第五周

图像处理（一）

- OSTU二值化算法
- 线扫描
- 旋转变换
- 临近插值法

OSTU二值化算法

- 大津算法：

算法假定该图像根据双模直方图包含两类像素：

前景像素和背景像素

于是它要计算能将两类分开的最佳阈值，使得它们的类内方差最小；由于两两平方距离恒定，所以即它们的类间方差最大。

- 求最大类间方法：枚举

- 方差：

$$\mathrm{Var}(X) = \sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2$$

假设：

前景像素所占比例为w_0，期望为u_0

背景像素所占比例为w_1，期望为u_1

整个图像的期望$u = w_0 * u_0 + w_1 * u_1$

分割的像素点为t

根据方差公式：

$$g(t) = w_0 * (u_0 - u)^2 + w_1 * (u_1 - u)^2$$

当t使得g(t)最大时，即为图像最佳的阈值

- 由于$w_1=1-w_0$
- $u_1=(u-w_0*u_0)/(1-w_0)$

故$\boldsymbol{g(t)=w_0*(u_0-u)^2+w_1*(u_1-u)^2}$

可化简为

$$g(t)=\frac{w_0(t)*(u-u_0)^2}{(1-w_0(t))}$$

假设$u_a=u_0*w_0$ 则

$$g(t)=\frac{(u*w_0(t)-u_a(t))^2}{w_0(t)*(1-w_0(t))}$$

Matlab实现流程

1. 统计各灰度级像素在整幅图像中的个数：

Count=imhist（f）；f为输入的灰度图像

2. 计算每个灰度级在图像中所占的比例

[Row Col]=size(f);

Count=Count /（Row*Col）;Row和Col是f的长和宽

得到的Count是归一化的直方图

3. 去除两边不存在的灰度级

```
L=256
for i=1:L
    记录直方图（count）第一个不为0的点
end
for i=L:-1:1
    记录直方图（count）最后一个不为0的点
end
count = count(st+1:nd+1);
```

思考：为什么要去除两边的灰度级？

4. 计算计算前t个像素的累加概率$w_0(t)$和像素期望值$u_0(t)$

$$\mu_1(t) = \left[\Sigma_0^t p(i)\, x(i)\right] / \omega_1$$

```
for t = 1 : L
    w0 (t) = sum(count(1:t));
    u0(t) = sum(x(1:t)*count(1:t)) / w0(t);
end
```

