在上篇博客特征点检测学习_1(sift 算法) 中简单介绍了经典的 sift 算法，sift 算法比较稳定，检测到的特征点也比较多，其最大的确定是计算复杂度较高。后面有不少学者对其进行了改进，其中比较出名的就是本文要介绍的 surf 算法，surf 的中文意思为快速鲁棒特征。本文不是专门介绍 surf 所有理论（最好的理论是作者的论文）的，只是对 surf 算法进行了下整理，方便以后查阅。

网上有些文章对 surf 做了介绍，比如：

http://wuzizhang.blog.163.com/blog/static/78001208201138102648854/

surf 算法原理,有一些简单介绍.

http://blog.csdn.net/andkobe/article/details/5778739

对 surf 的某些细节做了通俗易懂的解释.

http://www.owlei.com/DancingWind/

这篇文章名字叫做《surf 原文翻译》，写得非常好，看完会对 surf 中采用的一些技术更加深入的理解，不过本文却不是翻译英文的，而是该作者自己的理解，对积分图，Hessian 矩阵等引入的原因都做了通俗的解释，推荐一看。

一、Surf 描述子形成步骤

1. 构造高斯金字塔尺度空间

其实 surf 构造的金字塔图像与 sift 有很大不同，就是因为这些不同才加快了其检测的速度。Sift 采用的是 DOG 图像，而 surf 采用的是 Hessian 矩阵行列式近似值图像。首先来看看图像中某个像素点的 Hessian 矩阵，如下：

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

即每一个像素点都可以求出一个 Hessian 矩阵。但是由于我们的特征点需要具备尺度无关性，所以在进行 Hessian 矩阵构造前，需要对其进行高斯滤波。这样，经过滤波后在进行 Hessian 的计算，其公式如下：

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

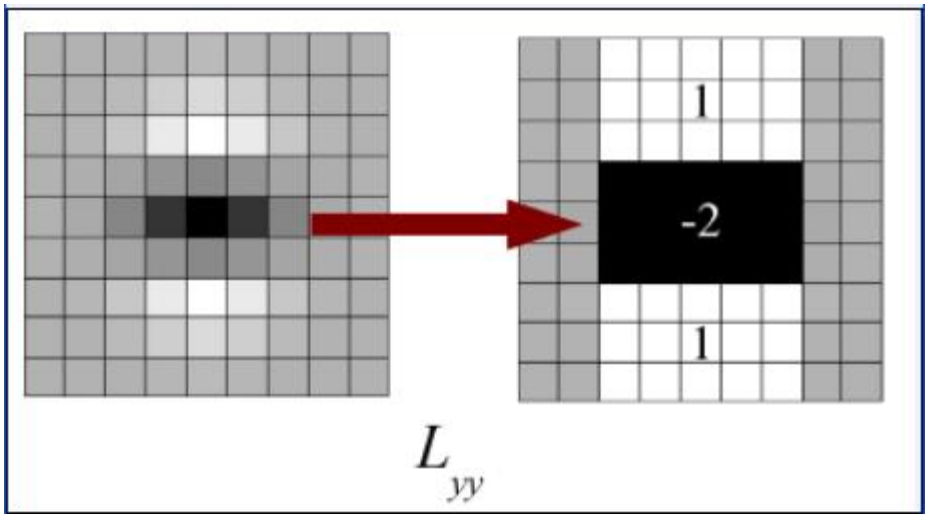
公式中的符号，估计有点数学基础的朋友都能够猜到，这里就不多解释了。

最终我们要的是原图像的一个变换图像，因为我们要在这个变换图像上寻找特征点，然后将其位置反映射到原图中，例如在 sift 中，我们是在原图的 HOG 图上寻找特征点的。那么在 surf 中，这个变换图是什么呢？从 surf 的众多资料来看，就是原图每个像素的 Hessian 矩阵行列式的近似值构成的。其行列式近似公式如下：

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

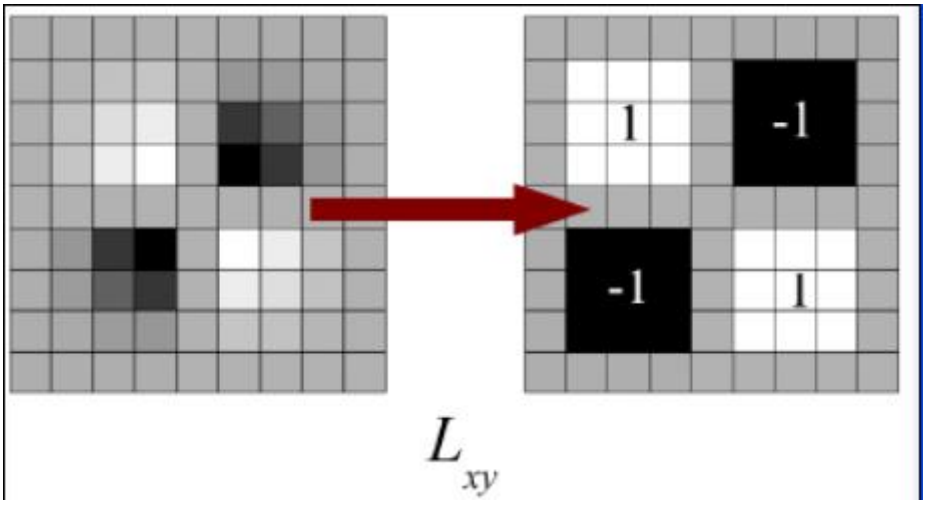
其中 0.9 是作者给出的一个经验值，其实它是有一套理论计算的，具体去看 surf 的英文论文。

