图像处理

一，Harris角点检测

- 理论（1998年提出）
  - 何为角点（在邻域内的各个方向上灰度变化值足够高的点，是图像边缘曲线上曲率极大值的点。）

    > 轮廓之间的角点
    >
    > 对于同一场景，即使视角发生变化，通常具有稳定性质的特征
    >
    > 该点附近区域的像素点无论是在梯度方向上还是梯度幅值上都有较大的变化

  - 基本思想

    > 使用一个固定窗口在图像上进行任意方向的滑动，比较滑动前与滑动后的两种情况，窗口中像素灰度的变化程度，
    >
    > 如果存在任意方向上的滑动，都有较大的灰度变化，那么我们可以认为该窗口中存在角点。

  - 数学表示

    $$E(u,v)=\sum_{x,y}[I(x+u,y+v)-I(x,y)]^2$$

    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。

    所以通常窗口函数有如下两种形式：

  - 进一步演化

    因为泰勒展开有：

    $$f(x+u,y+v)\approx f(x,y)+uf_x(x,y)+vf_y(x,y)$$

    于是上式$E(u,v)$可以写作下面：

    $$E(u,v)\cong[u,v]\quad M\quad\begin{bmatrix}u\\v\end{bmatrix}$$

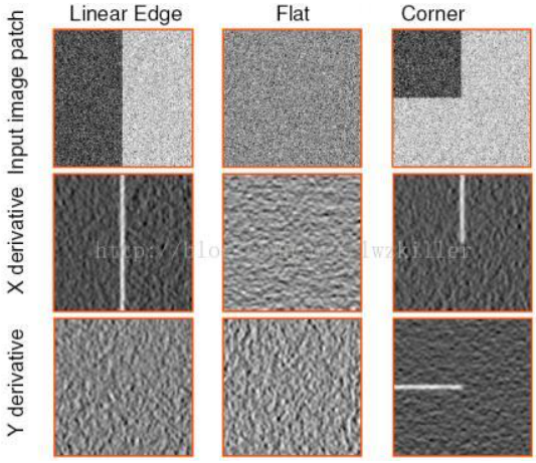
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Windowing function - computing a weighted sum (simplest case, w=1)

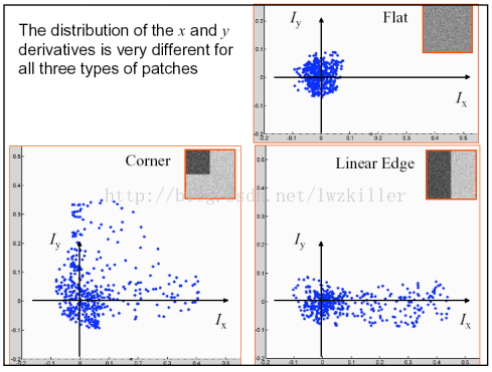
Note: these are just products of components of the gradient, Ix, Iy

- 矩阵M的关键性

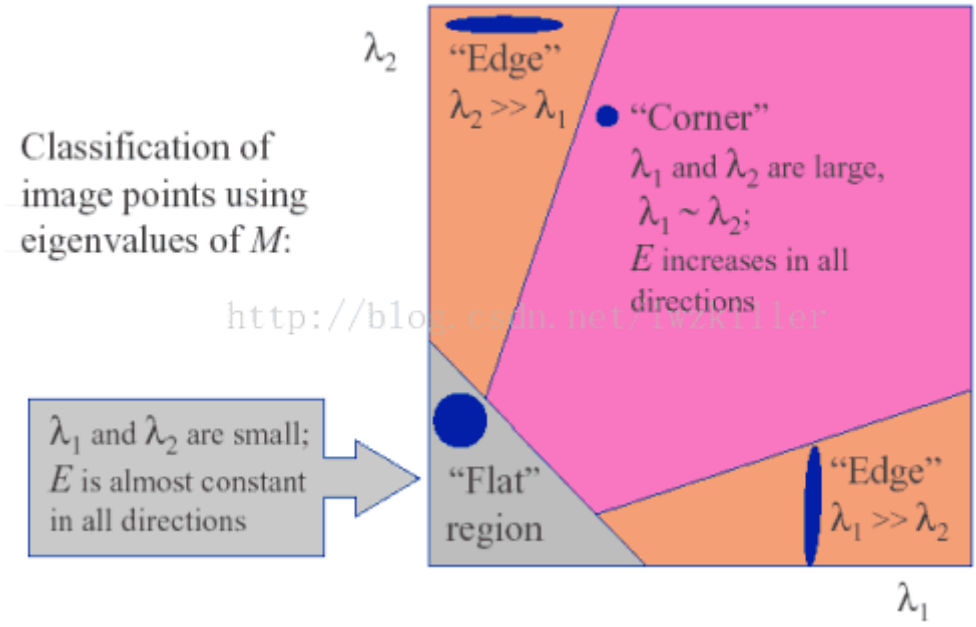
  难道我们是直接求上述的E(u,v)值来判断角点吗？Harris角点检测并没有这样做，而是通过对窗口内的每个像素的x方向上的梯度与y方向上的梯度进行统计分析。这里以Ix和Iy为坐标轴，因此每个像素的梯度坐标可以表示成(Ix,Iy)。针对平坦区域，边缘区域以及角点区域三种情形进行分析：



下图是对这三种情况窗口中的对应像素的梯度分布进行绘制：



如果使用椭圆进行数据集表示，则绘制图示如下：



虽然我们利用E(u,v)来描述角点的基本思想，然而最终我们仅仅使用的是矩阵M。让我们看看矩阵M形式，是不是跟协方差矩阵形式很像，像归像，但是还是有些不同，哪儿不同？一般协方差矩阵对应维的随机变量需要减去该维随机变量的均值，但矩阵M中并没有这样做，所以在矩阵M里，我们先进行各维的均值化处理，那么各维所对应的随机变量的均值为0，协方差矩阵就大大简化了，简化的最终结果就是矩阵M，是否明白了？我们的目的是分析数据的主要成分，相信了解PCA原理的，应该都了解均值化的作用。

