

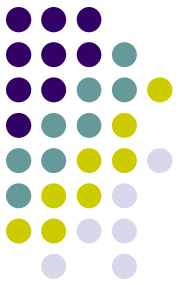
提纲



- 基础知识
- 点、线、边缘检测
 - 背景知识
 - 孤立点的检测
 - 线检测
 - 边缘模型
 - 基本边缘检测
 - 高级边缘检测
- 边缘连接和边界检测

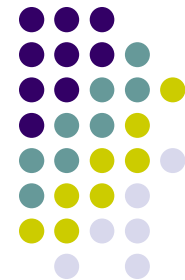


背景



- 边缘检测的结果不完美
 - 噪声
 - 不均匀照明导致的边缘间断
 - 虚假的灰度值不连续
- 边缘连接
 - 将边缘像素组合成有意义的边缘或区域边界
 1. 局部处理
 2. 区域处理
 3. 全局处理（使用霍夫变换）

提纲



- 边缘连接和边界检测
 - 局部处理
 - 区域处理
 - 全局处理

局部处理



1. 分析每个候选点 (x, y) 邻域内像素的特点
2. 将依据某准则相似的点连接起来

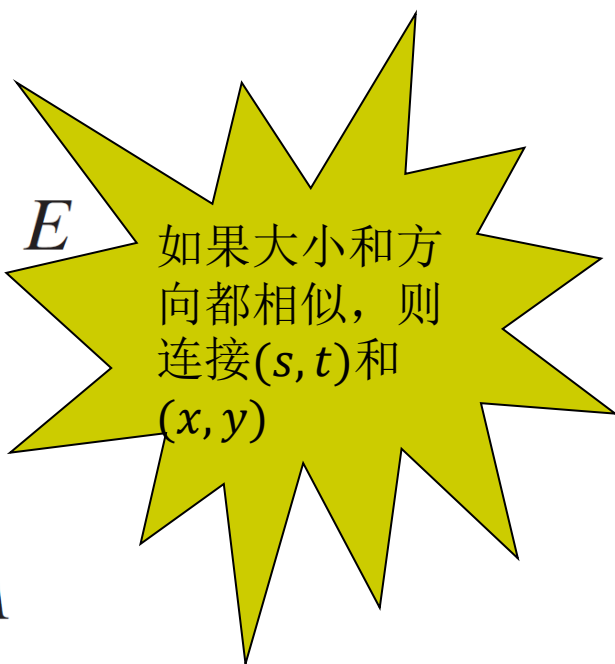
a. 基于梯度大小判断相似

$$|M(s, t) - M(x, y)| \leq E$$

- (s, t) 在 (x, y) 的邻域内

b. 基于梯度方向判断相似

$$|\alpha(s, t) - \alpha(x, y)| \leq A$$



如果大小和方向都相似，则连接 (s, t) 和 (x, y)

局部处理



- 简化算法（计算简单）

1. 计算输入图像 $f(x, y)$ 的梯度大小和方向

- 梯度大小 $M(x, y)$ ，梯度方向 $\alpha(x, y)$

2. 依据下式生产二值图像

$$g(x, y) = \begin{cases} 1 & \text{if } M(x, y) > T_M \text{ AND } \alpha(x, y) = A \pm T_A \\ 0 & \text{otherwise} \end{cases}$$

