图像的特征提取与增强实验

实验人：李东嵘 **16342080**

实验目的：先用直方图技术对图像进行直方图均衡化，再用边缘提取算子提取图像边缘，最后用**hough**变换检测图像中的圆。

实验平台：windows10，MATLAB **R2016a**

在本次实验中，我们将先实验直方图均衡化技术。这之后，我们会手动实现一个卷积滤波算法，并我各种3x3滤波器进行实验。最后，我们使用Hough变换提取图像中的圆。

1.图像的直方图均衡化

我们将先编写程序实现直方图均衡化。

就一般而言，许多灰度图像的直方图分布并不是均匀的。它们往往存在较亮或较暗的问题，而直方图均衡化可以另图像整体灰度更加平均，因而使亮处/暗处细节得到充分的暴露。

具体而言，我们假设原图像灰度范围为0-L-1, r为原图像灰度级的分布（由经验分布确定）。那么我们的问题变成了需要找到变换T,使得变换后的灰度级随机变量$s = T(r)$的分布服从均匀分布。这一要求可以由以下变换实现：

$$s = T(r) = (L-1)\int_0^r p_r(\omega)d\omega$$

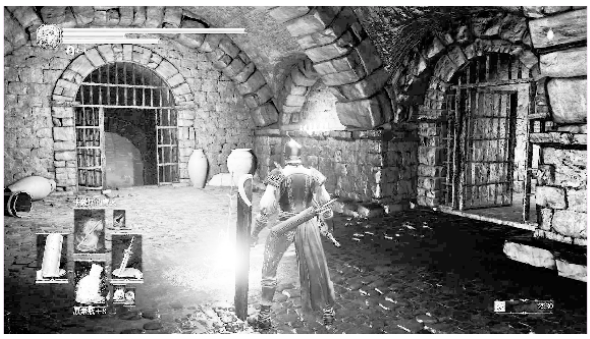
此时，s的分布函数$p_s(s) = p_r(r)\,|\,dr/ds\,|$，而$|\frac{ds}{dr}| = (L-1)\frac{d}{dr}[\int_0^r p_r(\omega)d\omega]$，于是$p_s(s) = p_r(r)\,|\,\frac{1}{(L-1)p_r(r)}\,| = \frac{1}{L-1}$，即s服从均匀分布。在实际操作中，我们用求和代替积分。

于是，我们用HistUni()函数实现直方图均衡化。代码将在附录1中给出。我们用一些图片来测试算法的效果。

```
jail=imread('c:/users/a/desktop/jail.jpg');
jail=rgb2gray(jail);
```

我们可以看到，这幅图像十分昏暗，暗处细节非常不明显，右边铁栅门内核天花板处的轮廓全部隐藏在了阴影中。

```
output=HistUni(jail);
figure();
subplot(2,1,1)
imshow(jail)
subplot(2,1,2)
imshow(output)
```



测试效果：

我们可以看到，图像的整体亮度有了明显的均衡，暗处砖石板的纹路被完全暴露了出来。我们再用题目中的图片进行测试，可以得到以下结果：

两幅图中，我们依然可以看到，原图得到了有效的灰度均衡。

2.边缘检测算子与边缘检测

在这一部分，我们将实现卷积滤波函数，并用边缘检测算子对原图像进行边缘提取。

出于对程序可扩展性的考虑，我们将卷积滤波拆分成两个子程序（见附录2），其中Convolute()函数实现了两个邻域的卷积滤波，而函数fil()则专门用于实现图像的卷积滤波，其接收一副图像和一个滤波器作为参数，并通过调用Convolute()实现全局滤波。

卷积滤波的原理我们将不再赘述。我们将接下来直接测试各个算子的效果：

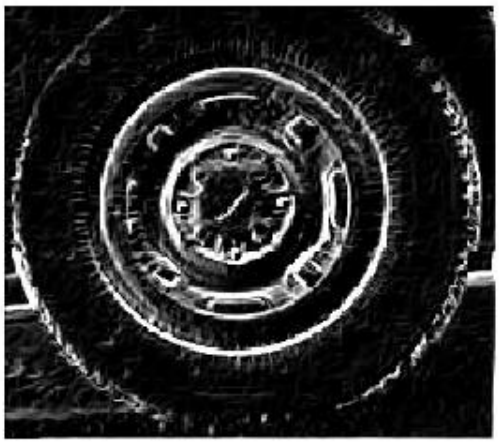
```
%Sobel算子
sobelx=[-1,-2,-1;0,0,0;1,2,1];
sobely=[-1,0,1;-2,0,2;-1,0,1];
%Prewitt算子
px=[-1,-1,-1;0,0,0;1,1,1];
py=[-1,0,1;-1,0,1;-1,0,1];
```