如果我们对协方差矩阵M进行对角化，很明显，特征值就是主分量上的方差，这点大家应该明白吧？不明白的话可以复习下PCA原理。如果存在两个主分量所对应的特征值都比较大，说明什么？像素点的梯度分布比较散，梯度变化程度比较大，符合角点在窗口区域的特点；如果是平坦区域，那么像素点的梯度所构成的点集比较集中在原点附近，因为窗口区域内的像素点的梯度幅值非常小，此时矩阵M的对角化的两个特征值比较小；如果是边缘区域，在计算像素点的x、y方向上的梯度时，边缘上的像素点的某个方向的梯度幅值变化比较明显，另一个方向上的梯度幅值变化较弱，其余部分的点都还是集中原点附近，这样M对角化后的两个特征值理论应该是一个比较大，一个比较小，当然对于边缘这种情况，可能是呈45°的边缘，致使计算出的特征值并不是都特别的大，总之跟含有角点的窗口的分布情况还是不同。

注：M为协方差矩阵，需要大家自己去理解下，窗口中的像素集构成一个矩阵（2*n，假设这里有n个像素点），使用该矩阵乘以该矩阵的转置，即是协方差矩阵。

因此可以得出下列结论：

> 特征值都比较大时，即窗口中含有角点

> 特征值一个较大，一个较小，窗口中含有边缘

> 特征值都比较小，窗口处在平坦区域

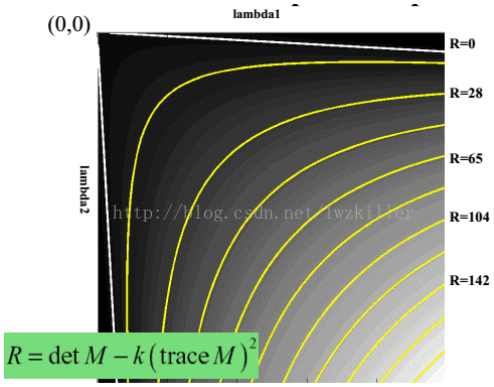
- 如何度量角点响应？

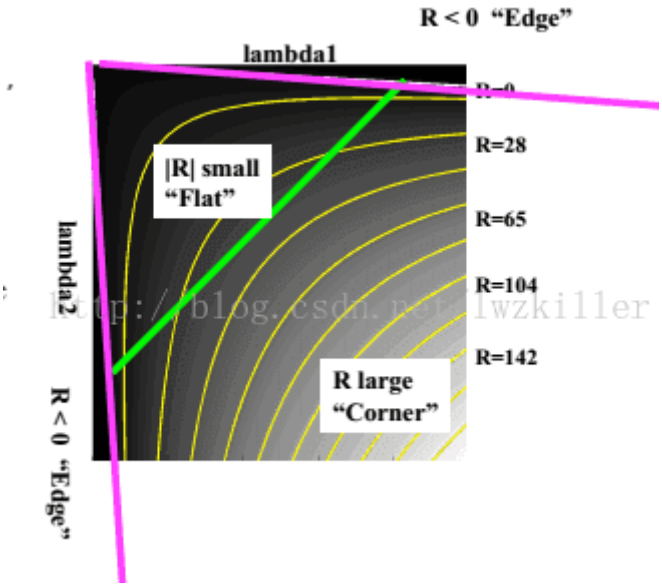
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

其中k是常量，一般取值为0.04~0.06，这个参数仅仅是这个函数的一个系数，它的存在只是调节函数的形状而已。

但是为什么会使用这样的表达式呢？一下子是不是感觉很难理解？其实也不难理解，函数表达式一旦出来，我们就可以绘制它的图像，而这个函数图形正好满足上面几个区域的特征。 通过绘制函数图像，直观上更能理解。绘制的R函数图像如下：





- API
  - cornerHarris()
- Harris角点的优点
  - 计算简单
  - 提取的点特征均匀且合理
  - 稳定：Harris算子对图像旋转、亮度变化、噪声影响和视点变换不敏感

- Harris 算子的局限性
  - 对尺度很敏感，不具有尺度不变性
  - 提取的角点精度是像素级的
  - 需要设计角点匹配算法

二，Shi-Tomas角点检测

※Shi-Tomasi 方法在很多情况下可以得到比 Harris 算法更好的结果。

- 原理

  Harris 角点检测中每个窗口的分数公式是将矩阵 M 的行列式与 M 的迹相减：

  $$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

  由于 Harris 角点检测算法的稳定性和 k 值有关，而 k 是个经验值，不好设定最佳值。

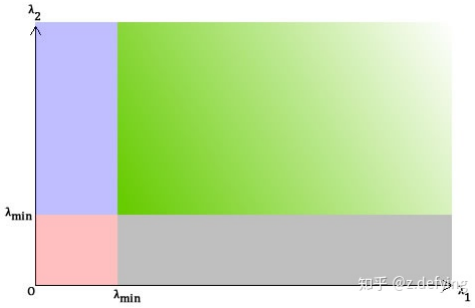
  Shi-Tomasi 发现，角点的稳定性其实和矩阵 M 的较小特征值有关，于是直接用较小的那个特征值作为分数。这样就不用调整k值了。

  所以 Shi-Tomasi 将分数公式改为如下形式：

  $$R = min(\lambda_1, \lambda_2)$$

  和 Harris 一样，如果该分数大于设定的阈值，我们就认为它是一个角点。

  我们可以把它绘制到 λ1 ～ λ2 空间中，就会得到下图：



- API

  goodFeaturesToTrack()

  Note the value returned is different from Harris.

