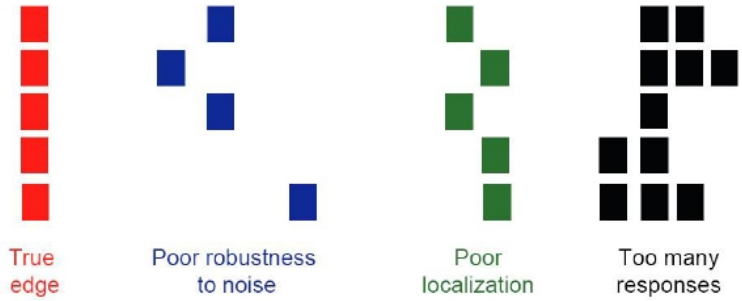
图像边缘检测

边缘的要求

边缘的特点&要求

- **检测准确：** 假阳（false positive)、假阴（false negative)最少
- **定位准确：** 与真实边缘对齐
- **单像素响应：** 边缘宽度为单像素



假阳少：尽量保证检测出来的边缘像素都是真正的边缘像素

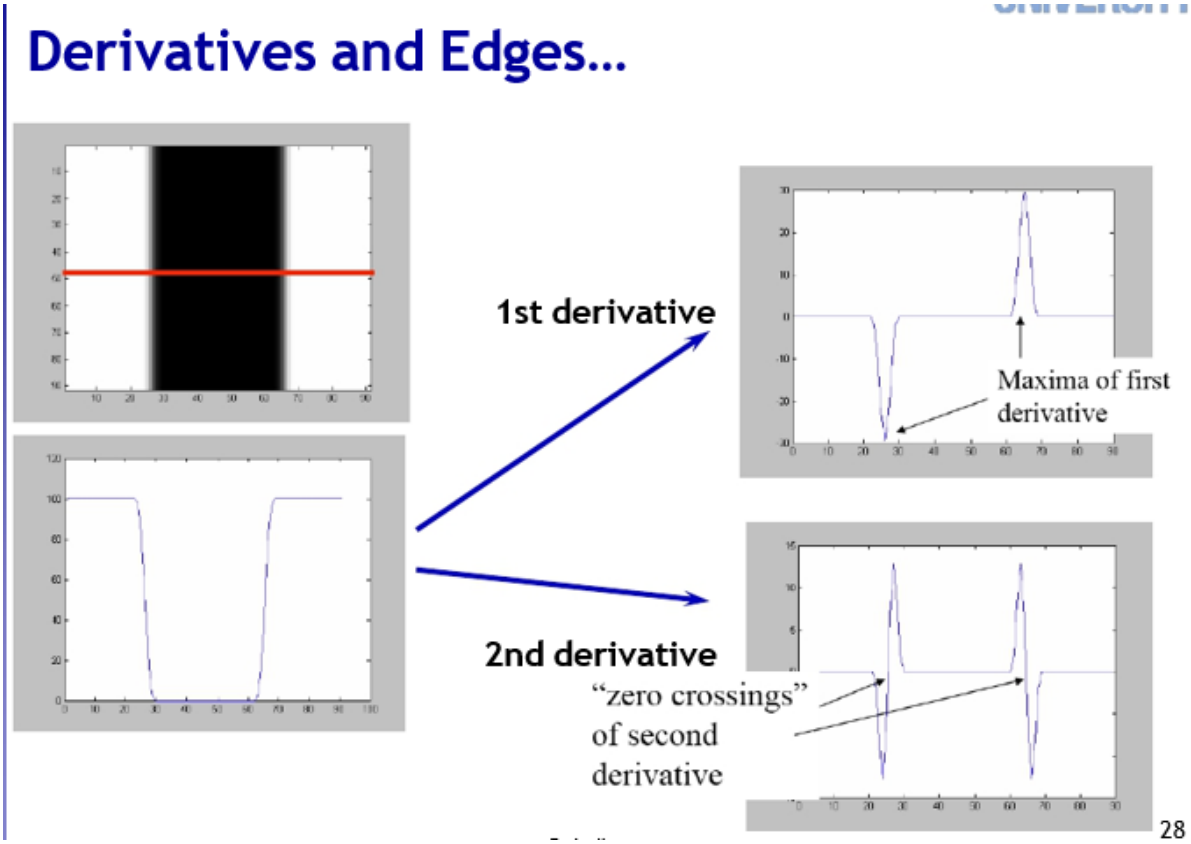
假阴少：尽量保证不是边缘的像素都不会被检测出来

边缘检测的基本步骤

基本步骤

梯度与边缘的关系

- 对于一阶梯度来说，边界出现在一阶导的最大最小处
- 对于二阶梯度来说，边界出现在二阶导为 0 的位置



- 那么我们如何计算梯度呢?
  - 对于一个二维函数，计算公式如下

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

  - 对于离散的数据，我们可以使用有限差分来近似估计
    - 注意到，分母是 1 ，所有该式子就是后一个的值减去前一个的值

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

图像梯度

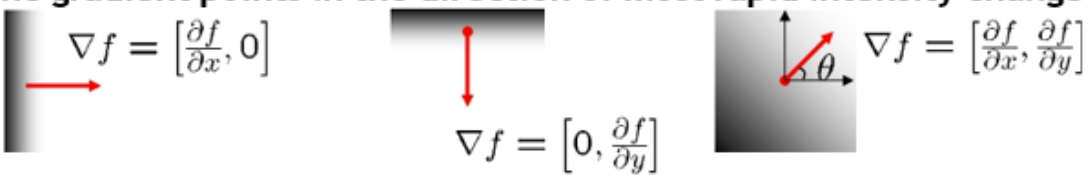
- 表达式

The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- 梯度指向强度变化最快的方向

The gradient points in the direction of most rapid intensity change



- 梯度方向

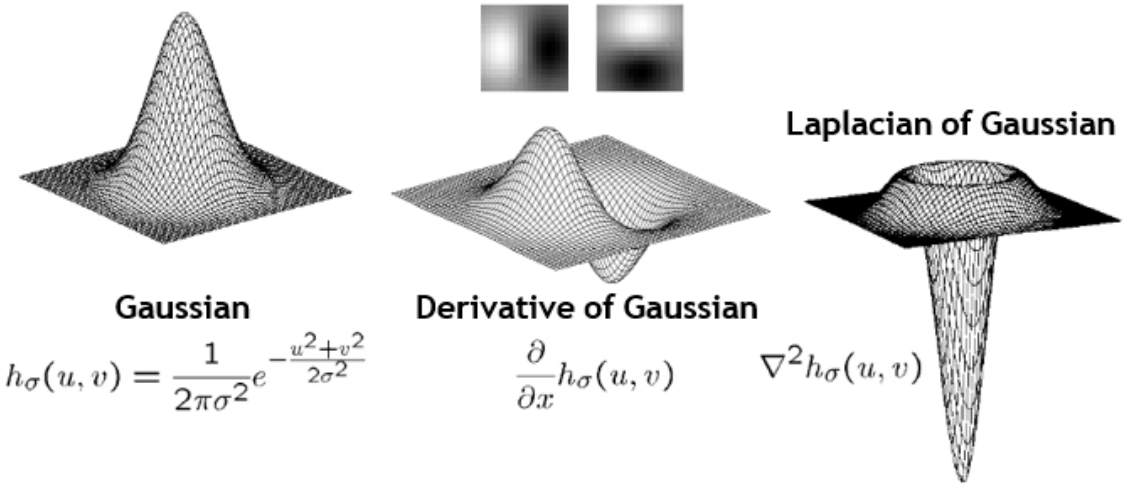