由于求 Hessian 时要先高斯平滑，然后求二阶导数，这在离散的像素点是用模板卷积形成的，这 2 中操作合在一起用一个模板代替就可以了，比如说 y 方向上的模板如下：



该图的左边即用高斯平滑然后在 y 方向上求二阶导数的模板，为了加快运算用了近似处理，其处理结果如右图所示，这样就简化了很多。并且右图可以采用积分图来运算，大大的加快了速度，关于积分图的介绍，可以去查阅相关的资料。

同理，x 和 y 方向的二阶混合偏导模板如下所示：



上面讲的这么多只是得到了一张近似 hessian 行列式图，这例比 sift 中的 DOG 图，但是在金字塔图像中分为很多层，每一层叫做一个 octave，每一个 octave 中又有几张尺度不同的图片。在 sift 算法中，同一个 octave 层中的图片尺寸(即大小)相同，但是尺度(即模糊程度)不同，而不同的 octave 层中的图片尺寸大小也不相同，因为它是由上一层图片降采样得到的。在进行高斯模糊时，sift 的高斯模板大小是始终不变的，只是在不同的 octave 之间改变图片的大小。而在 surf 中，图片的大小是一直不变的，不同的 octave 层得到的待检测图片是改变高斯模糊尺寸大小得到的，当然了，同一个 octave 中个的图片用到的高斯模板尺度也不同。Surf 采用这种方法节省了降采样过程，其处理速度自然也就提上去了。

2. 利用非极大值抑制初步确定特征点

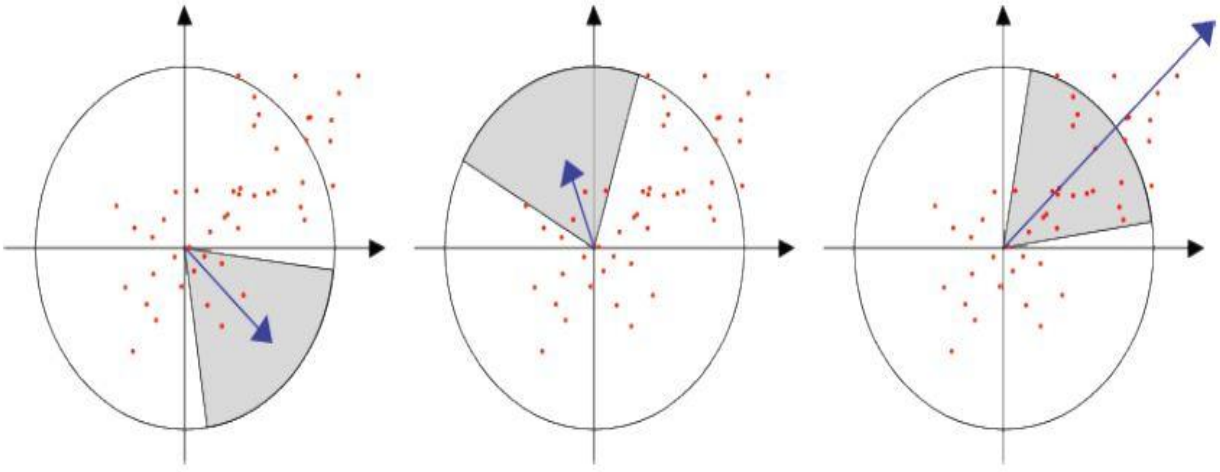
此步骤和 sift 类似，将经过 hessian 矩阵处理过的每个像素点与其 3 维领域的 26 个点进行大小比较，如果它是这 26 个点中的最大值或者最小值，则保留下来，当做初步的特征点。

