第六次编程作业

PB16150288 黄志鹏

实验目的

1. 利用霍夫变换做直线边缘检测
2. 对图像进行全局阈值分割
3. 对另一个图像进行Otsu图像分割并对比较朴素的算法
4. 对图像进行分块可变图像分割，并对比前两种算法

实验原理

1. 直线的霍夫变换

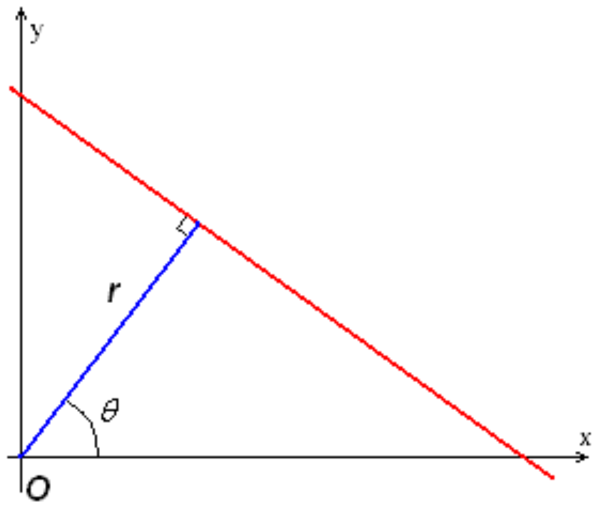
最简单的霍夫变换是在图像中识别直线。在[平面直角坐标系(x-y)中，一条直线可以用方程式

$$y = m_0 x + b_0$$

表示，b_0是直线的截距，m_0是直线的斜率。而(m_0, b_0)可以视为参数空间(m, b)中的一点。当直线垂直于x轴时，斜率为无限大，若用电脑数值计算时会很不方便，因此提出使用[Hesse normal form]来表示直线的参数：

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

r是从原点到直线的距离，θ是r→$\vec{r}$和x轴的夹角。利用参数空间(r, θ)解决了原本参数空间(m, b)发散的问题，进而能够比较每一个线段的参数，有人将(r, θ)平面称为二维直线的霍夫空间(Hough space)。



给定一个点(x_0, y_0)，通过该点的所有直线的参数(r, θ)的集合，会在(r, θ)平面上形成一个三角函数，可由下方程式证明：

$$r = x_0\cos\theta + y_0\sin\theta \Rightarrow r = \sqrt{x_0^2+y_0^2}\left(\frac{x_0}{\sqrt{x_0^2+y_0^2}}\cos\theta + \frac{y_0}{\sqrt{x_0^2+y_0^2}}\sin\theta\right) \Rightarrow r = \sqrt{x_0^2+y_0^2}(\cos\phi\cos\theta+\sin\phi\sin\theta)$$

$$\Rightarrow r = \sqrt{x_0^2+y_0^2}\cos(\theta-\phi)$$

因此，给定很多点，判断这些点是否共线([concurrent lines)的问题，经由霍夫变换之后，变成判断一堆曲线(每一个点在(r,θ)平面上代表一条曲线)是否 在(r,θ)平面上相交于同一点的问题(concurrent curves)。

2. 全局阈值分割

```
1.为全局阈值选择一个初始估计值T(图像的平均灰度)。
2.用T分割图像。产生两组像素：G1有灰度值大于T的像素组成，G2有小于等于T像素组成。
3.计算G1和G2像素的平均灰度值m1和m2;
4.计算一个新的阈值:T = (m1 + m2) / 2;
5.重复步骤2和4,直到连续迭代中的T值间的差为零。
```

3. Otsu图像分割

1. 计算每个强度级的直方图和概率
2. 设置 $\omega_i(0)$ 和 $\mu_i(0)$ 的初始值
3. 遍历所有可能的阈值

$$t = 1\ldots$$

   最大强度
   1. 更新 ω_i 和 μ_i
   2. 计算 $\sigma_b^2(t)$
4. 所需的阈值对应于最大的

$$\sigma_b^2(t)$$

5. 你可以计算两个最大值（和两个对应的）。

$$\sigma_{b1}^2(t)$$

   是最大值而 是更大的或相等的最大值
6. 所需的阈值 =

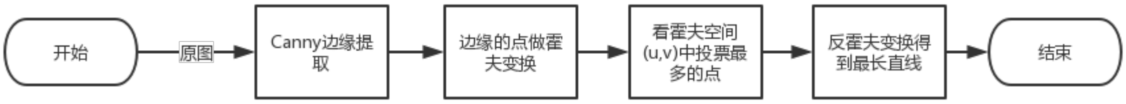
$$\frac{\text{threshold}_1 + \text{threshold}_2}{2}$$