- 该改进是计算适合跟踪的优质特征（good features to track），使得特征分布更均匀，其检测精度是亚像素级别的。

三，自定义角点检测

- 自定义角点检测器
  - 基于Harris 与 Shi-Tomasi 角点检测
  - 通过计算矩阵M得到$\lambda_1 \lambda_2$两个特征值根据他们得到的角点响应值
  - 然后设置自己的阈值实现计算出阈值得到有效角点相应的角点位置
- 相关API
  - cornerEigenValsAndVecs()：ロバストなエッジ検出やコーナー検出に利用できます.
  - cornerMinEigenVal()

四，亚像素级别的角点检测

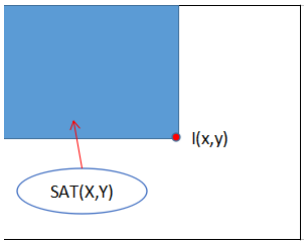
- 检测出来的角点坐标不一定都是整数，亦有可能是小数，为了精准定位这个角点有了
- 应用：
  - 跟踪
  - 三维重建
  - 相机校正
- 亚像素定位
  - 差值方法
  - 基于图像矩计算
  - 曲线拟合（高斯曲线， 多项式， 椭圆曲面）
- 相关API
  - cornerSubPix

五，积分图计算

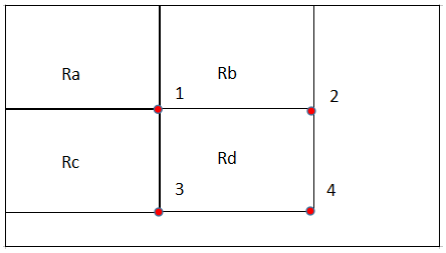
- 积分图原理

  图像是由一系列的离散像素点组成, 因此图像的积分其实就是求和. 图像积分图中每个点的值是原图像中该点左上角的所有像素值之和.

  首先建立一个数组 A 作为积分图像，其宽高与原图像相等. 然后对这个数组赋值，每个点存储的是`该点与图像原点所构成的矩形`中所有像素的和。

  定义了积分图的概念，就可以很方便的计算任意区域内的像素和，如下图所示：

  积分图数组初始化之后, 我们就得到了一张积分图:

  点1的积分 $SAT1 = Sum(Ra)$,
  点2的积分 $SAT2 = Sum(Ra) + Sum(Rb)$,
  点3的积分 $SAT3 = Sum(Ra) + Sum(Rc)$,
  点4的积分 $SAT4 = Sum(Ra) + Sum(Rb) + Sum(Rc) + Sum(Rd)$

  那么为了计算某个矩形像素和，比如区域 `Rd` 内所有点的像素值之和（积分）可以表示为：

  $$Sum(Rd) = SAT1 + SAT4 - SAT2 - SAT3$$

  所以无论矩形的尺寸大小，只需查找积分图像 4 次就可以快速计算任意矩形内像素值的和, 即算法复杂度为 O(4)。

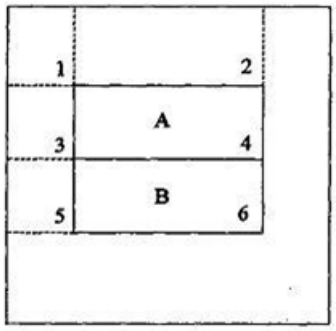
- 积分图应用

- Haar-like特征计算

  以如下一种 Haar-like 边缘特征为例

  假设需要计算的这种 Haar-like 特征在图中的位置如下所示：

  那么，A，B区域所构成的 Haar-like 边缘特征是：
  $Harr_{A-B} = Sum(A) - Sum(B)$

  $= [SAT4 + SAT1 - SAT2 - SAT3] - [SAT6 + SAT3 - SAT4 - SAT5]$

  显然，对一个灰度图而言，事先将其积分图构建好，当需要计算灰度图某个区域内所有像素点的像素值之和的时候，利用积分图，通过查表运算，可以迅速得到结果。

- 实现自适应阈值化

  自适应阈值是一种局部方法。它的原理是根据每个像素的邻域（如 5x5）计算阈值，如将每个像素的值与指定的邻域的平均值进行比较，如果某像素的值与它的局部平均值差别很大，就会被当作异常值在阈值化过程中被分离。

  如若不采用积分图像，则每个像素比较时，都需要进行 5 x 5 次加法运算；而采用积分图像，运算复杂度不随邻域大小而改变，每次只需计算 2 次加法和 2 次减法。

- BoxFilter快速计算

  参见SURF特征检测

- 滑动窗口

  与BoxFilter类似，略

- API

  - integral()

    可以求得:

    $sum(X,Y) = \sum_{x<X,y<Y} image(x,y)$

    $sqsum(X,Y) = \sum_{x<X,y<Y} image(x,y)^2$

    $tilted(X,Y) = \sum_{y<Y,abs(x_X+1)\leq Y-y-1} image(x,y)$ 旋转45度。

六，Haar-like特征

- 综述

  Haar特征是一种用于目标检测或识别的图像特征描述子，Haar特征通常和AdaBoost分类器组合使用，而且由于Haar特征提取的实时性以及AdaBoost分类的准确率，使其成为人脸检测以及识别领域较为经典的算法。

- 特征

  - 高类间变异性：每个特征之间差别很大
  - 低类间变异性：同意特征之间的结果应该相互吻合
  - 局部强度差
  - 不同尺度
  - 计算效率高

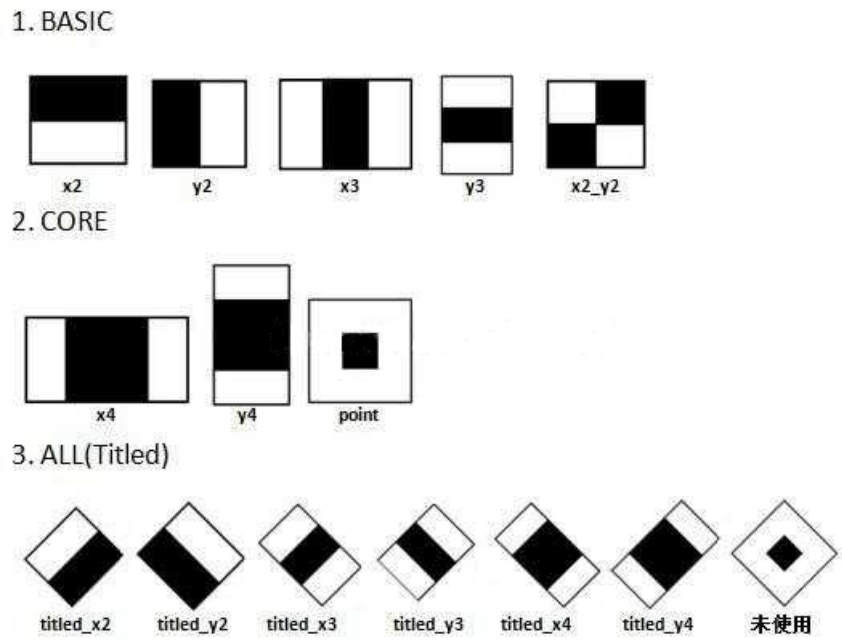
- Haar特征

  特征模板内有白色和黑色两种矩形，并定义该模板的特征值为白色矩形像素和减去黑色矩形像素和。

  按照OpenCV代码，Haar特征值=整个Haar区域内像素和×权重 + 黑色区域内像素和×权重

  $$featureValue(x) = weight_{all} \times \sum_{Pixel \in all} Pixel + weight_{black} \times \sum_{Pixel \in black} Pixel$$