3. 精确定位极值点

这里也和 sift 算法中的类似，采用 3 维线性插值法得到亚像素级的特征点，同时也去掉那些值小于一定阈值的点。

4. 选取特征点的主方向。

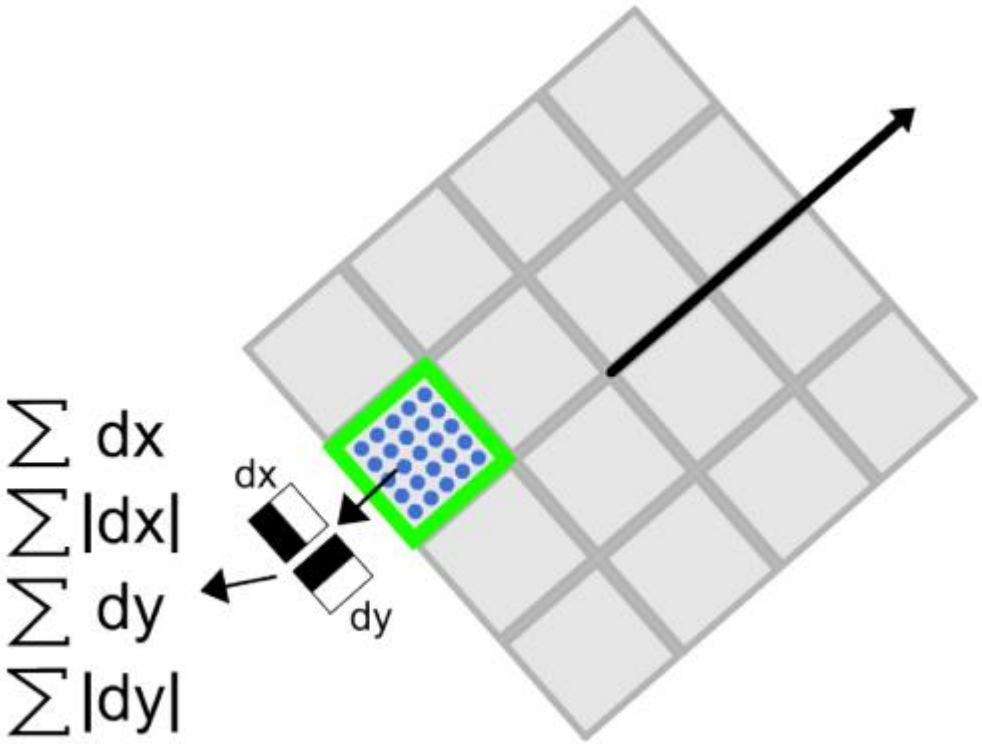
这一步与 sift 也大有不同。Sift 选取特征点主方向是采用在特征点领域内统计其梯度直方图，取直方图 bin 值最大的以及超过最大 bin 值 80%的那些方向做为特征点的主方向。而在 surf 中，不统计其梯度直方图，而是统计特征点领域内的 harr 小波特征。即在特征点的领域(比如说，半径为 6s 的圆内，s 为该点所在的尺度)内，统计 60 度扇形内所有点的水平 haar 小波特征和垂直 haar 小波特征总和，haar 小波的尺寸变长为 4s，这样一个扇形得到了一个值。然后 60 度扇形以一定间隔进行旋转，最后将最大值那个扇形的方向作为该特征点的主方向。该过程的示意图如下：



5. 构造 surf 特征点描述算子

在 sift 中，是在特征点周围取 16*16 的邻域，并把该领域化为 4*4 个的小区域，每个小区域统计 8 个方向梯度，最后得到 4*4*8=128 维的向量，该向量作为该点的 sift 描述子。

