计算机视觉中的特征点提取算法比较多，但SIFT除了计算比较耗时以外，其他方面的优点让其成为特征点提取算法中的一颗璀璨的明珠。SIFT算法的介绍网上有很多比较好的博客和文章，我在学习这个算法的过程中也参看网上好些资料，即使评价比较高的文章，作者在文章中对有些比较重要的细节、公式来历没有提及，可能写博客的人自己明白，也觉得简单，因此就忽略了这些问题，但是对刚入门的人来说，看这些东西，想搞清楚这些是怎么来的还是比较费时费力的。比如SIFT算法中一个重要的操作：求取描述子的主方向。好多文章只是一提而过或忽略，然后直接给出一个公式，SIFT算法的原作者也提使用抛物线插值，但是具体怎么插的就不太详尽了，对于初学者来说更是不知所云。因此本文打算在参看的文章上对有关这些细节给出一些比较详细的说明，还有本文尽量对操作过程配备对应图片或示意图说明，同时附上robwhesss开源SIFT C代码对应程序块并给予注解，方便理解。

一、 SIFT算法

1、算法简介

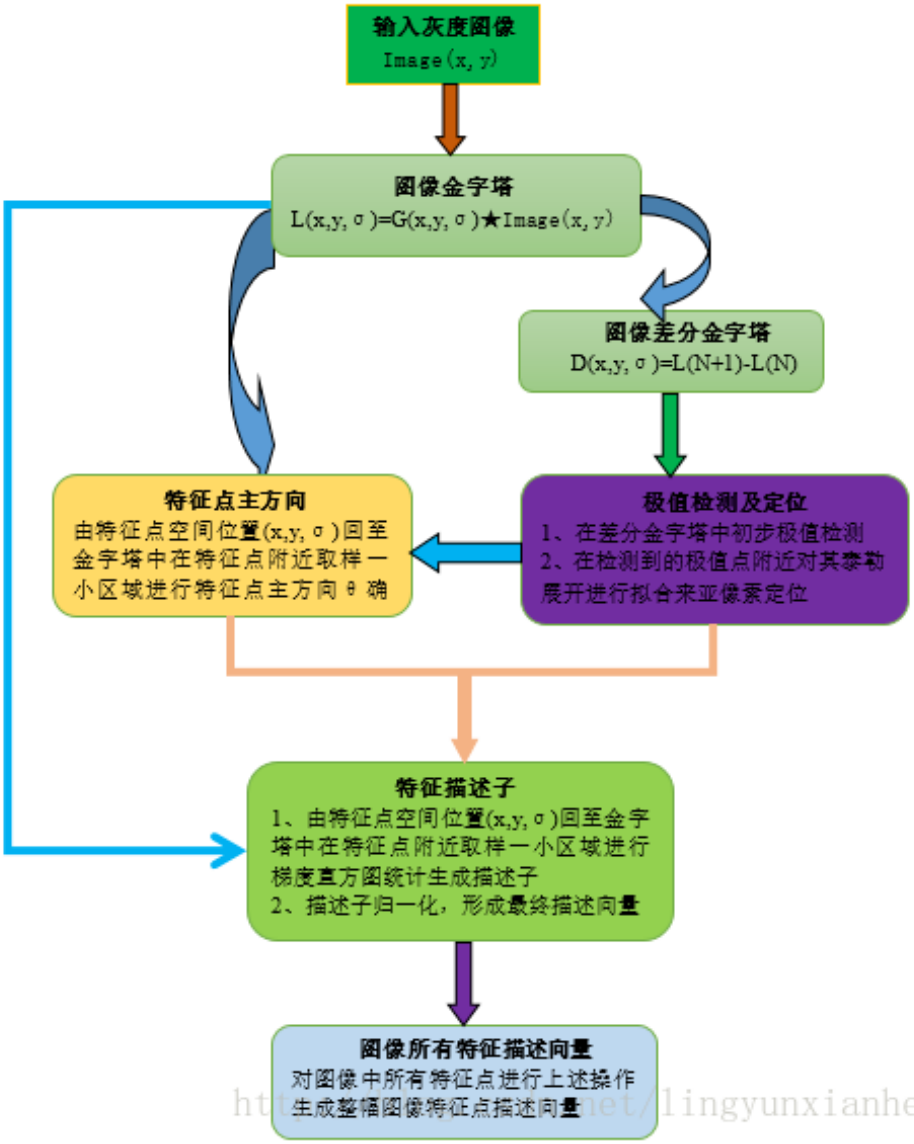
尺度不变特征转换即SIFT (Scale-invariant feature transform)是一种计算机视觉的算法。它用来侦测与描述影像中的局部性特征，它在空间尺度中寻找极值点，并提取出其位置、尺度、旋转不变量，此算法由 David Lowe在1999年所发表，2004年完善总结。

其应用范围包含物体辨识、机器人地图感知与导航、影像缝合、3D模型建立、手势辨识、影像追踪和动作比对。

局部影像特征的描述与侦测可以帮助辨识物体，SIFT特征是基于物体上的一些局部外观的兴趣点而与影像的大小和旋转无关。对于光线、噪声、些微视角改变的容忍度也相当高。基于这些特性，它们是高度显著而且相对容易撷取，在母数庞大的特征数据库中，很容易辨识物体而且鲜有误认。使用 SIFT特征描述对于部分物体遮蔽的侦测率也相当高，甚至只需要3个以上的SIFT物体特征就足以计算出位置与方位。在现今的电脑硬件速度下和小型的特征数据库条件下，辨识速度可接近即时运算。SIFT特征的信息量大，适合在海量数据库中快速准确匹配。

SIFT算法的实质是在不同的尺度空间上查找关键点(特征点)，并计算出关键点的方向。SIFT所查找到的关键点是一些十分突出，不会因光照，仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等。

