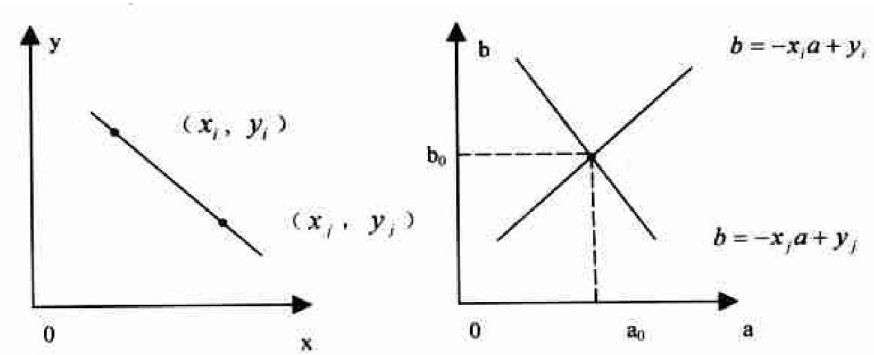
Hough变换提取直线——原理

图像坐标空间中过点(x_i, y_i)和点(x_j, y_j)的直线上的每一点在参数空间$a-b$上各自对应一条直线，这些直线都相交于点(a_0, b_0),而a_0、b_0就是图像坐标空间$x-y$中点(x_i, y_i)和点(x_j, y_j)所确定的直线的参数。
反之，在参数空间相交于同一点的所有直线，在图像坐标空间都有共线的点与之对应。根据这个特性，给定图像坐标空间的一些边缘点，就可以通过Hough变换确定连接这些点的直线方程。

具体计算时，可以将参数空间视为离散的。建立一个二维累加数组$A(a, b)$,第一维的范围是图像坐标空间中直线斜率的可能范围，第二维的范围是图像坐标空间中直线截矩的可能范围。开始时$A(a, b)$初始化为0，然后对图像坐标空间的每一个前景点(x_i, y_i),将参数空间中每一个a的离散值代入式子(2)中，从而计算出对应的b值。每计算出一对(a, b),都将对应的数组元素$A(a, b)$加1，即$A(a, b) = A(a, b) + 1$。所有的计算结束之后，在参数计算表决结果中找到$A(a, b)$的最大峰值，所对应的a_0、b_0就是源图像中共线点数目最多(共$A(a, b)$个共线点)的直线方程的参数；接下来可以继续寻找次峰值和第3峰值和第4峰值等等，它们对应于原图中共线点略少一些的直线。



（a）图像坐标空间　　（b）参数空间

图　直角坐标中的 Hough 变换

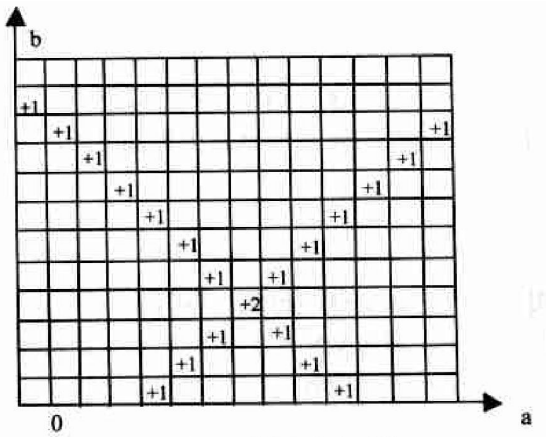


图　参数空间表决结果

Hough变换提取直线——原理

- 转换到极坐标空间：

极坐标中用如下参数方程表示一条直线。

$$\rho = xcos\theta + ysin\theta (4)$$

其中，ρ代表直线到原点的垂直距离，θ代表x轴到直线垂线的角度，取值范围为$\pm 90^{\circ}$，如图所示。
与直角坐标类似，极坐标中的Hough变换也将图像坐标空间中的点变换到参数空间中。
在极坐标表示下，图像坐标空间中共线的点变换到参数空间中后，在参数空间都相交于同一点，此时所得到的ρ、θ即为所求的直线的极坐标参数。与直角坐标不同的是，用极坐标表示时，图像坐标空间的共线的两点(x_i, y_i)和(x_j, y_j)映射到参数空间是两条正弦曲线，相交于点(ρ_0, θ_0),如上图所示。

具体计算时，与直角坐标类似，也要在参数空间中建立一个二维数组累加器A，只是取值范围不同。对于一副大小为$D \times D$的图像，通常ρ的取值范围为$[-\sqrt{2}D/2, \sqrt{2}D/2]$,$\theta$的取值范围为$[-90^{\circ}, 90^{\circ}]$。计算方法与直角坐标系中累加器的计算方法相同，最后得到最大的A所对应的(ρ, θ)。

