计算机视觉和模式识别 作业2

*13331231*

*孙圣*

*计应2班*

一、使用说明

MAC OSX系统：通过sh execute.sh直接编译运行即可（需要安装opencv）,默认对第1张图片进行处理。要对其他图片进行测试，执行./a.out ./Dataset/X.jpg即可。

Windows系统，在cmd中输入execute.exe ./Dataset/X.jpg。

二、实验过程

首先通过imread()读入图像，利用data成员变量来判断是否是图像文件。

由于图片较大，很难在屏幕中完全显示，因此调用resize()函数将宽和高都缩小为原来的五分之一，便于观察结果。

之后调用cvtColor()将图像进行二值化处理，转换为灰度图像。因为A4纸本身是白色的，而纸上的文字和其他周围的物体大多都接近黑色，因此二值化后可以更加明显的将A4纸区别出来。

然后调用blur()函数，将图像模糊化，这样可以进一步减少噪声的干扰。

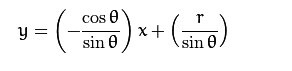
设定Canny的参数，包括低阈值和高阈值。对于这两个参数比较有效的调整应该在整个程序完成之后，根据最终识别出来的直线的数量来变更。

调用完Canny()函数之后，得到的只是边缘上一个一个离散的点，因此要利用投票的机制来确定A4纸边缘的直线。因此要设定Hough变换的参数，主要调节的是每次旋转的角度和判断为直线的阈值。之后调用HoughLines()函数，产生的直线信息（极坐标表示：theta,rho）保存在vector中。

由于检测出来直线是无序的，为了之后计算出四个边缘点方便，要根据theta值对直线进行排序，即对边在vector中是连续出现的。

之后对vector中的直线进行遍历。根据观察，有些情况下A4纸的一条边上会检测出两条直线，这样就需要完成去重的工作：对于除了第一条直线之外的每一条直线，都和前一条直线的参数进行比较。如果rho和theta的差都在一定阈值范围内的话，即两条直线相隔很近，则忽略掉该条直线。其实，这个处理方法有一定的误差，更好的办法是对两条接近的直线取平均，作为最终的边缘直线。

完成了去重之后，对于每条直线，要从极坐标转换为直角坐标的形式。公式如图：



求出了斜率和截距之后，可以在直线上生成两个点，根据这两个点绘制直线。

但是，当theta为0度，即直线平行于y轴时，上述公式变成了

*y = -inf x + inf*

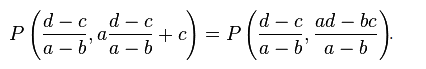
这显然不正确。因此要先判断sin(theta)是否为0，如果为0，则输出

x = rho这条直线。

最后，对最终的四条直线再进行一次遍历，联立



求出交点：



对于平行于y轴的直线，直接将x值代入到另一个直线方程即可。

三、Hough变换原理

对于课上所讲述的基本的霍夫变换，即使用直角坐标中的(m, b)，这存在有两个问题：1.m和b的范围无法确定，导致存储空间的浪费；2.斜率m可能为无穷大，此时无法在累加器中表示。

因此，更加常用的方法是利用极坐标(theta,rho)来表示。

对于任意一个点(x0,y0)，我们都可以找到一个直线系来表示。



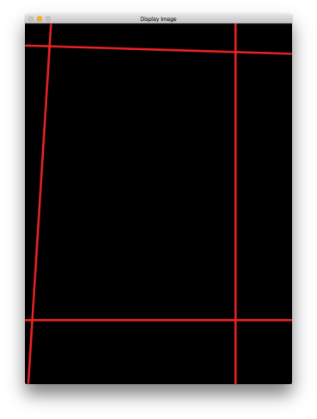
通过不断的旋转直线，即改变theta的值，我们可以得到一个关于theta和r的正弦曲线。