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- 边缘强度是由梯度大小决定的

The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

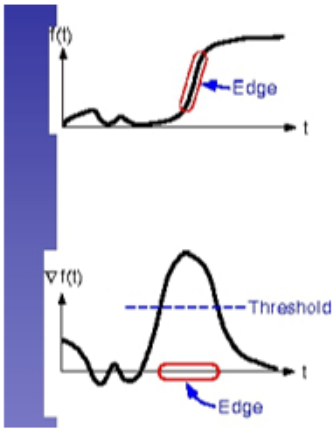
- 二维边缘检测函数



- **∇^2 is the Laplacian operator:**

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- 整体流程
  - 计算边缘响应
  - 利用响应图，设置阈值，来确定图像边缘
    - 使用阈值是为了粗略保留边缘
    - 非极大值抑制是为了使边缘的宽度变得更窄

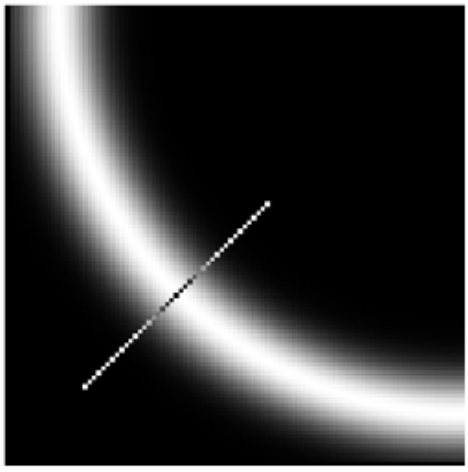


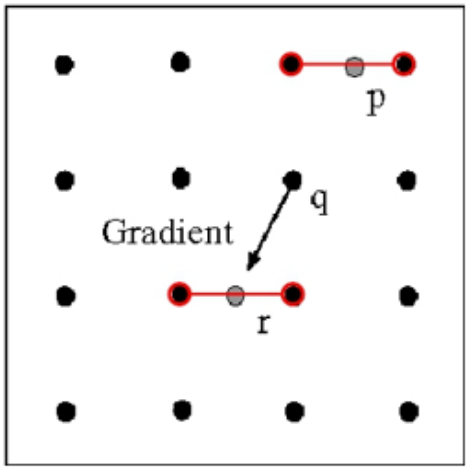
- 滞后阈值法

  保持两个阈值 T_{high} 和 T_{low}

  - 使用 T_{high} 来寻找强边以开始边缘链（大于该值的确定为边）
  - 使用 T_{low} 寻找弱边以继续边缘链（小于该值大确定不为边，介于两值中间的，若与边相连，则确定为边）
  - 阈值的典型比率大致为 $\frac{T_{high}}{T_{low}} = 2$

- 为了保证边缘的单像素响应，我们使用极大值抑制的方法
  - 只保留沿梯度方向上响应值的极大值点





Canny边缘检测器（过程懂了，理论没懂）

- 可能是计算机视觉领域使用最广泛的边缘检测器
- 理论模型：

过程

- 用高斯函数的一阶导过滤图像
- 寻找梯度的大小和方向
- 非极大值抑制

- 使用滞后阈值法并进行连接

基元检测

基元定义：直线，圆等基本几何元素

使用边缘检测来检测线?

