[OpenCV]經典霍夫變換原理

新機器視覺 昨天

以下文章來源於古月居 ， 作者Eragonl

本文主要講述的是霍夫變換的一些內容，並加入一些在生活中的應用，希望能對讀者對於霍夫變換的內容有所了解。

首先我先說的是，霍夫變換是一個特徵提取技術。其可用於隔離圖像中特定形狀的特徵的技術，應用在圖像分析、計算機視覺和數字圖像處理領域。目的是通過投票程序在特定類型的形狀內找到對象的不完美實例。這個投票程序是在一個參數空間中進行的，在這個參數空間中，候選對像被當作所謂的累加器空間中的局部最大值來獲得，所述累加器空間由用於計算霍夫變換的算法明確地構建。此處我們主要介紹的是比較基本的霍夫變換在直線中的應用，例如在圖像中檢測直線（線段），Hough變換的主要優點是對於噪聲有良好的魯棒性。

基礎原理介紹

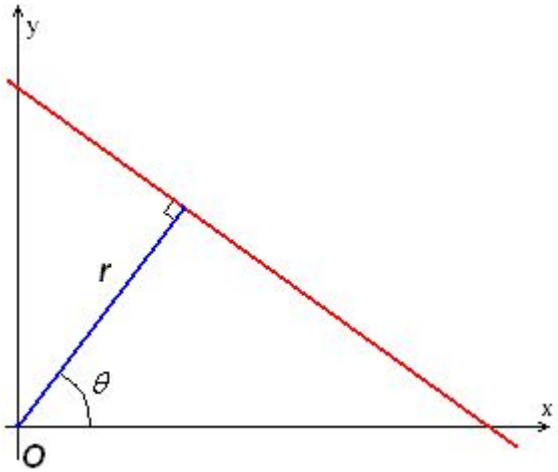
正如我們上面所介紹的那樣，霍夫變換最簡單的是檢測直線。我們知道，直線的方程表示可以由斜率k和截距b表示（這種表示方法，稱為斜截式，也就是高中的時候學習到的一種常用形式），如下所示：

$$y=kx+b$$

如果我们用参数空间表示则为(b,k),即,我们可以用斜率和截距就能表示一条直线。但这种形式会产生一个问题，那就是当我们的直线斜率k为无限大的时候（即垂线），这会使得该直线无法使用斜截式来进行表示，此处我们需要使用到另一种直线的表现形式：黑塞法线式（或者简称为法线式）：