2、SIFT算法流程图



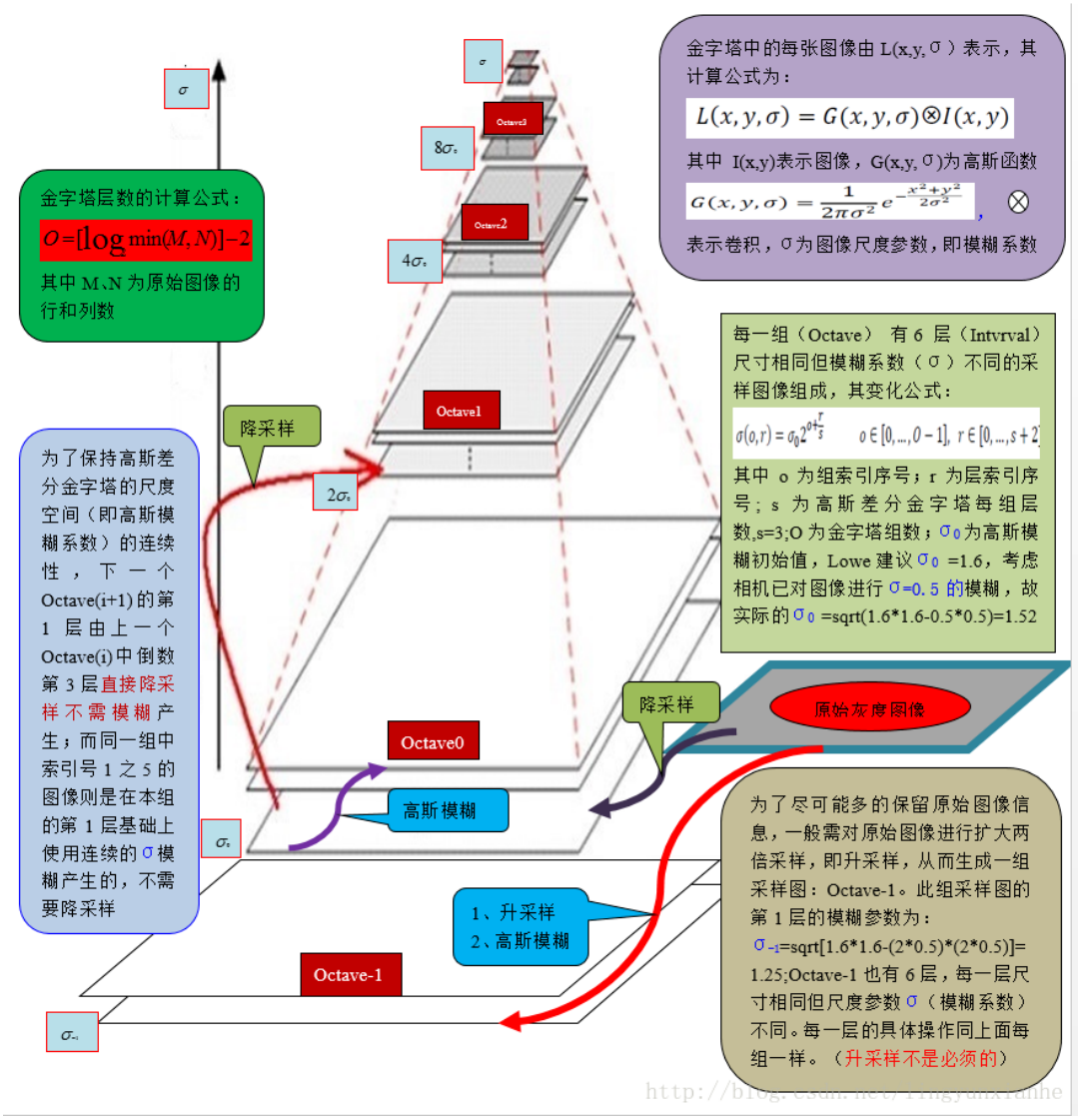
二、SIFT算法操作步骤

1、图像金字塔

1.1、高斯金字塔

图像高斯金字塔(Gaussian Pyramid)是采用高斯函数对图像进行模糊以及降采样处理得到。其形成过程可如下图所示：

其中高斯模糊系数计算公式如下：



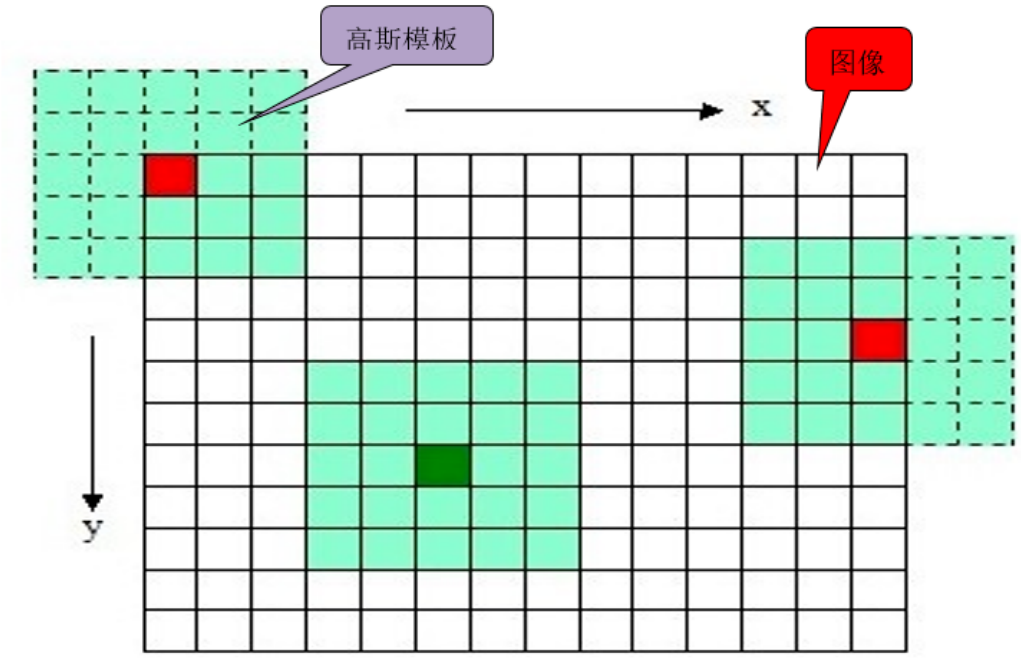
其中高斯模糊系数计算公式如下：

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

1.1.1、高斯函数与图像卷积

根据3σ原则，使用NxN的模板在图像每一个像素点处操作，其中N=[（6σ+1）]且向上取最邻近奇数。

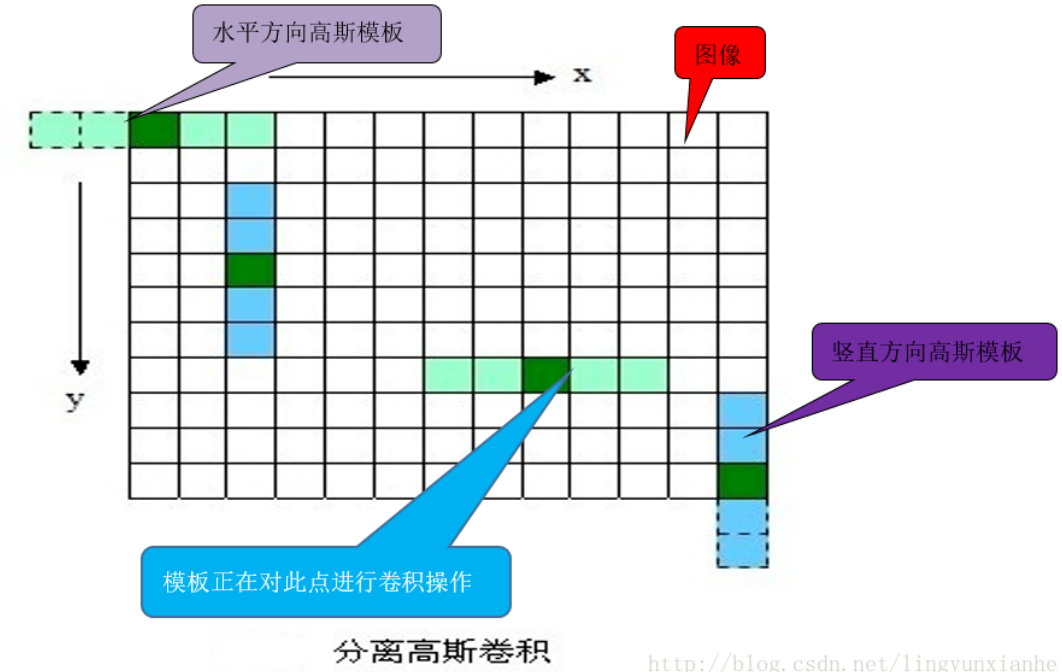
其操作如下图：



二维高斯卷积

1.1.2、分离高斯卷积

上面这样直接与图像卷积，速度比较慢，同时图像边缘信息也会损失严重。后来，后来、、、，不知哪位学者发现，可以使用分离的高斯卷积（即先用1xN的模板沿着X方向对图像卷积一次，然后用Nx1的模板沿着Y方向对图像再卷积一次，其中N=[（6σ+1）]且向上取最邻近奇数），这样既省时也减小了直接卷积对图像边缘信息的严重损失。