困难

- 会产生额外边缘点，选择哪些点？选择哪条线?
- 线的一些部分检测出来的，一些没有检测出来，怎么办?
- 检测的边缘点和方向存在噪音，如何检测真正的参数?

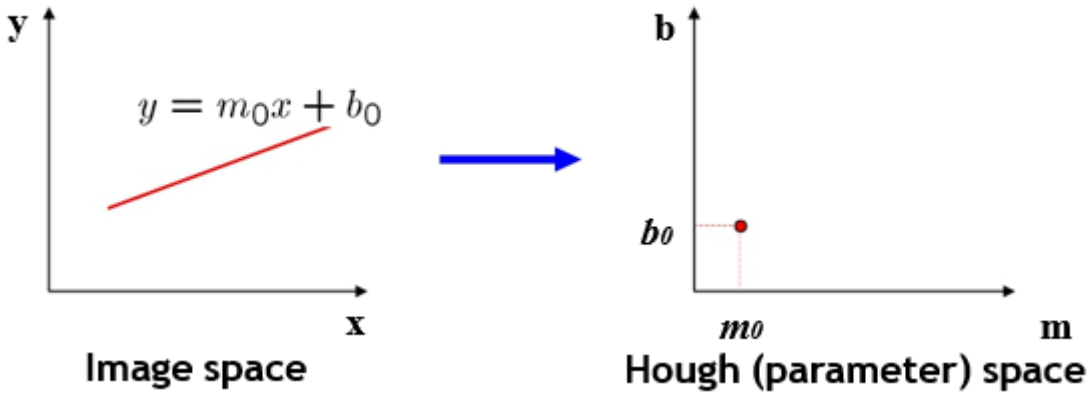
霍夫变换

任务：给定点集，检测包含的直线（可能存在多条直线）

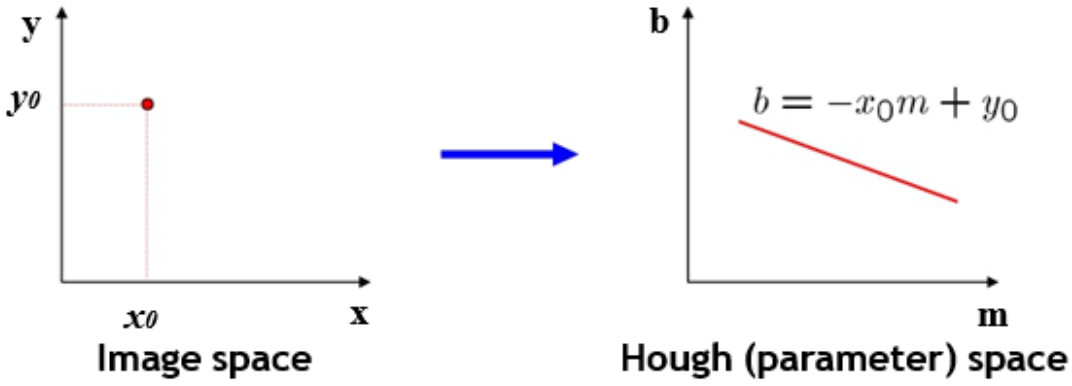
思想：基于投票的思想

- 每个点投票所有可能经过它的直线
- 在直线参数空间取票数较多的点

图像空间中的一条线，可以映射成霍夫空间中的一个点。因为$y = mx + b$

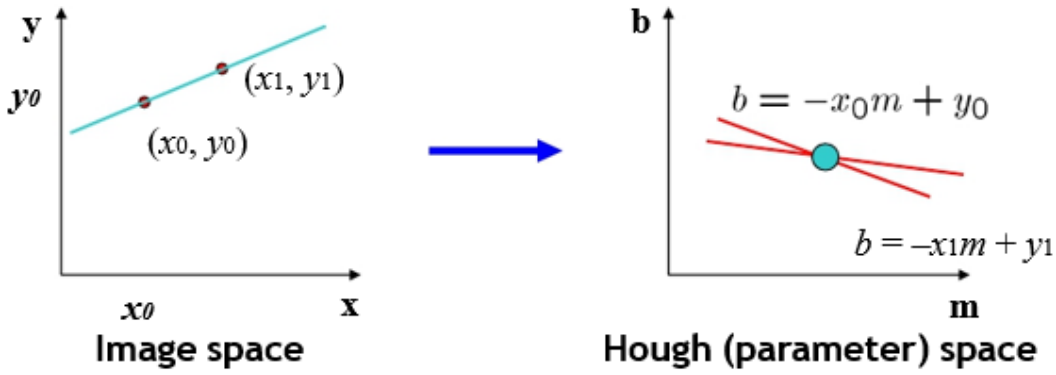


图像空间中的一个点，可以映射为霍夫空间中的一条线。因为$y = mx + b$



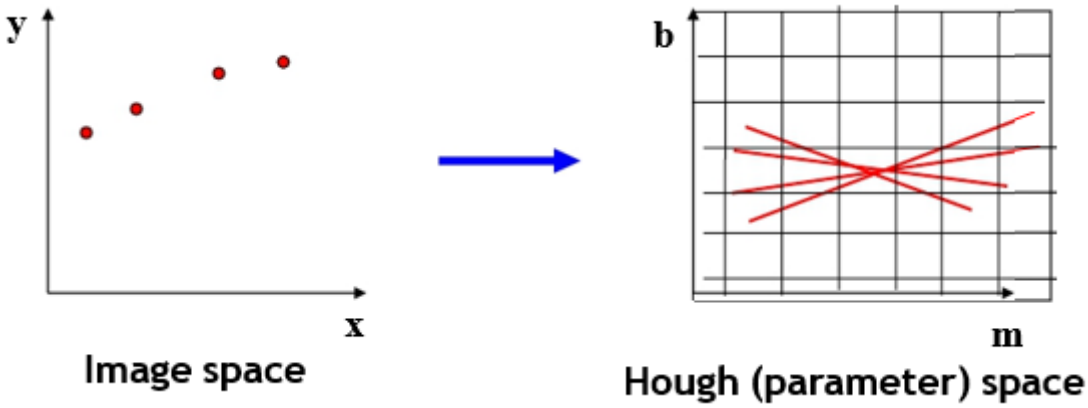
那么图像空间中的一条线的参数，在霍夫空间中怎么寻找呢?

- 答案：是图像空间中在该线上的点，在霍夫空间中对应的线的交点（这就是投票，线上的点会在霍夫空间中为这个线的参数投一票）



对于图像空间中最明显的线，我们如何找到最可能的参数(m, b)呢?

- 让图像空间中的每个边缘点，在霍夫空间中为一组可能的参数投票
- 得票最多的参数就代表了图像空间中的线