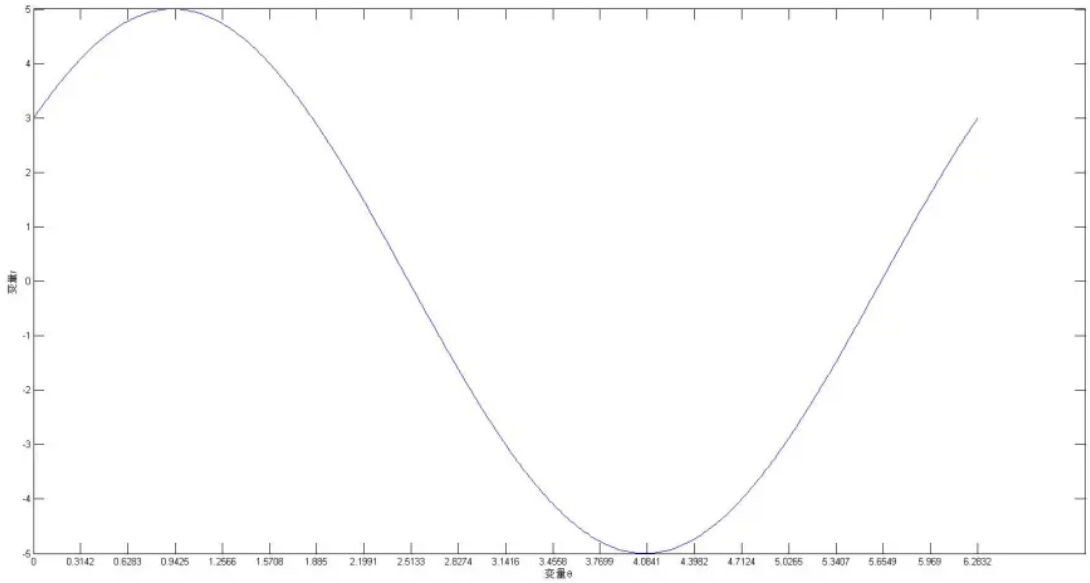
$$r=x\cos\theta+y\sin\theta$$

其中r是原点到直线上最近点的距离(其他人可能把这记录为ρ，下面也可以把r看成参数ρ，两者含义相同)，θ是x轴与连接原点和最近点直线之间的夹角。如下图图1所示。



从而我们可以将图像的每一条直线与一对参数(r,θ)相关联。由参数(r,θ)构成的平面有时被称为霍夫空间，用于表示二维直线所构成的集合。

我们经过Hough变换后，我们圆来笛卡尔坐标系中的一个点可以映射到Hough空间中去，如下图图2所示。



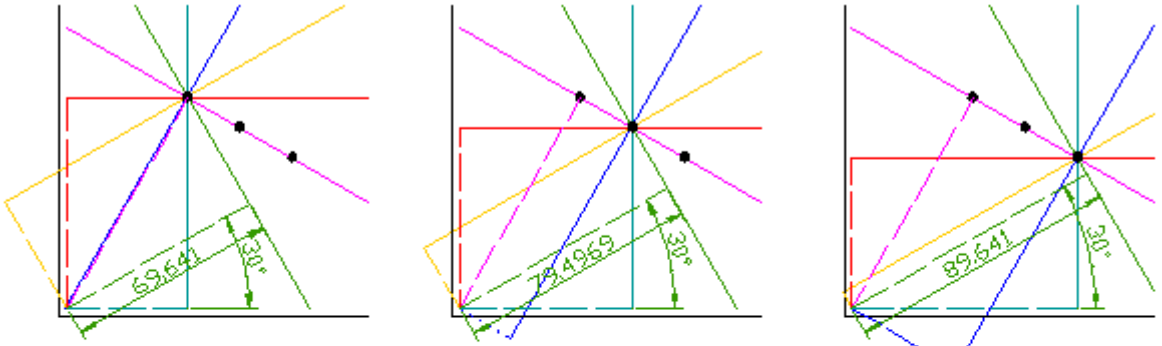
（注：图2表示的是原坐标系中，过该点的所有可能直线的集合在Hough空间中的表示。）

图2显示了经过原笛卡尔坐标系中的定点（3,4），通过该点的所有可能直线的(r,θ)的关系。显示了在极坐标对极径极角平面绘出所有通过该定点的直线，将得到一条正弦曲线。正弦曲线的形状取决于，点到所定义原点的距离r。通常，r越大，正弦曲线的振幅越大，反之则会越小。

所以我们可以得到一个结论，给定平面中的单个点，那么通过该点的所有直线的集合对应于(r,θ)平面中的正弦曲线，这对于该点是独特的。一组两个或更多点形成一条直线将产生在该线的(r,θ)处交叉的正弦曲线。因此，检测共线点的问题可以转化为找曲线相交点的问题。

例：

考虑下面三个点，这里显示为黑点。（下图为图3）



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

（**注：**此处也展示了霍夫变换的几个基本步骤：首先，对每个点均绘制不同角度的线条，这些线全部经过各自的对应点并显示为实线。其次，对于每条实线，找到经过原点的对应垂线并显示为虚线。然后找到虚线的长度和角度。这些值显示在图表下方的表格中。这对被转换的三个点中的每一个都重复该过程。然后将结果绘制成图，有时称为霍夫空间图，如下图图4所示）