分离高斯卷积

1.1.3、高斯金子塔源码分析

```
1. for (o = 0; o < octvs; o++)//金字塔组数为octvs，
2. for (i = 0; i < intvls + 3; i++)//每一组有intvls + 3 层，intvls一般为3
3.          {
4. if (o == 0 && i == 0)//如果是第一组第1层
5.                  gauss_pyr[o][i] = cvCloneImage(base);//base 为原始灰度图像经过升采样或降采样得到的图像
6. /* base of new octvave is halved image from end of previous octave */
7. elseif (i == 0)//建立非第一组的第1层
8.                  gauss_pyr[o][i] = downsample(gauss_pyr[o - 1][intvls]);//降采样图像
9. /* blur the current octave's last image to create the next one */
10. else//建立非第一组的非第1层
11.                 {
12.                 gauss_pyr[o][i] = cvCreateImage(cvGetSize(gauss_pyr[o][i - 1]),IPL_DEPTH_32F, 1);
13.                 cvSmooth(gauss_pyr[o][i - 1], gauss_pyr[o][i],CV_GAUSSIAN, 0, 0, sig[i], sig[i]);// sig[i]为模糊系数
14.                 }//cvSmooth 为平滑处理函数，也即模糊处理。CV_GAUSSIAN 为选用高斯函数对图像模糊
15. return gauss_pyr;//返回建好的金字塔
```

1.2、高斯差分金字塔

2002年Mikolajczyk在详细的实验比较中发现尺度归一化的高斯拉普拉斯函数的极大值和极小值同其它的特征提取函数，例如：梯度，Hessian或Harris角特征比较，能够产生最稳定的图像特征。而Lindeberg早在1994年就发现高斯差分函数（简称DOG算子）与尺度归一化的高斯拉普拉斯函数非常近似。如下式：

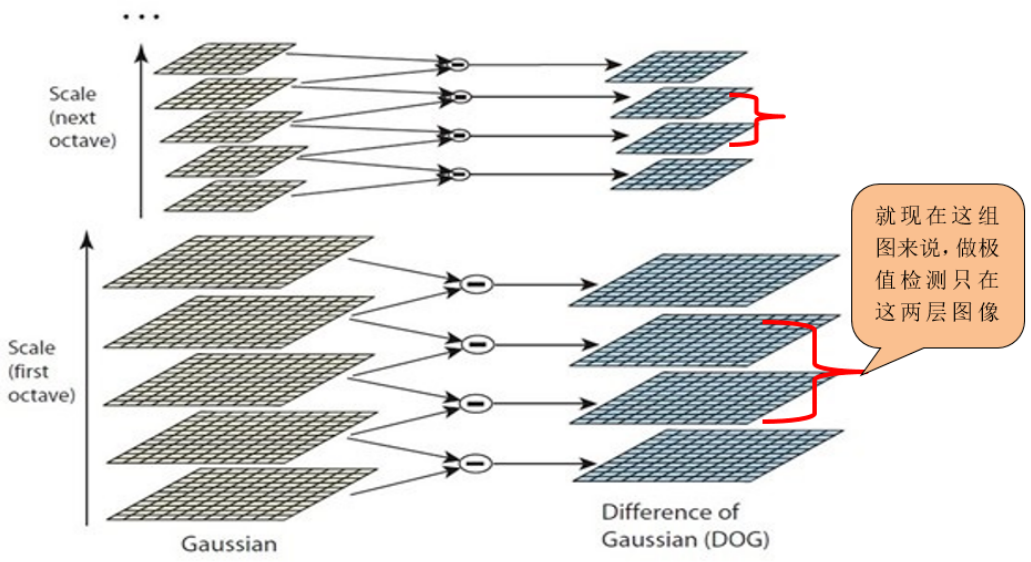
$$G(x,y,k\sigma)-G(x,y,\sigma)\approx(k-1)\sigma^2\nabla^2G$$

其中k-1是个常数，并不影响极值点位置的求取。

1.2.1、差分金字塔的建立

差分金字塔的是在高斯金字塔的基础上操作的，其建立过程是：在高斯金子塔中的每组中相邻两层相减（下一层减上一层）就生成高斯差分金字塔.

高斯差分金字塔其操作如下图：



高斯差分金字塔的生成

1.2.2、差分金字塔源码分析

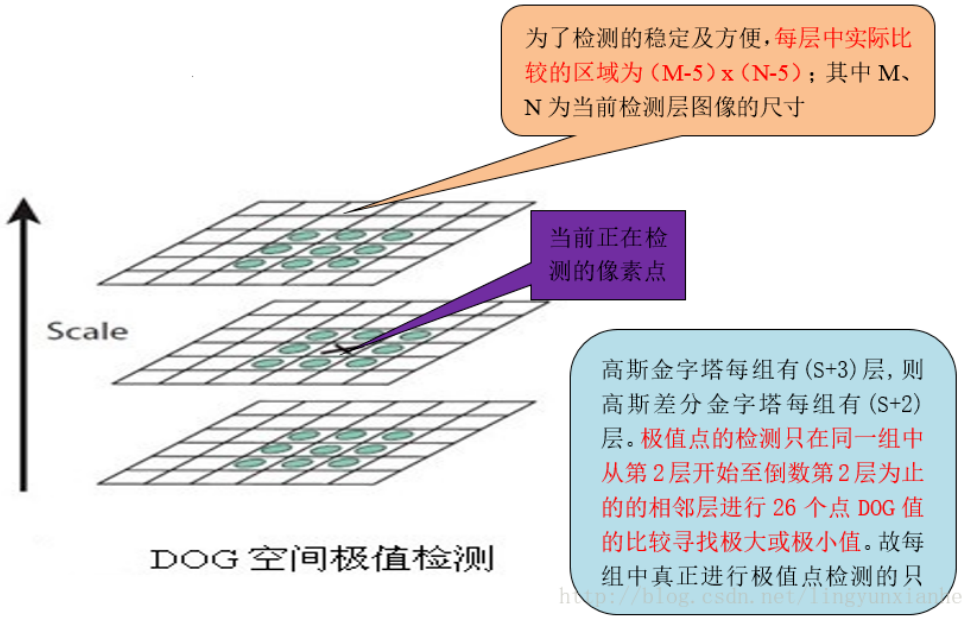
```
1. for (o = 0; o < octvs; o++)//octvs为高斯金字塔组数
2. for (i = 0; i < intvls + 2; i++)//因为相减，故高斯金字塔中每组有（intvls + 2）层图像
3.          {
4.          dog_pyr[o][i] = cvCreateImage(cvGetSize(gauss_pyr[o][i]),IPL_DEPTH_32F, 1);
5.          cvSub(gauss_pyr[o][i + 1], gauss_pyr[o][i], dog_pyr[o][i], NULL);//cvSub为opencv
内置相减函数
6.          }
7. return dog_pyr;//返回高斯差分金字塔
```

2、空间极值点（即关键点）检测

关键点是由DOG空间的局部极值点组成的，关键点的初步探查是通过同一组内各DoG相邻两层图像之间比较完成的。为了寻找DoG函数的极值点，每一个像素点要和它所有的相邻点比较，看其是否比它的图像域和尺度域的相邻点大或者小。如图下图所示，中间的检测点和它同尺度的8个相邻点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。