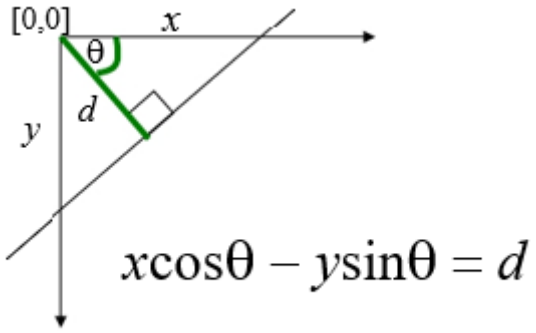


线的极坐标表达

传统的(m, b)参数空间的问题：会产生无限的值，且对垂直的线没有定义（写不成$y = mx + b$）的形式

用(d, θ)表示

- d:线和原点之间的垂直距离
- θ:垂线与 x 轴的夹角



  - 图像空间中的点在霍夫空间中对应正弦波段

极坐标下的霍夫变换算法

θ 从 0 到 180 ， 相当于遍历了可能经过该点的所有直线，每条直线的参数是(d, θ) 。若两点在一条直线上，他们都会为这个(d, θ)投票

Using the polar parameterization:

$$x\cos\theta - y\sin\theta = d$$

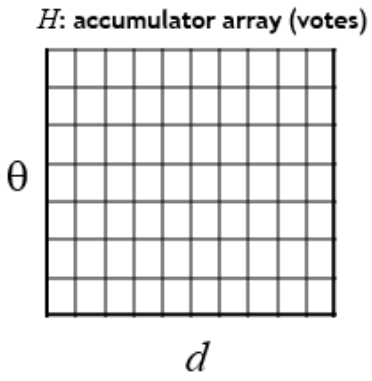
Basic Hough transform algorithm

1. Initialize $H[d,\theta] = 0$.
2. For each edge point (x,y) in the image

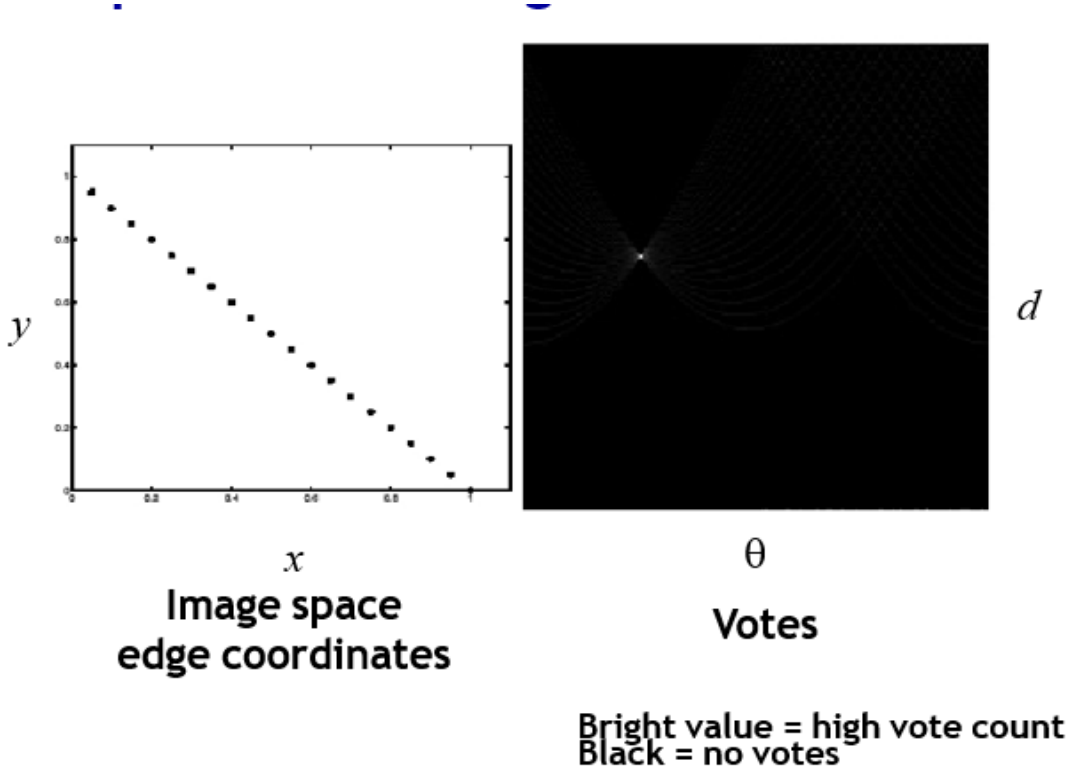
   for $\theta = 0$ to 180 // some quantization

