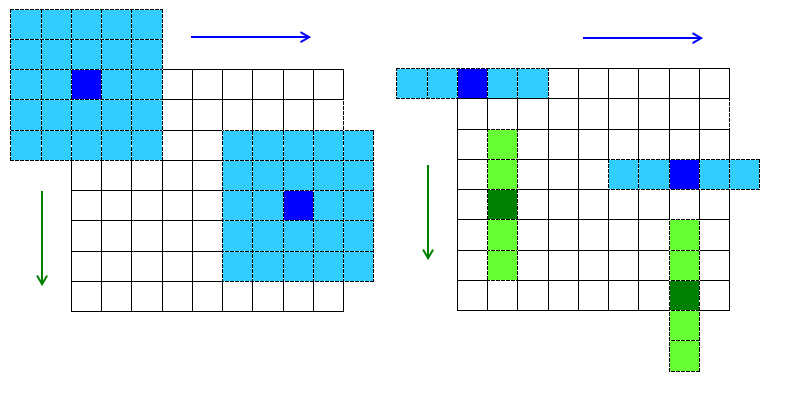
二维空间高斯函数是等高线从中心成正态分布的同心圆：



高斯模版是圆对称的，且卷积的结果使原始像素值有最大的权重，距离中心越远的相邻像素值权重也越小。

在实际应用中，在计算高斯函数的离散近似时，在大概3σ***距离***之外的像素都可以看作不起作用，这些像素的计算也就可以忽略。所以，通常程序只计算(6σ+1)\*(6σ+1)就可以保证相关像素影响。

高斯模糊另一个很厉害的性质就是线性可分：使用二维矩阵变换的高斯模糊可以通过在水平和竖直方向各进行一维高斯矩阵变换相加得到，O(N^2\*m\*n)次乘法就缩减成了O(N\*m\*n)+O(N\*m\*n)次乘，减少计算量。（N为高斯核大小，m,n为二维图像高和宽）



一个5\*5的高斯模版如下所示：



参考：

<https://blog.csdn.net/xiaowei_cqu/article/details/8069548>

<https://blog.csdn.net/samkieth/article/details/50407655>

# ORB算子

ORB（Oriented FAST and Rotated BRIEF）是一种快速特征点提取和描述的算法。

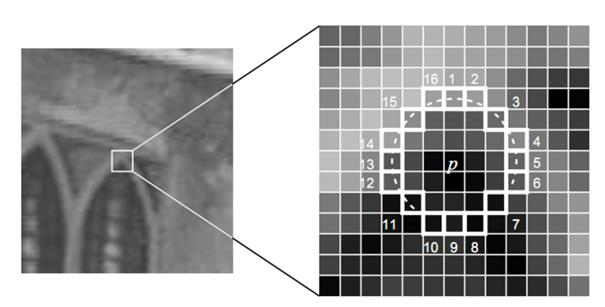
## oFast算法

ORB算法的特征提取是由FAST算法改进的，这里称为oFAST（FAST keypoint Orientation）。在使用FAST提取出特征点之后，给其定义一个特征点斱向，以此来实现特征点的旋转不变性。

* + 1. **Fast算法**

判断特征点：从图像中选取一点P，以P为圆心画一个半径为3像素的圆。圆周上如果有连续N个像素点的灰度值比P点的灰度值大或小，则认为P为特征点 。 这就是大家经常说的FAST-N。有FAST-9、FAST-10、FAST-11、FAST-12，大家使用比较多的是FAST-9和FAST-12。

快速算法：为了加快特征点的提取，快速排出非特征点，首先检测1、5、9、13位置上的灰度值，如果P是特征点，那么这四个位置上有3个或3个以上的的像素值都大于或者小于P点的灰度值。如果不满足，则直接排出此点。



* + 1. **筛选最优点**

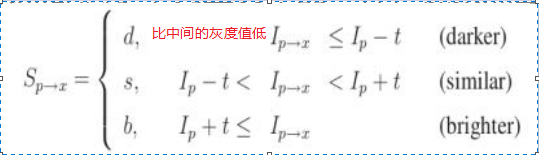
通过Fast算法检测出的特征点有很多误检，我们采用机器学习的方法选取最优特征点。简单来说就是使用ID3算法训练一个决策树，将特征点圆周上的16个像素输入决策树中，以此来筛选出最优的FAST特征点。具体步骤如下：