2.1、极值点检测过程

2.1.1、极值点检测示意



2.1.2、极值点检测源码分析

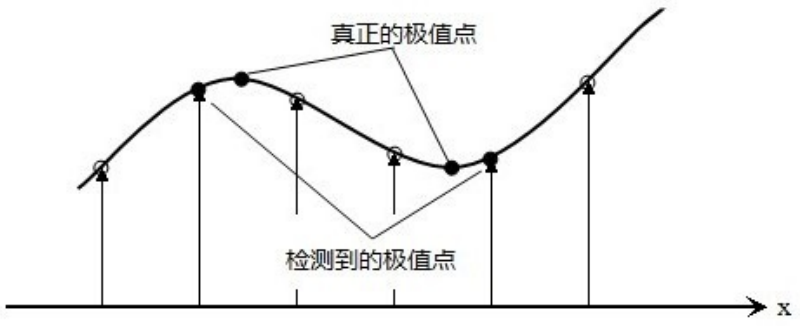
```
1. if (val > 0)//极大值检测{
2. for (i = -1; i <= 1; i++)
3. for (j = -1; j <= 1; j++)
4. for (k = -1; k <= 1; k++)
5. if (val < pixval32f(dog_pyr[octv][intvl + i], r + j, c + k))//pixval32f为提取图像像素位置上的灰度值
6. return0;}
7. else/* check for minimum */
8.          {
9. for (i = -1; i <= 1; i++)
10. for (j = -1; j <= 1; j++)
11. for (k = -1; k <= 1; k++)
12. if (val > pixval32f(dog_pyr[octv][intvl + i], r + j, c + k))//r c为图像的行数和列数，dog_pyr为高斯差分图
13. return0;
```

2.2、关键点定位

以上方法检测到的极值点是离散空间的极值点，以下通过拟合三维二次函数来精确确定关键点的位置和尺度，同时去除低对比度的关键点和不稳定的边缘响应点(因为DoG算子会产生较强的边缘响应)，以增强匹配稳定性、提高抗噪声能力。

2.2.1、关键点精确定位

离散空间的极值点并不是真正的极值点，下图显示了二维函数离散空间得到的极值点与连续空间极值点的差别。利用已知的离散空间点插值得到的连续空间极值点的方法叫做子像素插值。



离散空间与连续空间极值点的差别

为了提高关键点的稳定性，需要对尺度空间DoG函数进行曲线插值。利用DoG函数在尺度空间的Taylor展开式(插值函数)为：

任意一极值点在其 $X0 = (x0,\ y0,\ \sigma 0)$ 处泰勒展开并舍掉 2 阶以后的项结果如下：

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

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

Hessian 矩阵

其中 f 的一阶偏导数，二阶偏导数，以及二阶混合偏导数由下面几个公式求（h=1）得：

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

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

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

上面算式的矩阵表示如下：

$$D(X) = D + \frac{\partial D^T}{\partial X}X + \frac{1}{2}X^T\frac{\partial^2 D}{\partial X^2}X$$

其中，X求导并让方程等于零，可以得到极值点的偏移量为：

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

对应极值点，方程的值为：

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

其中，X^代表相对插值中心的偏移量，当它在任一维度上的偏移量大于0.5时（即x或y或σ），意味着插值中心已经偏移到它的邻近点上，所以必须改变当前关键点的位置。同时在新的位置上反复插值直到收敛；也有可能超出所设定的迭代次数或者超出图像边界的范围，此时这样的点应该删除，在Lowe中进行了5次迭代。另外，过小的点易受噪声的干扰而变得不稳定，所以将 小于某个经验值(Lowe论文中使用0.03，Rob Hess等人实现时使用0.04/S)的极值点删除。同时，在此过程中获取特征点的精确位置(原位置加上拟合的偏移量)以及尺度（σ）。

2.2.2、消除边缘响应

一个定义不好的高斯差分算子的极值在横跨边缘的地方有较大的主曲率，而在垂直边缘的方向有较小的主曲率。DOG算子会产生较强的边缘响应，需要剔除不稳定的边缘响应点。获取特征点处的Hessian矩阵，主曲率通过一个2x2 的Hessian矩阵H求出（D的主曲率和H的特征值成正比）：

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

假设H的特征值为α和β（α、β代表x和y方向的梯度）且α>β。令α=rβ则有：

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

其中*Tr(H)*求取*H*的对角元素和；*Det(H)*为求*H*的行列式值。

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

则公式(r+1)^2/r的值在两个特征值相等时最小，随着的增大而增大。值越大，说明两个特征值的比值越大，即在某一个方向的梯度值越大，而在另一个方向的梯度值越小，而边缘恰恰就是这种情况。所以为了剔除边缘响应点，需要让该比值小于一定的阈值，因此，为了检测主曲率是否在某域值*r*下，只需检测:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r},$$

论文建议*r=10*, *OpenCv*也采用*r=10*

2.2.3、精确定位中的泰勒插值源码分析

```
1. while (i < SIFT_MAX_INTERP_STEPS)//SIFT_MAX_INTERP_STEPS=5为最大迭代次数，避免长时迭代
```

```
2.          {
3.          interp_step(dog_pyr, octv, intvl, r, c, &xi, &xr, &xc);// 泰勒展开拟合，xi,xr,xc依次为x、y、σ方向偏移量,
4. if (ABS(xi) < 0.5  &&  ABS(xr) < 0.5  &&  ABS(xc) < 0.5)//如果当前偏移量绝对值中的每个值均小于0.5，退出迭代
5. break;
6.          c += cvRound(xc);//计算行坐标，cvRound 为四舍五入。
7.          r += cvRound(xr);
8.          intvl += cvRound(xi);
9. if (intvl < 1 ||//不在计算的图像层中
10.                 intvl > intvls ||//高斯差分每组的层数为intvls
11.                 c < SIFT_IMG_BORDER ||//靠近图像边缘5个像素的区域不做检测，SIFT_IMG_BORDER=5,
12.                 r < SIFT_IMG_BORDER ||
13.                 c >= dog_pyr[octv][0]->width - SIFT_IMG_BORDER ||//靠近图像边缘5个像素的区域不做检测
14.                 r >= dog_pyr[octv][0]->height - SIFT_IMG_BORDER)
15.                 {
16. returnNULL;
17.                 }
18.         i++;//迭代计数
19.         }
1. staticvoidinterp_step(IplImage*** dog_pyr, int octv, int intvl, int r, int c,double* xi, double* xr, double* xc)
2.     {
3.     CvMat* dD, *H, *H_inv, X;
4. double x[3] = { 0 };
5.     dD = deriv_3D(dog_pyr, octv, intvl, r, c);//一阶偏导数
6.     H = hessian_3D(dog_pyr, octv, intvl, r, c);//Hessian 矩阵即二阶导数组成的矩阵
7.     H_inv = cvCreateMat(3, 3, CV_64FC1);
8.     cvInvert(H, H_inv, CV_SVD);//求Hessian矩阵的逆矩阵
9.     cvInitMatHeader(&X, 3, 1, CV_64FC1, x, CV_AUTOSTEP);
10.    cvGEMM(H_inv, dD, -1, NULL, 0, &X, 0); //cvGEMM为矩阵乘法，//第一个矩阵的系数；//H_inv、dD第一二个矩阵//-1矩阵前的常数//X结果矩阵
11.    cvReleaseMat(&dD);
12.    cvReleaseMat(&H);
13.    cvReleaseMat(&H_inv);
14.    *xi = x[2];
15.    *xr = x[1];
16.    *xc = x[0];
```