  Haar特征值反映了图像的灰度变化情况。例如：脸部的一些特征能由矩形特征简单的描述，如：眼睛要比脸颊颜色要深，鼻梁两侧比鼻梁颜色要深，嘴巴比周围颜色要深等。但矩形特征只对一些简单的图形结构，如边缘、线段较敏感，所以只能描述特定走向（水平、垂直、对角）的结构。

  （1）如何快速计算那么多的特征？---积分图大显神通；

  （2）哪些矩形特征才是对分类器分类最有效的？---如通过AdaBoost算法来训练

- 归一化以达到光照不变性

  为了避免光照变化，需要归一化。

七，SIFT特征检测（Scale-invariant feature transform）

- 特点
  - SIFT特征是图像的局部特征，对旋转，尺度缩放，亮度变化保持不变性，对视角变化，仿射变换，噪声宜保持一定的稳定性。
  - 信息量丰富，适用于海量特征数据库中进行批评。
  - 多量性，少数物体也可产生大量的SIFT特性。
  - 高速性， 经优化的SIFT匹配算法甚至可以达到实时性。
- 步骤
  - 检测尺度空间的极值点
  - 精确定位特征点
  - 设定特征点的方向和参数
  - 生成特征点的描述子
- 图像金字塔
  利用图像金字塔的方法，我们可以得到一系列大小不一的图像，由大到小，从下到上构成的塔状模型。假设高斯金字塔的第l 层图像为Gl，则有：

$$G_l(i,j)=\sum_{m=-2}^{2}\sum_{n=-2}^{2}\omega(m,n)G_{l-1}(2i+m,2j+n)$$
$$(1\le l\le N,0\le i\le R_l,0\le j\le C_l)$$

  其中ω如下

$$\omega=\frac{1}{256}\begin{pmatrix}1&4&6&4&1\\4&16&24&16&4\\6&24&36&24&6\\4&16&24&16&4\\1&4&6&4&1\end{pmatrix}=\frac{1}{16}\begin{pmatrix}1&4&6&4&1\end{pmatrix}\times\frac{1}{16}\begin{pmatrix}1\\4\\6\\4\\1\end{pmatrix}$$

  写成上面的形式是为了说明，**二维窗口的卷积算子，可以写成两个方向上的1维卷积核（二项核）的乘积。**

  上面卷积形式的公式实际上完成了2个步骤：1）高斯模糊；2）降维。

  按上述步骤生成的$G_0,G_1\dots G_n$就构成了图像的高斯金字塔，其中G_0为金字塔的底层(与原图像相同)，G_n为金字塔的顶层。可见高斯金字塔的当前层图像是对其前一层图像先进行高斯低通滤波，然后做隔行和隔列的降采样(去除偶数行与偶数列)而生成的。

八， SURF特征检测

- SURF特征的基本介绍（Speeded Up Robust Features）
  - 特征检测
  - 尺度空间
  - 选择不变性（可以计算Orientation）
  - 特征向量
- 步骤

  1， **尺度空间的极值检测：搜索所有尺度空间上的图像，通过Hessian来识别潜在的对尺度和选择不变的兴趣点。**

构建Hessian矩阵的目的：生成图像稳定的边缘点(突变点)，跟Canny、拉普拉斯边缘检测的作用类似，为特征提取做准备。

- Hessian Matrix

$$H(I(x,y)) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial xy} \\ \frac{\partial^2 I}{\partial xy} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix}$$

黑塞矩阵(Hessian Matrix)是由一个多元函数的二阶偏导数构成的方阵，描述了函数的局部曲率。

在构建Hessian矩阵前需要对图像进行**高斯滤波**，经过滤波后的Hessian矩阵表达式为：

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

其中L(x,y,σ)=G(σ)∗I(x,y)，代表着图像的高斯尺度空间，是由图像和不同的高斯卷积得到。

- 补充

我们知道在离散数学图像中，一阶导数是相邻像素的灰度差：
$L_x = L(x+1,y) - L(x,y)$

二阶导数是对一阶导数的再次求导：
$L_{xx} = L(x+1,y) - 2L(x,y) + L(x-1,y)$

反过来看Hessian矩阵的判别式，

其实就是当前点对水平方向二阶偏导数乘以垂直方向二阶偏导数再减去当前水平、垂直二阶偏导的二次方：

$\Delta(H)$=$L_{xx} * L_{yy} - L_{xy} * L_{xy}$

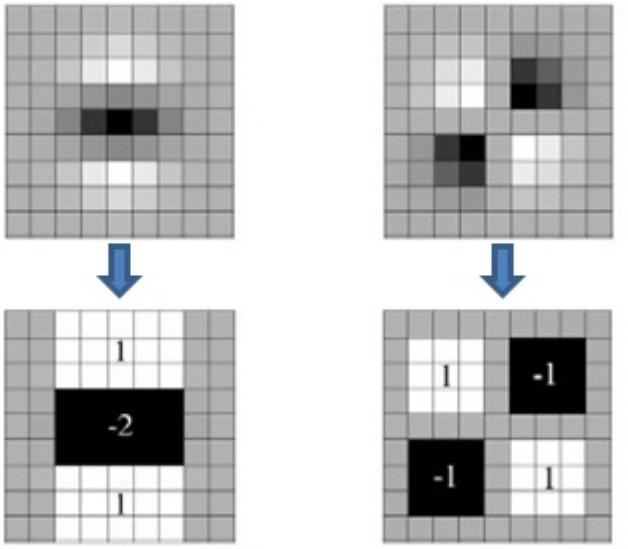
通过这种方法可以为图像中每个像素计算出其H行列式的决定值，并用这个值来判别图像局部特征点。