1. 选取进行角点提取的应用场景下的一组训练图像。

2. 使用FAST角点检测算法找出训练图像上的所有角点。

3. 对于每个角点，将其周围的16个像素存储成一个向量。对所有像素都这样做构建一个特征向量。

4. 每一个角点的16像素点都属于下列三类中的一种，像素点因此被分成三个子集: P d , P s , P b



6. 定义一个新的布尔变量 K p ，如果是角点就设置为True，否则就设置为False。

7. 使用机器学习中ID3算法来查询每一个子集。

8. 递归计算所有的子集直到它的熵为0

被构建好的决策树用于其它图像的FAST检测。

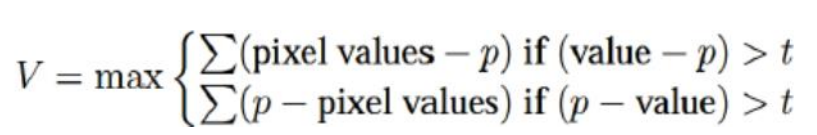
* + 1. **非极大值抑制**

非极大值抑制用以解决临近位置多个特征点的问题。

1. 计算特征点出的FAST得分 值s（像素点与周围16个像素点差值的绝对值之和）

2. 以特征点p为中心的一个邻域（如3x3戒5x5）内，若有多个特征点，则判断每个特征点的s值

3. 若p是邻域所有特征点中响应值最大的，则保留；否则，抑制。若邻域内只有一个特征点，则保留。得分计算公式如下（公式中用V表示得分，t表示阈值）：



* + 1. **提取尺度不变性（金字塔方式）**

1. 设置一个比例因子scaleFactor（opencv默认为1.2）和金字塔的层数nlevels（Opencv默认为8）
2. 将原图像按比例因子缩小成nlevels幅图像。
3. 缩放后的图像为：I’= I/scaleFactor k (k=1,2,…, nlevels)。nlevels幅不同比例的图像提取特征点总和作为这幅图像的oFAST特征点。
   * 1. **提取旋转不变性**

以当前点为圆心，R为半径做圆，圆心到质心的方向作为当前点的方向特征。

* + - 1. **质心**

（待补充）

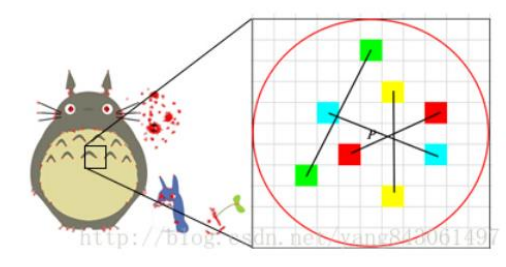
## rBRIEF算法

ORB算法的特征描述是由BRIEF算法改进的，这里称为rBRIEF（Rotation-Aware Brief）。也就是说，在BRIEF特征描述的基础上加入旋转因子从而改进BRIEF算法。

* + 1. **BRIEF算法**

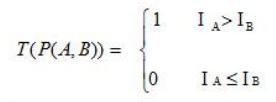
BRIEF是对已检测到的特征点进行描述，它是一种二进制编码的描述子，摈弃了利用区域灰度直方图描述特征点的传统方法，大大的加快了特征描述符建立的速度，同时也极大的降低了特征匹配的时间，是一种非常快速，很有潜力的算法。

BRIEF的基本原理及时在特征点周围区域内，取N对像素点比较灰度值；N对像素点分为2组，p i，q I （i=1,2,…,n）；如果 I ( p i )> I ( q i )，则生成二进制串中的1，否则为0。

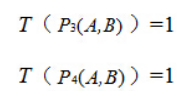
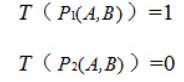


算法步骤如下：

1. 以关键点P为圆心，以d为半径做圆O。
2. 在圆O内某一模式选取N个点对。这里为斱便说明，N=4，实际应用中N可以取512.假设当前选取的4个点对如上图所示分别标记为：。
3. 定义T操作：IA大于IB为1，IA小于等于IB为0，其中IA表示A点的灰度值。



1. 分别对已选取的点对进行T操作，将得到的结果进行组合。

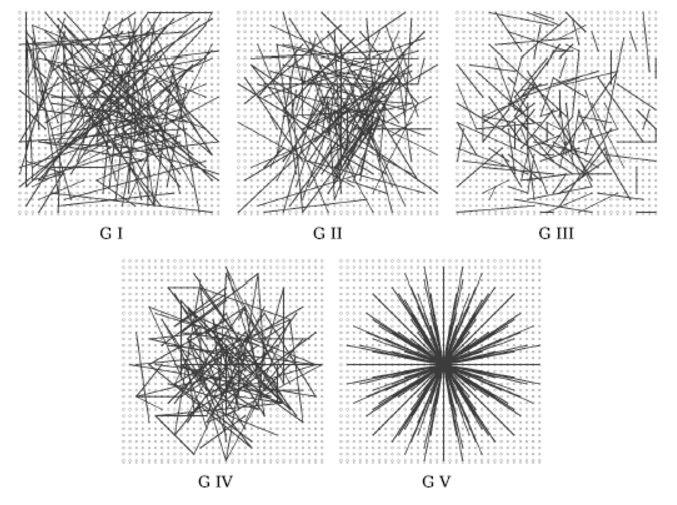
假如：

则最终的描述子为：1011

* + 1. **BRIEF算法对点选择方法**

BRIEF论文中给出的关于在特征点SxS的区域内选取点对的方法：

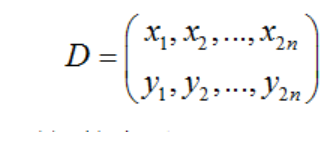
1. 在图像块内平均采样；
2. p 和 q 都符合(0,S 2 /25)的高斯分布；
3. p 符合(0,S 2 /25)的高斯分布，而 q 符合(0,S 2 /100)的高斯分布；
4. 在空间量化极坐标下的离散位置随机采样；
5. 把 p 固定为(0,0)， q 在周围平均采样。