   $d = x\cos\theta - y\sin\theta$

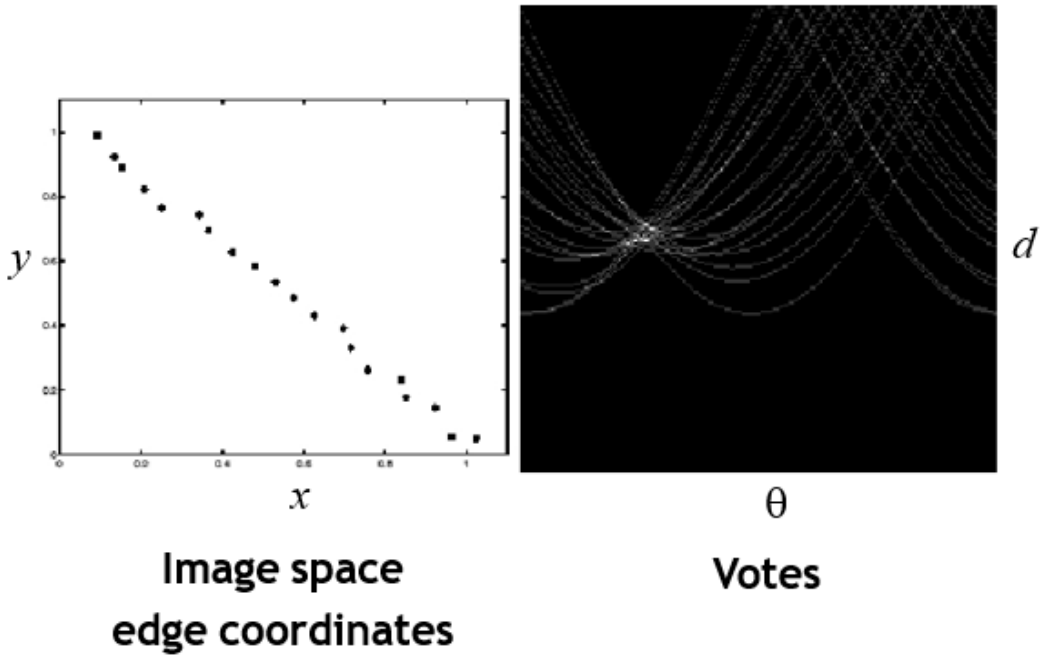
   H[d, θ] += 1
3. Find the value(s) of (d,θ) where $H[d,\theta]$ is maximal.
4. The detected line in the image is given by $d = x\cos\theta - y\sin\theta$



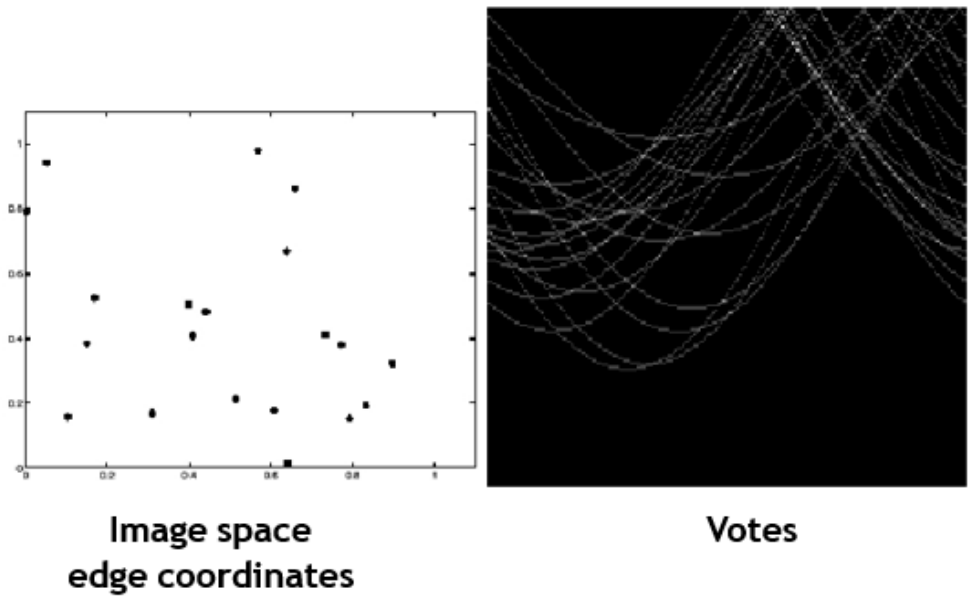
- 结果图



  - 这种情况下效果不好



- 这里数据看起来更像是随机分布的，但是我们仍然找到了峰值



- 改进

Extension 1: Use the image gradient

1. same
2. for each edge point $I[x,y]$ in the image

   compute unique (d,θ) based on image gradient at (x,y)