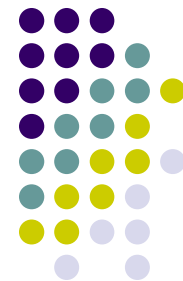
- T_M 为阈值、 A 为特定角度、 T_A 为允许的带宽

3. 逐行扫描，填充长度不超过 K 的空隙

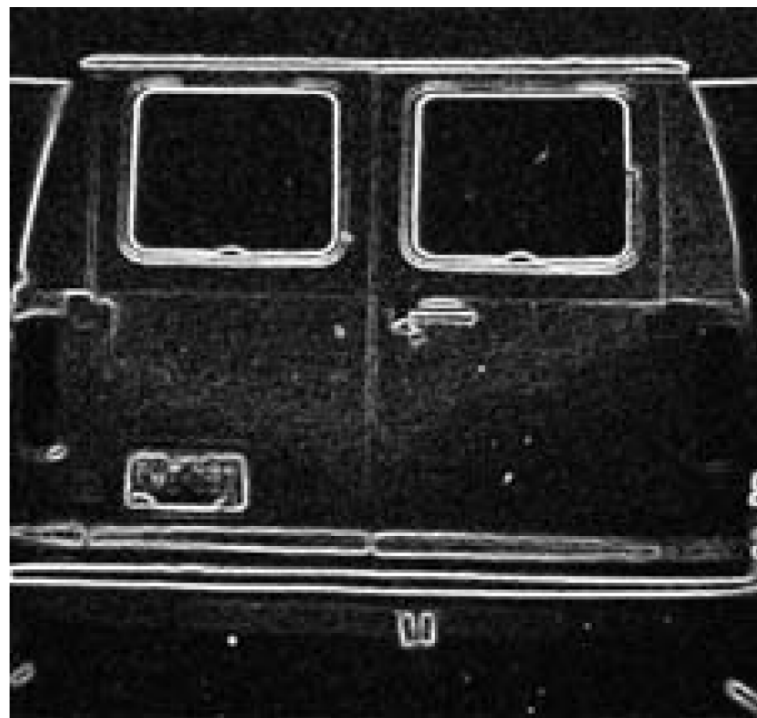
4. 以角度 θ 旋转 $g(x, y)$ ，重复第3步，再反旋转

举例

- 寻找车牌



汽车尾部图像



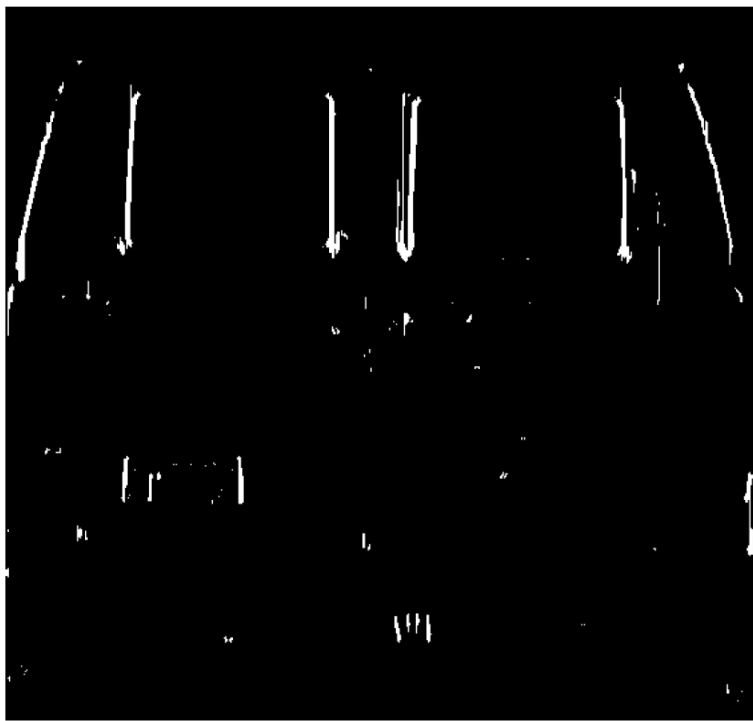
梯度大小图像

举例

- 寻找车牌



水平连接的边缘像素



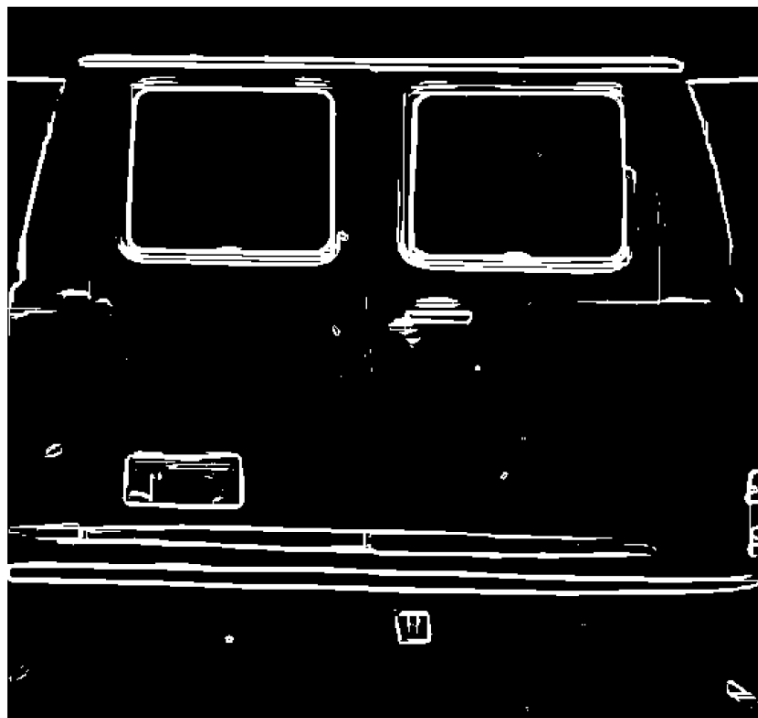
垂直连接的边缘像素

举例

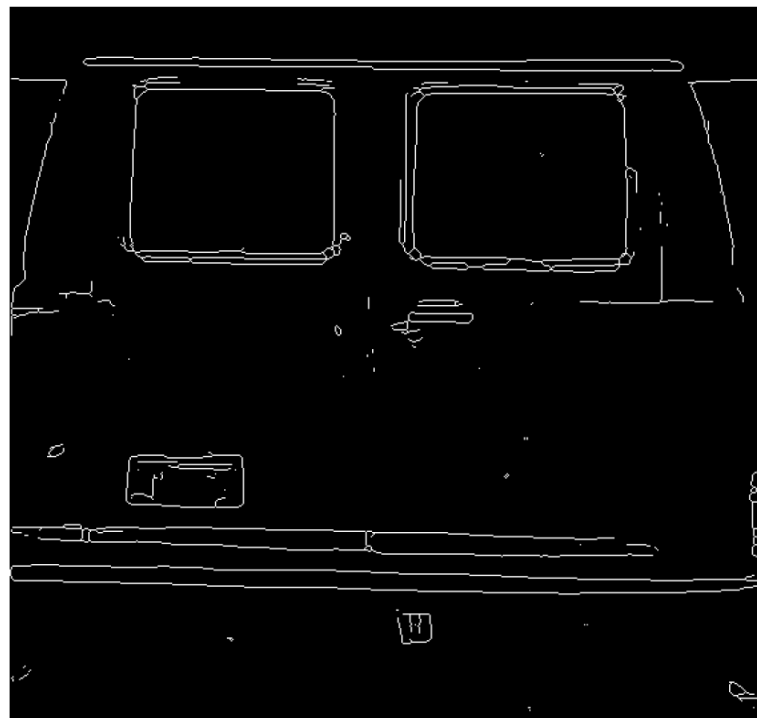
● 寻找车牌

采用比较大的
夹角范围

美国车牌的
长宽比是2:1



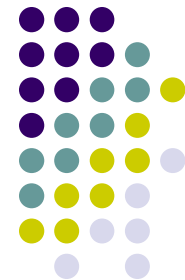
合并后的图像



细化后的图像



提纲



- 边缘连接和边界检测
 - 局部处理
 - 区域处理
 - 全局处理

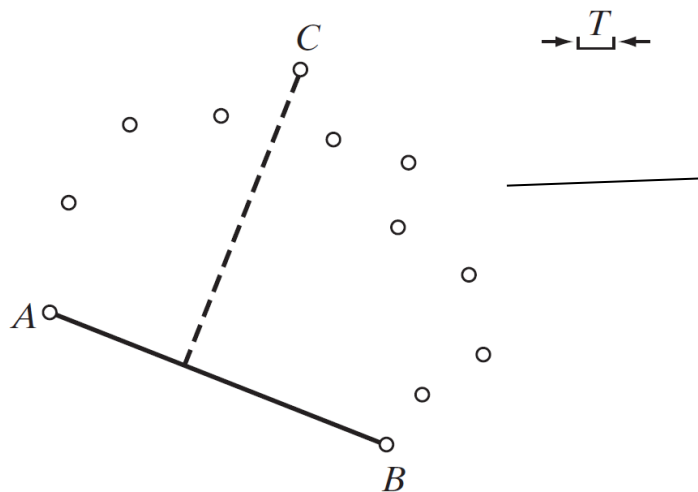
区域处理



- 前提
 - 感兴趣区域的位置已知
 - 预先知道属于边界的像素点
- 目标：基于区域连接像素，近似区域边界
- 方法
 - 函数近似
 - 为已知点拟合一条2维曲线
 - 多边形近似
 - 实现容易、捕捉基本形状特征、表示简单