在 surf 中，也是在特征点周围取一个正方形框，框的边长为 20s(s 是所检测到该特征点所在的尺度)。该框带方向，方向当然就是第 4 步检测出来的主方向了。然后把该框分为 16 个子区域，每个子区域统计 25 个像素的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 haar 小波特征为水平方向值之和，水平方向绝对值之和，垂直方向之和，垂直方向绝对值之和。该过程的示意图如下所示：



这样每个小区域就有 4 个值，所以每个特征点就是 16*4=64 维的向量，相比 sift 而言，少了一半，这在特征匹配过程中会大大加快匹配速度。

二、特征点的匹配过程

surf 特征点的匹配过程和 sift 类似，这里不做详细介绍

三、实验部分

本次实验采用网上流行的 open surf，用 c++完成的，用到了 opencv 库，下载地址为：http://www.chrisevansdev.com/computer-vision-opensurf.html

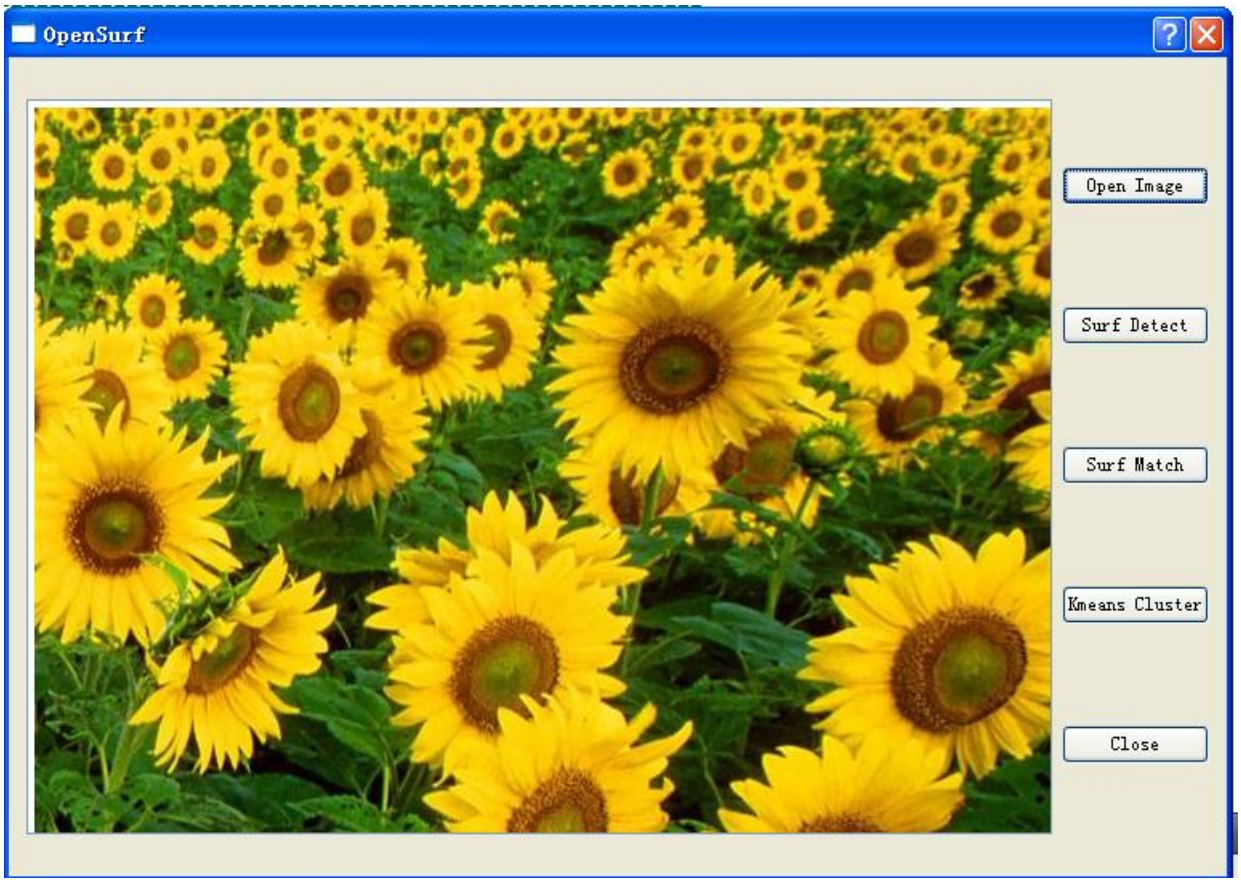
该代码的作者给出的主函数实现了 6 中功能，包括静态图片特征点的检测，视频中特征点的检测，图片之间的匹配，视频与图片之间的匹配，特征点聚类等 6 中功能。本次实验就简单的测试了图片的检测，匹配和特征点聚类 3 个功能。并加入了简单的界面。

