实验二 图像预处理实验

1913416 阎丝雨

一、实验目的

掌握图像预处理的基本方法和主要思路，编程实现图像的边缘检测、模板匹配和亮度调整等图像预处理方法。

二、实验原理、实验步骤、程序代码、结果显示

（1） 图像的边缘检测

图的边缘是指在局部不连续的特征。边缘检测是为了将其周围像素灰度有阶跃变化的像素检测出来，这些像素组成的集合就是该图像的边缘。比较常用的边缘检测方法就是考察每个像素在某个领域内灰度的变化，然后利用边缘临近一阶或二阶方向导数变化规律检测边缘，即边缘检测局部算法。

- **图像的读取、灰度化**

```
imag = imread('npy.jpg');  %读取图片
imag = rgb2gray(imag);          %转化为灰度图
figure(1),subplot(151),imshow(imag),title('原图');
[high,width] = size(imag);    % 获得图像的高度和宽度
```

- **编程利用一阶微分算子 sobel 算子实现边缘检测**

将图像的每一个点都用 sobel 算子做卷积：一个用来检测垂直边缘，一个用来检测水平边缘，图像的每一个像素的横向及纵向灰度值通过勾股定理结合，来计算该点灰度的大小，如果梯度 G 大于某一阀值　则认为该点(x,y)为边缘点。

```
%%sobel边缘检测
F2 = double(imag);
U = double(imag);
uSobel = imag;
T=250;
for i = 2:high - 1   %sobel边缘检测
    for j = 2:width - 1
        Gx = (U(i+1,j-1) + 2*U(i+1,j) + F2(i+1,j+1)) - (U(i-1,j-1) + 2*U(i-1,j) + F2(i-1,j+1));
        Gy = (U(i-1,j+1) + 2*U(i,j+1) + F2(i+1,j+1)) - (U(i-1,j-1) + 2*U(i,j-1) + F2(i+1,j-1));
        uSobel(i,j) = sqrt(Gx^2 + Gy^2);
        if uSobel(i,j)<T
            uSobel(i,j)=0;
        else
            uSobel(i,j)=255;
        end
    end
end
subplot(152);imshow(im2uint8(uSobel)),title('sobel算子边缘检测图像');
```

- **编程利用二阶微分算子拉普拉斯算子实现边缘检测以及图像锐化**

边缘检测：一阶微分法能够用来检测边缘是否存在。那么二阶微分法，也就是拉普拉斯算子就可以增强图像的细节，找到图像的边缘。

锐化：拉普拉斯锐化图像是根据图像某个像素的周围像素到此像素的突变程度有关，也就是说它的依据是图像像素的变化程度。当邻域中心像素灰度低于它所在的领域内其它像素的平均灰度时，此中心像素的灰度应被进一步降低，当邻域中心像素灰度高于它所在的邻域内其它像素的平均灰度时，此中心像素的灰度应被进一步提高，以此实现图像的锐化处理。

图像锐化的实质：

锐化图像 = 原图像 + 检测出的边缘

注：有时候会把噪音增强，应当锐化前对图像进行平滑处理。

```
%%拉普拉斯边缘检测
w=fspecial('gaussian',[5 5]);
I=imfilter(imag,w,'replicate');%平滑

[m,n]=size(I);
Ig=I;
for i=2:m-1
    for j=2:n-1
        Ig(i,j)=(1+4).*I(i,j)-(I(i+1,j)+I(i-1,j)+I(i,j+1)+I(i,j-1));
        %Ig(i,j)=sum(sum(Ig));
    end
end
subplot(153),imshow(uint8(Ig)),title('拉普拉斯边缘检测图像');
Ig_=Ig+I;
subplot(154),imshow(uint8(Ig_)),title('拉普拉斯锐化后的图像');
```

- **Canny 边缘检测**

Canny 算子是一个具有滤波，增强，检测的多阶段的优化算子，在进行处理前，Canny 算子先利用高斯平滑滤波器来平滑图像以除去噪声，Canny 分割算法采用一阶偏导的有限差分来计算梯度幅值和方向，在处理过程中，Canny 算子还将经过一个非极大值抑制的过程，最后 Canny 算子还采用两个阈值来连接边缘。Canny 方法不容易受噪声干扰，能够检测到真正的弱边缘。

```
h=fspecial('gaussian',5);%高斯滤波
I2=imfilter(imag,h,'replicate');
GcannyBW=edge(I2,'canny');%高斯滤波后使用Canny算子进行边缘检测
subplot(155);imshow(im2uint8(GcannyBW)),title('canny算子边缘检测');
```

- **运行结果**

- **结果分析**

Canny 算子的边缘检测结果最优,能检测到弱边缘并将边缘连接，相比之下 sobel 算子的边缘检测就难以检测出弱边缘，拉普拉斯算子的程度更低。但拉普拉斯算子检测出边缘后与原图像相合得到锐化图像。

(2) 对比图像复原与图像增强

图像增强是对图像的某些特征，如边缘、轮廓、对比度等进行强调或锐化，以便于显示、观察或进一步分析与处理。通过对图像的特定加工，将被处理的图像转化为对具体应用来说视觉质量和效果更“好”或更“有用”的图像。

图像复原是图像处理重要的研究领域。在成像过程中，由于成像系统各种因素的影响，可能使获得的图像不是真实景物的完善影像。图像在形成、传播和保存过程中使图像质量下降的过程，称为图像退化。图像复原就是重建退化的图像，使其最大限度恢复景物原貌的处理。

