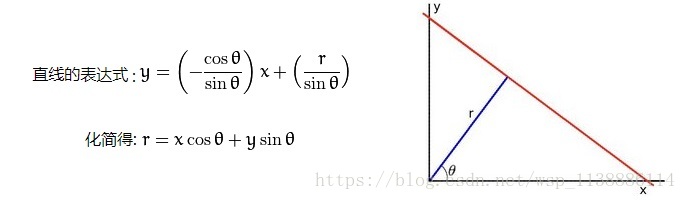
# 1.霍夫线检测

## 目标：

* 理解霍夫变换的概念
* 如何使用它来检测图像中的线条
* 函数：cv.HoughLines()，cv.HoughLinesP()

**一、直线表示**

一条直线在图像二维空间可由两个变量表示

* 在 笛卡尔坐标系: 可由参数: (m , b m,b*m*,*b*) 斜率和截距表示.
* 在 极坐标系: 可由参数: (r , θ r,\theta*r*,*θ*) 极径和极角表示
* 对于霍夫变换, 我们将用 极坐标系 来表示直线. 因此, 直线的表达式可为下图：  
  
* 一般来说对于点 (x 0 , y 0 x\_{0}, y\_{0}*x*0​,*y*0​), 我们可以将通过这个点的一族直线统一定义为:  
  r θ = x 0 ⋅ c o s θ + y 0 ⋅ c o s θ r\_{\theta} = x\_0 ·cos\theta + y\_0 ·cos\theta*rθ*​=*x*0​⋅*cosθ*+*y*0​⋅*cosθ*  
  这就意味着每一对 (r θ , θ r\_{\theta},\theta*rθ*​,*θ*) 代表一条通过点 (x 0 , y 0 x\_{0}, y\_{0}*x*0​,*y*0​) 的直线.

**二、霍夫直线检测**

Hough变换是经典的检测直线的算法。  
用来检测图像中的直线，也可以用来检测图像中简单的结构。

OpenCV的中用函数 HoughLines(标准) 和 HoughLinesP(基于统计) 来检测图像中的直线.

* 基本的版本是cv2.HoughLines。其输入一幅含有点集的二值图（由非0像素表示），其中一些点互相联系组成直线。通常这是通过如Canny算子获得的一幅边缘图像。
* cv2.HoughLines()输出的是[float, float]形式的ndarray，  
  检测到的线(ρ , θ)中浮点点值的参数。
* 下面的例子首先使用Canny算子获得图像边缘，然后使用Hough变换检测直线。其中HoughLines函数的参数3和4对应直线搜索的步长。  
  在本例中：函数将通过步长为1的半径和步长为π/180的角来搜索所有可能的直线。最后一个参数是经过某一点曲线的数量的阈值，超过这个阈值，就表示这个交点所代表的参数对(ρ , θ)在原图像中为一条直线。
* """
* cv2.HoughLines()
* dst: 输出图像. 它应该是个灰度图 (但事实上是个二值化图)
* lines: 储存着检测到的直线的参数对 (r,\theta) 的容器
* rho : 参数极径 r 以像素值为单位的分辨率. 我们使用 1 像素.
* theta: 参数极角 \theta 以弧度为单位的分辨率. 我们使用 1度 (即CV\_PI/180)
* threshold: 设置阈值： 一条直线所需最少的的曲线交点
* srn and stn: 参数默认为0
* cv2.HoughLinesP(dst, lines, 1, CV\_PI/180, 50, 50, 10 )
* dst: 输出图像. 它应该是个灰度图 (但事实上是个二值化图)
* lines: 储存着检测到的直线的参数对 (x\_{start}, y\_{start}, x\_{end}, y\_{end}) 的容器
* rho : 参数极径 r 以像素值为单位的分辨率. 我们使用 1 像素.
* theta: 参数极角 \theta 以弧度为单位的分辨率. 我们使用 1度 (即CV\_PI/180)
* threshold: 设置阈值： 一条直线所需最少的的曲线交点。超过设定阈值才被检测出线段，值越大，基本上意味着检出的线段越长，检出的线段个数越少。
* minLinLength: 能组成一条直线的最少点的数量. 点数量不足的直线将被抛弃.
* maxLineGap: 能被认为在一条直线上的两点的最大距离。
* """