举例

- 已知 A 、 B 是曲线的端点

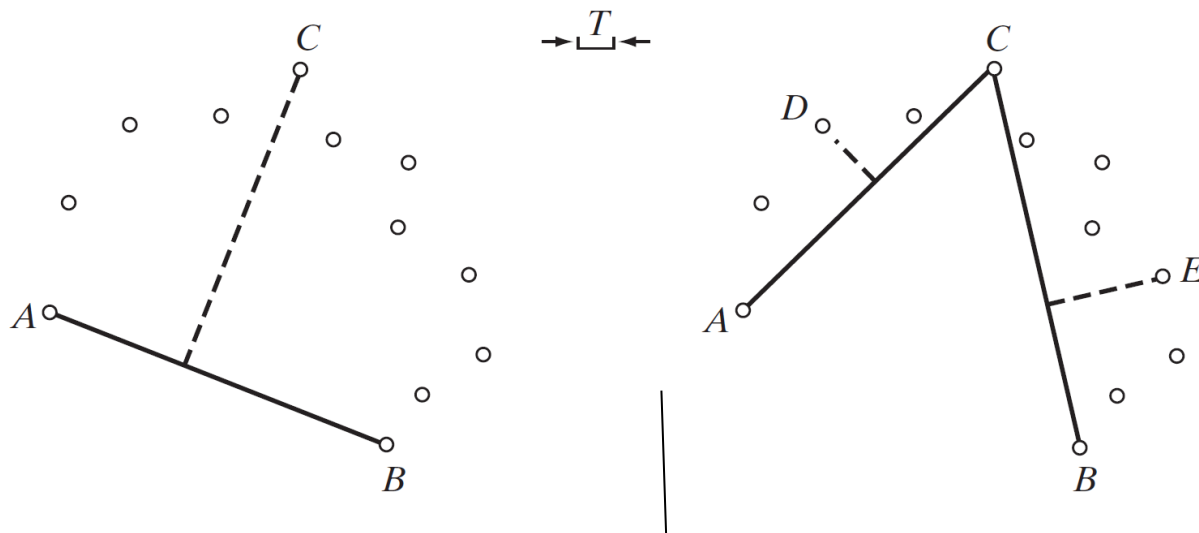


- 1、用直线连接 A 、 B
- 2、计算所有点离直线 AB 的距离
- 3、找到最远的点 C
- 4、如果距离大于阈值 T ，把 C 当做一个顶点



举例

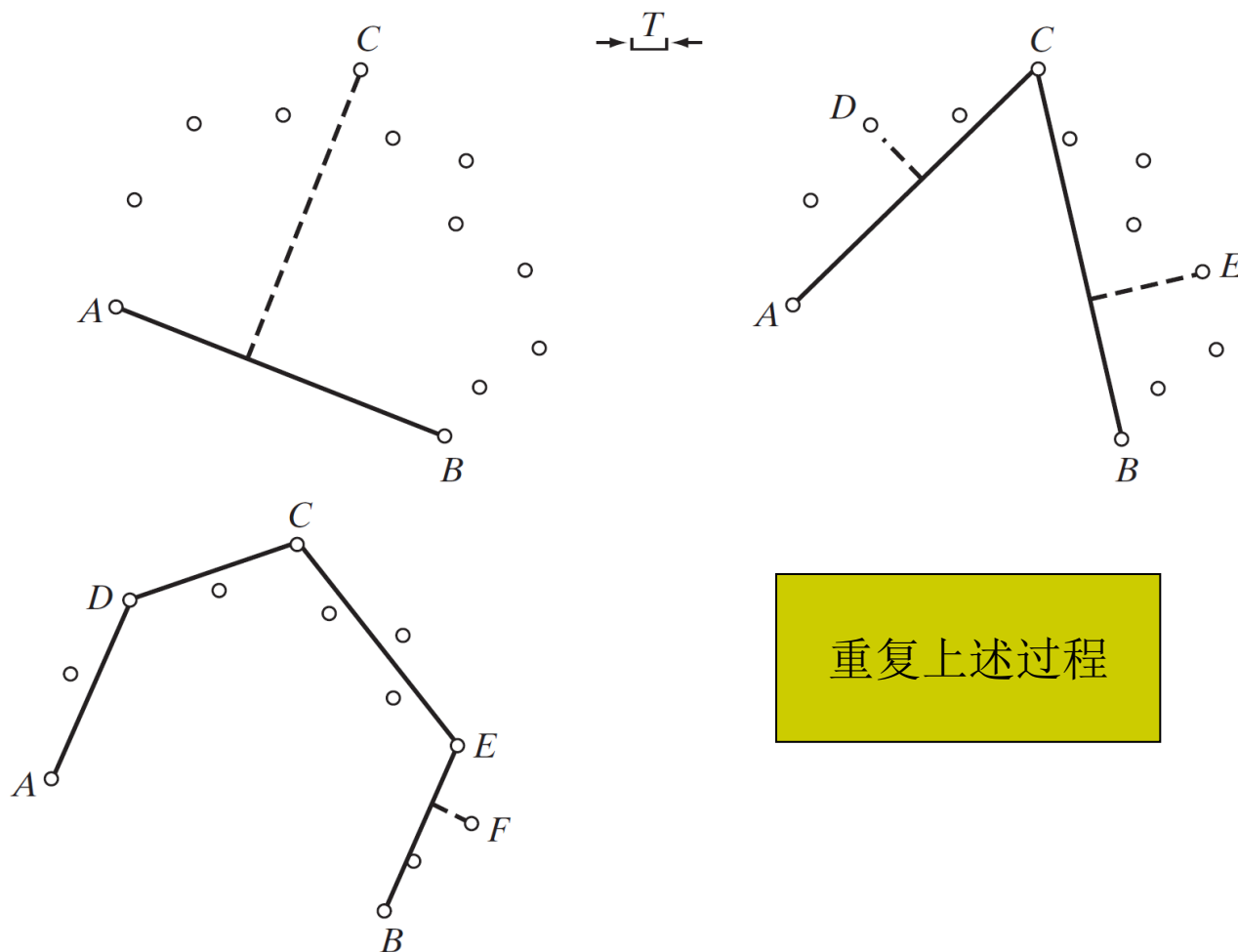
- 已知 A 、 B 是曲线的端点



- 1、用直线连接 A 、 C
- 2、计算 AC 之间的点离直线 AC 的距离
- 3、找到最远的点 D
- 4、如果距离大于阈值 T ，把 D 当做一个顶点

举例

- 已知 A 、 B 是曲线的端点

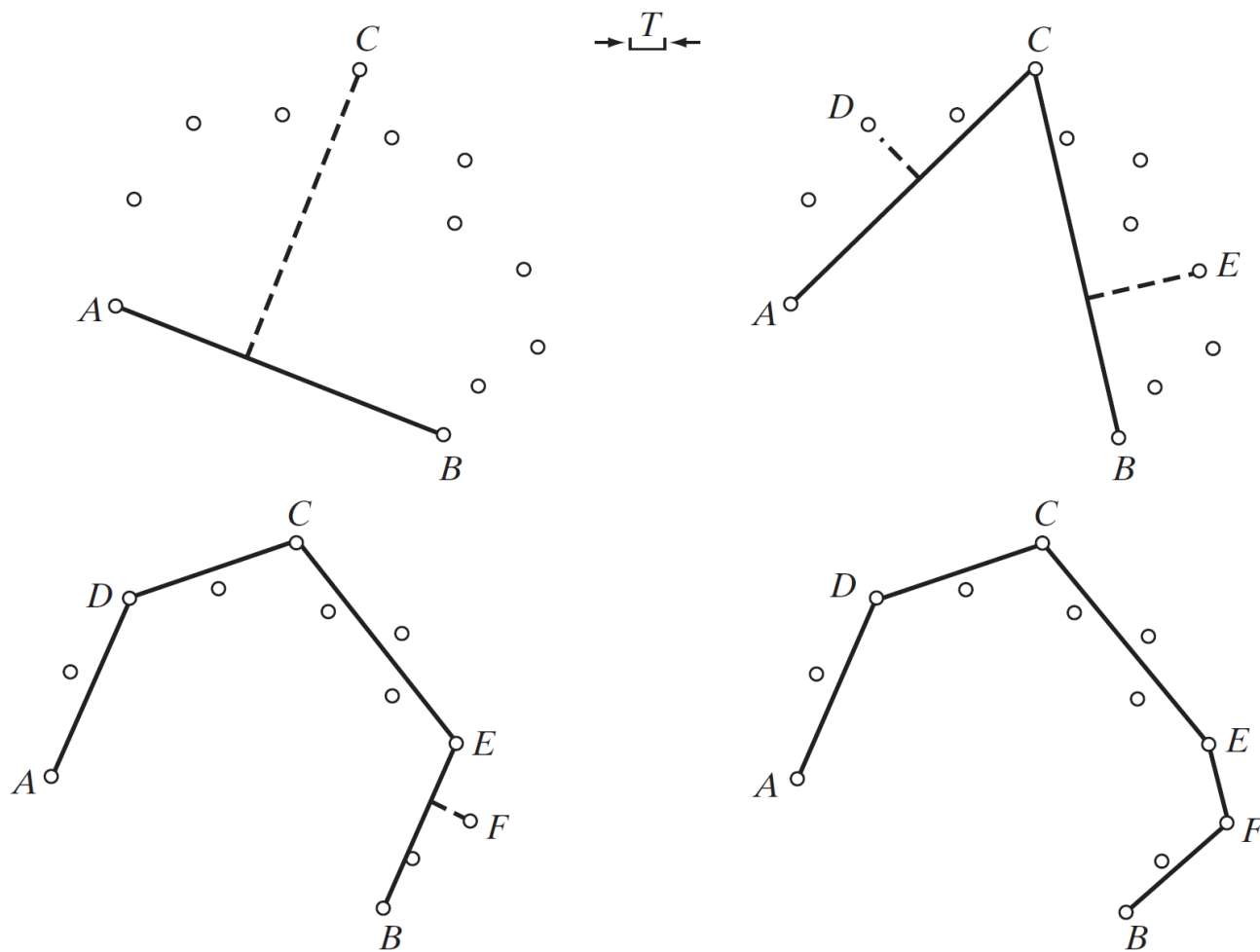


重复上述过程

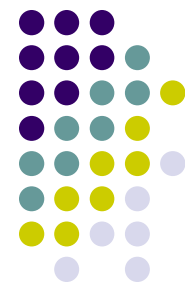


举例

- 已知 A 、 B 是曲线的端点



算法设计



- 前提
 - 两个起始点
 - 所有的点必须排序
 - 顺时针、逆时针
- 判断曲线类型
 - 边界线段（开放曲线）
 - 存在两个间距较大的连续点（可作为起始点）
 - 边界（闭合曲线）
 - 连续点之间的距离比较均匀
 - 两端的点为起始点