17. }

3、关键点方向分配

为了使描述符具有旋转不变性，需要利用图像的局部特征为给每一个关键点分配一个基准方向。使用图像梯度的方法求取局部结构的稳定方向。

3.1、特征点的梯度

3.1.1、梯度的计算

对于在DOG金字塔中检测出的关键点点，采集其所在高斯金字塔图像3σ领域窗口内像素的梯度和方向分布特征。梯度的模值和方向如下：

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

$$\theta(x,y)=\tan^{-1}((L(x,y+1)-L(x,y-1))/L(x+1,y)-L(x-1,y)))$$

上面一阶偏导是利用下面的差分公式来近似的（其中 h=1）

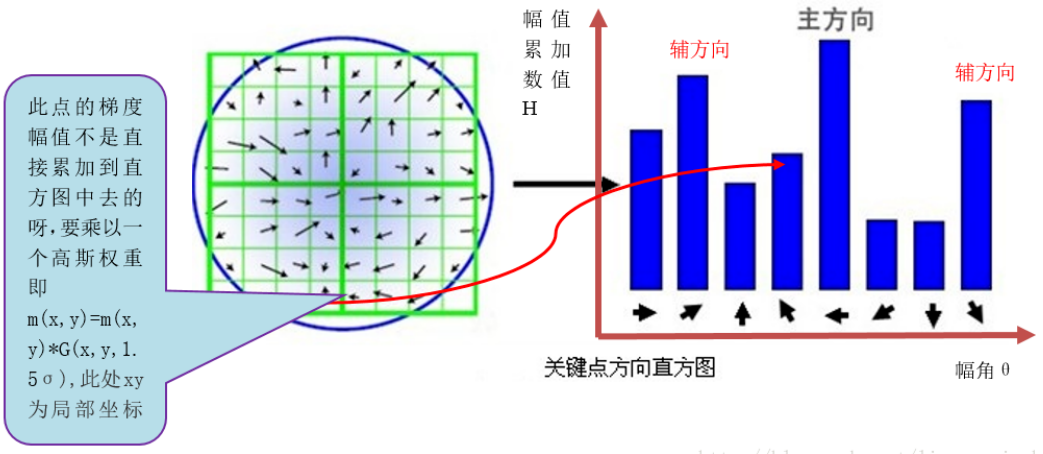
$$\frac{\partial f}{\partial x}=\frac{f(i,j+1)-f(i,j-1)}{2h},\quad \frac{\partial f}{\partial y}=\frac{f(i+1,j)-f(i-1,j)}{2h}$$

我们发现 L 的一阶偏导有一个 1/2 的系数，但实际操作时不需这样做不会影响结果

L为关键点所在的尺度空间值，按Lowe的建议，梯度的模值m(x,y)按 σ=1.5σ_oct 的高斯分布加成，按尺度采样的3σ原则，领域窗口半径为 3x1.5σ_oct。

3.1.1、梯度直方图

在完成关键点的梯度计算后，使用直方图统计领域内像素的梯度和方向。梯度直方图将0~360度的方向范围分为36个柱(bins)，其中每柱10度。如图5.1所示，直方图的峰值方向代表了关键点的主方向，(为简化，图中只画了八个方向的直方图)。



3.2、特征点主方向的确定

方向直方图的峰值则代表了该特征点处邻域梯度的方向，以直方图中最大值作为该关键点的主方向。为了增强匹配的鲁棒性，只保留峰值大于主方向峰值80%的方向作为该关键点的辅方向。因此，对于同一梯度值的多个峰值的关键点位置，在相同位置和尺度将会有

多个关键点被创建但方向不同。仅有15%的关键点被赋予多个方向，但可以明显的提高关键点匹配的稳定性。实际编程实现中，就是把该关键点复制成多份关键点，并将方向值分别赋给这些复制后的关键点，并且，离散的梯度方向直方图要进行插值拟合处理，来求得更精确的方向角度值。

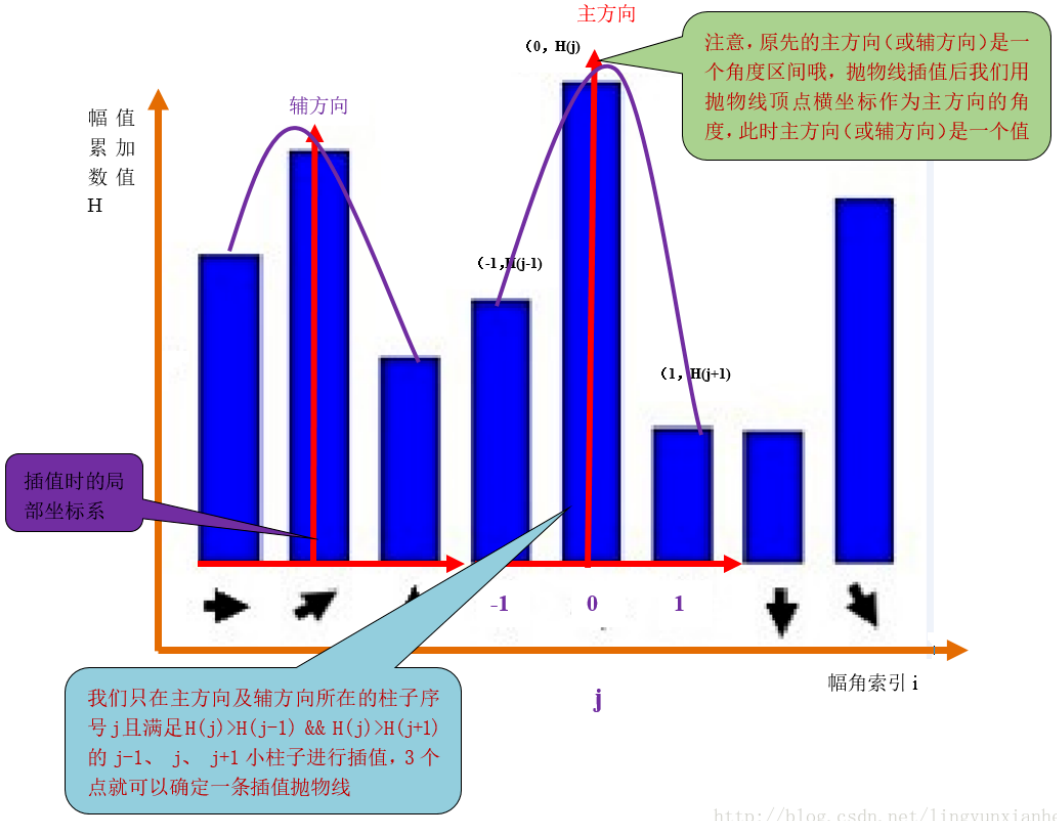
3.2.1、梯度图像的平滑处理

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。Opencv 所使用的平滑公式为：

$$H(i)=\frac{h(i-2)+h(i+2)}{16}+\frac{4\times(h(i-1)+h(i+1))}{16}+\frac{6\times h(i)}{16}$$

其中i∈[0,35], 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0o=360o，如果出现$h(j)$，j超出了$(0,\cdots,35)$的范围，那么可以通过圆周循环的方法找到它所对应的、在0o=360o之间的值，如$h(-1)=h(35)$。