图像复原的概念与图像增强相似。但图像增强可以针对本来完善的图像，经过某一处理，使其适合于某种特定的应用，是一个主观的过程。图像复原的目的也是改善图像质量，但图像复原更偏向于利用退化过程的先验知识使已被退化的图像恢复本来面目，更多的是一个客观过程。

- **加入运动模糊**

```
%产生运动模糊
len=20;%设置运动位移为35个像素
theta=25;%设置运动角度为45度
psf=fspecial('motion',len,theta);%建立二维运动仿真滤波器psf
mf=imfilter(imag,psf,'circular','conv');%用psf产生退化图像
figure(2)
subplot(131),imshow(mf),title('加入运动模糊的图像');
```

- **图像增强**

利用拉普拉斯算子对图像进行锐化，期以增强边界来消除模糊。

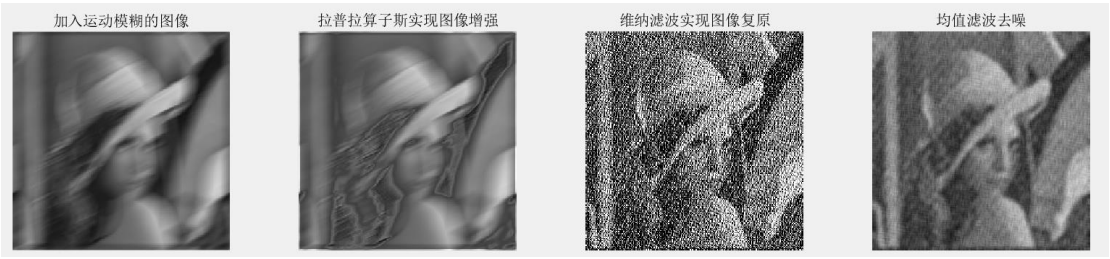
```
%%sobel边缘检测
F2 = double(imag);
U = double(imag);
uSobel = imag;
T=250;
for i = 2:high - 1   %sobel边缘检测
    for j = 2:width - 1
        Gx = (U(i+1,j-1) + 2*U(i+1,j) + F2(i+1,j+1)) - (U(i-1,j-1) + 2*U(i-1,j) + F2(i-1,j+1));
        Gy = (U(i-1,j+1) + 2*U(i,j+1) + F2(i+1,j+1)) - (U(i-1,j-1) + 2*U(i,j-1) + F2(i+1,j-1));
        uSobel(i,j) = sqrt(Gx^2 + Gy^2);
        if uSobel(i,j)<T
            uSobel(i,j)=0;
        else
            uSobel(i,j)=255;
        end
    end
end
subplot(152);imshow(im2uint8(uSobel)),title('sobel算子边缘检测图像');
```

- **图像复原**

选择改进的逆滤波算法-维纳滤波，进行图像复原。得到初步结果噪声过多，于是又进行了均值滤波去噪。

```
%维纳滤波实现图像复原
mnrl=deconvwnr(mf,psf,0);
subplot(143),imshow(uint8(mnrl)),title('维纳滤波实现图像复原');
filt = 1/25 * ones(5);
C = imfilter(mnrl,filt,'symmetric','same');
subplot(144),imshow(uint8(C)),title('均值滤波去噪');
```

- **运行结果**



- **结果分析**

图像增强是处于主观的目的强盗图像的某一特性从而达到目的，而图像复原是针对图像整体，对图像进行复原，回到最初的状态，图像复原比图像增强更适用于运动模糊的去除，此外，尽管维纳滤波具有去噪的作用，但处理后的图像仍具噪声，因此应当再进行去噪处理。

（3）图片的亮度和对比度调整

对图像各像素点(i, j)做以下运算：

f (i , j) = a * f (i , j) + b f(i, j) = a*f(i, j) + bf(i,j)=a*f(i,j)+b

则 a 改变的是对比度， b 改变的是亮度

- **改变图像对比度与亮度的函数**

```
function [imag_] = adjust(imag,a,b)
%a调整对比度，b调整亮度
[w, h] = size(imag);
for i = 1:w
    for j = 1:h
        imag_(i, j) = round(imag(i, j).*(1+a)+b);
        if (imag_(i, j) > 255)
            imag_(i, j) = 255;
        end
    end
end
end
```

- **改变图像对比度与亮度**

利用保持变量唯一的方法，当改变对比度时，保持亮度不变。改变亮度时，保持对比度不变。

```
%改变对比度与亮度
figure(3);
imag_1=adjust(imag,0.5,0);
subplot(231),imshow(uint8(imag_1)),title('对比度为0.5');

imag_2=adjust(imag,1,0);
subplot(232),imshow(uint8(imag_2)),title('对比度为1');

imag_3=adjust(imag,2,0);
subplot(233),imshow(uint8(imag_3)),title('对比度为2');

%改变亮度
imag_4=adjust(imag,0.6,20);
subplot(234),imshow(uint8(imag_4)),title('亮度为20');

imag_5=adjust(imag,0.6,60);
subplot(235),imshow(uint8(imag_5)),title('亮度为60');

imag_6=adjust(imag,0.6,100);
subplot(236),imshow(uint8(imag_6)),title('亮度为100');
```

- **运行结果**

- **结果分析**

由公式可知，对比度与亮度辅车相依，在改变其中一个的时候，另一个也会随之变换，从运行结果中也可以看出。但在调整对比度时主要仍为图像的对比度参数发生变化。若想得到更明显的变化，可以采用伽马变换的方法。