区域处理算法



1. 令 P 是一个已排序、不重复的二值图像中的序列。指定两个起始点 A 和 B 。它们是多边形的两个起始顶点。
2. 指定一个阈值 T ，以及两个空堆栈“开” (OPEN) 和“闭” (CLOSED)。
3. 如果 P 中的点对应于一条闭合曲线，则把 B 放到“开”和“闭”中，并把 A 放到“开”中。
如果对应于一条开放曲线，则把 A 放到“开”中，而把 B 放到“闭”。
4. 计算从“闭”中最后一个顶点到“开”中最后一个顶点的线的参数。

区域处理算法



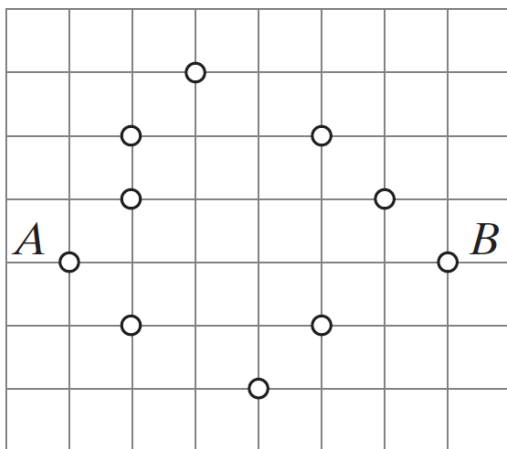
5. 寻找属于序列 P 、且在步骤4中直线的两个顶点之间的点；计算这些点与直线的距离，选择具有最大距离 D_{max} 的点 V_{max} 。
6. 如果 $D_{max} > T$ ，则把 V_{max} 作为一个新顶点放在“开”堆栈的末尾。转到步骤4。
7. 否则，从“开”中移除最后一个顶点，并把它作为“闭”的最后一个顶点插入。
8. 如果“开”非空，转到步骤4。
9. 否则，退出。“闭”中的顶点就是拟合 P 中的点的多边形的顶点。

举例



- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B

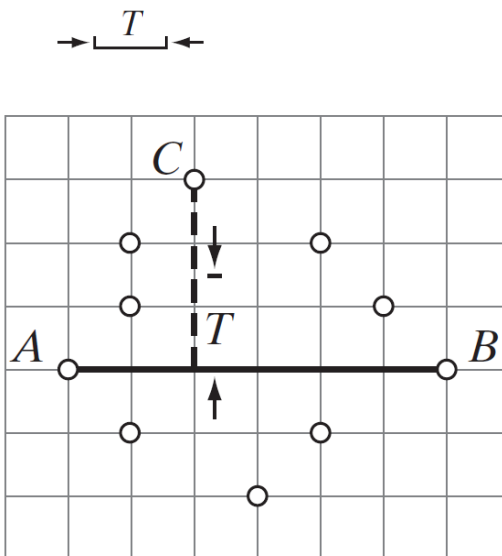
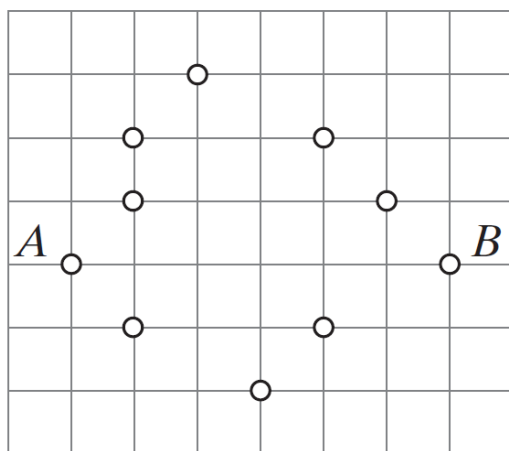


举例



- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B B	B, A B, A	— (BA)	A, B C

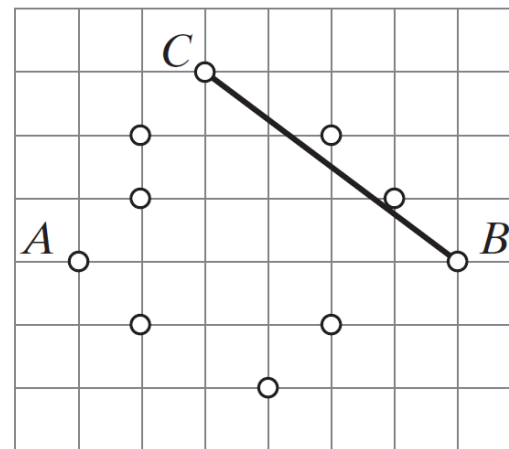
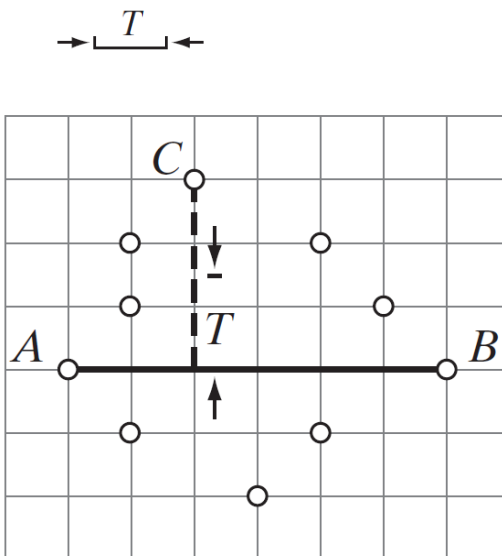
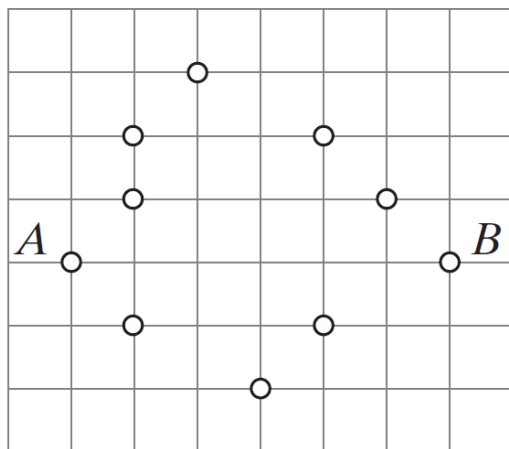