论文中指出，第二种算法能获得较好的匹配结果，在旋转不是非常厉害的图像里，用BRIEF生成的描述子的匹配质量非常高，作者测试的大多数情况中都超越了SURF。但在旋转大于30°后，BRIEF的匹配率快速降到0左右，因此我们需要对BRIEF进行改进。

* + 1. **Steered BRIEF（旋转不变性改进）**

在使用oFast算法计算出的特征点中包括了特征点的方向角度。假设原始的BRIEF算法在特征点SxS（一般S取31）邻域内选取n对点集。

****

经过旋转角度θ旋转（θ是特征点的方向角度吗，如何取？），得到新的点对:



(注意这里是对点对进行的旋转，旋转后的点对会落在新图像的新位置上，形成新的点对)

在新的点集位置上比较点对的大小形成二进制串的描述符。这里需要注意的是，在使用oFast算法是在不同的尺度上提取的特征点。因此，在使用BRIEF特征描述时，要将图像转换到相应的尺度图像上，然后在尺度图像上的特征点处取SxS邻域，然后选择点对并旋转，得到二进制串描述符。

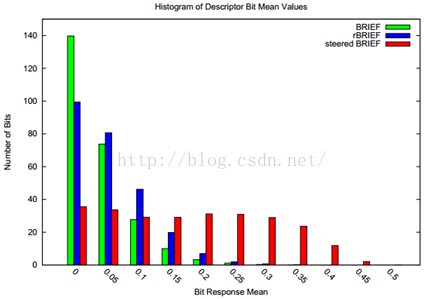
* + 1. **BRIEF改进特征点描述子的相关性**

使用steeredBRIEF方法得到的特征描述子具有旋转不变性，但是却在另外一个性质上不如原始的BRIEF算法。是什么性质呢，是描述符的可区分性，或者说是相关性。这个性质对特征匹配的好坏影响非常大。描述子是特征点性质的描述。描述子表达了特征点不同于其他特征点的区别。我们计算的描述子要尽量的表达特征点的独特性。如果不同特征点的描述子的可区分性比较差，匹配时不容易找到对应的匹配点，引起误匹配。

好的特征描述是，所有特征点上的每一位上的值的平均值非诚接近0.5。相关性计算如下图：



ORB论文中，作者用不同的方法对100k个特征点计算二进制描述符，对这些描述符进行统计，如下表所示（其中X轴代表位均值与0.5的距离，y轴是相应均值下的特征点数量统计）：



对BRIEF和steered BRIEF两种算法的比较可知，BRIEF算法落在0上的特征点数较多，因此BRIEF算法计算的描述符的均值在0.5左右，每个描述符的方差较大，可区分性较强。而steeredBRIEF失去了这个特性。至于为什么均值在0.5左右，方差较大，可区分性较强的原因，这里大概分析一下。这里的描述子是二进制串，里面的数值不是0就是1，如果二进制串的均值在0.5左右的话，那么这个串有大约相同数目的0和1，那么方差就较大了。用统计的观点来分析二进制串的区分性，如果两个二进制串的均值都比0.5大很多，那么说明这两个二进制串中都有较多的1时，在这两个串的相同位置同时出现1的概率就会很高。那么这两个特征点的描述子就有很大的相似性。这就增大了描述符之间的相关性，减小之案件的可区分性。

下面我们介绍解决上面这个问题的方法：rBRIEF。

原始的BRIEF算法有5种种取点对的方法，原文作者使用了方法2。为了解决描述子的可区分性和相关性的问题，ORB论文中没有使用5种方法中的任意一种，而是使用统计学习的方法来重新选择点对集合。

首先建立300k个特征点测试集。对于测试集中的每个点，考虑其31x31邻域。这里不同于原始BRIEF算法的地方是，这里在对图像进行高斯平滑之后，使用邻域中的某个点的5x5邻域灰度平均值来代替某个点对的值，进而比较点对的大小。这样特征值更加具备抗噪性。另外可以使用积分图像加快求取5x5邻域灰度平均值的速度。

从上面可知，在31x31的邻域内共有(31-5+1)x(31-5+1)=729个这样的子窗口，那么取点对的方法共有M=265356种，我们就要在这Ｍ种方法中选取256种取法，选择的原则是这256种取法之间的相关性最小。怎么选取呢？

1）在300k特征点的每个31x31邻域内按M种方法取点对，比较点对大小，形成一个300kxM的二进制矩阵Q。矩阵的每一列代表300k个点按某种取法得到的二进制数。

2）对Q矩阵的每一列求取平均值，按照平均值到0.5的距离大小重新对Q矩阵的列向量排序，形成矩阵T。

3）将T的第一列向量放到R中。

4）取T的下一列向量和R中的所有列向量计算相关性，如果相关系数小于设定的阈值，则将T中的该列向量移至R中。

5）按照4）的方式不断进行操作，直到R中的向量数量为256。

综上所述，rBRIEF算法就是