Hession矩阵判别式中的L(x,y)是原始图像的高斯卷积，由于高斯核服从正太分布，从中心点往外，系数越来越小，为了提高运算速度，SURF算法使用了盒式滤波器来替代高斯滤波器L，所以在Lxy上乘了一个加权系数0.9，目的是为了平衡因使用盒式滤波器近似所带来的误差，则H矩阵判别式可表示为：

$\Delta(H)$=$L_{xx} * L_{yy} - 0.9 * L_{xy}^2$

盒式滤波器和高斯滤波器的示意图如下：

Box Filter:



那么为什么盒式滤波器可以提高运算速度呢？

这就涉及到积分图的使用，盒式滤波器对图像的滤波转化成计算图像上不同区域间像素的加减运算问题，这正是积分图的强项，只需要简单积分查找积分图就可以完成。

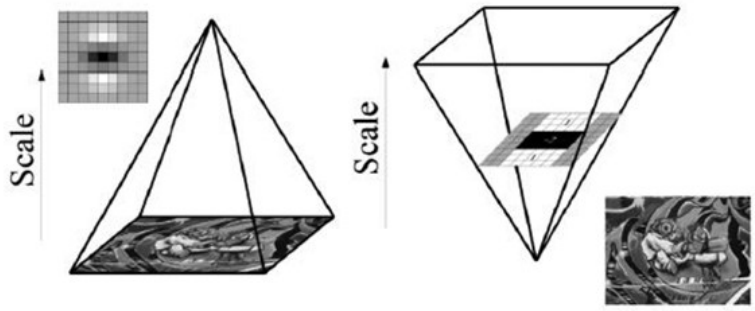
2，**在不同尺度空间发现关键点，非最大信号抑制**

- 尺度空间

  同SIFT算法一样，SURF算法的尺度空间由O组S层组成。

  不同的是,SIFT算法下一组图像的长宽均是上一组的一半，同一组不同层图像之间尺寸一样，但是所使用的尺度空间因子(高斯模糊系数σ)逐渐增大；

  而在SURF算法中，不同组间图像的尺寸都是一致的，不同的是不同组间使用的盒式滤波器的模板尺寸逐渐增大，同一组不同层图像使用相同尺寸的滤波器，但是滤波器的尺度空间因子逐渐增大。如下图所示：



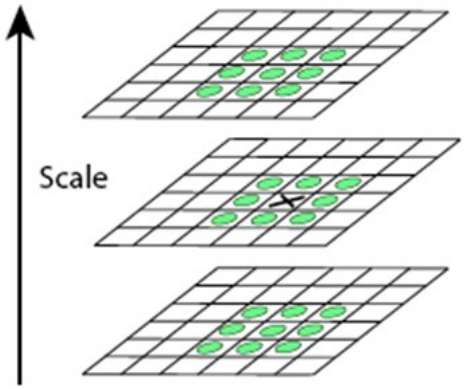
3，**特征点过滤并进行精确定位**

- SURF特征点的定位过程和SIFT算法一致

  将经过Hessian矩阵处理的每个像素点(即获得每个像素点Hessian矩阵的判别式值)与其图像域（相同大小的图像）和尺度域（相邻的尺度空间）的所有相邻点进行比较，当其大于（或者小于）所有相邻点时，该点就是极值点。

  如图所示，中间的检测点要和其所在图像的3×3邻域8个像素点，以及其相邻的上下两层3×3邻域18个像素点，共26个像素点进行比较。

  初步定位出特征点后，再经过滤除能量比较弱的关键点以及错误定位的关键点，筛选出最终的稳定的特征点。



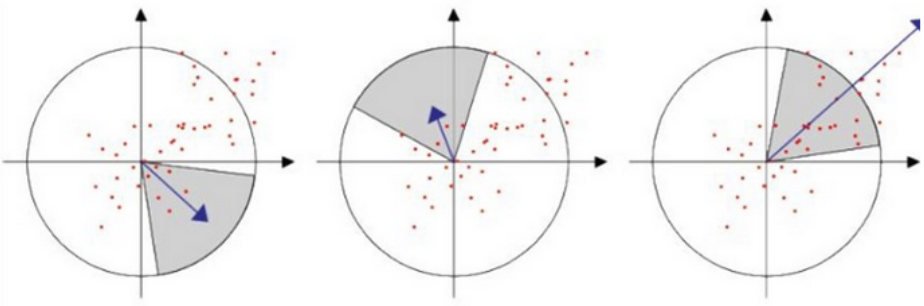
4，计**算特征点主方向**

- SIFT算法

  特征点主方向是采用在特征点邻域内统计其梯度直方图，横轴是梯度方向的角度，纵轴是梯度方向对应梯度幅值的累加，取直方图bin最大的以及超过最大80%的那些方向作为特征点的主方向。

- SURF算法

  采用的是统计特征点圆形邻域内的Harr小波特征，即在特征点的圆形邻域内，统计60度扇形内所有点的水平、垂直Harr小波特征总和，然后扇形以0.2弧度大小的间隔进行旋转并再次统计该区域内Harr小波特征值之后，最后将值最大的那个扇形的方向作为该特征点的主方向。该过程示意图如下：