举例



- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—

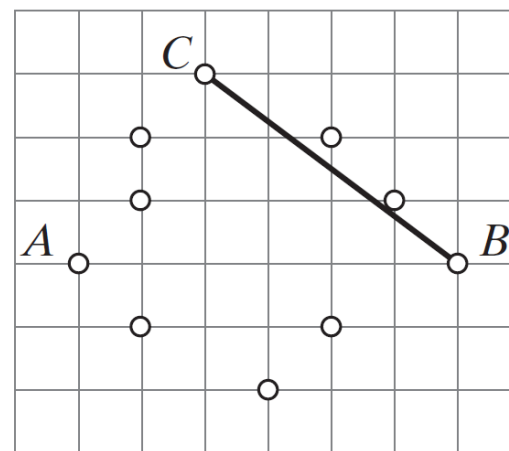
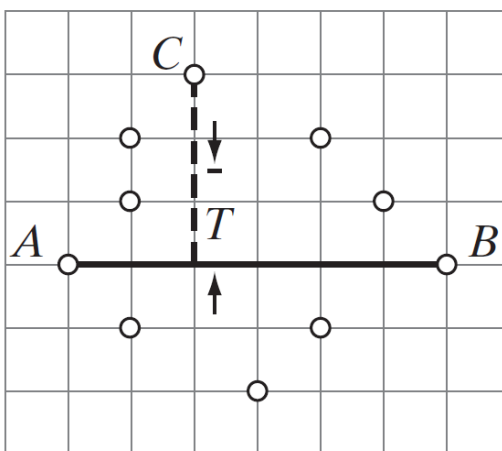
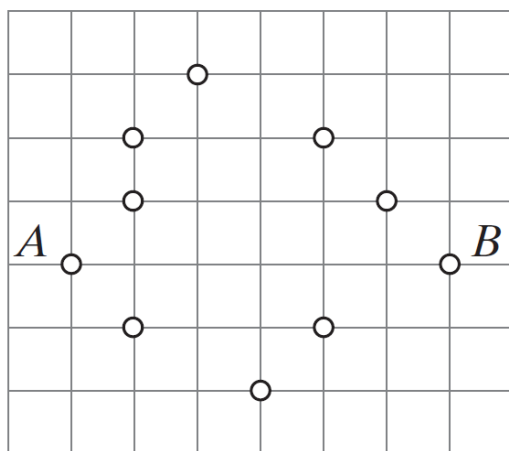


举例



- 闭合曲线
- 顺时针排序
- A 、 B 为起点

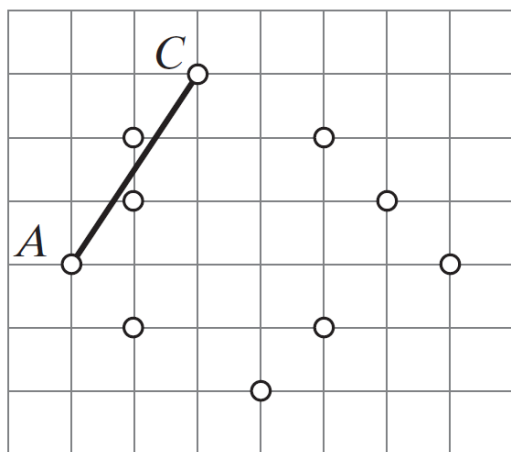
CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—



举例

- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—

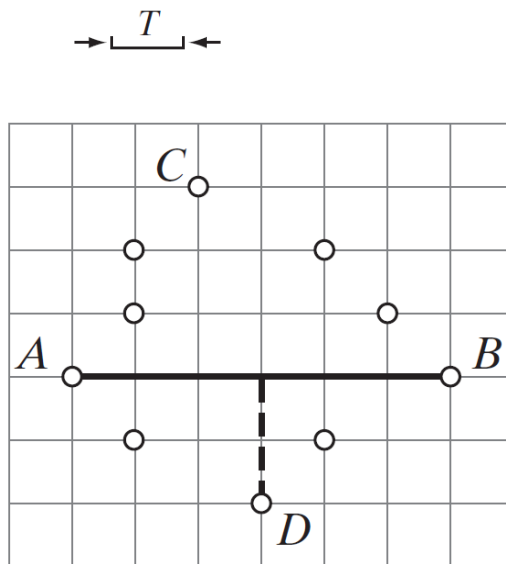
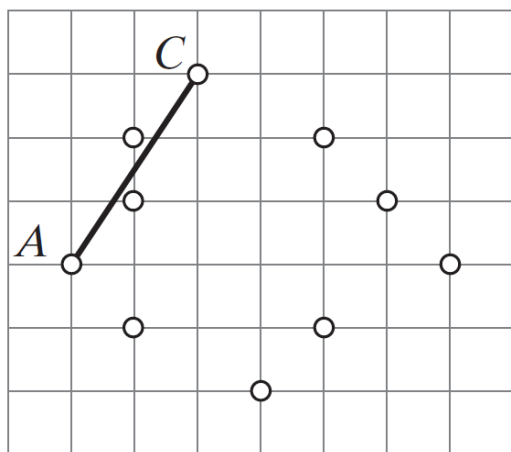


$\rightarrow \overbrace{\quad}^T \leftarrow$

举例

- 闭合曲线
- 顺时针排序
- A 、 B 为起点

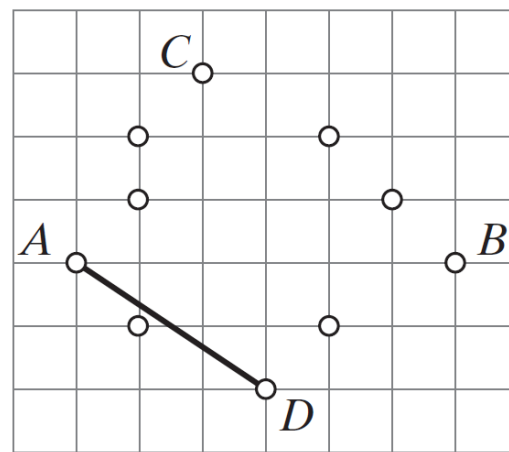
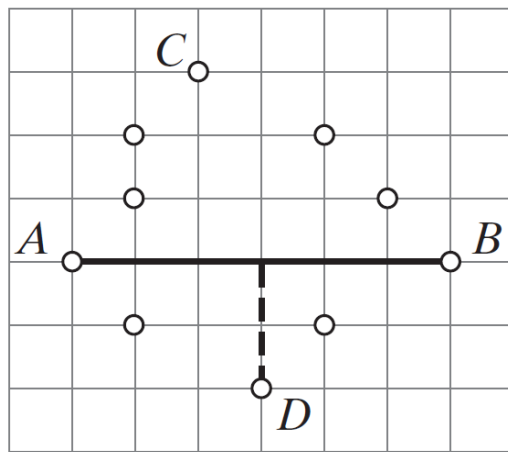
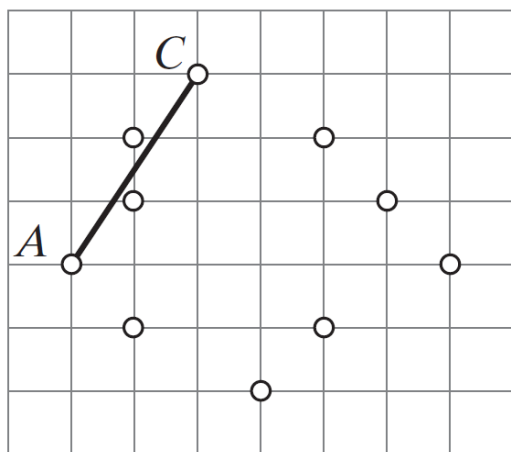
CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D