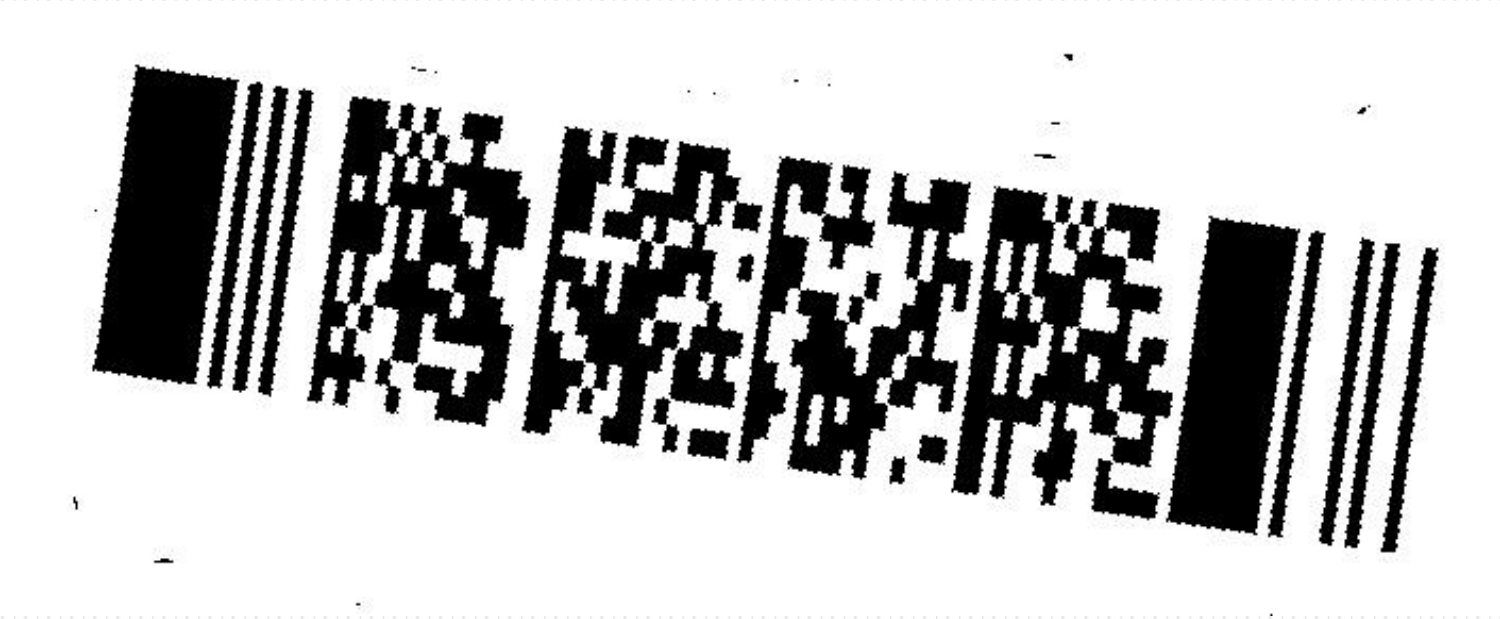
5. 计算g(t)，得到

```
for t = 1 : L
    g(t) = w0 (t)*(u - u0 (t))^2/(1 - w0 (t));
end
```

线扫描

由于拍摄条件的限制，我们得到的PDF417码图形会存在不同类型的失真。

当图形只存在旋转失真且图片中干扰较小时，可以利用扫描PDF417左边线是方法得到图形的旋转角度。



Matlab实现流程

1. 建立一个零矩阵leftline

```
leftline=zeros（row, 2）        %row是图片的行数
%leftline(i,j)记录位于二维码某一直线的点
```

