图像处理学习笔记（三&五）-特征子和配准

本次作业采用SURF算法，在图像配准中，特征点的描述往往不是位置这么简单，而是使用了一个N维向量来描述一个特征点，这些描述子之间可以通过定义距离公式来比较相近程度。

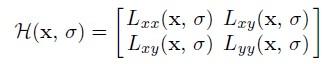
SurfDescriptorExtractor 是一个提取SURF特征点以及其描述的类。

**SURF算法原理**

surf特征是一种尺度不变的特征点，下面是SURF算法的大致流程。

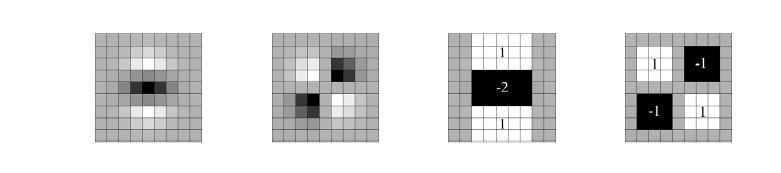
1.特征点的提取

1）利用Hessian矩阵，计算特征值α

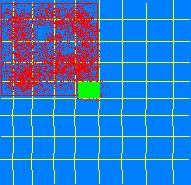


其中Lxx(x, σ)是高斯滤波后图像g(σ)的在x方向的二阶导数,其他的Lyy(x, σ)、Lxy(x, σ)都是g(σ)的二阶导数。为了减小计算量，原文使用了一个简单的方法，并利用了积分图像的优势（大大的减少计算量），方法其实很简单就是在模糊的基础上将原本的模块近似下。

一般计算图像的二阶导时，利用下面的公式d2f(x)/dx2=(f(x+1)-f(x))-(f(x)-f(x-1))=-2\*f(x)+f(x+1)+f(x-1)。以9X9滤波器为例，如下图所示，左边两幅图分别为灰度图像在中心点（黑色点）处的二阶导数d2f(x)/dx2和d2f(x)/dxdy的模板对应的值, 近似后变成右边的两幅图，图中灰色部分像素值为0。可是这样计算特征值不是也很复杂么？当然，所以作者提供了一种新思路--使用积分图像。



积分图像，顾名思义，即指当前像素点所在位置距原点（0,0）所包围面的所有灰度之和。下图绿色的部分为当前像素点，红色为积分区域。



这样计算图像中任意一块矩形区域的灰度之和Sx只需要利用矩形4个顶点(Xi,Yi)(i=1,2,3,4? 顺序为从上之下，先左后右)的积分值S（x,y）即可。

Sx=S(X1,Y1)+S(X4,Y4)-S(X2,Y2)-S(X3,Y3)

至此，大家应该知道近似二阶导数的高斯模板并引入积分图像的好处了吧，只需要在函数定义之前计算各个坐标点的积分图像，然后就能方便的求出hessian的特征值。不过由于函数模板的近似，这里需要修正下特征值α的求解公式：



这里Dxx和Dxy就是根据图一得到的，而Dyy和Dxx类似，只需要倒置一下模板即可。

2）根据是否为领域极大值判断特征点

这里要引入图像堆的概念，说简单点，就是一组大小相同的图像，这些图像都是根据不同大小高斯滤波二阶导模板。按照模板大小从小到大将Pi沿z轴方向排布，这样中间层的每个像素点的领域就为3X3X3（包括上下两层）。若该点的特征值α为这27个点中的最大值，那么可以认为该点为Feature points--特征点（图像依据这些特征点的匹配进行更多的操作，比如拼接，比较相似性等等）。

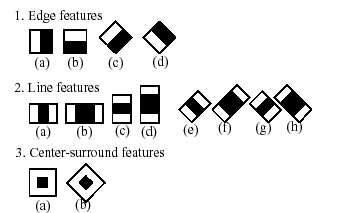
2.特征点的匹配

1）寻找特征向量

欲进行特征点的匹配，必须提取出特征点的特征向量再利用两个向量的相似程度认为两个点是否为两幅图像相互对应的点。

第一步：计算特征点的主方向

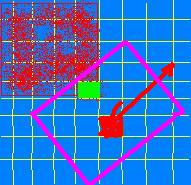
以特征点为圆心半径为6像素建立圆领域，计算得出里面有109个像素点。计算这些点的harr特征harrx和harry。那么该怎样计算任意一点的harr特征值？



选取edge features前两个作为harrx和harry值，这个方向有些类似与梯度方向，不过这里的领域显然更广。至于计算么，依旧是利用积分图像。

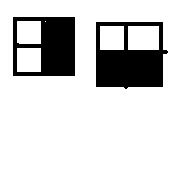
对这109个像素点分别求出各自的向量的方向angle=acrtan(harry/harrx) ，根据最近邻原则将这些 angle划分到60，120，...，300，360等6个值上。划分在同一范围上的像素点分别将他们的harrx和harry相加即可。不过为了体现相邻像素点的更大影响，还需要考虑高斯权重系数。这样得到最大的harrx和最大的harry,组成了主方向向量。