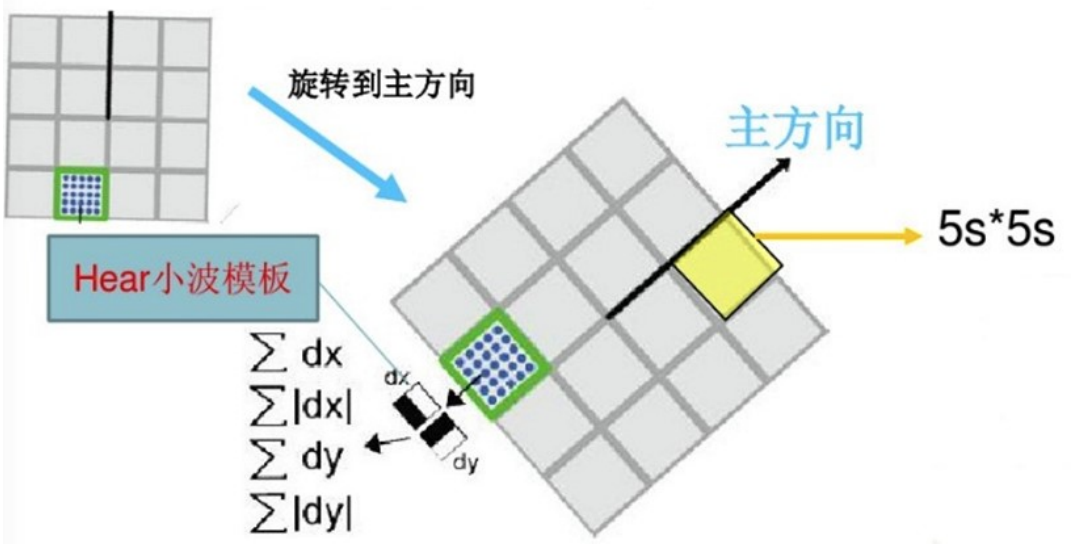
5，**生成特征描述**

- SIFT算法

  为了保证特征矢量的旋转不变性，先以特征点为中心，在附近邻域内将坐标轴旋转θ（特征点的主方向）角度，然后提取特征点周围4×4个区域块，统计每小块内8个梯度方向，这样一个关键点就可以产生128维的SIFT特征向量。

- SURF算法

  也是提取特征点周围4×4个矩形区域块，但是所取得矩形区域方向是沿着特征点的主方向，而不是像SIFT算法一样，经过旋转θ角度。每个子区域统计25个像素点水平方向和垂直方向的Haar小波特征，这里的水平和垂直方向都是相对主方向而言的。该Harr小波特征为水平方向值之和、垂直方向值之和、水平方向值绝对值之和以及垂直方向绝对之和4个方向。该过程示意图如下：



把这4个值作为每个子块区域的特征向量，所以一共有4×4×4=64维向量作为SURF特征的描述子，比SIFT特征的描述子减少了一半。

6，**特征点匹配**

在完成采集后，还需要建立图像的特征点数据库，每个特征点的数据结构包括：位置坐标、尺度、方向、特征向量（128或64维）。为新图像的每个特征点在数据库中逐个匹配，即根据特征向量的欧氏距离在数据库中寻找其最近邻和次近邻特征点，若最近邻距离或次近邻距离大于某一阈值，则特征匹配成功。

综上所述，可知SURF采用Henssian矩阵获取图像局部最值还是十分稳定的，但是在求主方向阶段太过于依赖局部区域像素的梯度方向，有可能使得找到的主方向不准确，后面的特征向量提取以及匹配都严重依赖于主方向，即使不大偏差角度也可以造成后面特征匹配的放大误差，从而匹配不成功；另外图像金字塔的层取得不足够紧密也会使得尺度有误差，后面的特征向量提取同样依赖相应的尺度，在这个问题上我们只能采用折中解决方法：取适量的层然后进行插值。

- 相关API

  因为需要付费，故而收在了contrib中。

  #include<opencv2/xfeatures2d.hpp>

  using namespace cv::xfeatures2d;

  Ptr detector = SURF::create(minHessian);

  。。。

九，HOG(histogram of oriented gradients)特征检测

（1）主要思想

在一副图像中，局部目标的表象和形状（appearance and shape）能够被梯度或边缘的方向密度分布很好地描述。（本质：梯度的统计信息，而梯度主要存在于边缘的地方）。

（2）具体的实现方法是：

- 灰度图像转换
- 采用Gamma校正法对输入图像进行颜色空间的标准化（归一化）；

  目的是调节图像的对比度，降低图像局部的阴影和光照变化所造成的影响，同时可以抑制噪音的干扰；

原理：通过非线性变换，让图像从暴光强度的线性响应变得更接近人眼感受的响应，即将漂白（相机曝光）或过暗（曝光不足）的图片，进行矫正

```
$I(x, y) = I(x,y)^{gamma}$  （一般可取gamma=1/2）
```

- 梯度计算（Sobel算子）
  求导操作不仅能够捕获轮廓，人影和一些纹理信息，还能进一步弱化光照的影响。
- 将图像划分成小cells（例如6*6像素/cell）

  目的是为局部图像区域提供一个编码，同时能够保持对图像中人体对象的姿势和外观的弱敏感性。

  加权投影：权重->梯度大小，binsw->梯度方向
- 统计每个cell的梯度直方图（不同梯度的个数），即可形成每个cell的descriptor
- 将每几个cell组成一个block，一个block内所有cell的特征descriptor串联起来便得到该block的HOG特征descriptor。

  归**一化**

  由于局部光照的变化以及前景-背景对比度的变化，使得梯度强度的变化范围非常大。这就需要对梯度强度做归**一化**。归一化能够进一步地对光照、阴影和边缘进行压缩。

  作者采取的办法是：把各个细胞单元组合成大的、空间上连通的区间（blocks）。这样，一个block内所有cell的特征向量串联起来便得到该block的HOG特征。这些区间是互有重叠的，这就意味着：每一个单元格的特征会以不同的结果多次出现在最后的特征向量中。我们将归一化之后的块描述符（向量）就称之为HOG描述符。
- 特征数据与检测窗口
- 匹配方法

（3）提高性能：

把这些局部直方图在图像的更大的范围内（我们把它叫区间或block）进行对比度归一化（contrast-normalized），所采用的方法是：先计算各直方图在这个区间（block）中的密度，然后根据这个密度对区间中的各个细胞单元做归一化。通过这个归一化后，能对光照变化和阴影获得更好的效果。

（4）优点：

与其他的特征描述方法相比，HOG有很多优点。首先，由于HOG是在图像的局部方格单元上操作，所以它对图像几何的和光学的形变都能保持很好的不变性，这两种形变只会出现在更大的空间领域上。其次，在粗的空域抽样、精细的方向抽样以及较强的局部光学归一化等条件下，只要行人大体上能够保持直立的姿势，可以容许行人有一些细微的肢体动作，这些细微的动作可以被忽略而不影响检测效果。因此HOG特征是特别适合于做图像中的人体检测的。