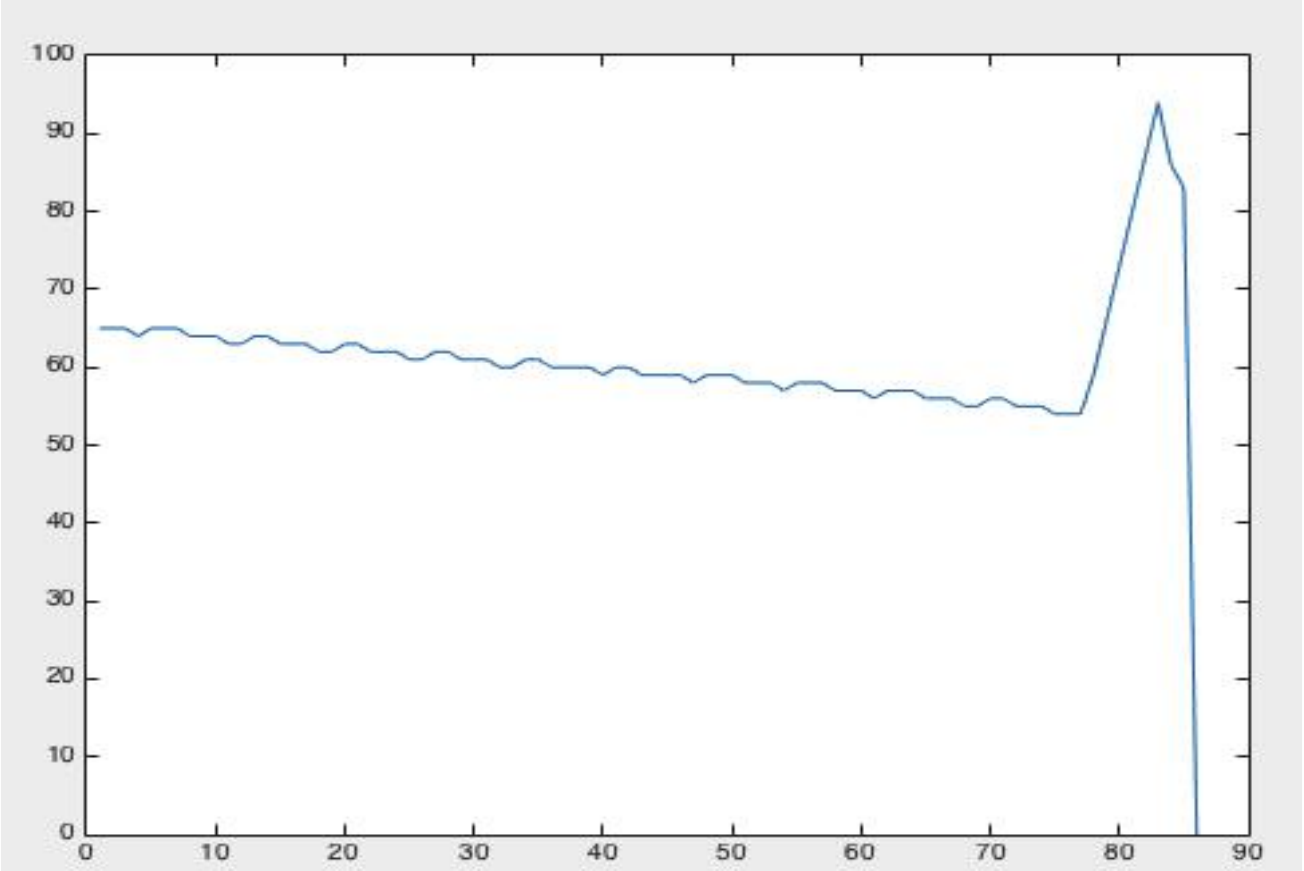
2. 开始历遍图像

3. 当某像素点的灰度值为0且该点的所在的列在上一个记录的点的所在的列的距离小于10时，在leftline中记录该点的行数与列数

```
for i = 1:row
    for j=1:col
        if 检测到灰度值为0的像素
        if  j-leftline(k-1, 2) < 10
            leftline(k,1) = i;
            leftline(k,2) = j;
            k = k+1;
        end
    end
    break
 end
end
```

4. 利用plot()函数 在屏幕上描绘扫描线

plot(leftline(1:k, 2))



旋转变换

- 旋转角度：

根据线扫面，得到二维码左侧边一条直线上点的坐标。坐标集合的每两个点可以计算一次斜率。取这些斜率的中点，作为旋转角度

```
n= int(k/2)
for i=1:n
    tanarray(i) = (leftline(i+n, 2)-leftline(i, 2))/(leftline(i+n, 1)-leftline(i, 1));
end
tanarray=sort(tanarray);
ang=atan(0-tanarray(int(n/2))) %ang 是旋转角度
```

- 旋转后照片尺寸：size=(col*sin(ang)+row*cos(ang))*(col*cos(ang)+row*sin(ang)　　%row*col是原图片的尺寸

- 旋转后的像素值符合以下公式

```
X=(i-row/2)*cos(ang)+(j-col/2)*sin(ang)+
    (col*sin(ang)+row*cos(ang))/2
Y=(i-row/2)*sin(ang)+(j-col/2)*cos(ang)+
    (col*cos(ang)+row*sin(ang))/2
dst(X,Y)=src(i,j);
```

- 新建一个size大小的矩阵存放旋转后图像，做变换

```
ost = ones(new_row, new_col)*255;
dst(X,Y)=src(i,j);
```

邻近插值法

- 由于旋转后图片会大小大量毛刺，故需要用领近插值法来去除

- 插值规则是，如果该点的水平或垂直方向上的邻近点与该点的灰度值不同，则把该点的灰度值重新赋值为邻近点的值

迷人风景只有站在一定高度才能领略，望诸君努力，谢谢！