## 直线检测实例1

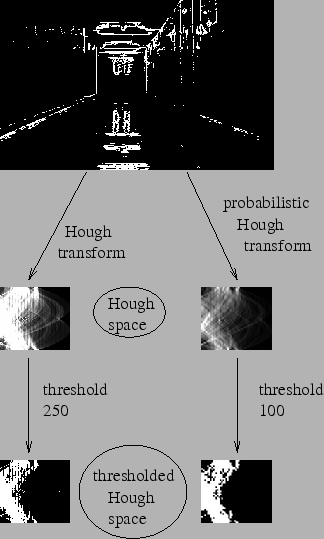
|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 概率Hough变换

在霍夫变换中，你可以看到即使对于具有两个参数的行，也需要大量计算。概率Hough变换是我们看到的Hough变换的优化。它没有考虑所有要点。相反，它只需要一个足以进行线检测的随机点子集。我们必须降低门槛。 请参见下图，其中比较霍夫空间中的霍夫变换和概率霍夫变换。（图片提供：Franck Bettinger的主页）

[](https://camo.githubusercontent.com/16f5211baefd8106825fd83e1adfc3dade59f25ce83f2202cd245ff7c46f996e/68747470733a2f2f646f63732e6f70656e63762e6f72672f342e302e302f686f7567686c696e6573342e706e67)

OpenCV实现基于使用Matas，J。和Galambos，C。和Kittler，J.V。[122]的渐进概率Hough变换的线的鲁棒检测。 使用的函数是cv.HoughLinesP()。 它有两个新的论点。

* minLineLength - 最小线长。 短于此的线段将被拒绝。
* maxLineGap - 线段之间允许的最大间隙，将它们视为一条线。

最好的是，它直接返回行的两个端点。在前面的例子中，你只得到了行的参数，你必须找到所有的点。在这里，一切都是直接而简单的。

**import cv2 as cv**

**import numpy as np**

**img = cv.imread('../data/sudoku.png')**

**gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)**

**edges = cv.Canny(gray,50,150,apertureSize = 3)**

**lines = cv.HoughLinesP(edges,1,np.pi/180,100,minLineLength=100,maxLineGap=10)**

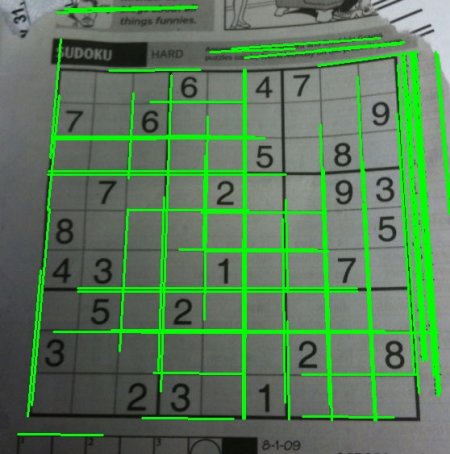
**for line in lines:**

**x1,y1,x2,y2 = line[0]**

**cv.line(img,(x1,y1),(x2,y2),(0,255,0),2)**

**cv.imwrite('houghlines5.jpg',img)**

窗口将如下图显示：



# 可以修改一下参数，使他效果更好

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

# 2.霍夫圆检测

## 目标：

* 学习使用Hough变换来查找图像中的圆圈
* 函数：cv.HoughCircles()
* **cv2.HoughCircles(image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]]) → circles**
* 参数说明:
* image- 8位，单通道， **灰度**输入图像。  
  circles- 找到的圆的输出向量。每个向量被编码为3元素的浮点向量 （x，y，半径）。  
  circle\_storage - 在C函数中，这是一个将包含找到的圆的输出序列的内存存储。  
  method- 使用检测方法。目前，唯一实现的方法是 CV\_HOUGH\_GRADIENT，基本上是 21HT，在[Yuen90]中有描述 。  
  dp - 累加器分辨率与图像分辨率的反比。例如，如果 dp = 1，则累加器具有与输入图像相同的分辨率。如果 dp = 2，则累加器的宽度和高度都是一半。  
  minDist -检测到的 **圆的中心之间的最小距离**。如果参数太小，除了真正的参数外，可能会错误地检测到多个邻居圈。如果太大，可能会错过一些圈子。  
  param1 - 第一个方法特定的参数。在CV\_HOUGH\_GRADIENT的情况下， 两个传递给Canny（）边缘检测器的阈值较高（较小的两个小于两倍）。  
  param2 - 第二种方法参数。在CV\_HOUGH\_GRADIENT的情况下  
  ，它是检测阶段的圆心的累加器阈值。越小，可能会检测到越多的虚假圈子。首先返回对应于较大累加器值的圈子。  
  minRadius -最小圆半径。  
  maxRadius - 最大圆半径。

## 实例

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |