计算机科学与技术　　学院

计算机视觉 课程实验报告

学号：	姓名：	班级：
实验题目：Harris 角点检测实现		
实验内容： 实验 8.1　Harris 角点检测： 实现 Harris 角点检测算法，并与 OpenCV 的 cornerHarris 函数的结果进行比较。		

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

一、角点检测相关介绍

1、经典的 Harris 角点检测

基本算法步骤：

1. 利用 Soble 计算出 XY 方向的梯度值得到 Ix 和 Iy
2. 计算出 Ix^2, Iy^2, Ix*Iy
3. (可选)利用高斯函数对 Ix^2, Iy^2, Ix*Iy 进行滤波，或用全一窗口
4. 计算局部特征结果矩阵 M 的特征值和响应函数的值

   R(i,j)=Det(M)-k(trace(M))^2　　(0.04<=k<=0.06)
5. 将计算出响应函数的值 C 进行局部极大值抑制，滤除一些不是角点的点，防止重叠，同时要满足大于设定的阈值

2、改进的 Shi-Tomasi 角点检测

(1)经典的 Harris 角点检测方法不难看出，该算法的稳定性和 k 有关，而 k 是个经验值，不好把握，浮动也有可能较大。鉴于此，改进的 Harris 方法（）直接计算出两个特征值，通过比较两个特征值直接分类，这样就不用计算 Harris 响应函数了。

(2)另一方面，我们不再用非极大值抑制了，而选取容忍距离,容忍距离内只有一个特征点：该算法首先选取一个具有最大最小特征值（把每个 harris 矩阵的最小特征值作为其衡量，认为其更有代表性）的点（即：max（min（e1，e2））， e1，e2 是 harris 矩阵的特征值）作为角点，然后依次按照最大最小特征值顺序寻找余下的角点，当然和前一角点距离在容忍距离内的新角点被忽略，防止重叠。

在 opencv 中的实现为：goodFeatureTrack（）

```
void goodFeaturesToTrack(InputArray image, OutputArray corners,
                         int maxCorners, double qualityLevel, double minDistance,
                         InputArray mask=noArray(), int blockSize=3,
                         bool useHarrisDetector=false, double k=0.04 );
```

3、FAST 角点检测算法

该算法检测的角点定义为在像素点的周围邻域内有足够多的像素点与该点处于不同的区

域。应用到灰度图像中，即有足够多的像素点的灰度值大于该点的灰度值或者小于该点的灰度值，便是角点。算法原理比较简单，但实时性很强。

Eg:若某像素点圆形邻域圆周上有 3/4 的点和该像素点不同（编程时不超过某阈值 th），则认为该点就是候选角点。*（这一思路可以使用机器学习的方法进行加速。对同一类图像，例如同一场景的图像，可以在 16 个方向上进行训练，得到一棵决策树，从而在判定某一像素点是否为角点时，不再需要对所有方向进行检测，而只需要按照决策树指定的方向进行 2-3 次判定即可确定该点是否为角点。）*

和 Harris 算法类似，该算法需要非极大值抑制。

补充：SUSAN 提取算子

基本原理是，与每一图像点相关的局部区域具有相同的亮度。如果某一窗口区域内的每一像元亮度值与该窗口中心的像元亮度值相同或相似，这一窗口区域将被称之为“USAN”。计算图像每一像元的“USAN”，为我们提供了是否有边缘的方法。位于边缘上的像元的“USAN”较小（像素值一般是连续变化的），位于角点上的像元的“USAN”更小。因此，我们仅需寻找最小的“USAN”，就可确定角点。该方法由于不需要计算图像灰度差，因此，具有很强的抗噪声的能力。

Fast 在 opencv 中的实现：FastFeatureDetector fast(参数列表)；

```
//示例代码
Mat  image, image1 = cv::imread ("test.jpg");
cv::cvtColor (image1,image,CV_BGR2GRAY);
//快速角点检测
std::vector<cv::KeyPoint> keypoints;
cv::FastFeatureDetector fast(40,true);
fast .detect (image,keypoints);
drawKeypoints(image,keypoints,image,cv::Scalar::all(255),cv::DrawMatchesFlags::
DRAW_OVER_OUTIMG);
```