   $H[d,\theta]$ += 1
3. same
4. same

(Reduces degrees of freedom)

Extension 2

- Give more votes for stronger edges (use magnitude of gradient)

Extension 3

- Change the sampling of (d,θ) to give more/less resolution

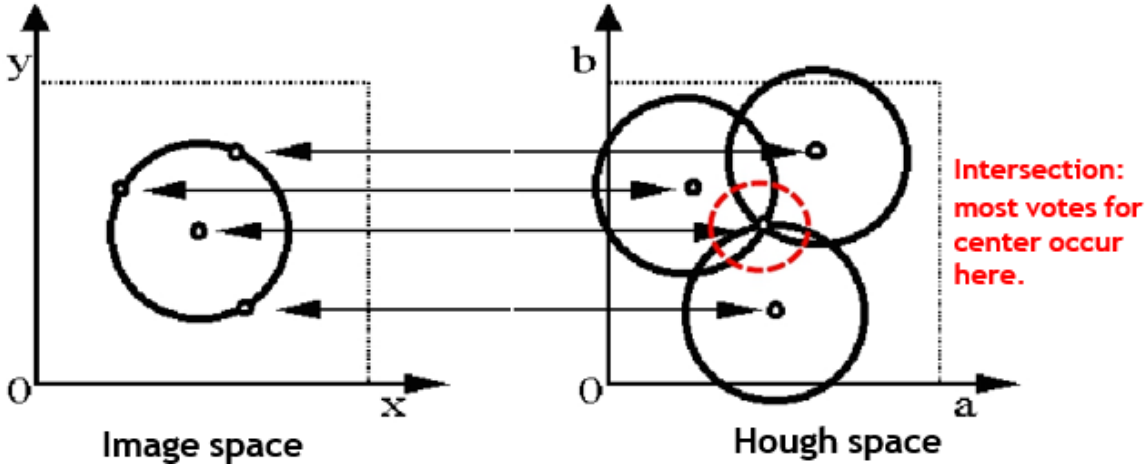
Extension 4

- The same procedure can be used with circles, squares, or any other shape...

推广到圆的检测

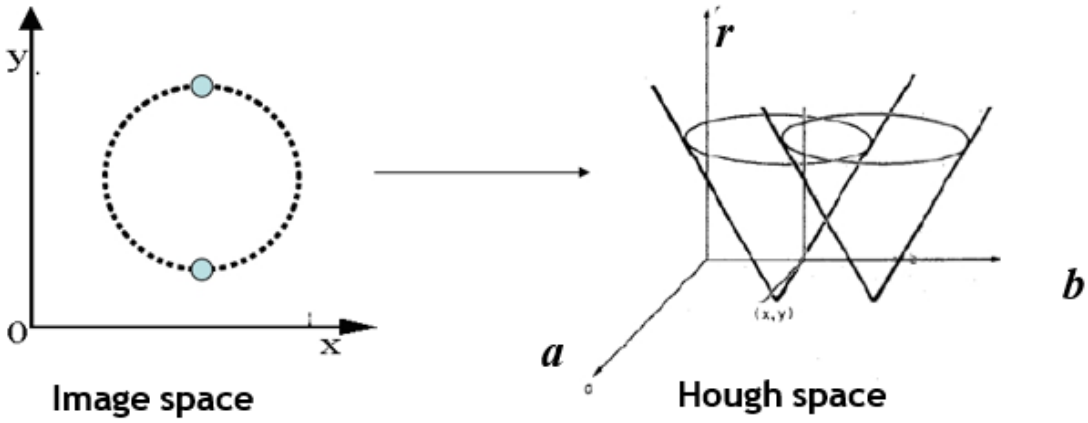
- 知道半径，不知道梯度
  - 图像空间中圆上的一个点(x,y)，在霍夫空间中对应以(x,y)为圆心的一个圆。同一个圆上的点在霍夫空间中形成的圆会有交点