开发环境为：opencv2.4.2+Qt4.8.2+open surf+windosxp

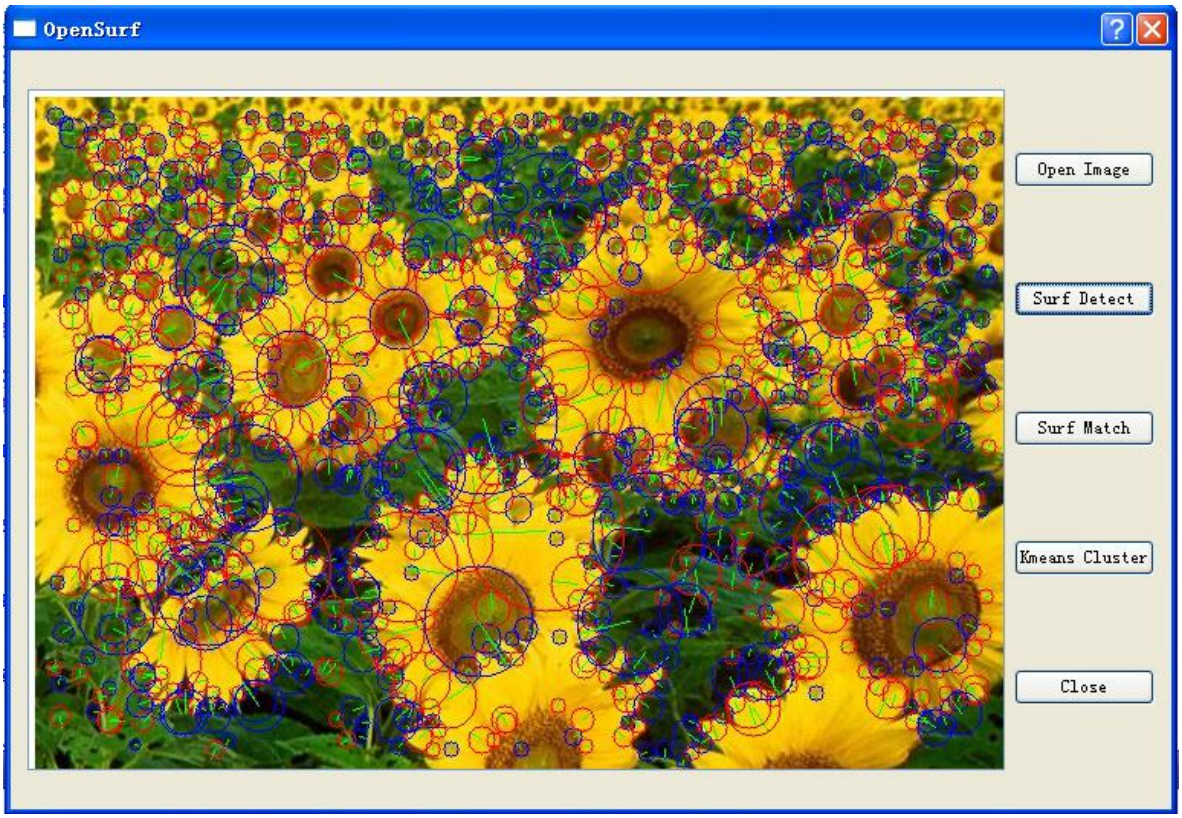
实验分为下面 3 个部分来描述。

Surf 特征点检测和描述

打开软件，单击 Open Image 按钮，选择一张待检测的图片，效果如下：



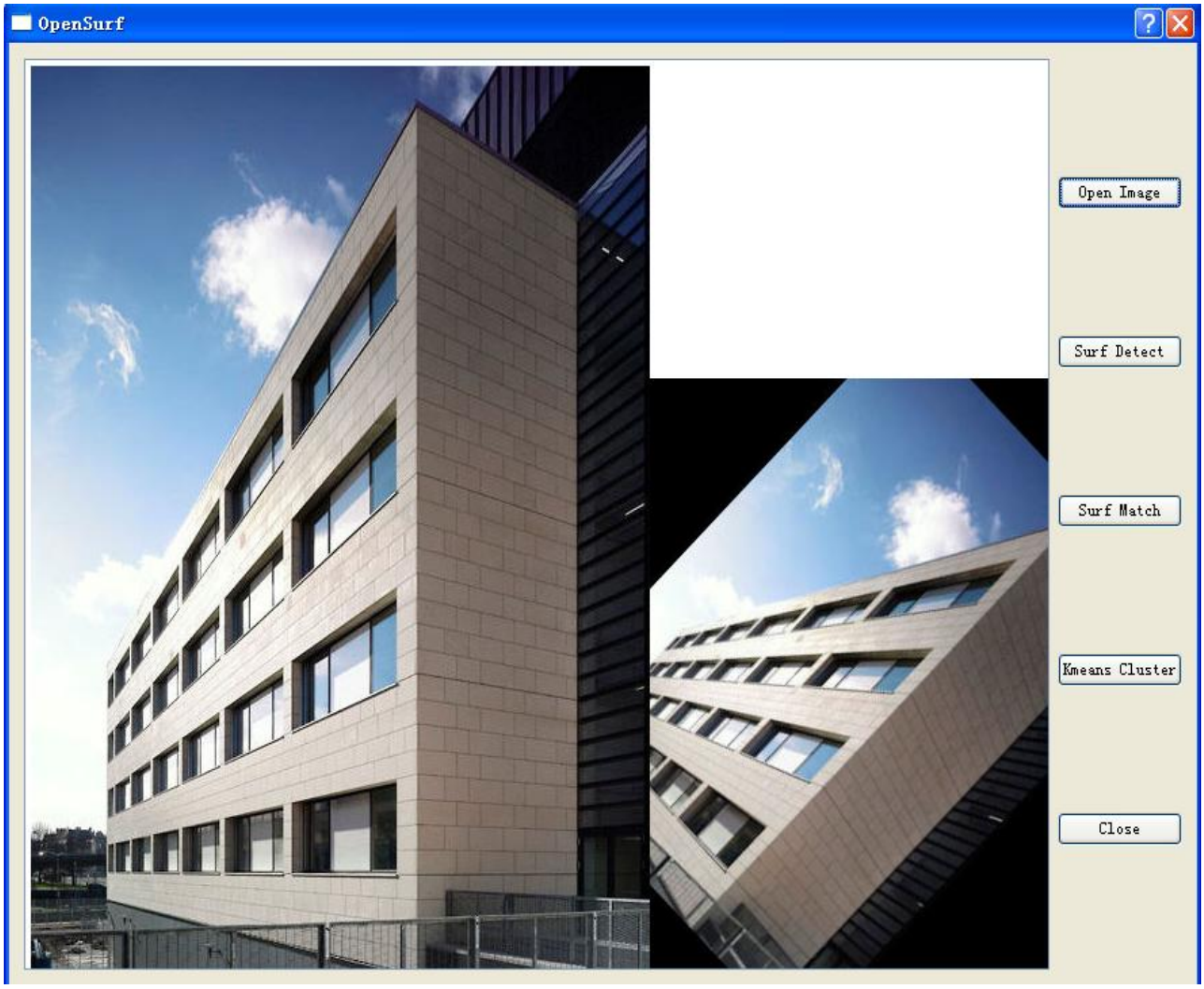
单击 Surf Detect 按钮，程序会对该图片进行特征点检测，并显示特征结果，包括特征点的主方向，尺度等信息。效果如下：