生成滤波器模板后，我们调用之前写好的卷积滤波函数，先实现sobel算子滤波：

```
gx=fil(wheels,sobelx);
gy=fil(wheels,sobely);
```

```
g=sqrt(gx.^2+gy.^2);
```

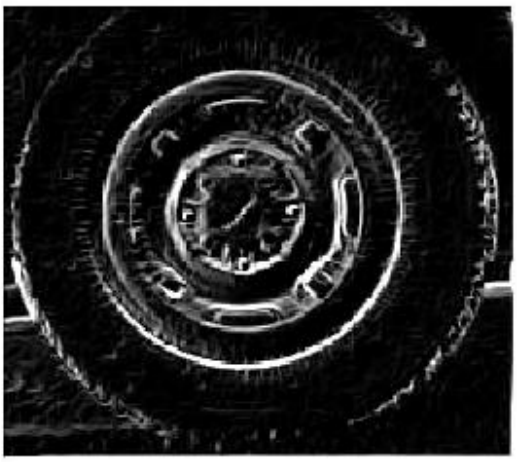
边缘图像与原图的对比：



我们可以看到，提取边缘的效果拔群。我们接下来再用Prewitt算子进行测试：

```
fx=fil(wheels,px);
fy=fil(wheels,py);
f=sqrt(fx.^2+fy.^2);
```

我们再显示原图和提取边缘图的对比：



效果依旧不俗。

3.用Hough变换实现圆检测

在边界提取完成后，一个很自然的想法是如何对边界图像进行图形检测。我们可以考虑使用hough变换来对图像中的圆进行检测。

Hough变换的应用是广泛的。我们考虑图像F上的一个点，若它在某个圆C上，我们可以设出它的参数方程，并将这个点投影到参数空间(a,b,r)上，其中a，b代表圆心坐标，r代表半径。由于具体参数有无穷多个，因此该点再参数空间上的投影是一族三维空间内的同心圆。而在参数空间的无穷多个圆中，任意两个相交的圆代表这两族圆所代表的点有可能在同一圆上。因此，当参数空间中某个点上累计的交点数量超过了某个阈值，则可以认为这些圆族所代表的点落在同一个圆上。

我们用matlab中的霍夫变换函数imfindcircles()来寻找圆：

```
gw=im2bw(g);%二值化边缘图像
figure(3)%显示原图像与检测到的圆的叠加
imshow(wheels)
c=[];
r=[];
```

```
for i=1:10%多次调节检测范围，使imfindcircles()可以找到绝大多数圆
    [center,radius]=imfindcircles(gw,[10*i,10*i+100]);
    c=[c;center];
    r=[r;radius];
end
```

```
警告: You just called IMFINDCIRCLES with a large radius range. Large radius ranges reduce
algorithm accuracy and increase computational time. For high accuracy, relatively small
radius range should be used. A good rule of thumb is to choose the radius range such that
Rmax < 3*Rmin and (Rmax - Rmin) < 100. If you have a large radius range, say [20 100],
consider breaking it up into multiple sets and call IMFINDCIRCLES for each set separately,
like this:

[CENTERS1, RADII1, METRIC1] = IMFINDCIRCLES(A, [20 60]);
[CENTERS2, RADII2, METRIC2] = IMFINDCIRCLES(A, [61 100]);
```


```
警告: You just called IMFINDCIRCLES with a large radius range. Large radius ranges reduce
algorithm accuracy and increase computational time. For high accuracy, relatively small
radius range should be used. A good rule of thumb is to choose the radius range such that
Rmax < 3*Rmin and (Rmax - Rmin) < 100. If you have a large radius range, say [20 100],
consider breaking it up into multiple sets and call IMFINDCIRCLES for each set separately,
like this:

[CENTERS1, RADII1, METRIC1] = IMFINDCIRCLES(A, [20 60]);
[CENTERS2, RADII2, METRIC2] = IMFINDCIRCLES(A, [61 100]);
```



```
警告: You just called IMFINDCIRCLES with a large radius range. Large radius ranges reduce
algorithm accuracy and increase computational time. For high accuracy, relatively small
radius range should be used. A good rule of thumb is to choose the radius range such that
Rmax < 3*Rmin and (Rmax - Rmin) < 100. If you have a large radius range, say [20 100],
consider breaking it up into multiple sets and call IMFINDCIRCLES for each set separately,
like this:

[CENTERS1, RADII1, METRIC1] = IMFINDCIRCLES(A, [20 60]);
[CENTERS2, RADII2, METRIC2] = IMFINDCIRCLES(A, [61 100]);
```