4. 分块可变的Otsu图像分割

先对图像进行分块，每一块做otsu方法分割，然后再合并。

实验过程

1. 霍夫变换检测最长的直线




```
clear
clc
close all;
f=imread('2.png');
figure;
f=rgb2gray(f);
imshow(f);
[gv,t]=edge(f,'canny');

figure;
imshow(gv);

figure;
[H,theta,rho]=hough(gv,'Theta',-90:0.2:90-0.2);
imshow(H,[],'XData',theta,'YData',rho,'initialMagnification','fit');

axis on, axis normal
xlabel('\theta'),ylabel('\rhp');

[x,y]=lineextraction(f);
% Plot the line in the image
figure; imshow(f, [min(min(f)) max(max(f))]), hold on
plot([x(1) y(1)], [x(2) y(2)],'LineWidth',2,'Color','blue');
plot(x(1),x(2),'x','LineWidth',2,'Color','red');
plot(y(1),y(2),'x','LineWidth',2,'Color','red');
hold off
```

2. 全局阈值分割

```
clc;
clear;
close all;
count =0;
f=imread('2.tif');
f=imresize(f,0.5);
T=mean2(f);
done =false;
while ~done
    count =count +1;
```

```
    g= f > T;
    Tnext=0.5*(mean(f(g))+mean(f(~g)));
    done = abs(T-Tnext) < 0.5;
    T = Tnext;
end

g= im2bw(f,T/300);
imshow(f)
figure;imhist(f);
figure;imshow(g);
```

3. Ostu 阈值分割

```
clc;
clear;
close all;
count =0;
f=imread('3.tif');

T=mean2(f);
done =false;
while ~done
    count =count +1;
    g= f > T;
    Tnext=0.5*(mean(f(g))+mean(f(~g)));
    done =abs(T-Tnext) < 0.5;
    T = Tnext;
end

g= im2bw(f,T/255);
imshow(f)
figure;imhist(f);
figure;imshow(g);

f2=imhist(f);
[T,M]=myotsu(f2);
g=im2bw(f,T);
figure,imshow(g)
```

4. 分块可变的阈值分割

```
clc;
clear;
close all;
```

```
count =0;
%f=imread('4.tif');
f=imread('5.tif');
[m,n]=size(f);

im_1_1=f(1:ceil(m/2),1:ceil(n/3));
im_1_2=f(ceil(m/2)+1:m,1:ceil(n/3));
im_2_1=f(1:ceil(m/2),ceil(n/3)+1:ceil(n/3*2));
im_2_2=f(ceil(m/2)+1:m,ceil(n/3)+1:ceil(n/3*2));
im_3_1=f(1:ceil(m/2),ceil(n/3*2)+1:n);
im_3_2=f(ceil(m/2)+1:m,ceil(n/3*2)+1:n);

T=mean2(f);
done =false;
while ~done
    count =count +1;
    g= f > T;
    Tnext=0.5*(mean(f(g))+mean(f(~g)));
    done =abs(T-Tnext) < 0.5;
    T = Tnext;
end

g= im2bw(f,T/255);
imshow(f)
figure;imhist(f);
figure;imshow(g);

f2=imhist(f);
[T,M]=myotsu(f2);
g=im2bw(f,T);
figure,imshow(g);

f1=im_1_1;
f2=imhist(f1);
[T,M]=myotsu(f2);
g_1_1=im2bw(f1,T);

f1=im_1_2;
f2=imhist(f1);
[T,M]=myotsu(f2);
g_1_2=im2bw(f1,T);

f1=im_2_1;
f2=imhist(f1);
[T,M]=myotsu(f2);
g_2_1=im2bw(f1,T);

f1=im_2_2;
f2=imhist(f1);
```

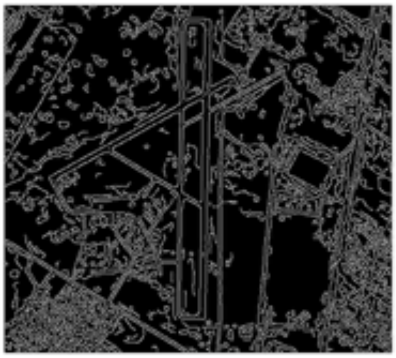
```
[T,M]=myotsu(f2);
g_2_2=im2bw(f1,T);

f1=im_3_1;
f2=imhist(f1);
[T,M]=myotsu(f2);
g_3_1=im2bw(f1,T);

f1=im_3_2;
f2=imhist(f1);
[T,M]=myotsu(f2);
g_3_2=im2bw(f1,T);

g_new=g;
g_new(1:ceil(m/2),1:ceil(n/3))=g_1_1;
g_new(ceil(m/2)+1:m,1:ceil(n/3))=g_1_2;
g_new(1:ceil(m/2),ceil(n/3)+1:ceil(n/3*2))=g_2_1;
g_new(ceil(m/2)+1:m,ceil(n/3)+1:ceil(n/3*2))=g_2_2;
g_new(1:ceil(m/2),ceil(n/3*2)+1:n)=g_3_1;
g_new(ceil(m/2)+1:m,ceil(n/3*2)+1:n)=g_3_2;

figure;
imshow(g_new);
```