For a fixed radius r, unknown gradient direction



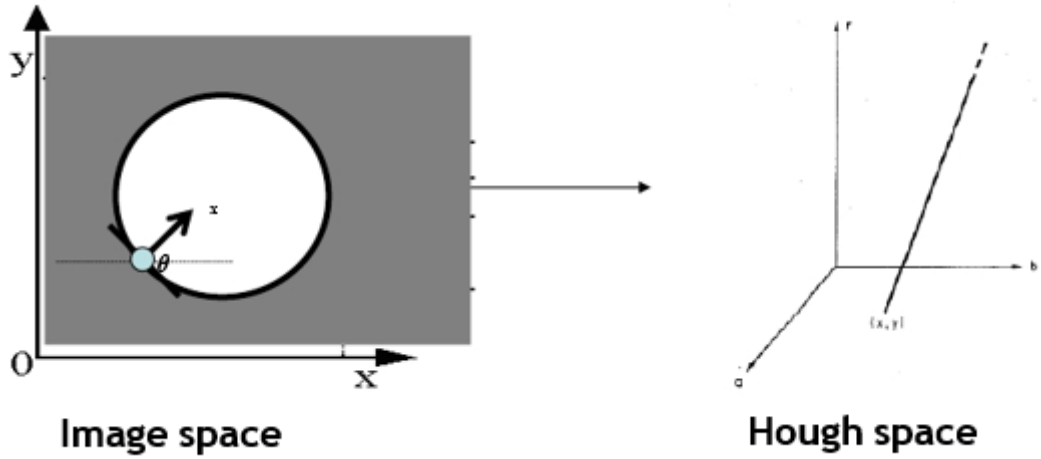
- 不知道半径，不知道梯度

  遍历半径，同一个圆上的点在霍夫空间中形成的圆锥会有交点



- 知道梯度，不知道半径

  圆上的点只有对应梯度的线经过圆心，每个点画一条线，投票最多的就是圆心



- 算法流程：圆心是(a,b)

For every edge pixel (x,y) :

  For each possible radius value r:

    For each possible gradient direction θ:

      // or use estimated gradient

      $a = x - r\cos(\theta)$

      $b = y + r\sin(\theta)$

      $H[a,b,r] \mathrel{+}= 1$

    end

end

- 投票的技巧
  - 首先最小化不相关的标记（取具有显著梯度的边缘点）
  - 选择一个好的网格/离散化
    - 太粗了：当太多的不同线条对应于一个桶时，会得到很大的票数（并不是同一条线上的点头的票，而是因为给它投票的点太多了，它的票数就大）

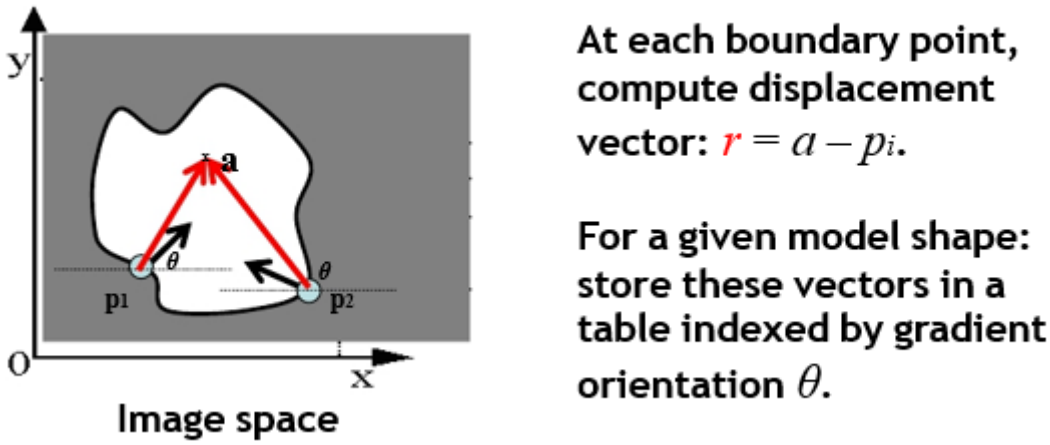
        - 太细：由于一些不完全相邻的点为不同的桶投票，从而错过了一些线。
    - 也为邻居投票（在累加器阵列中平滑）
    - 利用边缘的方向来减少参数的自由度
    - 要读回哪些点为 "获胜 "的山峰投票。 在投票上保留标签。

优缺点

- 优点
    - 所有的点都是独立处理的，所以可以应对遮挡
    - 对噪声有一定的稳健性：噪声点不太可能 对任何单一仓的贡献一致
    - 可以在一次处理中检测到一个模型的多个实例
- 缺点
    - 搜索时间的复杂性随着模型参数的数量呈指数级增长
    - 非目标形状会在参数空间中产生虚假的峰值
    - 很难选择一个好的网格大小