3.2.2、梯度直方图抛物线插值



假设我们在第i个小柱子要找一个精确的方向，那么由上面分析知道：

设插值抛物线方程为$h(t)=at^2+bt+c$,其中a、b、c为抛物线的系数，t为自变量，t∈[-1,1]，此抛物线求导并令它等于0。

即$h(t)'=0$得$t_{max}=-b/(2a)$

现在把这三个插值点带入方程可得：

现在把这三个插值点带入方程可得：

$$\begin{cases} h(-1)=a-b+c \\ h(0)=c \\ h(1)=a+b+c \end{cases} \Rightarrow \begin{cases} a=[h(1)+h(-1)]/2-h(0) \\ b=[h(1)-h(-1)]/2 \\ c=h(0) \end{cases}$$

由上式知：

$t_{max}=-b/(2a)=\frac{h(-1)-h(1)}{2[h(-1)+h(1)-2h(0)]}$ （局部坐标系中的取值）

$i'=i+\frac{h(i-1)-h(i+1)}{2[h(i-1)+h(i+1)-2h(i)]}$ （小柱子在直方图中的索引号）

3.2.3、抛物线插值源码分析

```
1. #define interp_hist_peak( l, c, r ) ( 0.5 * ((l)-(r)) / ((l) - 2.0*(c) + (r)) )//插值计算式，l为左侧柱子值，r为左侧柱子值
2. staticvoidadd_good_ori_features(CvSeq* features, double* hist, int n,
3. double mag_thr, struct feature* feat)//精确主方向及辅方向
4.     {
5. structfeature* new_feat;
6. double bin, PI2 = CV_PI * 2.0;//CV_PI=pi
7. int l, r, i;
8. for (i = 0; i < n; i++)// 直方图有n=36个小柱子
9.         {
10.         l = (i == 0) ? n - 1 : i - 1;//把小柱子看成是循环的，角度的取值为0-360即一个圆周
11.         r = (i + 1) % n;
12. //只对小柱子的值大于等于主峰80%且此小柱子比左右两边小柱子都高的柱子进行抛物线插值
13. if (hist[i] > hist[l] && hist[i] > hist[r] && hist[i] >= mag_thr)// mag_thr为>=80%的最高峰值
14.             {
15.             bin = i + interp_hist_peak(hist[l], hist[i], hist[r]);//interp_hist_peak 插值函数
16.             bin = (bin < 0) ? n + bin : (bin >= n) ? bin - n : bin;//角度取值约束在0-360之间，且是连续循环的
17.             new_feat = clone_feature(feat);//幅值特征点
18.             new_feat->ori = ((PI2 * bin) / n) - CV_PI;//?
19.             cvSeqPush(features, new_feat);
20. free(new_feat);
21.             }
```

至此，图像的关键点已检测完毕，每个关键点有三个信息：位置、所处尺度、方向。由此可以确定一个SIFT特征区域。

4、特征点描述符

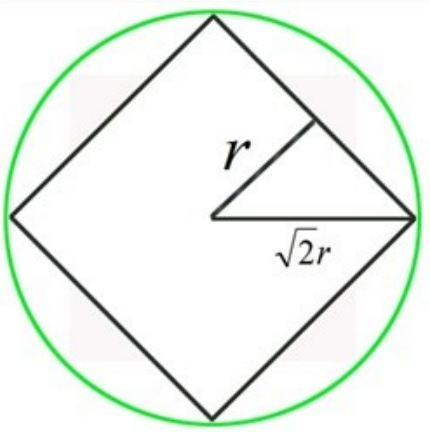
通过以上步骤，对于每一个关键点，拥有三个信息：位置、尺度以及方向。接下来就是为每个关键点建立一个描述符，使其不随各种变化而改变，比如光照变化、视角变化等等。并且描述符应该有较高的独特性，以便于提高特征点正确匹配的概率。

4.1、特征的生成过程

4.1.1、确定计算描述子所需的区域

将关键点附近的区域划分为d*d(Lowe建议d=4)个子区域，每个子区域作为一个种子点，每个种子点有8个方向。考虑到实际计算时，需要采用三线性插值，所需图像窗口边长为3x3xσ_oct x(d+1)。在考虑到旋转因素(方便下一步将坐标轴旋转到关键点的方向)，如下图6.1所示，实际计算所需的图像区域半径为：

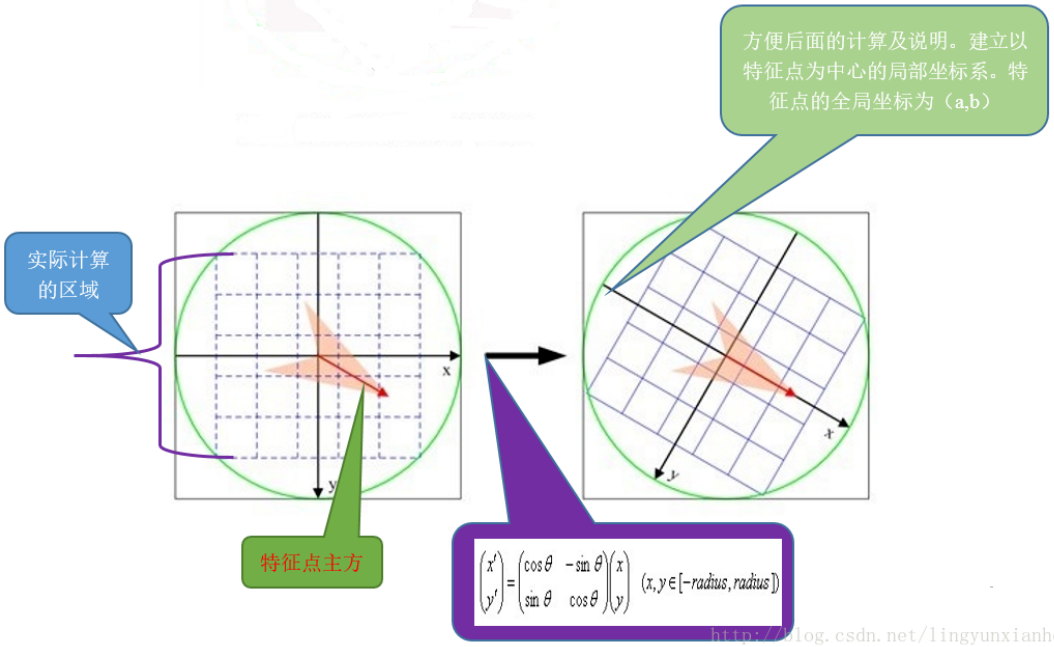
$$radius = \frac{3\sigma\_oct \times \sqrt{2} \times (d+1)}{2}$$



旋转引起的领域半径变化

4.1.2、坐标轴旋转至主方向

将坐标轴旋转为关键点的方向，以确保旋转不变性。



4.1.3、梯度直方图的生成

将邻域内的采样点分配到对应的子区域内，将子区域内的梯度值分配到8个方向上，计算其权值。

旋转后的采样点 落在子区域的下标为

旋转后的采样点 (x',y') 落在子区域的下标为

$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \frac{1}{3\sigma\_oct} \begin{pmatrix} x' \\ y' \end{pmatrix} + \frac{d}{2} \quad -0.5$$

坐标归一化，此后在特征点局部坐标系中把 3σ 长度做为 1 个单元长度，加 d/2 则是把坐标系由特征点处平移至左上角的边界点上，再减 0.5 则是回移坐标系至描述子区间中的第一个子区间的中心处。why 要这样做？一切的一切都是方便后面的三线性插值