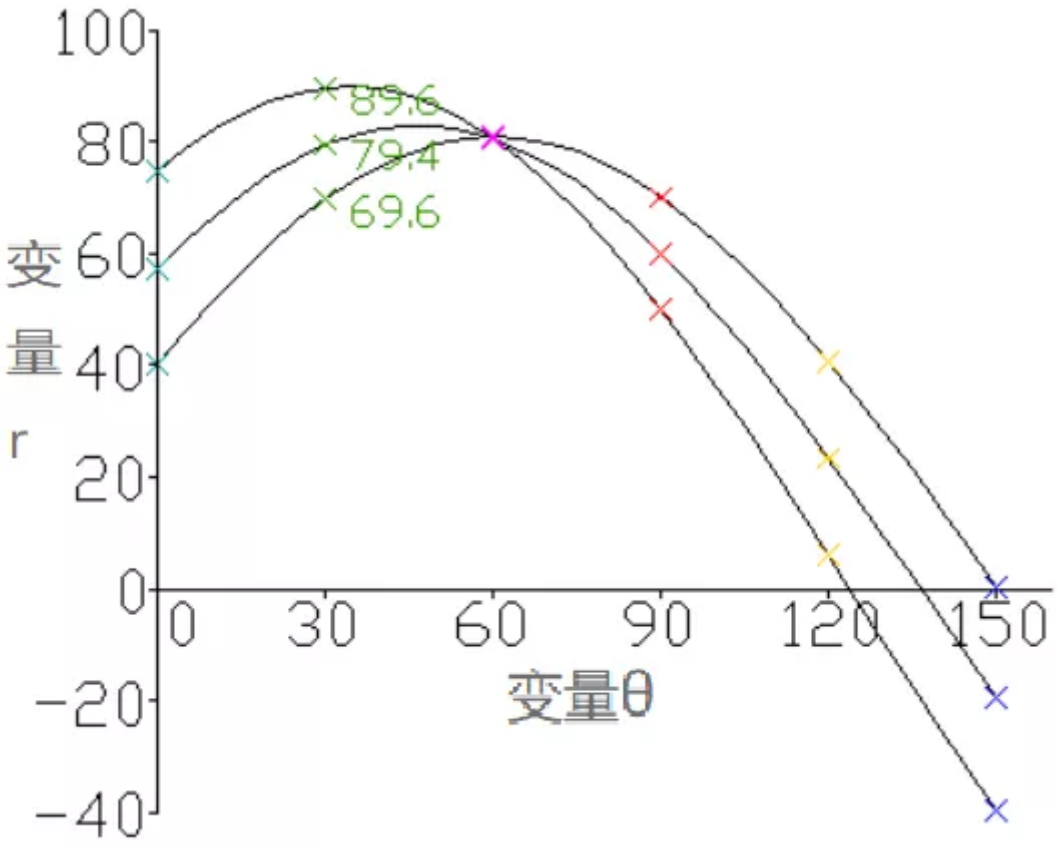


图4中曲线相交的点给出的距离和角度表示各个测试点相交的线。在图4中，所示的线条在粉红点相交; 这对应于图3中的实线粉红线，其穿过三个点。

分析上下文，边缘段的点（一个或多个）的坐标(xi,yi)在图像中是已知的，并且因此作为参数线等式中的常量，而r与θ是未知变量是我们要寻找的。如果我们绘制由(r,θ)每个定义的可能值(xi,yi)，笛卡尔图像空间中的点映射到霍夫参数空间中的曲线（正弦曲线）。这个点到曲线的变换是直线的霍夫变换。当在霍夫参数空间中查看时，在笛卡尔图像空间中共线的点变得很明显，因为它们产生在相同(r,θ)点相交的曲线。

霍夫变换提取直线

我们通过将霍夫参数空间量化为有限间隔或累加器单元来实现变换。随着算法的运行，每个算法都把(xi,yi)转换为一个离散化的 (r,θ)曲线，并且沿着这条曲线的累加器单元被递增。累加器阵列中产生的峰值表示图像中存在相应的直线的相应证明。

此时需要注意的是，现在我们考虑的是直线的霍夫变换。累加器阵列的维度是二维的（也就是r和θ)。

那么对于图像来说，(x,y)处的每个像素及其邻域，霍夫变换算法被用于确定该像素是否有足够的直线证据。如果是，它将计算该线的参数(r,θ)，然后查找参数落入的累加器箱，并增加该箱的值（投票值）。通过查找具有最高值的箱，通常通过查找累加器空间中的局部最大值，可以提取最可能的线，并且读出它们的（近似的）几何定义。

找到这些峰值的最简单方法是通过应用某种形式的阈值，但其他技术可能在不同情况下产生更好的结果。由于返回的行不包含任何长度信息，因此通常有必要在下一步中查找图像的哪些部分与哪些行匹配。此外，由于边缘检测步骤中存在缺陷误差，通常会在累加器空间中出现错误，这可能使得找到合适的峰值以及适当的线条变得非常重要。

线性霍夫变换的最终结果是类似于累加器的二维阵列（矩阵），该矩阵的一个维度是量化角度θ，另一个维度是量化距离r。矩阵的每个元素的值等于位于由量化参数 (r,θ)表示的线上的点或像素的总和。所以具有最高值的元素表示输入图像中代表最多的直线。我们也可以把累计器单元的结果认为是投票值。换句话说，将每个交点看成一次投票，也就是说A(r,θ)=A(r,θ)+1，所有点都如此进行计算后，可以设置一个阈值，投票大于这个阈值的可以认为是找到的直线。

霍夫变换提取圆

而当我们需要去进行圆检测的时候，我们累加器是三维累加器，在圆检测的情况下，我们可以知道的是其对应的参数方程为：

$$(x-a)2+(y-b)2=r2$$

其中a和b是圆心的坐标并且是r半径。在这种情况下，算法的计算复杂度开始增加，因为我们现在在参数空间和三维累加器中有三个坐标。（通常，累加器阵列的计算和大小随着参数数量的增加而多项式增加，因此，基本霍夫技术仅适用于简单曲线。）