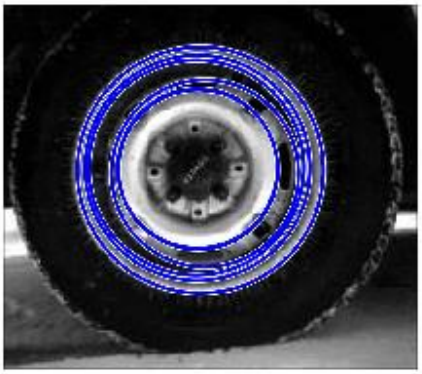


```
警告: You just called IMFINDCIRCLES with a large radius range. Large radius ranges reduce
algorithm accuracy and increase computational time. For high accuracy, relatively small
radius range should be used. A good rule of thumb is to choose the radius range such that
Rmax < 3*Rmin and (Rmax - Rmin) < 100. If you have a large radius range, say [20 100],
consider breaking it up into multiple sets and call IMFINDCIRCLES for each set separately,
like this:

[CENTERS1, RADII1, METRIC1] = IMFINDCIRCLES(A, [20 60]);
[CENTERS2, RADII2, METRIC2] = IMFINDCIRCLES(A, [61 100]);
```


```
viscircles(c,r,'EdgeColor','b');
```

效果如下:



至此，我们成功地用Hough变换完成了圆的识别。

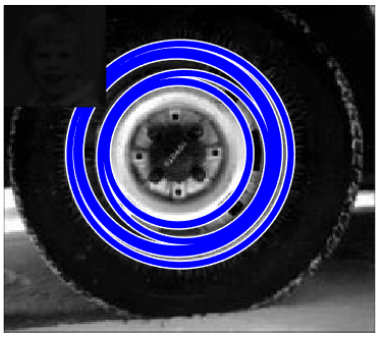
4.APPENDIX

4.1.直方图均衡化：

```
%function output=HistUni(image)

imax=max(max(image));
```

未定义与 'matlab.graphics.primitive.Image' 类型的输入参数相对应的函数 'max'。



```
imin=min(min(image));

[height,width]=size(image);

%构造原灰度的经验分布
probability=zeros(256,1);

for i=1:height
    for j=1:width
        probability(image(i,j)+1)=probability(image(i,j)+1)+1;
    end
end

%归一化
probability=probability/(height*width);

output=zeros(height,width);

%用求和代替积分
for i=1:height
    for j=1:width
    output(i,j)=(255)*(sum(probability(1:image(i,j))));
    end
end

%归一化后输出图像
output=im2double(output);
output=output/255;
```

```
%end
```

4.2.边缘检测

4.2.1邻域卷积函数:

```
%function c=Convolute(x,y)

x=x(:);
y=y(:);
x=flip(x);

c=x'*y;

if c<0
    c=0;
end


%end
```

4.2.2 卷积滤波函数

```
%function output=fil(image,f)

image=im2double(image);

[height,width]=size(image);

output=zeros(height,width);

%将图像嵌入一个周围留有一圈的背景板，方便考虑卷积滤波的边界情况
background=zeros(height+2,width+2);

background(2:height+1,2:width+1)=image;

for i=1:height
    for j=1:width
        %获取各个像素点的小邻域
        neighbour=background(i:i+2,j:j+2);
        %调用Convolute进行滤波
        output(i,j)=Convolute(f,neighbour);

    end
end

%end
```