参考

十，LBP（Local Binary Patterns）

LBP（Local Binary Pattern，局部二值模式）是一种用来描述图像局部纹理特征的算子；

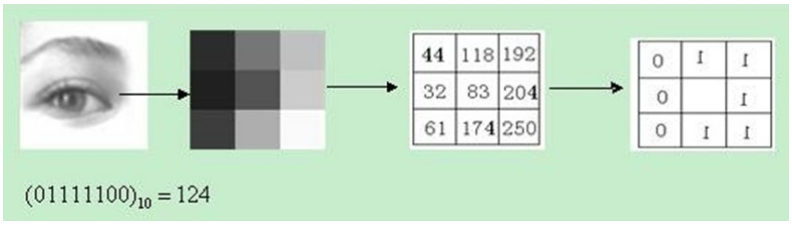
它具有旋转不变性和灰度不变性等显著的优点。

用于纹理特征提取。而且，提取的特征是图像的局部的纹理特征；

- LBP特征描述

基础款

原始的LBP算子定义为在3*3的窗口内，以窗口中心像素为阈值，将相邻的8个像素的灰度值与其进行比较，若周围像素值大于中心像素值，则该像素点的位置被标记为1，否则为0。这样，3*3邻域内的8个点经比较可产生8位二进制数（通常转换为十进制数即LBP码，共256种），即得到该窗口中心像素点的LBP值，并用这个值来反映该区域的纹理信息。



顺时针方向构造LBP码。

中心对比度：$1_{mean} - 0_{mean}$ 即编码为1位置的像素的均值 - 编码为0的位置的像素均值。

进阶**款**

基本的 LBP算子的最大缺陷在于它只覆盖了一个固定半径范围内的小区域，这显然不能满足不同尺寸和频率纹理的需要。

为了适应不同尺度的纹理特征，并达到灰度和旋转不变性的要求，Ojala等对 LBP 算子进行了改进，将 3×3邻域扩展到任意邻域，并用圆形邻域代替了正方形邻域，改进后的 LBP 算子允许在半径为 R 的圆形邻域内有任意多个像素点。从而得到了诸如半径为R的圆形区域内含有P个采样点的LBP算子；

- 圆形LBP算子（灰度不变性）

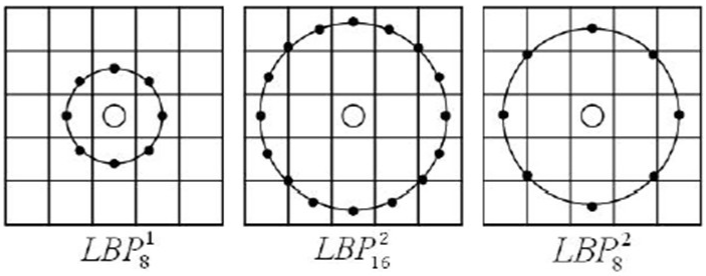


图2.2 几种LBP算子

采样点的计算方法：

$$x_p = X_c + R * cos(\frac{2\pi p}{P})$$
$$y_p = y_c + R * sin(\frac{2\pi p}{P})$$

R是采样半径，p是第p个采样点，P是采样数目。**注意：**这里是逆时针方向， 我们编码规则要求是顺时针。

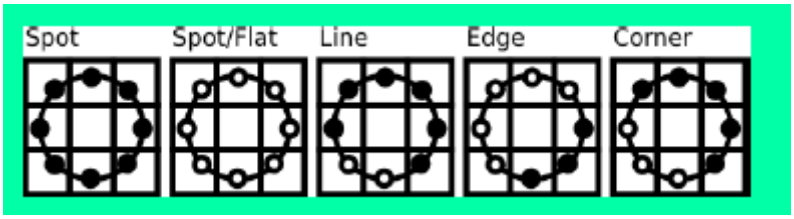
插值

由于计算的值可能不是整数，即计算出来的点不在图像上，我们使用计算出来的点的插值点。

插值方法有很多，Opencv使用的是双线性插值，双线性插值的公式如下：

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}$$

不同算子检测到的不同特征：



通过LBP特征的定义可以看出，LBP特征对光照变化是鲁棒的，其效果如下图所示：

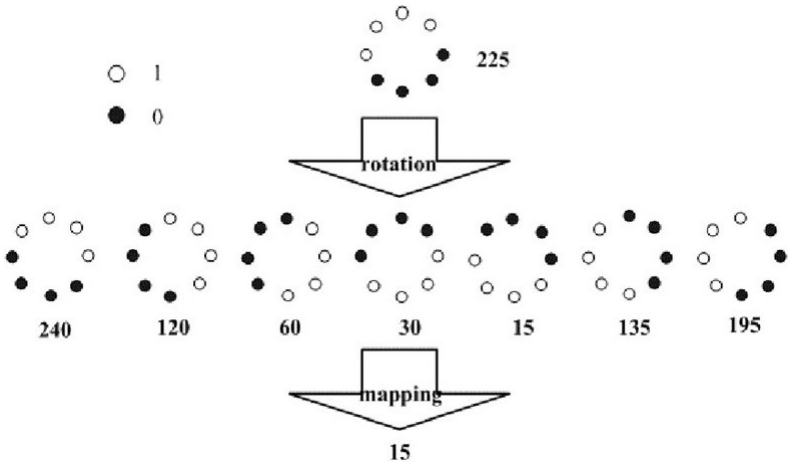
一些结论

从实验结果可以看出，半径越小，图像纹理越精细

邻域数目越小，图像亮度越低，合理，因此4位的灰度值很小

- LBP旋转不变模式（旋转不变性）

  首先不断的旋转圆形邻域内的LBP特征，根据选择得到一系列的LBP特征值，从这些LBP特征值选择LBP特征值最小的作为中心像素点的LBP特征。



- LBP等价模式（Uniform Pattern）

  由于一个LBP特征有多种不同的二进制形式，对于半径为R的圆形区域内含有P个采样点的LBP算子将会产生2P种模式。很显然，随着邻域集内采样点数的增加，二进制模式的种类是以指数形式增加的。因此，需要对原始的LBP模式进行降维，使得数据量减少的情况下能最好的表示图像的信息。