实验结果

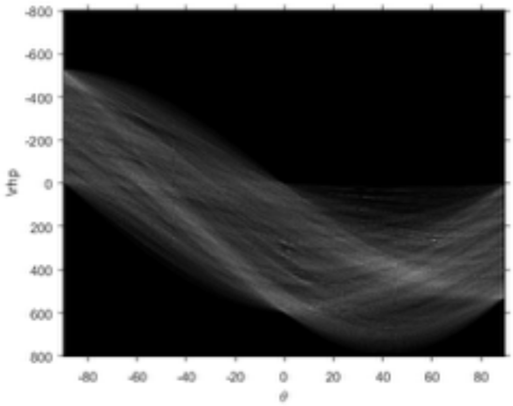
1. 霍夫变换

原图：

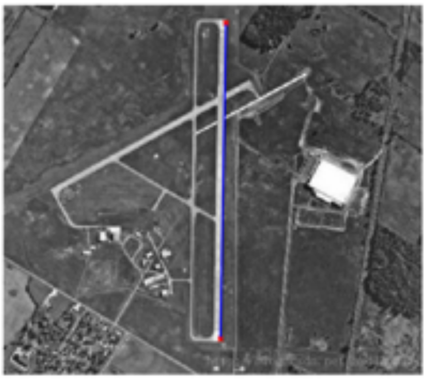
边界提取：



霍夫变化:



最长直线提取结果:



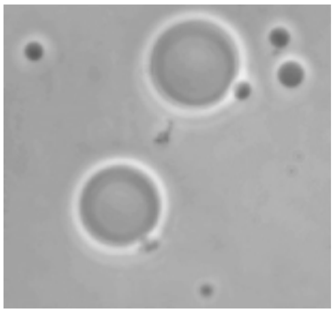
2. 全局阈值分割

原图:

全局阈值分割：

3. Ostu阈值分割

原图：



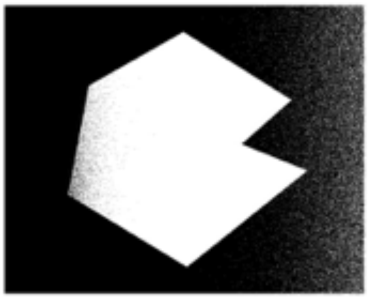
全局阈值分割：

Ostu阈值分割：

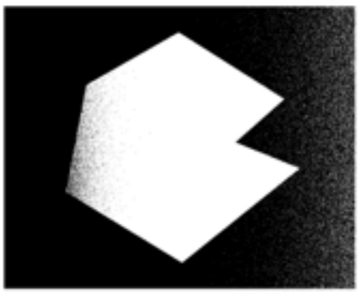
4. 分块可变的阈值分割

原图：

基本全局阈值分割：



Ostu阈值分割：



分块可变阈值分割：

结果分析

4.1 霍夫变换

由实验结果可知，经过坎宁算子边缘提取，以及霍夫变换后，确实能够提取出直线，说明实验结果良好。

4.2 全局阈值处理

由实验结果可知，在图像比较友好的情况下，即有明显的两个峰值，用基本阈值分割算法能够得到较好的阈值分割结果。

其中，参数T/K 决定了图片处理的速度和清晰度。

4.3 Otsu 方法

由直方图见，由于亮度没办法像之前 一样分成两个明显的峰值，相反，较为密集，普通全局阈值处理算法会错把左下角部分的背景当成前景。而Otsu 算法能够规避这一问题，体现了他的优越性。这也从侧面反映了实验符合我们的预期，所以是成功的。

4.4 分块可变阈值分割

由直方图和实验结果可知，在有噪声，且图像整体较暗的情况下，基本全局阈值算法效果比较一般，而Otsu 算法对此的性能提升，也好不到哪去，主要体现在错把噪声当成前景或者背景。而使用分块的Otsu 方法，则能得到比较好的结果。