二、具体 Harris 角点检测实现介绍

1、图像的梯度计算：

直接调用 opencv 的 sobel 算子进行滤波：

```
Ix:Sobel(img, dst, CV_64FC1, 0, 1, 3);
Iy:Sobel(img, dst, CV_64FC1, 1, 0, 3);
```

2、Ixx、Iyy、Ixy 的计算及局部特征结果矩阵 M 的特征值和响应函数的值：

通过第一步的大的 Ix 和 Iy，分别遍历对应元素值进行相乘得到新的所需 Mat，针对是否进行高斯的滤波（使用 opencv 库函数 GaussianBlur），设计两个计算的函数接口：

```
Gauss:   Mat computeImage(Mat& ixx, Mat& iyy, Mat& ixy, int wsize);
NoGauss: Mat computeImage(Mat& ix, Mat& iy, int wsize, int para);
```

两种的 Harris 检测主要框架代码如下：

```
//使用全一的窗口函数
void myHarrisCorner_ave(Mat &srcImg) {
    Mat image,src_grayImg,Ix,Iy,I_xx,I_yy,I_xy,R,filter_R,result;
```

```
    cvtColor(srcImg, src_grayImg, COLOR_BGR2GRAY);
    image = srcImg.clone();

    int wsize = 3;//窗口大小
    sobelGradient(src_grayImg, Ix,1);
    sobelGradient(src_grayImg,Iy, 2);
    I_xx = computeImage(Ix, Iy, wsize, 1);
    I_yy = computeImage(Ix, Iy, wsize, 2);
    I_xy = computeImage(Ix, Iy, wsize, 4);
    //计算响应值
     R = computeImage(Ix, Iy, wsize, 3);
     //局部非极大值抑制
    filter_R = filterR(R, 10);
    mixP(filter_R, image, 2);
    imshow("Ave", image);
    imwrite("myHarris_Ave05.jpg", image);
}
//使用高斯平滑
void myHarrisCorner_Gauss(Mat &srcImg) {
    Mat image, src_grayImg, Ix, Iy, I_xx, I_yy, I_xy, R, filter_R, result;
    cvtColor(srcImg, src_grayImg, COLOR_BGR2GRAY);
    image = srcImg.clone();
    int wsize = 3;//窗口大小
    sobelGradient(src_grayImg, Ix, 1);
    sobelGradient(src_grayImg, Iy, 2);
    I_xx = computeImage(Ix, Iy, wsize, 1);
    GaussianBlur(I_xx, I_xx, Size(3, 3), 0, 0);
    I_yy = computeImage(Ix, Iy, wsize, 2);
    GaussianBlur(I_yy, I_yy, Size(3, 3), 0, 0);
    I_xy = computeImage(Ix, Iy, wsize, 4);
    GaussianBlur(I_xy, I_xy, Size(3, 3), 0, 0);
    //计算响应值
    R = computeImage(I_xx, I_yy,I_xy, wsize);
     //局部非极大值抑制
    filter_R = filterR(R, 10);
    //显示结果
    mixP(filter_R, image, 2);
    imshow("Gauss", image);
    imwrite("myHarris_gauss05.jpg", image);
}
```

三、四种方法的结果对比分析

对于 opencv 自带的 Harris 角点检测、改进的 Shi-Tomasi 角点检测、FAST 角点检测算法、自己实现的 Harri 的使用高斯平滑以及不使用的五种输入同样的测试图片结果如下，

每一种都调好参数后便固定不变测试对于不同大小角度的图片：（图片顺序为：①Fast、②Harris(opencv)、③myHarris(noGauss)、④myHarris(Gauss)、⑤ST）

1、第一组分辨率：692*910

2、第二组分辨率：240*319

3、第三组：340*256（翻转）

4、第四组：480*316

通过对上诉几组图片结果的观察发现，其中 Shi-Tomasi 的效果应该是最好的，角点标记的很全面，而且没有像 Harris 那样阈值很难调节，对不同图片变化很大（使用高斯的阈值大概为 10^12 很大！），而 Fast 得有点就在于简单速度快，但是精准不够，opencv 库内的 Harris 角点检测个人感觉很难用，阈值很难调，而且可能是为了不检测出错，检测出来的角点数量比较少，效果不是很好。