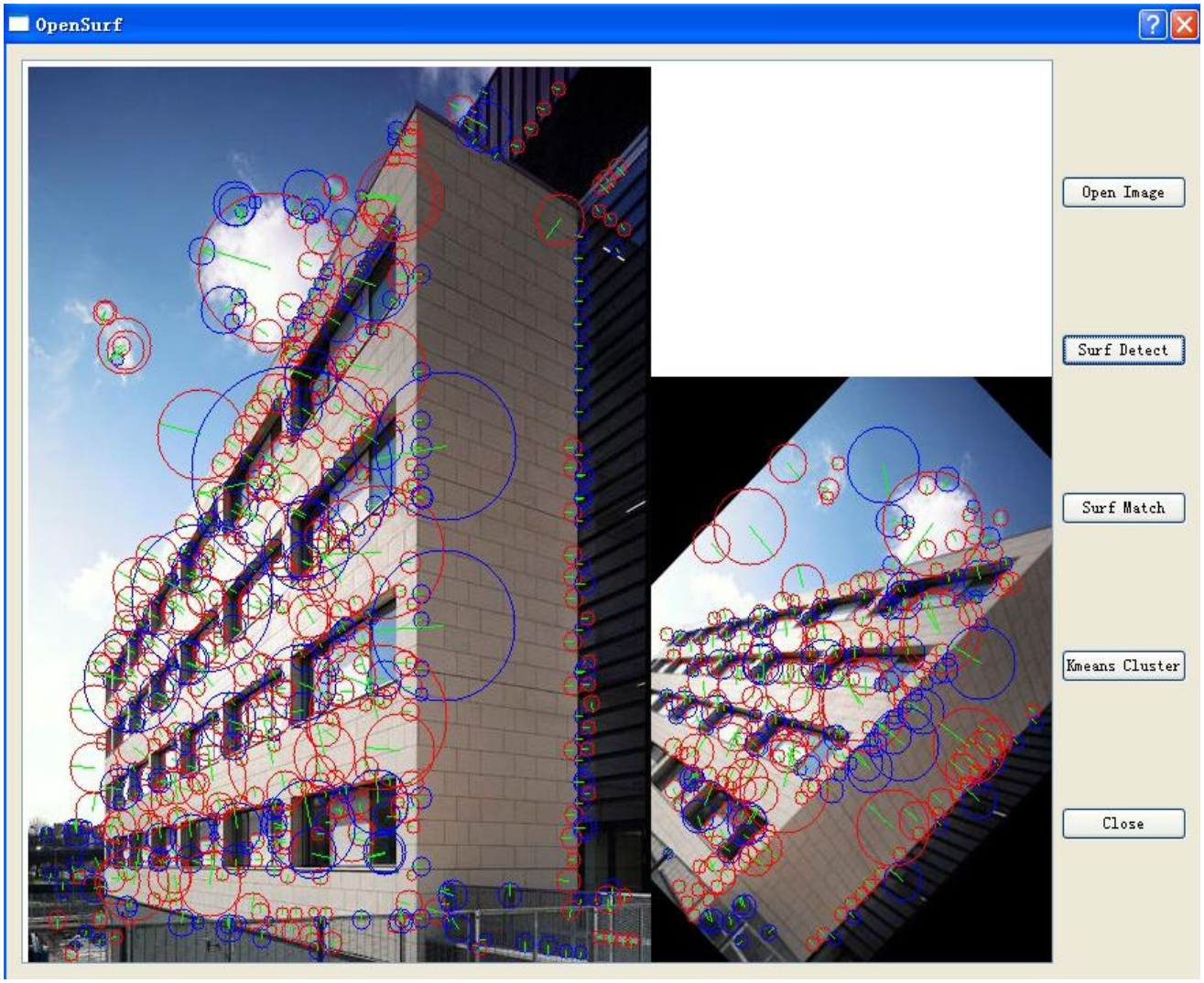
单击 Close 按钮退出程序。

Surf 特征点匹配

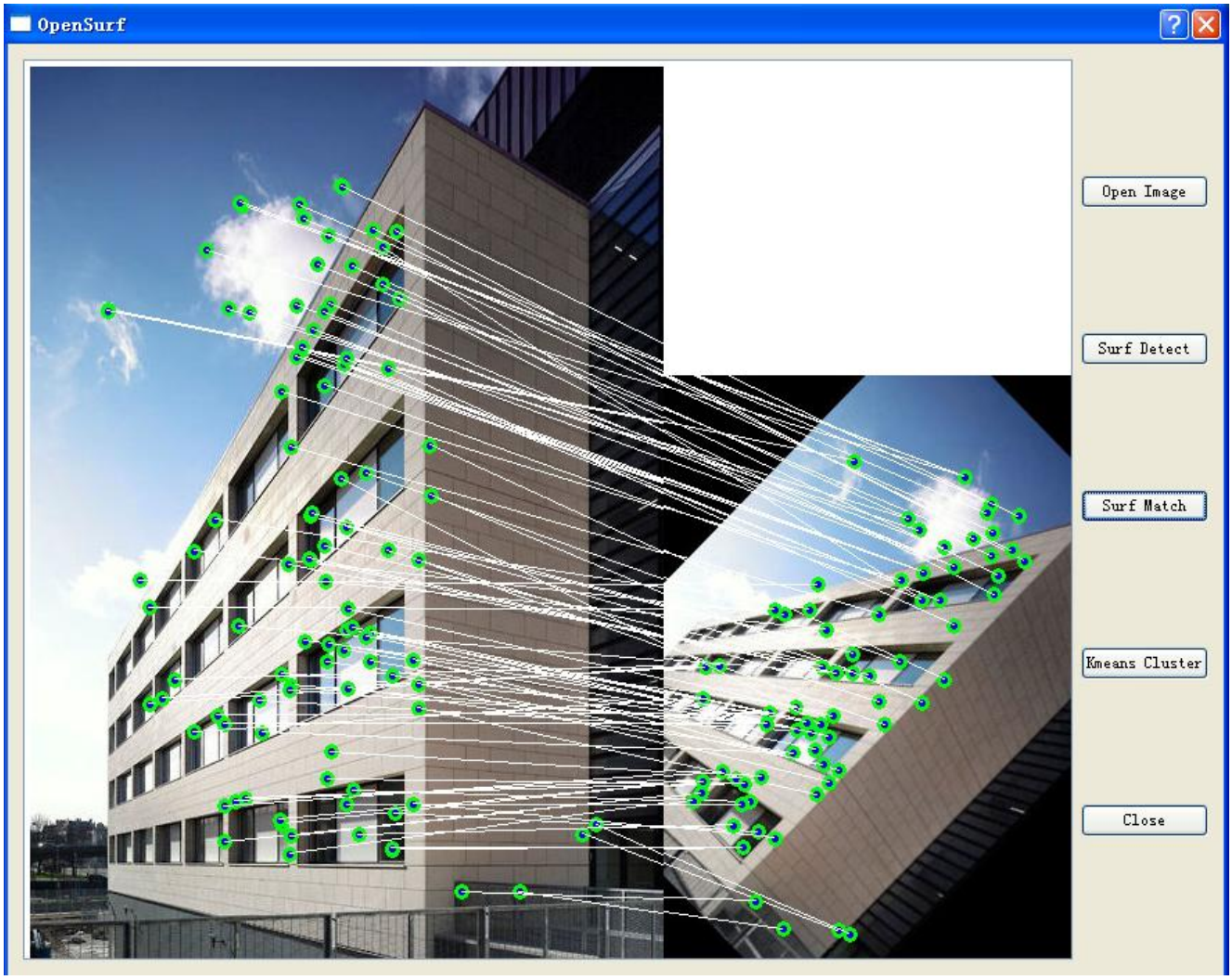
打开软件，单击 Open Image 按钮，依次打开 2 幅待匹配的图片，这 2 幅图片要有相同的内容，只是尺度，旋转，光照等不同。打开图片后如下：