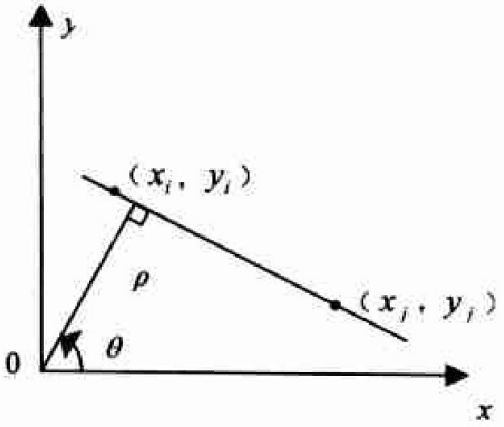


图　直线的参数式表示

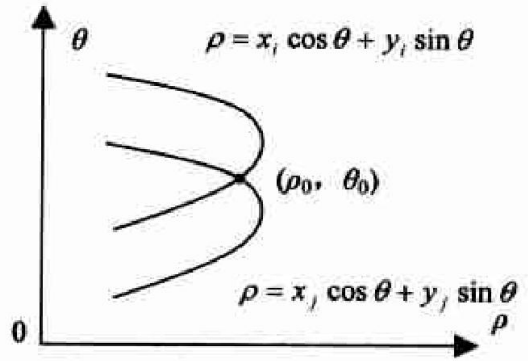
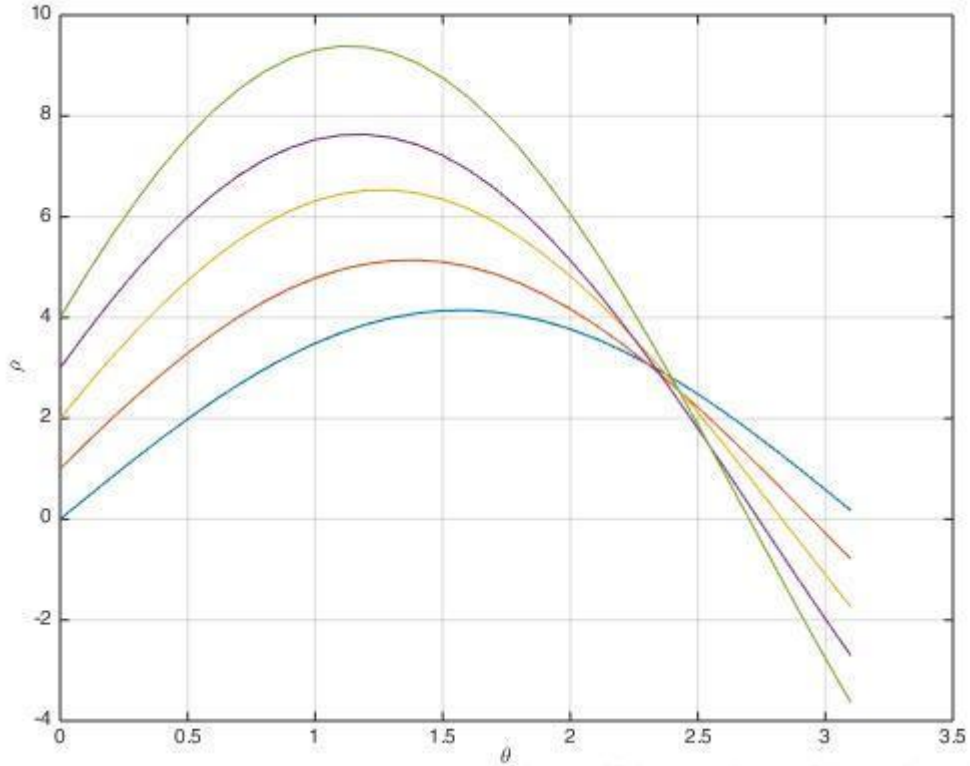
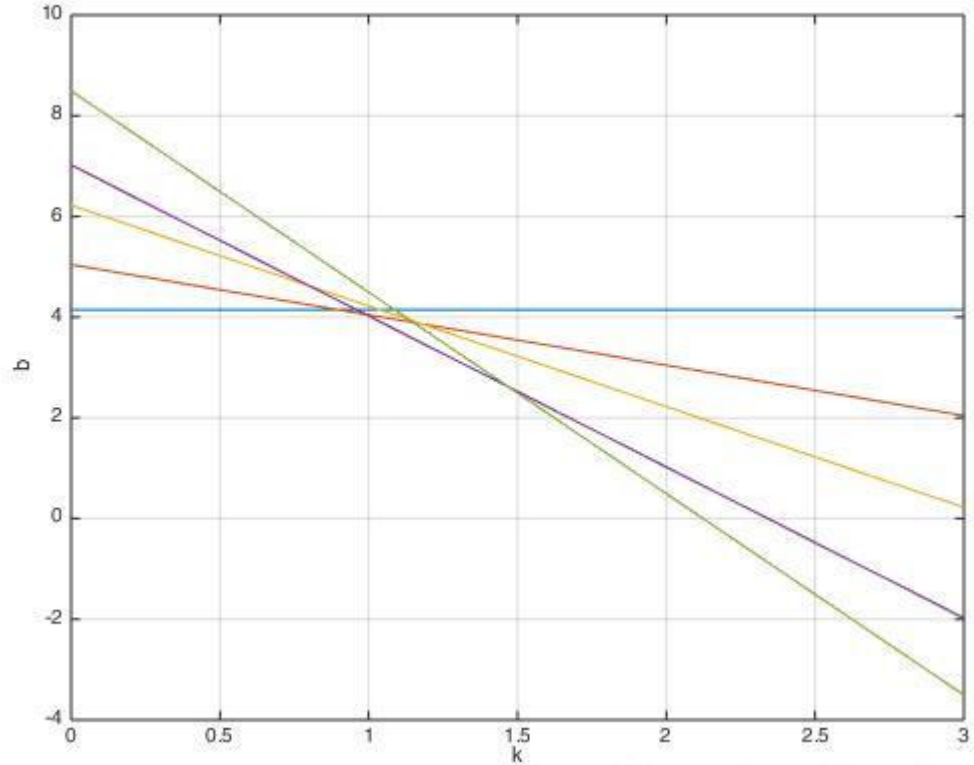