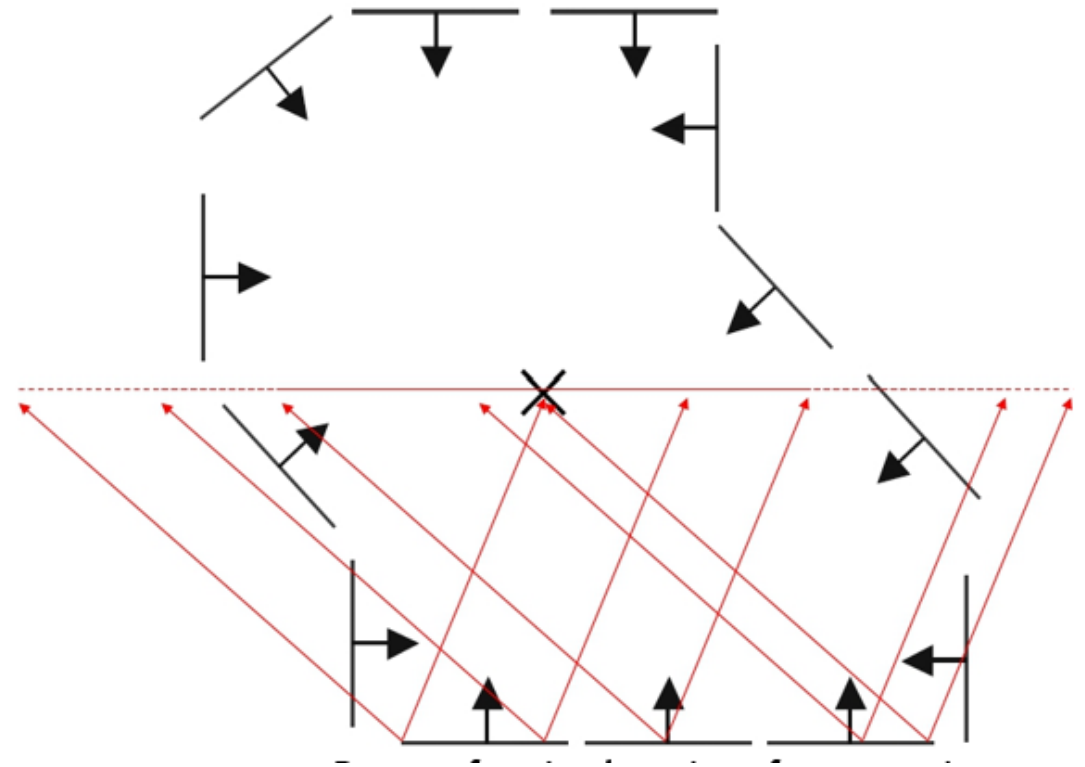
推广到任意形状

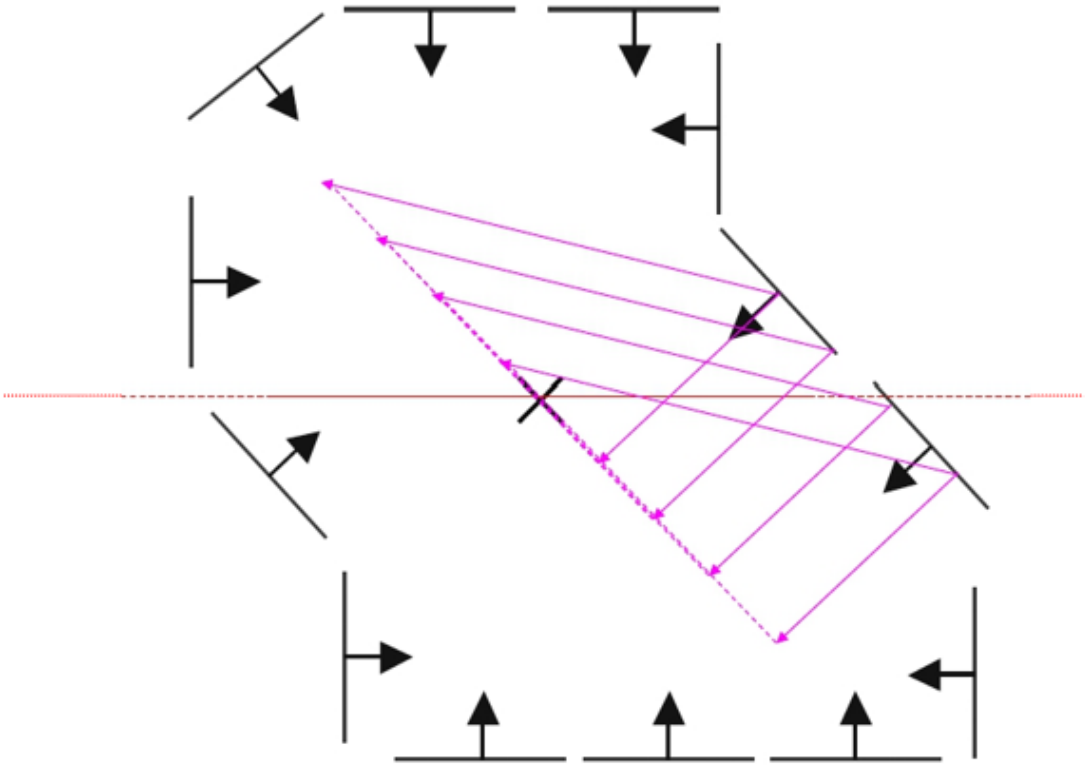
- 任务：通过几个边界点和一个参考点来检测任意形状



- 步骤
    - 对于每个边界点
        - 使用梯度方向作为索引将其存入表中（一个梯度方向可能会对应很多vector）
        - 使用向量 r 来对参考点的位置进行投票

一条边上的每个点都会对参考点的位置进行投票，因此该位置得票多

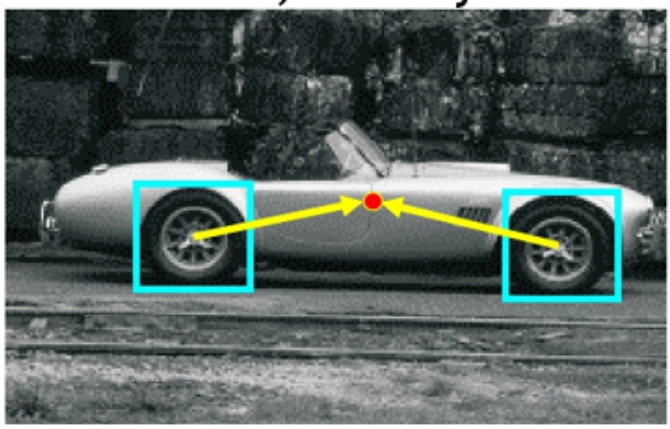




应用

可以通过“视觉代码进行索引”，而不是通过梯度方向

- Instead of indexing displacements by gradient orientation, index by “visual codeword”.



Training image

Visual codeword with displacement vectors