对于每一个点都进行以上的操作，即可得到许多曲线。曲线的交点越多(Voting)，则证明具有该系数的直线上含有更多的点。

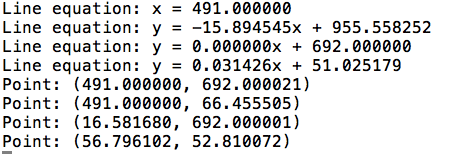
最后，只需要设计一个阈值，便能够排除掉噪声的影响，得到我们所需要的直线。

四、实验结果

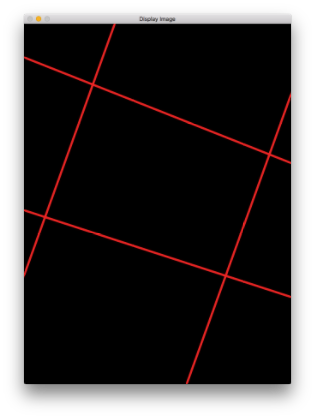
1.jpg



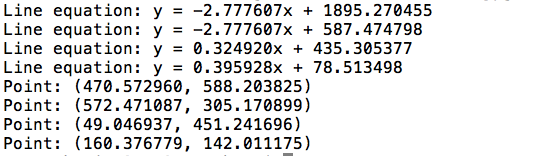
直线方程和交点：



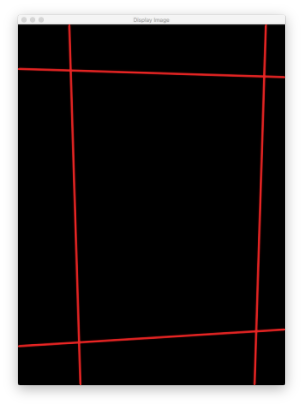
2.jpg



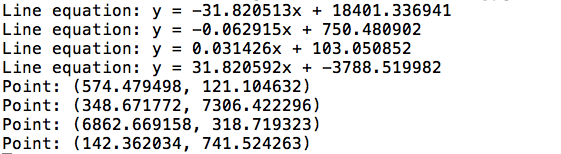
直线方程和交点：



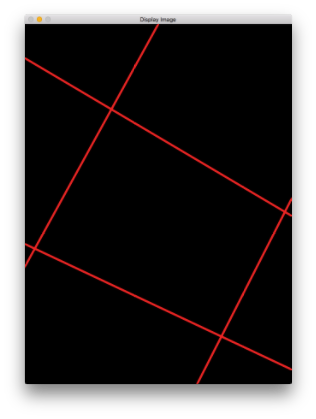
3.jpg



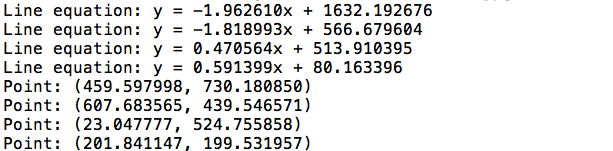
直线方程和交点：



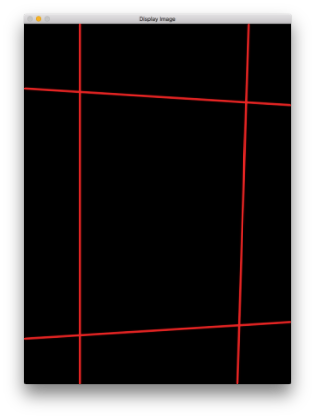
4.jpg



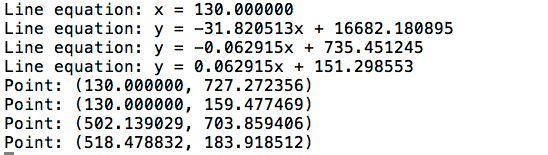
直线方程和交点：



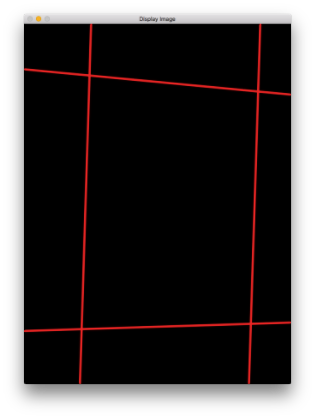
5.jpg



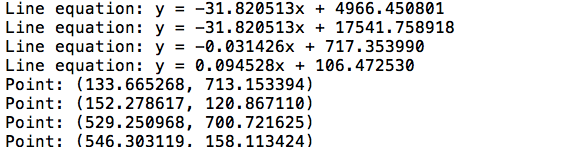
直线方程和交点：



6.jpg



直线方程和交点：



参考资料：

[1] Canny边缘检测

<http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html>

[2] Hough变换检测直线

<http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html>

[3] 求相交直线的交点

<https://en.wikipedia.org/wiki/Line–line_intersection>