举例

- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D
B, C, A	B, D	(AD)	—



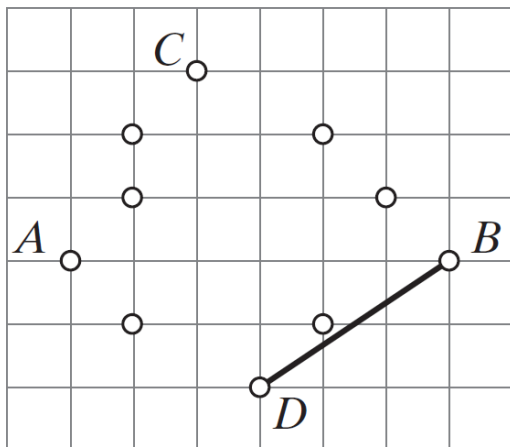
$\rightarrow \overline{T} \leftarrow$

举例

- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D
B, C, A	B, D	(AD)	—
B, C, A, D	B	(DB)	—

→ T ←

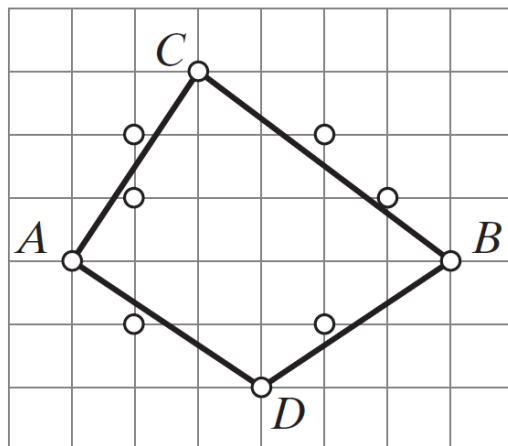
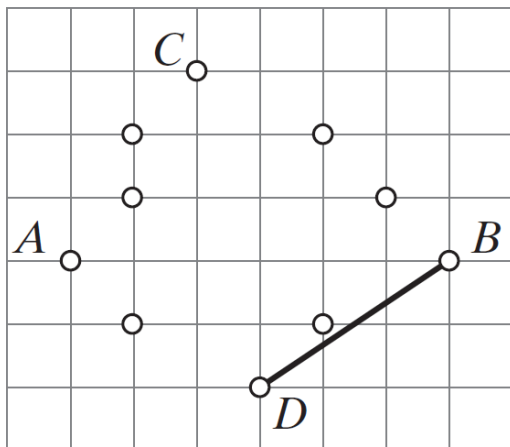


举例

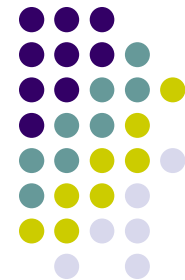
- 闭合曲线
- 顺时针排序
- A 、 B 为起点

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D
B, C, A	B, D	(AD)	—
B, C, A, D	B	(DB)	—
B, C, A, D, B	Empty	—	—

→ T ←



提纲



- 边缘连接和边界检测
 - 局部处理
 - 区域处理
 - 全局处理

全局处理



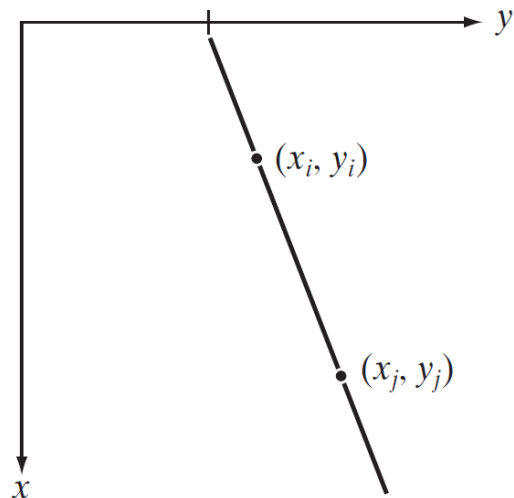
- 考虑没有边缘先验知识的情况
- 利用全局性质判断是否为边缘像素
 1. 指定感兴趣的几何形状
 2. 判断边缘像素集合是否满足该形状
- 问题：给定 n 个点，寻找共线的像素
 1. 考虑所有可能的直线 $n(n-1)/2$
 2. 寻找靠近每一条直线的像素集合
 - 复杂度 $O(n^3)$

霍夫变换

- xy -平面

- 直线方程

$$y_i = ax_i + b$$

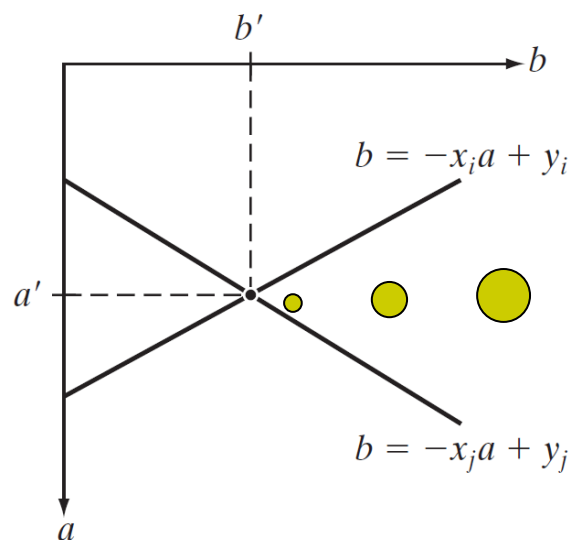


- ab -平面

- 参数方程

$$b = -ax_i + y_i$$

- 每个点对应一条直线



交点对应于
上图中直线
的参数

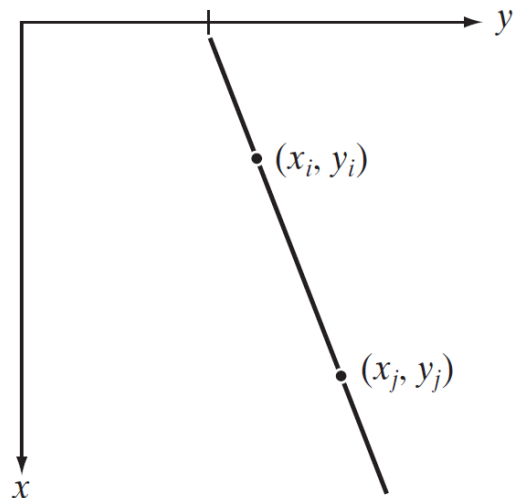


霍夫变换

- xy -平面

- 直线方程

$$y_i = ax_i + b$$

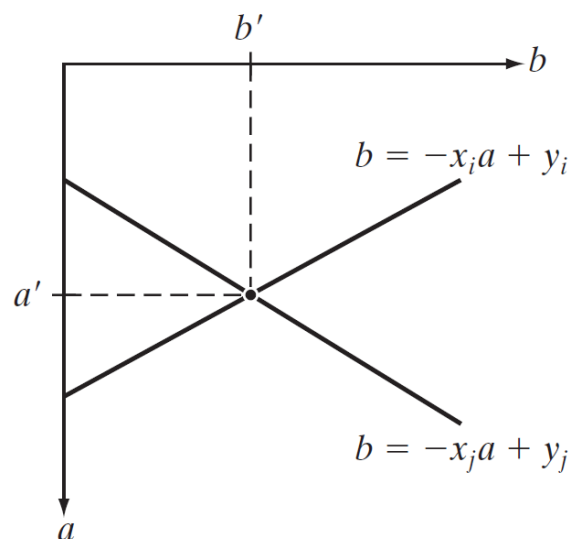


- ab -平面

- 参数方程

$$b = -ax_i + y_i$$

- 每个点对应一条直线



简单的想法:

1. 画出所有 ab -平面中的直线
2. 寻找最多直线的交点

困难:

1. ab -平面是无界的

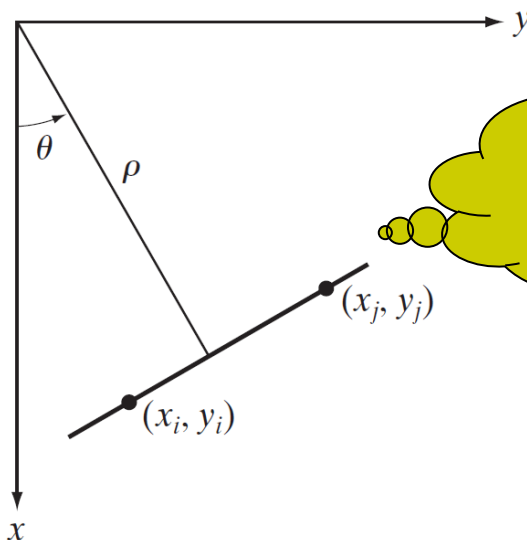


霍夫变换

- xy -平面

- 法线方程

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



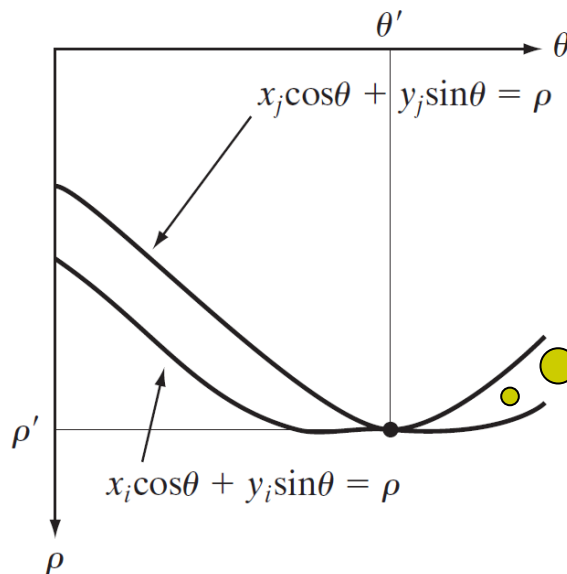
$$\langle (x, y), (\cos \theta, \sin \theta) \rangle = \rho$$

- $\rho\theta$ -平面

- 参数方程

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

- 每个点对应一条正弦曲线

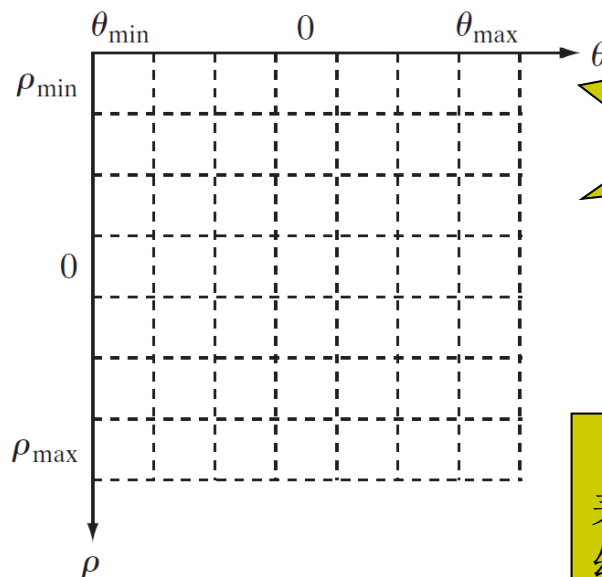


交点对应于
上图中直线的
参数

霍夫变换

- 划分累加单元

- $-90^\circ \leq \theta \leq 90^\circ$
- $-D \leq \rho \leq D$
 - D 是对角长度



复杂度和
 n 呈线性

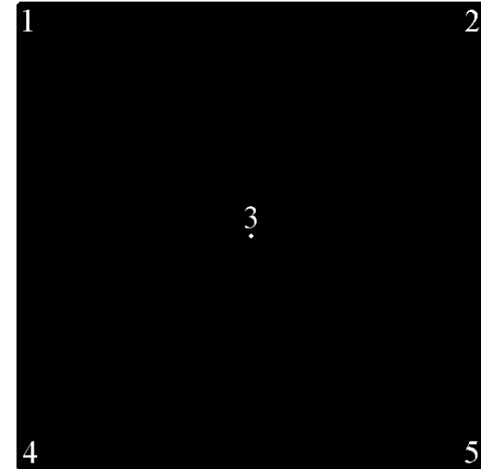
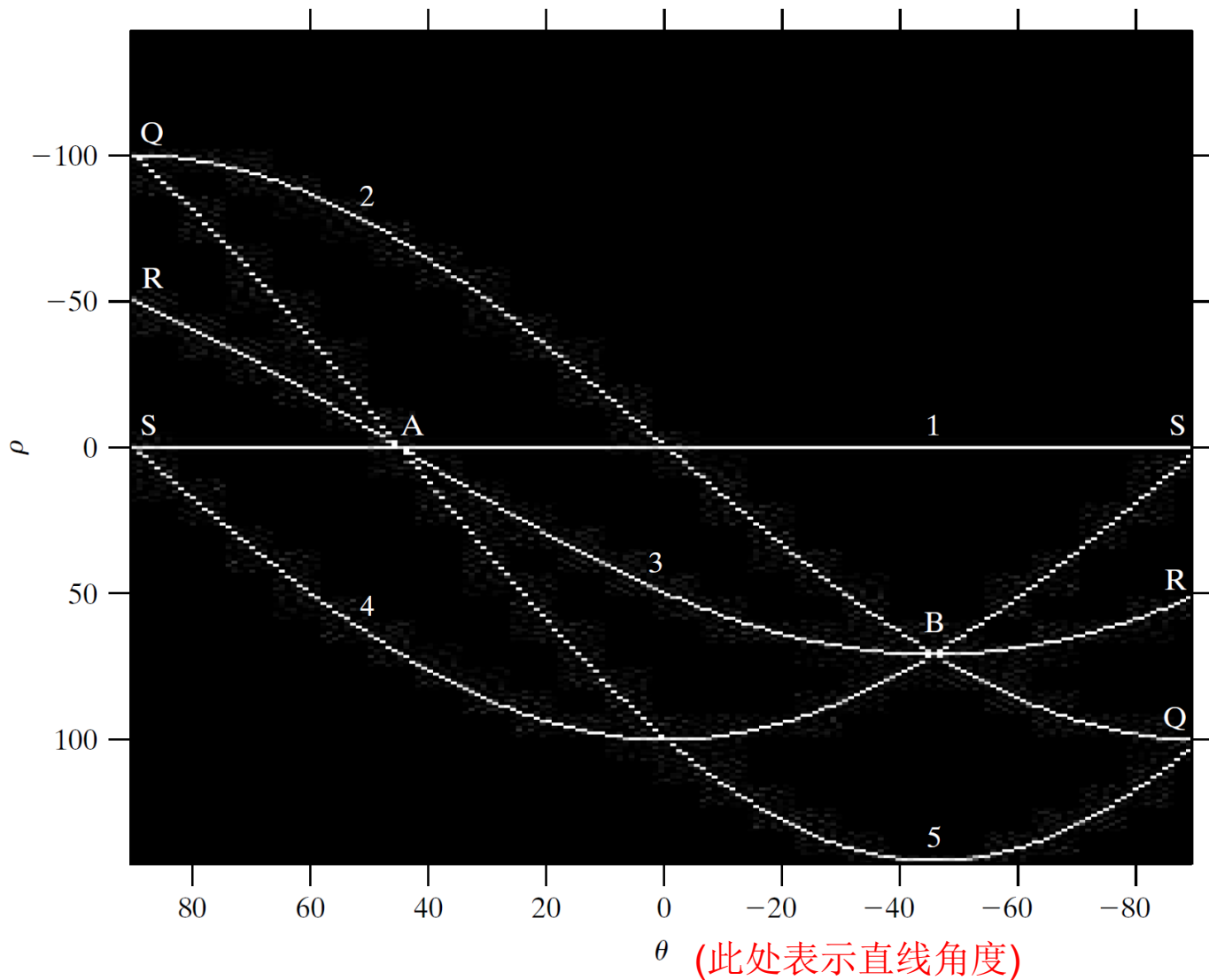
$A(i, j) = P$
表示有 P 个点属于直
线 $x\cos\theta_j + y\sin\theta_j =$
 ρ_i