  Ojala等认为，在实际图像中，绝大多数LBP模式最多只包含两次从1到0或从0到1的跳变。因此，Ojala将“等价模式”定义为：当某个LBP所对应的循环二进制数从0到1或从1到0最多有两次跳变时，该LBP所对应的二进制就称为一个等价模式类。除等价模式类以外的模式都归为另一类，称为混合模式类。过这样的改进，二进制模式的种类大大减少，而不会丢失任何信息。

  模式数量由原来的2^p种减少为 $C_8^2 * 2 + 2 = p(p-1) + 2$种，其中$p$表示邻域集内的采样点数。

  对于3×3邻域内8个采样点来说，二进制模式由原始的256种减少为58种，即：它把值分为59类，58个uniform pattern为一类，其它的所有值为第59类。这样直方图从原来的256维变成59维。这使得特征向量的维数更少，并且可以减少高频噪声带来的影响。

  跳变小于2次的为等价模式类，共58个，他们对应的值按照从小到大分别编码为1—58，即它们在LBP特征图像中的灰度值为1—58，而除了等价模式类之外的混合模式类被编码为0，即它们在LBP特征中的灰度值为0，因此等价模式LBP特征图像整体偏暗。

- LBP用于人脸检测的原理

显而易见的是，上述提取的LBP算子在每个像素点都可以得到一个LBP“编码”，那么，对一幅图像（记录的是每个像素点的灰度值）提取其原始的LBP算子之后，得到的原始LBP特征依然是“一幅图片”（记录的是每个像素点的LBP值）。

LBP的应用中，如纹理分类、人脸分析等，一般都不将LBP图谱作为特征向量用于分类识别，而是采用LBP特征谱的统计直方图作为特征向量用于分类识别。

因为，从上面的分析我们可以看出，这个“特征”跟位置信息是紧密相关的。直接对两幅图片提取这种“特征”，并进行判别分析的话，会因为“位置没有对准”而产生很大的误差。后来，研究人员发现，可以将一幅图片划分为若干的子区域，对每个子区域内的每个像素点都提取LBP特征，然后，在每个子区域内建立LBP特征的统计直方图。如此一来，每个子区域，就可以用一个统计直方图来进行描述；整个图片就由若干个统计直方图组成。

- 步骤

  （1）首先将检测窗口划分为16×16的小区域（cell）；

  （2）对于每个cell中的一个像素，将相邻的8个像素的灰度值与其进行比较，若周围像素值大于中心像素值，则该像素点的位置被标记为1，否则为0。这样，3*3邻域内的8个点经比较可产生8位二进制数，即得到该窗口中心像素点的LBP值；

  （3）然后计算每个cell的直方图，即每个数字（假定是十进制数LBP值）出现的频率；然后对该直方图进行归一化处理。

  （4）最后将得到的每个cell的统计直方图进行连接成为一个特征向量，也就是整幅图的LBP纹理特征向量；

  然后便可利用SVM或者其他机器学习算法进行分类了。

十一，特征描述子

- 什么是特征

  描述一个特征其实就是描述特征与他周围内容的相互关系。

  我们描述特征是为了能够更好的匹配特征，使得我们认为描述相同的特征是同一个特征的是可信的（概率高的）。所以我们的描述必须是有代表性的，具有排他性的（discriminative）,而不是模棱两可泛泛而谈的。

- 特征不变性

  - 旋转不变性 rotation normalization
  - 尺度不变性 scale normalization
  - 放射不变性 affine normalization
  - 投影不变性 projected normalization

  特征匹配的方法是先找出特征显著的特征点（Feature Detect），然后再分别描述两个特征点（Feature Descriptor），最后比较两个描述的的相似程度来判断是否为同一个特征（Feature Match）。而在特征描述之前如果能够做到确定特征的方向，则可以实现旋转不变性（Rotation invarient），如果能确定尺度，则可以实现尺度不变性（Scale invarient）。

方法	提点方法	确定方向	确定尺度	描述方法
SIFT	DoG的最值点位置，再通过二次拟合来确定位置	特征领域的梯度直方图的最值方向	通过建立确定尺度空间，尺度空间中DoG最值所在尺度为特征尺度	在特征周围去一个Region，分为4*4的sub region，对每个sub region使用八方向的梯度表示，总共128维
SURF	Hessian矩阵行列式的最值	特征领域对Haar wavelet的最大相应方向	尺度空间中Hessian矩阵行列式最值所在尺度	在特征周围取一个region分为4*4的sub region，对每个sub region计算Haar wavelet响应，分别取X方向响应，X方向响应绝对值之和，Y方向响应，Y方向响应绝对值之和，共64维
BRIEF	无	无	无	在特征点周围随机抽取特征点对，比较两个点的像素强度，分别记为1：0，取256组组成256位的二进制字符串
ORB	使用FAST提点，使用HarrisCorner去除非角点	使用Intensity centroid方法来确定方向	无	通过贪心方法抽取符合正态分布的随机点对，其他同BRIEF
BRISK	使用FAST或AGAST提点	使用邻域随机抽样点对，对远点对做梯度确定方向	尺度空间中FAST提点最显著的尺度	使用短距离点对进行强度匹配，组成512位的二进制字符串

- Brute-Force（暴力匹配）

十二，FLANN特征匹配

快速最近邻逼近搜索函数库（Fast Approximate Nearest Neighbor Search Library）

十三，平面对象识别

- **findHomography** 发现两个平面的透视变换，生成变换矩阵
- **perspectiveTransform** 透视变换

十四，A(cceleration)KAZE局部匹配

- 特点
  - AOS构造尺度空间：非线性的构造尺度空间，更加精确。
  - Hessian矩阵特征点检测
  - 方向指定基于一阶微分图像
  - 描述子生成
- 与SIFT，SURF比较
  - 更加稳定
  - 非线性尺度空间
  - AKAZE速度更加快
  - 比较新的算法

十五，Brisk特征检测与匹配

- 介绍

  （Binary Robust Invariant Scalable Keypoints）

  - 构建尺度空间
  - 特征点检测
  - FAST9-16寻找特征点
  - 特征点定位
  - 关键点描述子

十六，级联分类器--人脸检测

\install\etc下的训练后的模型使用。

-- END