图　笛卡儿坐标映射到参数空间



放大



不严格相较于一点
解决办法：将参数空间格点化， 例如步长为1，落在同一个格子里的都算相交

边界跟踪


```
clear all
I = imread('rice.png');              %导入图像
figure(1),
subplot(1,3,1);
imshow(I),title('原始图像')
BW = im2bw(I, graythresh(I));        %生成二值图像
subplot(1,3,2);
imshow(BW),title('二值图像')
 [B,L] = bwboundaries(BW,'noholes'); %提取边界，并返回边界元胞数组B 和区域标志数组L
subplot(1,3,3);
imshow(label2rgb(L, @jet, [.5 .5 .5]))  %以不同的颜色标志不同的区域
title('彩色标记图像')
hold on
for k = 1:length(B)
   boundary = B{k};
   plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)  %在图像上叠画边界
end
```

L中存放区域编号

L　409 x 412

L(:)　168508 x 1 即 HW x 1

area_ids 76 x 1

find(L(:)==area_ids(3))：　919 x 1

```
 9 -    [B,L] = bwboundaries(BW,'noholes');      %提取边界，并返回边界元胞数组B 和区域标志数组L
10 -    area_ids=unique(L);
11 -    area = zeros(length(area_ids),1);
12 -    for i=1:length(area_ids)
13 -        area(i)=length(find(L(:)==area_ids(i)));
14 -    end
15      %对区域面积进行排序
16      % ...
```

area_ids(1)=0
最大的是背景，要先排除掉

边界跟踪

```
clear all
I = imread('rice.png');                %导入图像
figure(1),
subplot(1,3,1);
imshow(I),title('原始图像')
BW = im2bw(I, graythresh(I));          %生成二值图像
subplot(1,3,2);
imshow(BW),title('二值图像')
 [B,L] = bwboundaries(BW,'noholes'); %提取边界，并返回边界元胞数组B 和区域标志数组
subplot(1,3,3);
imshow(label2rgb(L, @jet, [.5 .5 .5]))  %以不同的颜色标志不同的区域
title('彩色标记图像')
hold on
for k = 1:length(B)
   boundary = B{k};
   plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)   %在图像上叠画边界
end
```

```
area_ids = unique(L);
for i=1:length(area_ids)
    area = length(find(L(:) == area_ids(i)));
end
```

```
9 -     [B,L] = bwboundaries(BW,'noholes');      %提取边界，并返回边界元胞数组B 和区域标志数组L
10 -    area_ids=unique(L);
11 -    area = zeros(length(area_ids),1);
12 -    for i=1:length(area_ids)
13 -        area(i)=length(find(L(:)==area_ids(i)));
14 -    end
15      %对区域面积进行排序
16      % ...
```

这里有好几层下标,不要搞乱了

L中存放区域编号

L 409 x 412

L(:) 168508 x 1 即 HW x 1

area_ids 76 x 1

find(L(:)==area_ids(3)): 919 x 1

区域面积

虚假的区域编号

Top3_ids=zeros(1,3);
% ... 排序算法

Top3_ids = [12, 14, 67]

真正的区域编号

area_id(12) = 35 → B{35}
area_id(14) = 28
area_id(67) = 56

Top3area_ids=zeros(1,3);
% ... 排序算法

Top3area_ids = [35, 28, 56]

课程实验

任意给出一幅彩色图像，编写代码实现：

（1）利用Hough变换检测图像中最长的3条直线并且标记出来；

（2）利用matlab自带的边界跟踪函数标记出面积最大的3个区域。