**代码**

**图像变换：**

**Mygui**

function pushbutton1\_Callback(~, ~, handles)

global im %定义一个全局变量im

global im2

[filename,pathname]=...

uigetfile({'\*.\*';'\*.bmp';'\*.tif';'\*.png'},'select picture'); %选择图片路径

str=[pathname filename]; %合成路径+文件名

im=imread(str); %读取图片

im2=im;

axes(handles.axes1); %使用第一个axes

imshow(im); %显示图片

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(~, ~, handles)

global BW

set(handles.axes2,'HandleVisibility','ON');

axes(handles.axes2);

[filename,pathname]=uiputfile({'\*.jpg';'\*.bmp';'\*.tif';'\*.\*'},'save image as');

file=strcat(pathname,filename);

BW=getimage(gca);

imwrite(BW,file);

set(handles.axes2,'HandleVisibility','Off');

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(~, ~, ~)

close(gcf) %关闭当前Figure窗口句柄

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(~, ~, handles)

global im

global BW %定义全局变量

axes(handles.axes2);

I1=double(im);

I2=fft2(I1);

I3=log(abs(I2));

BW=I3;

imshow(BW,[]);

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(~, ~, handles)

global im

global BW %定义全局变量

axes(handles.axes2);

I1=double(im);

I2=dct2(I1);

I3=log(abs(I2));

BW=I3;

imshow(BW);

% --- Executes on button press in pushbutton6.

function pushbutton6\_Callback(~, ~, handles)%原图-灰度

global im

global BW %定义全局变量

axes(handles.axes2);

BW=rgb2gray(im);

im=BW;

imshow(BW);

function pushbutton7\_Callback(~, ~, handles)%原图-二值

global im

global BW %定义全局变量

axes(handles.axes2);

BW=imbinarize(im);

im=BW;

imshow(BW);

% --- Executes on button press in pushbutton8.

function pushbutton8\_Callback(~, ~, handles)

global im

global BW %定义全局变量

I=im2double(im);

J=imnoise(I,'gaussian');

BW=J;

axes(handles.axes2);

imshow(BW);

% --- If Enable == 'on', executes on mouse press in 5 pixel border.

% --- Otherwise, executes on mouse press in 5 pixel border or over text4.

function text4\_ButtonDownFcn(~, ~,~)

% hObject handle to text4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton9.

function pushbutton9\_Callback(~, ~, handles)

global im

global BW %定义全局变量

I=im2double(im);

J=imnoise(I,'salt & pepper');

BW=J;

axes(handles.axes2);

imshow(BW);

% --- Executes on button press in pushbutton10.

function pushbutton10\_Callback(~, ~, ~)

global im

I=im2double(im);

I=double(I);

I5=2\*I-medfilt2(I);

imshow(I5,[]);%原图像－中值滤波图像

% --- Executes on button press in pushbutton11.

function pushbutton11\_Callback(~, ~, handles)

global im

axes(handles.axes2);

I=double(im);

J=medfilt2(I,[3,3]);

imshow(uint8(J));%邻域中值滤波效果')

% --- Executes on button press in pushbutton12.

function pushbutton12\_Callback(~, ~, handles)

global im

I=im2double(im);

h=fspecial('average');

I2= 2\*I-filter2(h,I);

axes(handles.axes2);

imshow(I2,[]);%原图像－均值滤波图像

% --- Executes on button press in pushbutton15.

function pushbutton15\_Callback(~, ~, handles)

% hObject handle to pushbutton15 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

axes( handles.axes2);

I=(handles.axes1);

if ndims(I)

m=fft2(I(:,:,1));

J=fftshift(m);

imshow(log(abs(J)),[]);

else

m=fft2(I);

J=fftshift(m);

imshow(log(abs(J)),[]);

end

% --- Executes on button press in pushbutton16.

function pushbutton16\_Callback(~, ~, handles)

global BW %定义全局变量

global im

axes(handles.axes2);

X=double(im);

A=fft2(X);

X2=ifft2(A);

X2=abs(X2);

BW=uint8(X2);

imshow(BW);

% --- Executes on button press in pushbutton17.

function pushbutton17\_Callback(~, ~, handles)

global im

axes(handles.axes2);

a=double(im);

A=dct2(a);

a2=idct2(A);

imshow(abs(a2),[]);

% --- Executes on button press in pushbutton18.

function pushbutton18\_Callback(~, ~, handles)

global im

axes(handles.axes2);

I=double(im);

h=fspecial('average',5);

J1=filter2(h,I,'valid');

imshow(uint8(J1));%filter2均值滤波效果');

% --- Executes on button press in gui2.

function gui2\_Callback(~, ~, ~)

global im

f=double(im);

set(gui2,'Visible','on'); %想打开的界面名称——gui.m和gui.fig

% --- Executes on button press in pushbutton20.

function pushbutton20\_Callback(~, ~, handles)

global im

axes(handles.axes2);

X=double(im);

A=fft2(X);

A1=fftshift(A);

%A为复数，所以要求其模，A的元素极大且差极大，所以取对数，取对数结果为浮点

%数且范围不定，所以利用imshow函数参数通过空的中括号[]将范围扩展至[0,255]

imshow(log(abs(A1)),[]);%傅里叶反变换频谱图

% --- Executes on button press in pushbutton21.

function pushbutton21\_Callback(~, ~, handles)

%傅立叶变换的平移特性

global im

axes(handles.axes2);

I=double(im);

[r,c]=size(I); %计算灰度图的大小，r表示行，c表示列，即通过size函数将灰度图I的行数存在矩阵的r中，列数存在矩阵的c中，这样就知道灰度图的大小是r×c

dst=zeros(r,c); %建立r×c的0矩阵（平移结果矩阵），初始化为零（黑色）

dx=80; %平移的x方向的距离，这里是竖直方向

dy=80; %平移的y方向的距离，这里是水平方向

tras=[1 0 dx;0 1 dy;0 0 1]; %平移变换矩阵

for i=1:r

for j=1:c

temp=[i;j;1]; %灰度图I要平移变换的点，这里用矩阵表示

temp=tras\*temp; %矩阵相乘，得到三行一列的矩阵temp，即平移后的矩阵

x=temp(1,1); %把矩阵temp的第一行第一列的元素给x

y=temp(2,1); %把矩阵temp的第二行第一列的元素给y

if(x>=1&&x<=r)&&(y>=1&&y<=c)%判断所变换后得到的点是否越界

dst(x,y)=I(i,j); %得到平移结果矩阵，点(x,y)是由点(i,j)平移而来的，有对应关系

end

end

end

A=fft2(dst);

A=fftshift(A);

imshow(log(abs(A)),[]);%title('平移后频谱);

% --- Executes on button press in pushbutton22.

function pushbutton22\_Callback(~, ~, handles)

%傅立叶变换的平移特性

global im

axes(handles.axes2);

I=double(im);

[r,c]=size(I); %计算灰度图的大小，r表示行，c表示列，即通过size函数将灰度图I的行数存在矩阵的r中，列数存在矩阵的c中，这样就知道灰度图的大小是r×c

dst=zeros(r,c); %建立r×c的0矩阵（平移结果矩阵），初始化为零（黑色）

dx=80; %平移的x方向的距离，这里是竖直方向

dy=80; %平移的y方向的距离，这里是水平方向

tras=[1 0 dx;0 1 dy;0 0 1]; %平移变换矩阵

for i=1:r

for j=1:c

temp=[i;j;1]; %灰度图I要平移变换的点，这里用矩阵表示

temp=tras\*temp; %矩阵相乘，得到三行一列的矩阵temp，即平移后的矩阵

x=temp(1,1); %把矩阵temp的第一行第一列的元素给x

y=temp(2,1); %把矩阵temp的第二行第一列的元素给y

if(x>=1&&x<=r)&&(y>=1&&y<=c)%判断所变换后得到的点是否越界

dst(x,y)=I(i,j); %得到平移结果矩阵，点(x,y)是由点(i,j)平移而来的，有对应关系

end

end

end

imshow(uint8(dst))%title('平移后');

% --- Executes on button press in pushbutton23.

function pushbutton23\_Callback(~, ~, handles)

%傅立叶变换的旋转特性

global im

axes(handles.axes2);

a=double(im);

A=fft2(a);

A=fftshift(A);

%对频谱图进行逆时针旋转90度

A2=imrotate(A,90,'bilinear');

imshow(log(abs(A2)),[]);% title旋转后的频谱图

% --- Executes on button press in pushbutton24.

function pushbutton24\_Callback(~, ~, handles)

%傅立叶变换的旋转特性

global im

axes(handles.axes2);

a=double(im);

A=fft2(a);

A=fftshift(A);

%对频谱图进行逆时针旋转90度

prompt={'度数：'};

def={'90'};

answer=inputdlg(prompt,'请输入：',2,def);

if ~isempty(answer)

a = str2double(answer);

A2=imrotate(A,a,'bilinear');

a2=fft2(fftshift(A2));

imshow(abs(a2),[]);% 旋转后的图

end

function pushbutton25\_Callback(~, ~, handles)

global im

global BW %定义全局变量

I=im2double(im);

J=imnoise(I,'poisson');

BW=J;

axes(handles.axes2);

imshow(BW);%泊松噪声

% --- Executes on button press in pushbutton26.

function pushbutton26\_Callback(~, ~, handles)

global im

global BW %定义全局变量

I=im2double(im);

J=imnoise(I,'speckle',0.1);

BW=J;

axes(handles.axes2);

imshow(BW);%随机噪声

**Gui2**

function pushbutton1\_Callback(~, ~, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global im

f=double(im);

F=fftshift(fft2(f));%对图像进行傅立叶变换并将频域中心偏移到中点

d0=20; %截止频率

[m,n]=size(f);

%理想低通滤波

%h=0;

for u=1:m

for v=1:n

if sqrt((u-m/2)\*(u-m/2)+(v-n/2)\*(v-n/2))<=d0

h=1;

else

h=0;

end

FH(u,v)=F(u,v)\*h;

end

end

fh=ifft2(FH);

%巴特沃斯低通滤波

nb=1;

for u=1:m

for v=1:n

hb=1/(1+(sqrt(2)-1)\*(sqrt((u-m/2)\*(u-m/2)+(v-n/2)\*(v-n/2))/d0)^(2\*nb));

FHB(u,v)=F(u,v)\*hb;

end

end

fhb=ifft2(FHB);

%指数低通滤波

ne=1;

for u=1:m

for v=1:n

he=exp(-((sqrt((u-m/2)\*(u-m/2)+(v-n/2)\*(v-n/2))/d0)^ne/sqrt(2)));

FHE(u,v)=F(u,v)\*he;

end

end

fhe=ifft2(FHE);

axes(handles.axes1);imshow(uint8(abs(fh)));%理想低通滤波')

axes(handles.axes2);imshow(uint8(abs(fhb)));%'巴特沃斯低通滤波')

axes(handles.axes3);imshow(uint8(abs(fhe)));%'指数低通滤波')

axes(handles.axes4);imshow(log(abs(FH)));%理想低通滤波频谱图');

axes(handles.axes5);imshow(log(abs(FHB)));%巴特沃斯低通滤波频谱图')

axes(handles.axes6);imshow(log(abs(FHE)));%指数低通滤波频谱图')

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(~, ~, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global im

f=double(im);

F=fftshift(fft2(f));%对图像进行傅立叶变换并将频域中心偏移到中点

d0=10; %截止频率

[m,n]=size(f);

%理想高通滤波

h=0;

for u=1:m

for v=1:n

if sqrt((u-m/2)\*(u-m/2)+(v-n/2)\*(v-n/2))<=d0

h=0;

else

h=1;

end

FH(u,v)=F(u,v)\*h;

end

end

fh=ifft2(FH);

%巴特沃斯高通滤波

nb=1;

for u=1:m

for v=1:n

hb=1/(1+(sqrt(2)-1)\*(d0/sqrt((u-m/2)\*(u-m/2)+(v-n/2)\*(v-n/2)))^(2\*nb));

FHB(u,v)=F(u,v)\*hb;

end

end

fhb=ifft2(FHB);

%指数高通滤波

ne=1;

for u=1:m

for v=1:n

he=exp(-((d0/sqrt((u-m/2)\*(u-m/2)+(v-n/2)\*(v-n/2)))^ne/sqrt(2)));

FHE(u,v)=F(u,v)\*he;

end

end

fhe=ifft2(FHE);

axes(handles.axes1);imshow(uint8(abs(fh)),[]);%理想高通滤波

axes(handles.axes2);imshow(uint8(abs(fhb)),[]);%巴特沃斯高通滤波')

axes(handles.axes3);imshow(uint8(abs(fhe)),[]);%指数高通滤波')

axes(handles.axes4);imshow(log(abs(FH)),[]);%理想高通滤波频谱图');

axes(handles.axes5);imshow(log(abs(FHB)),[]);%巴特沃斯高通滤波频谱图')

axes(handles.axes6);imshow(log(abs(FHE)),[]);%指数高通滤波频谱图

**图像分割：**

% 选择图片

function pushbutton1\_Callback(hObject, eventdata, handles)

global im;

[filename, pathname]=uigetfile({'\*.jpg';'\*.bmp';'\*.\*';'\*.png'},'打开图片');

str=[pathname filename];

im=imread(str);

axes(handles.axes1);

imshow(im);

% Roberts边缘检测

function pushbutton2\_Callback(hObject, eventdata, handles)

global im

global BW %定义全局变量

roberts1=[1,0;0,-1];

roberts2=[0,1;-1,0];

f1=imfilter(im,roberts1);

f2=imfilter(im,roberts2);

BW=max(abs(f1),abs(f2));

axes(handles.axes2); %使用第二个axes

imshow(BW,[]),title('Roberts边缘检测图像');

% Laplacian边缘检测

function pushbutton3\_Callback(hObject, eventdata, handles)

global im

global BW %定义全局变量

laplacian1=[0,-1,0;-1,4,-1;0,-1,0];

laplacian2=[-1,-1,-1;-1,8,-1;-1,-1,-1];

f1=imfilter(im,laplacian1);

f2=imfilter(im,laplacian2);

BW=max(abs(f1),abs(f2));

axes(handles.axes2); %使用第二个axes

imshow(BW,[]),title('Laplacian边缘检测图像');

%Prewitt边缘检测

function pushbutton4\_Callback(hObject, eventdata, handles)

global im

global BW %定义全局变量

prewitt1=[-1,0,1;-1,0,1;-1,0,1];

prewitt2=[1,1,1;0,0,0;-1,-1,-1];

f1=imfilter(im,prewitt1);

f2=imfilter(im,prewitt2);

BW=max(abs(f1),abs(f2));

axes(handles.axes2); %使用第二个axes

imshow(BW,[]),title('Prewitt边缘检测图像');

% Sobel边缘检测

function pushbutton5\_Callback(hObject, eventdata, handles)

global im

global BW %定义全局变量

sobel1=[-1,0,1;-2,0,2;-1,0,1];

sobel2=[1,2,1;0,0,0;-1,-2,-1];

f1=imfilter(im,sobel1);

f2=imfilter(im,sobel2);

BW=max(abs(f1),abs(f2));

axes(handles.axes2); %使用第二个axes

imshow(BW,[]),title('Sobel边缘检测图像');

% 退出

function pushbutton6\_Callback(hObject, eventdata, handles)

clc,clear,close all;

% 直线检测

function pushbutton7\_Callback(hObject, eventdata, handles)

global im f

thresh=[0.01,0.17];

sigma=2;

f=edge(double(im),'canny',thresh,sigma);

[H,theta,rho]=hough(f,'RhoResolution',0.5);

peak=houghpeaks(H,5); %用houghpeaks函数找到5个看起来很明显的Hough变换峰值

hold on

lines=houghlines(f,theta,rho,peak);

axes(handles.axes2); %使用第二个axes

imshow(f,[]),title('hough变换检测到的线段')

hold on

max\_len = 0;

%在原图像中，逐一标出houghlines找到的线段

for k = 1:length(lines)

xy = [lines(k).point1; lines(k).point2];

% point1,point2分别为线段的两个端点

plot(xy(:,1),xy(:,2),'LineWidth',1,'Color','green');

% 把线段的两个端点标出

plot(xy(1,1),xy(1,2),'x','LineWidth',1,'Color','yellow');

plot(xy(2,1),xy(2,2),'x','LineWidth',1,'Color','red');

len = norm(lines(k).point1 - lines(k).point2);

if ( len > max\_len)

% 如果该线段为最长线段，将该线段保存在xy\_long中

max\_len = len;

xy\_long = xy;

end

end

% 用其他颜色显示最长线段

plot(xy\_long(:,1),xy\_long(:,2),'LineWidth',1,'Color','cyan');

%阈值分割

function pushbutton8\_Callback(hObject, eventdata, handles)

global im g

%f=imread('rice.png');

%T=graythresh(im)

im=medfilt2(im); % 对原图像进行中值滤波

p=zeros(1,256);%灰度概率数组

[m,n]=size(im);

%计算灰度概率即直方图

for i=1:m

for j=1:n

p(im(i,j)+1)=p(im(i,j)+1)+1;

end

end

p=p/m/n;

u=sum(im(:))/m/n; % 计算图像的灰度平均值

wT=0; % wT表示小于阈值T的灰度概率

uT=0; % uT表示小于阈值T的灰度均值

T=0; % T表示最大类间方差阈值

max\_d2T=0; % max\_d2T 表示最大类间方差

for i=1:255

wT=wT+p(i+1); % 计算灰度级i为阈值的灰度概率

uT=uT+i\*p(i+1); % 计算灰度级i为阈值的灰度均值

d2T=(u\*wT-uT)\*(u\*wT-uT)/wT/(1-wT); % 计算灰度级i为阈值的类间方差

if max\_d2T<d2T

max\_d2T=d2T;

T=i;

end

end

% 用阈值T对图像进行分割

for i=1:m

for j=1:n

g(i,j)=im(i,j)>=T;

end

end

axes(handles.axes2); %使用第二个axes

imshow(g),title('阈值分割结果')

% 区域生长

function pushbutton9\_Callback(hObject, eventdata, handles)

% 定义原图像I，分割图像J，图像大小m，n及像素相似灰度差阈值d为全局变量

global im J m n d i

%I=imread('saturn.png');

im=rgb2gray(im);

im=imresize(im,[60,60]);

set(0,'recursionlimit',4000); %将matlab系统迭代上限设置为4000

im=double(im);

figure,subplot(2,2,1),imshow(uint8(im))

J=zeros(size(im));

[m,n]=size(im);

[sy,sx]=ginput(1); %选择目标种子点

sx=uint8(sx); sy=uint8(sy);

d=30; % 相似性准则为 灰度查小于d

J(sx,sy)=1;

% 调用种子生长递归函数

i=1;

seedsegment(sx,sy);

J1=im.\*J;

hold on

plot(sx,sy,'gs','linewidth',1),title('原图像及种子点位置');

subplot(2,2,2),imshow(J,[]);title('区域生长结果')

subplot(2,2,3),imshow(J1,[]),title('区域生长分割结果')

axes(handles.axes2); %使用第二个axes

imshow(J1,[]),title('区域生长分割结果')

% 四邻域种子生长函数

function seedsegment(x,y)

global I J m n d i

i=i+1;

if x<m % 向右生长

% 如果该像素不属于内部像素且满足生长准则，即灰度绝对差小于d

if J(x+1,y)==0 && abs(I(x+1,y)-I(x,y))<=d

J(x+1,y)=1;

% 继续向右生长

seedsegment(x+1,y);

end

end

if x>1 % 向左生长

if abs(I(x-1,y)-I(x,y))<=d && J(x-1,y)==0

J(x-1,y)=1;

seedsegment(x-1,y);

end

end

if y<n % 向下生长

if J(x,y+1)==0 && abs(I(x,y+1)-I(x,y))<=d

J(x,y+1)=1;

seedsegment(x,y+1);

end

end

if y>1 % 向上生长

if J(x,y-1)==0 && abs(I(x,y-1)-I(x,y))<=d

J(x,y-1)=1;

seedsegment(x,y-1);

end

end