- 统计每个单元内曲线的数目

- (i, j) 位置单元内曲线数目记为 $A(i, j)$
- (i, j) 位置单元对应的参数 (ρ_i, θ_j)
- 计算 $\rho = x\cos\theta_j + y\sin\theta_j$, 并离散化

举例

A处有3个点相交
对应的参数是 $(0, 45^\circ)$




B处有3个点相交
对应的参数是
 $(71, -45^\circ)$

Q, R, S在两端都出现

将霍夫变换用于边缘连接



1. 生成二值的**边缘图像**
 - 可采用之前介绍的任意算法
2. 划分 $\rho\theta$ -平面的**累加单元**
 - 粒度决定了精度、计算量
3. 统计每个累加单元的曲线数量
 - 寻找数值高的单元
4. 检验数值高累加单元对应的像素
 - 将距离小于某阈值的像素连接起来



可以扩展到直线以外的曲线

举例

- 寻找机场的主跑道
 - 中间位置、垂直方向

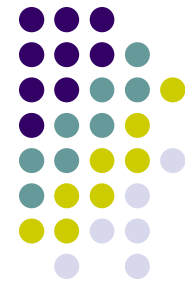


原图



举例

- 寻找机场的主跑道
 - 中间位置、垂直方向



原图

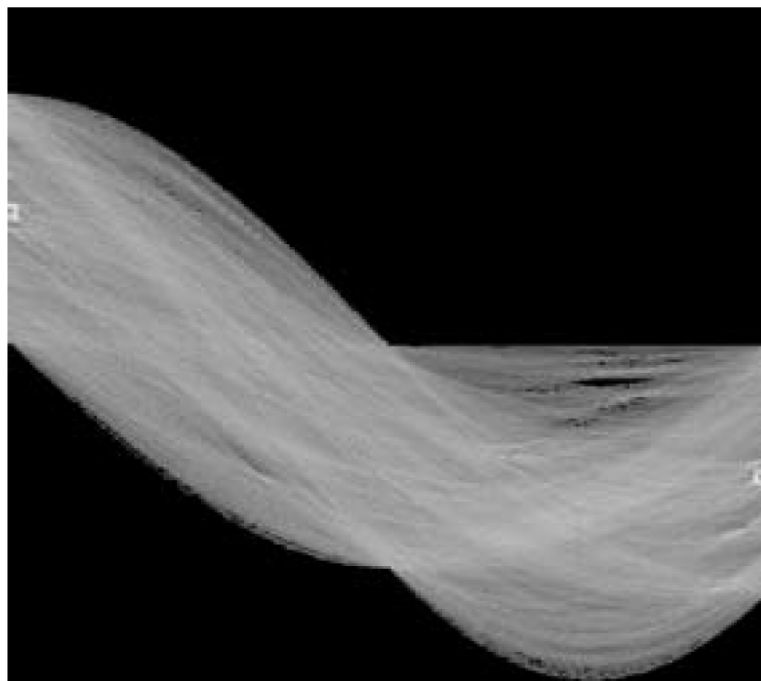


坎尼边缘检测器

举例

- 寻找机场的主跑道
 - 中间位置、垂直方向

方框为数值最大的单元



霍夫变换空间

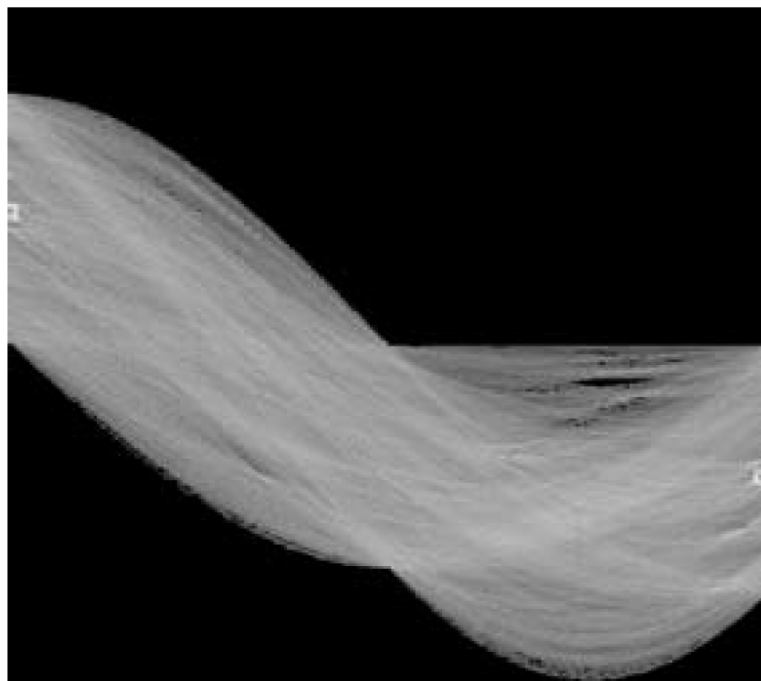


举例

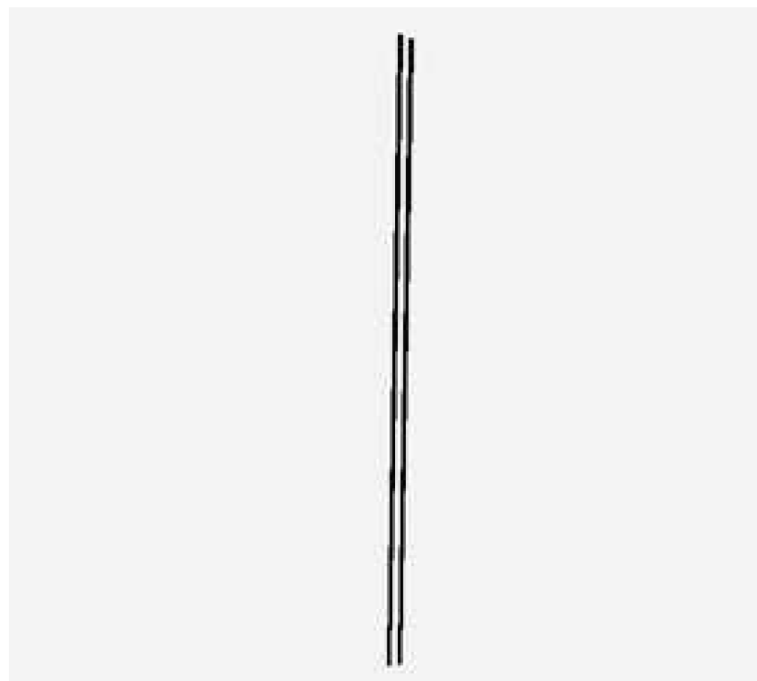
- 寻找机场的主跑道
 - 中间位置、垂直方向

方框为数值最大的单元

该单元对应的像素



霍夫变换空间



检测到的边缘

举例

- 寻找机场的主跑道
 - 中间位置、垂直方向



叠加到原图