单击 Surf Detect 按钮，和上面类似，会先对图片进行检测，效果如下：



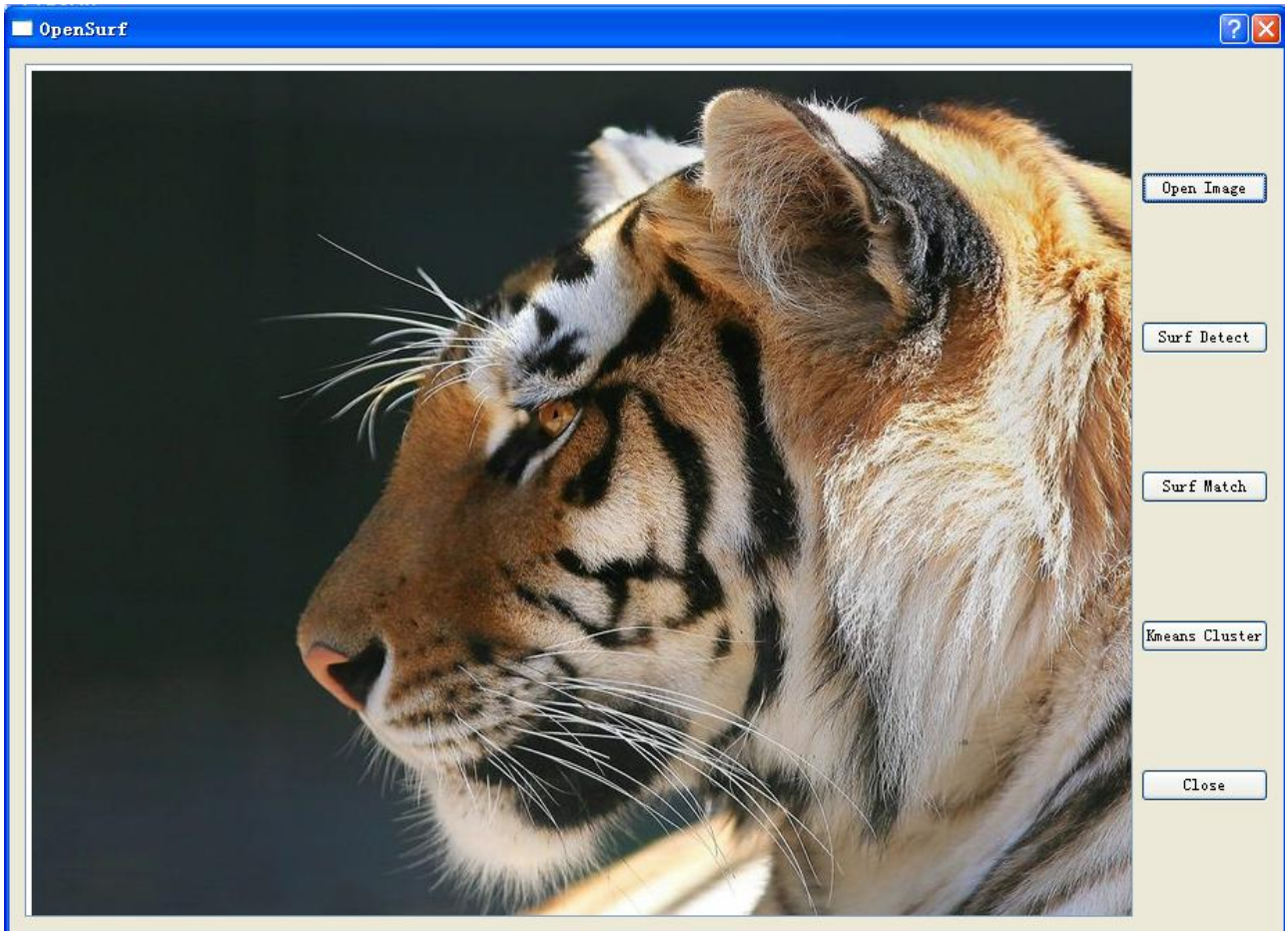
单击 Surf Match 按钮，程序会对检测到的图片进行特征点匹配，效果如下：