第二步.提取特征描述符

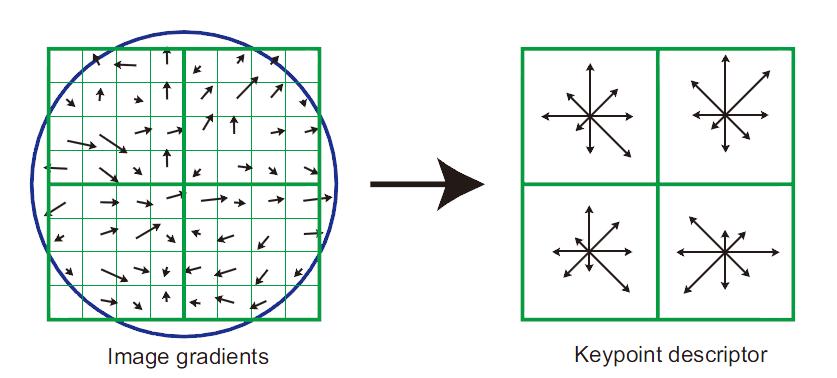


图中红色箭头为上面计算出来的主方向，按上图所示选取该红色特征点的8X8邻域（紫色边框内部）

计算得到4X4个像素块的梯度大小和方向（可以利用上面已经计算的harrx和harry），将8X8区域分割为2X2个区域T1,2,3,4，这样每个区域就包括了4个更小的由4个像素点组成的区域.



harrx和harry就是利用白色部分像素灰度值减去黑色部分像素灰度值即可得到（harrx,harry）方向向量。这样的向量一共有16个，将这些方向向量的方向角归并到上下左右斜上下8个方向上，并在T1,2,3,4中计算这8个方向的值。



那么这个4X8=32维向量即为所求的特征描述符。

3）特征点的匹配

采用最简单的两向量内积最大值为最匹配的点，设定一阈值，只有当这个最大值大于该阈值方可认为两特征点匹配。我们设定了阈值为400（minHessian = 400）。

用来计算的描述子提取被表达成一个高维空间的向量vector.所有实现 vector 特征描述子子提取的部分继承了 [DescriptorExtractor](http://www.opencv.org.cn/opencvdoc/2.3.2/html/modules/features2d/doc/common_interfaces_of_descriptor_extractors.html?highlight=surf#DescriptorExtractor) 接口.

DescriptorExtractor::compute

根据检测到的图像（第一种情况）或者图像集合（第二种情况）中的关键点(检测子)计算描述子.

C++:

 void DescriptorExtractor::compute(const Mat& image, vector<KeyPoint>& keypoints, Mat& descriptors) const

C++:

 void DescriptorExtractor::compute(const vector<Mat>& images, vector<vector<KeyPoint>>& keypoints, vector<Mat>& descriptors)const

|  |  |
| --- | --- |
| Parameters: | image – 图像.  images – 图像集. |

:param keypoints:输入的特征关键点. 不能被计算特征描述子的关键点被略过. 另外一些情况新的特征关键点增加进来, 比如: SIFT 增加了几个主方向上的特征关键点.

|  |  |
| --- | --- |
| Parameters: | descriptors – 计算特征描述子. In the second variant of the method descriptors[i] are descriptors computed for akeypoints[i]`. Row ``j is the keypoints (or keypoints[i]) is the descriptor for keypoint j-th keypoint. |

代码实现：

//-- Step 1: Detect the keypoints using SURF Detector

int minHessian = 400;

SurfFeatureDetector detector( minHessian );

…

//-- Step 2: Calculate descriptors (feature vectors)

SurfDescriptorExtractor extractor;

Mat descriptors\_1, descriptors\_2;

extractor.compute( img\_1, keypoints\_1, descriptors\_1 );

extractor.compute( img\_2, keypoints\_2, descriptors\_2 );

//-- Step 3: Matching descriptor vectors with a brute force matcher

BFMatcher matcher(NORM\_L2);

std::vector< DMatch > matches;

matcher.match( descriptors\_1, descriptors\_2, matches );

// 选择了25个配准点

std::nth\_element(matches.begin(),matches.begin()+24,matches.end());

matches.erase(matches.begin()+25,matches.end());

Surf特征子和配准的执行文件在D:\DIP\opencv2.4.9\opencv-build\bin\Debug>，参数是两个用来比较的图像文件，放在上级目录中，因此命令行：

Cd D:\DIP\opencv2.4.9\opencv-build\bin\Debug>

D:\DIP\opencv2.4.9\opencv-build\bin\Debug>SURF\_descriptor.exe ..\..\..\dfmz1.jpg ..\..\..\dfmz2.jpg

然后在两幅图上分别表示特征点，并选取两图中25对特征点连接成线对比。