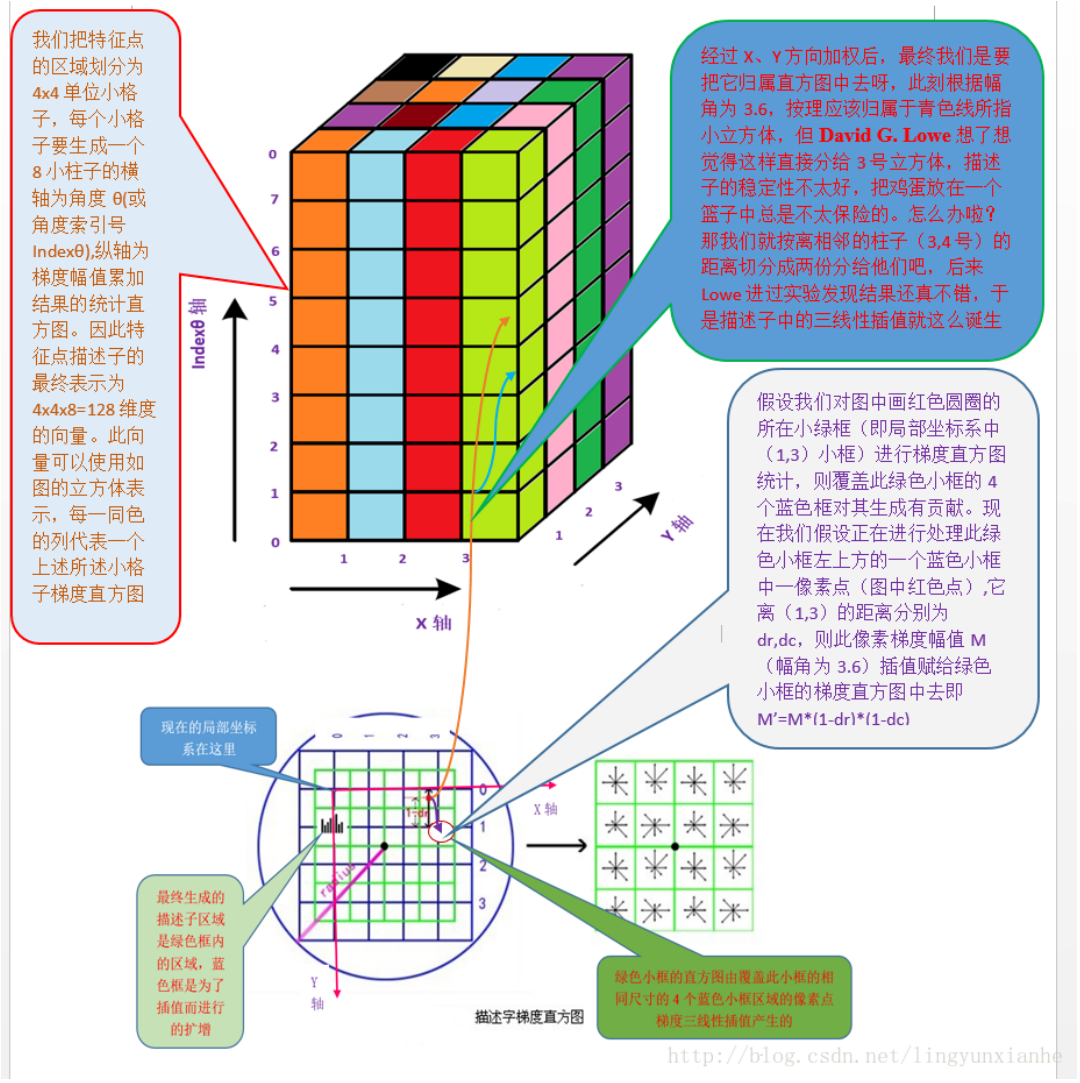
Lowe 建议子区域的像素的梯度大小按 $\sigma = 0.5d$ 的高斯加权计算，即

$$w = m(a+x, b+y) * e^{-\frac{(x')^2+(y')^2}{2\times(0.5d)^2}}$$

其中 a，b 为关键点在高斯金字塔图像中的位置坐标（也即全局坐标系中的位置）。

4.1.4、三线性插值

插值计算每个种子点八个方向的梯度。



采样点在子区域中的下标(x'',y'')　　(图中蓝色窗口内红色点)线性插值，计算其对每个种子点的贡献。如图中的红色点，落在第0行和第1行之间，对这两行都有贡献。对第0行第3列种子点的贡献因子为dr，对第1行第3列的贡献因子为1-dr，同理，对邻近两列的贡献因子为dc和1-dc，对邻近两个方向的贡献因子为do和1-do。则最终累加在每个方向上的梯度大小为：

$$weight = w * dr^{k} * (1-dr)^{1-k} * dc^{m} * (1-dc)^{1-m} * do^{n} * (1-do)^{1-n}$$

其中k，m，n为0（像素点超出了对要插值区间的四个邻近子区间所在范围）或为1（像素点处在对要插值区间的四个邻近子区间之一所在范围）。

4.1.5、特征描述子

如上统计的4*4*8=128个梯度信息即为该关键点的特征向量。

特征向量形成后，为了去除光照变化的影响，需要对它们进行归一化处理，对于图像灰度值整体漂移，图像各点的梯度是邻域像素相减得到，所以也能去除。得到的描述子向量为H=(h1,h2,.....,h128)，归一化后的特征向量为L=(L1,L2,.....,L128)，则