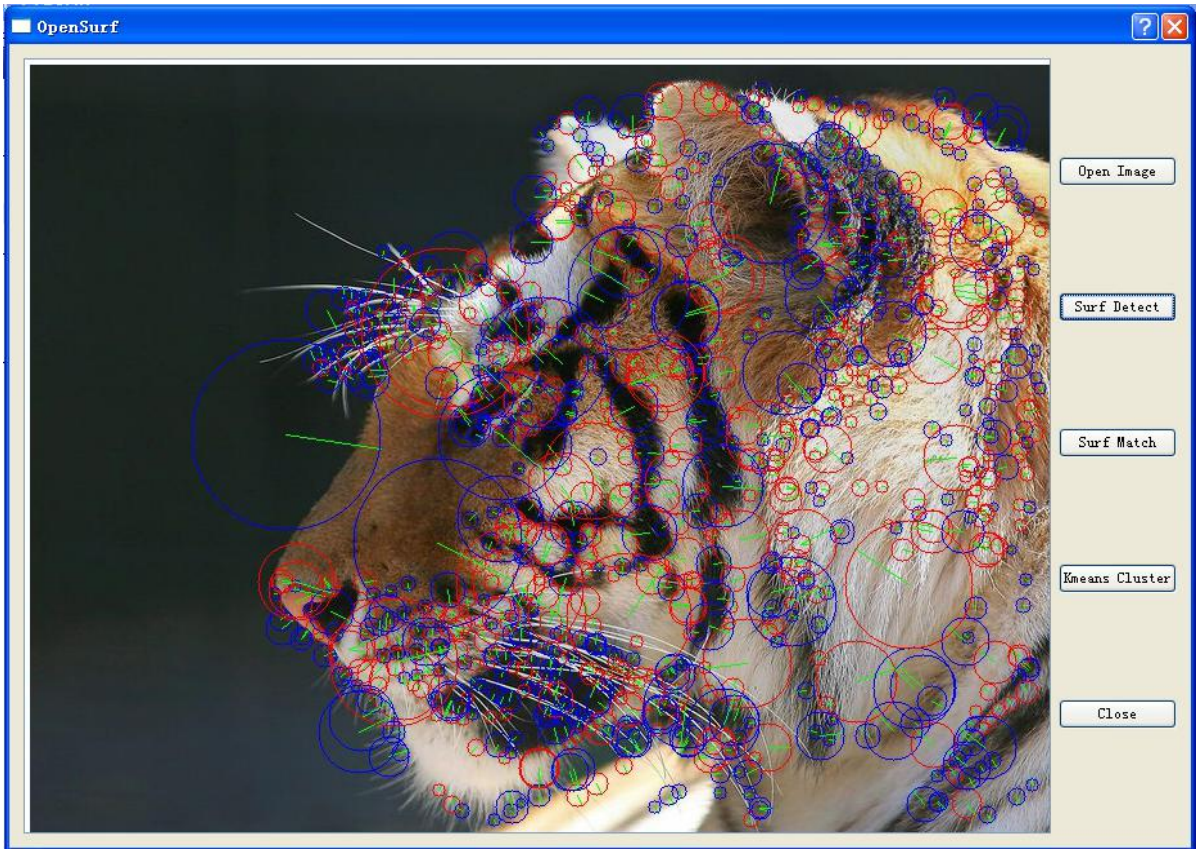
单击 Close 按钮退出程序。

Surf 特征点聚类

打开软件，单击 Open Image 按钮，选择一张待特征点分类的图片，如下所示：



单击 Surf Detect 按钮，首先对该图片进行 surf 特征点检测，如下：



单击 Kmeans Cluster 按钮，程序会对这些特征点集合进行聚类，并显示其结果，如下所示：

单击 Close 按钮退出程序。

总结：

Surf 在速度上比 sift 要快许多，这主要得益于它的积分图技术，已经 Hessian 矩阵的利用减少了降采样过程，另外它得到的特征向量维数也比较少，有利于更快的进行特征点匹配。

附录一：

和 RobHesson 运行时一样，这里的 open surf 运行时出现如下错误：

ipoint.obj:-1: error: LNK2019: 无法解析的外部符号 _cvFindHomography，该符号在函数 "int __cdecl translateCorners(class std::vector<struct std::pair<class Ipoint,class Ipoint>,class std::allocator<struct std::pair<class Ipoint,class Ipoint> > > &,struct CvPoint const * const,struct CvPoint * const)" (?translateCorners@@YAHAAV?$vector@U?$pair@VIpoint@@V1@@std@@V?$allocator@U?$pair@VIpoint@@V1@@std@@@2@@std@@QBUCvPoint@@QAU3@@Z) 中被引用

不过这次的原因是没有 opencv_calib3d242d.lib 库，因为本 open surf 在进行特征匹配时用到了 opencv 中的 3 维重建有关的函数 cvFindHomography(该函数是求 2 个图像间的单应矩阵)，所以很多人都会忘了添加这个库文件，就会导致这个错误。

如果用了 Qt 或 MFC 等界面设计代码时，编译该程序会报如下错误：

moc_open_surf.obj:-1: error: LNK2005: "public: void __thiscall Kmeans::SetIpoints(class std::vector<class Ipoint,class std::allocator<class Ipoint> > *)" (?SetIpoints@Kmeans@@QAEXPAV?$vector@VIpoint@@V?$allocator@VIpoint@@@std@@@std@@@Z) 已经在 main.obj 中定义

其实是 Open Surf 的作者可能没有考虑周全，它在 kmeans.h 文件中把 Kmeans 这个类的成员函数方法在头文件中实现了，其实这在标准 c++中是不支持的。解决方法就是把 kmeans.h 改造成 kemans.hpp(该方法我没有去试过)；另外一种方法就是新建一个 kmeans.cpp 文件，把成员函数的实现过程放在 cpp 文件中实现，我这次试验就是采用的这个方法。

附录二：

试验工程 code 下载。