它的算法步骤如下：

1.首先创建累加器空间，由每个像素单元格构成。最初每个单元格都设置为0。

2.然后对于每个图像中的边缘点(i,j)，按照圆方程(i−a)2+(j−b)2=r2将那些可能是一个圆中心的单元格值进行累加。这些单元格在等式中由字母a表示。

3.然后在前面的步骤中由每个可能找到的值a，区找到满足等式的所有可能值b。

4.搜索累加器空间中的局部最大值。这些单元格表示算法检测到的圆圈。

如果我们不知道事先定位的圆的半径，可以使用三维累加器空间来搜索具有任意半径的圆。当然，这在计算上更加昂贵。

該方法還可以檢測部分位於累加器空間外部的圓，只要該圓的區域內仍有足夠的圓。

霍夫變換在很多地方都有著應用，如果是在OpenCV（Python）下想要使用霍夫變換，只需要使用函數cv2.HoughLinesP函數，需要注意的是該函數並不是標準的霍夫變換，其為：概率霍夫變換，它只分析點的子集並估計這些點都屬於一條直線的概率，這是標準霍夫變換的優化版本。該函數計算代價少，執行更快，但準確度有一定程度的下降。

cv2.HoughLinesP函數的語法如下：

```
cv2.HoughLinesP(image,rho,theta,threshold,minLineLength,maxLineGap)
```

其參數分別解釋如下：

```
·image：要处理的二值图像；
·rho：线段的几何表示，表示取距离的间隔，一般取1；
·theta：线段的几何表示，表示取角度的间隔，一般取np.pi/180；
·threshold：阈值，低于该阈值的会被忽略；
·minLineLength：最小直线长度，小于该长度会被忽略；
·maxLineGap：最大线段间隙，大于此间隙才被认为是两条直线。
```

霍夫變換在自動駕駛中也有所應用，可以如下面一個簡單例子所示，其實現的是對我們畫面中的道路直線進行的檢測：

```python
import os
import re
import cv2
import numpy as np


# 初始化一个掩膜
def mask_create():
    img = cv2.imread('0.png')
    zero = np.zeros_like(img[:, :, 0])
    poly = np.array([[50, 270], [220, 160], [345, 160], [480, 270]])
    zero_fixed = cv2.fillConvexPoly(zero, poly, (255, 255, 255))
    return zero_fixed
```

```
14
15 # 掩膜计算，传入的图像需要是BGR图
16 def mask_calc(frame, mask):
17     img = cv2.bitwise_and(frame[:, :, 0], frame[:, :, 0], mask=mask)
18     return img
19
20 # 图像阈值操作，传入的图片需要是灰度图
21 def threshold(low, high, img):
22     ret, thresh = cv2.threshold(img, low, high, cv2.THRESH_BINARY)
23     return thresh
24
25 # 对图像进行霍夫变换，输入的图像需要是二值图,距离r为1,旋转角为1度,投票阈值为30,最远距离为200像素
26 # 并在原图上进行绘制图像
27 def hough(thresh, img):
28     lines = cv2.HoughLinesP(thresh, 1, np.pi/180, 30, maxLineGap=200)
29     try:
30         for line in lines:
31             x1, y1, x2, y2 = line[0]
32             img = cv2.line(img, (x1, y1), (x2, y2), (255, 255, 255), 3)
33     except:
34         return img
35     else:
36         return img
37 # 主函数
38 def mainn():
39     # 读取数据
40     col_frames = os.listdir('../frames/')
```

```
    # 排序
    col_frames.sort(key=lambda f: int(re.sub('\D', '', f)))
    # 读取画面每一帧
    for i in col_frames:
        img = cv2.imread(i)
        # 构建一个掩膜
        mask = mask_create()
        # 对原图像进行掩膜计算
        masked_frame = mask_calc(img, mask)
        thresh = threshold(135, 255, masked_frame)
        img = hough(thresh, img)
        cv2.imshow('img', img)
        if cv2.waitKey(40) == ord('q'):
            break
    cv2.destroyAllWindows()


mainn()
```

該代碼對我們在frames這個文件夾下的圖片進行直線檢測（文件夾下的圖片在源於行車記錄儀所拍下的行車記錄的部分採樣），運行結果如下圖所示。

本文到此為止，謝謝大家！

–THE END–

喜歡此內容的人還喜歡

數論重大突破：120年後，希爾伯特的第12個數學難題借助計算機獲得解決

程序員數學之美

殘差神經網絡究竟在幹啥?

計算機視覺life

用算法代替生物大腦，90後博士造出活的微型機器人登上Science子刊

量子位