$$L_i = \frac{h_i}{\sqrt{\sum_{j=1}^{128} h_j^2}} \quad i = 1,2,3,\ldots 128$$

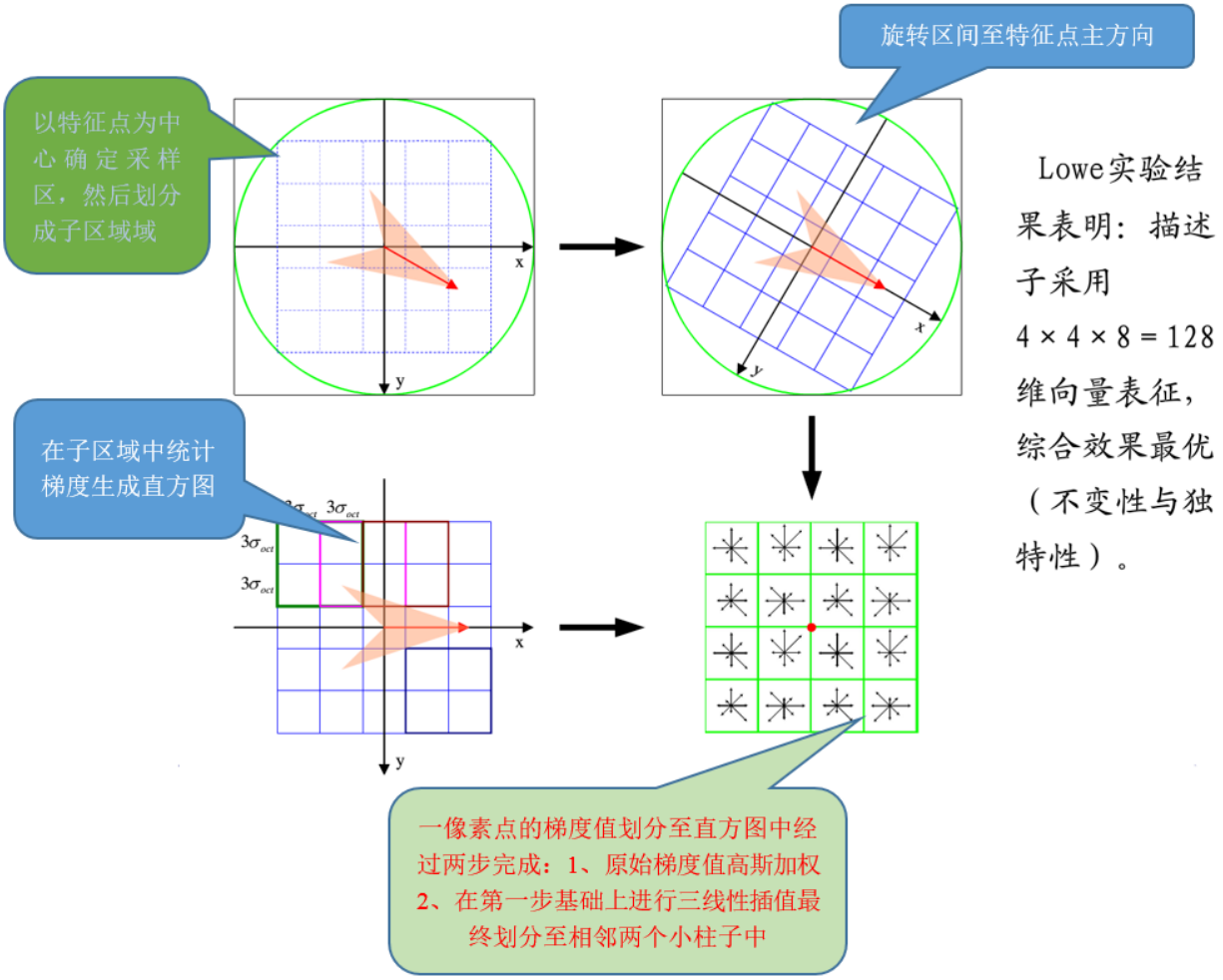
4.1.6、描述子的门限化

非线性光照，相机饱和度变化对造成某些方向的梯度值过大，而对方向的影响微弱。因此设置门限值(向量归一化后，一般取0.2)截断较大的梯度值(大于0.2的则就令它等于0.2，小于0.2的则保持不变)。然后再进行一次归一化处理，提高特征的鉴别性。

4.2、描述子相关分析

用一组图来概括描述子的生成过程

4.2.1、描述子生成总括



4.2.3、描述子三线性插值源码分析

```
1. staticvoidinterp_hist_entry(double*** hist, double rbin, double cbin,
2. double obin, double mag, int d, int n)
3.          {
4. double d_r, d_c, d_o, v_r, v_c, v_o;
5. double** row, *h;
6. int r0, c0, o0, rb, cb, ob, r, c, o;
7.          r0 = cvFloor(rbin);//向下取整
```

```
8.          c0 = cvFloor(cbin);
9.          o0 = cvFloor(obin);
10.         d_r = rbin - r0;//小数余项
11.         d_c = cbin - c0;
12.         d_o = obin - o0;
13. for (r = 0; r <= 1; r++)//沿着行方向不超过1个单位长度
14.             {
15.             rb = r0 + r;
16. if (rb >= 0 && rb < d)//如果此刻还在真正的描述子区间内
17.                 {
18.                 v_r = mag * ((r == 0) ? 1.0 - d_r : d_r);//d_r = rbin - r0;
19.                 row = hist[rb];
20. for (c = 0; c <= 1; c++)//沿着行方向不超过1个单位长度
21.                     {
22.                     cb = c0 + c;
23. if (cb >= 0 && cb < d)
24.                         {
25.                         v_c = v_r * ((c == 0) ? 1.0 - d_c : d_c);
26.                         h = row[cb];
27. for (o = 0; o <= 1; o++)//沿着直方图方向不超过1个单位角度
28.                             {
29.                             ob = (o0 + o) % n;  //n=8，8个小柱子
30.                             v_o = v_c * ((o == 0) ? 1.0 - d_o : d_o);
31.                             h[ob] += v_o;
32.                             }
33.                         }
34.                     }
35.                 }
```

通过上面的1至4个大步骤就可以完成SIFT算法对图像特征点的提取。至此SIFT算法完结。图像特征提取是图像匹配的基础，经过此算法提取出来的特征点用于后续的图像特征匹配和特征识别中，关于图像特征匹配相关内容将在后续讲解。

参考文献

1、sift算法详解及应用（课件）。（本文档简明扼要的简述了SIFT算法和图像匹配以及匹配修正。图文并茂，一览全貌）

http://wenku.baidu.com/view/87270d2c2af90242a895e52e.html?re=view

2、SIFT算法详解（sift操作过程理论通俗，尤其是高阶泰勒展开式及高阶导数分析的很好，对理解亚像素定位拟合中的图像具体编程操作很有用）

http://blog.csdn.net/zddblog/article/details/7521424

3、SIFT特征分析与源码解读（1模拟金字塔的过程解释的很详细，带有动画模拟；2 在寻找特征点进行亚像素定位拟合中的图像很形象）

http://blog.csdn.net/xw20084898/article/details/16832755

4、【OpenCV】SIFT原理与源码分析：关键点描述（对关键点描述子区域的取舍讲解的很详细）

http://blog.csdn.net/xiaowei_cqu/article/details/8113565

5、【OpenCV】SIFT原理与源码分析（对sift 算法采用分部分叙述且带有源码分析说明）

http://blog.csdn.net/xiaowei_cqu/article/details/8069548

6、opencv2.4.9sift源码分析(1赵春江的这篇文章是我目前看到分析sift算法比较全面的；2 尤其给出了使用三维直方图来分析三线性插值，对理解描述子的生成作用很大；3 给出了源码分析和演示结果）

http://wenku.baidu.com/view/d7edd2464b73f242336c5ffa.html

http://download.csdn.net/detail/zhaocj/8294793

7、九之再续：教你一步一步用c语言实现sift算法、下

（1算法中寻找主方向使用的抛物线插值拟合方法；2 描述子三次插值）

http://blog.csdn.net/v_JULY_v/article/details/6246213

8、RobHess的SIFT源码分析：综述(各个子程序详解及分析很细致，一概全貌）

http://blog.csdn.net/masibuaa/article/details/9191309

9、特征点检测学习 1(sift算法)（1这篇文章没有太多理论分析，但结合QT和OpenCV做出了生动的sift算法匹配演示，有图很直观生动呀，用程序配图一目了然；2 简述对robhess 的c版本sift代码在c++中的使用注意问题 ）

http://www.cnblogs.com/tornadomeet/archive/2012/08/16/2643168.html

10、OpenCV 中c版本sift源代码网址

http://blogs.oregonstate.edu/hess/code/sift/

11、【特征匹配】SIFT原理与C源码剖析（这个也不错，图文并茂，还带有 源码分析，总体来说是以程序带动问题分析）

http://blog.csdn.net/luoshixian099/article/details/47377611

12、插值与拟合（对多项式及其插值讲解还不错）

http://wenku.baidu.com/link?url=wWcqLrpokQrjZZKzFbuJ4QDbZXZkMByCu-KaVKrSyGD6fh9Bpk1kZOPitpkFpNBw_no8UoyWY2DGQg9I7aL_tO3oi7z5mUK7cN8Sca6dX-O

13、线性插值与抛物线插值（对这两种插值讲解的很详细，是目前发现最 好的一版

http://www.docin.com/p-711275966.html

14、奇异值分解（对奇异值怎么来的讲解比较细致）

http://blog.sina.com.cn/s/blog_53eb0fdf0101sfu1.html