1、编程实现基于区域生长法分割、区域分裂与合并法分割、迭代阈值法分割、均匀性度量法分割、聚类法分割、大津法分割等图像分割算法，并进行仿真结果分析。要求创建用户交互界面，能够实现图像读取、分割过程、分割结果显示等。（自选灰度或彩色图像）

(1) 结果分析

区域生长法计算简单，对于较均匀的连通目标有较好的分割效果。它的缺点是需要人为确定种子点，对噪声敏感。

区域分裂合并法对复杂图像的分割效果较好，但算法较复杂，计算量大，分裂还可能破坏区域的边界。

阈值分割的优点是计算简单、运算效率较高、速度快。阈值的选择需要根据具体问题来确定，一般通过实验来确定。对于给定的图像，可以通过分析直方图的方法确定最佳的阈值，当直方图明显呈现双峰情况时，可以选择两个峰值的中点作为最佳阈值。

均匀性度量法可以有效的将图像目标与背景分割开，以均值或方差作为均匀性度量指标。

聚类法收敛速度快，聚类效果较优。只对于明显区分的类效果好，初始聚类点选择不一样会导致不一样的结果，K 的数目是人为设定的，对噪声敏感，孤立点结果影响大。

大津法计算简单，不受图像亮度和对比度的影响，在图像处理上得到了广泛的应用。它按图像的灰度特性，将图像分成背景和前景两部分。方差是灰度分布均匀性的一种度量，背景和前景之间的类间方差越大，说明构成图像的两部分的差别越大，当部分前景错分为背景或部分背景错分为前景都会导致两部分差别变小。

(2) 结果图

1) 区域生长法分割

2) 区域分裂与合并法分割

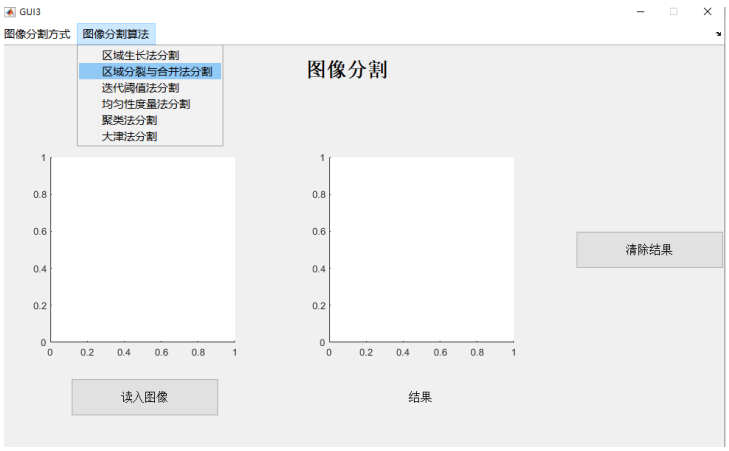
3） 迭代阈值法分割

4） 均匀性度量法分割

5） 聚类法分割

6） 大津法分割

(3) 交互界面



(4) 编程代码

```
function varargout = GUI3(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI3_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function GUI3_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = GUI3_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

%读入图像
function pushbutton1_Callback(hObject, eventdata, handles)
[filename,pathname,filterindex]=...
uigetfile({'*.*';'*.bmp';'*.tif';'*.png';'*.jpg';'*.jpeg'},'select picture');
str=[pathname filename];
s=str;
handles.filebig=filterindex;
if filterindex==0
return
else
im1=imread(str);
end
axes(handles.axes1);
imshow(im1);
handles.img=im1;
guidata(hObject,handles);

%清除界面
function pushbutton2_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla reset;
axes(handles.axes2);
cla reset;

%区域生长
function Untitled_16_Callback(hObject, eventdata, handles)
I = getimage(gca);
seed=[100,220];
thresh=16;
A=rgb2gray(I);
A=imadjust(A,[min(min(double(A)))/255,max(max(double(A)))/255],[]);
A=double(A);
B=A;
[r,c]=size(B);
n=r*c;
pixel_seed=A(seed(1),seed(2));
q=[seed(1) seed(2)];
top=1;
M=zeros(r,c);
M(seed(1),seed(2))=1;
count=1;
while top~=0
    r1=q(1,1);
    c1=q(1,2);
    p=A(r1,c1);
    dge=0;
    for i=-1:1
        for j=-1:1
            if r1+i<=r&&r1+i>0&&c1+j<=c&&c1+j>0
                if abs(A(r1+i,c1+j)-p)<=thresh&&M(r1+i,c1+j)~=1
                    top=top+1;
                    q(top,:)=[r1+i c1+j];
                    M(r1+i,c1+j)=1;
                    count=count+1;
                    B(r1+i,c1+j)=1;
```

```
            end
            if M(r1+i,c1+j)==0
                dge=1;
        else
            dge=1;
        end
    end
end
if dge~=1
    B(r1,c1)=A(seed(1),seed(2));
end
if count >=n
    top =1;
end
q=q(2:top,:);
top=top-1;
end
axes(handles.axes2);
imshow(B,[]);
handles.img=B,[];
guidata(hObject,handles);

%区域分裂与合并法
function Untitled_17_Callback(hObject, eventdata, handles)
I1 = getimage(gca);
S=qtdecomp(I1,0.25);
I2=full(S);
axes(handles.axes2);
imshow(I2);
handles.img=I2;
guidata(hObject,handles);

%迭代阈值法分割
function Untitled_18_Callback(hObject, eventdata, handles)
I = getimage(gca);
I=im2double(I);
level=0.1;
max_I=max(I(:));
min_I=min(I(:));
T1=1/2*(max_I+min_I);
[M,N]=size(I);
A_number=0;
B_number=0;
A_all=0;
B_all=0;
for i=1:M
    for j=1:N
        if(I(i,j)>=T1)
            A_number=A_number+1;
            A_all=A_all+I(i,j);
        else if(I(i,j)<T1)
             B_number=B_number+1;
             B_all=B_all+I(i,j);
            end
        end
    end
    A_ave= A_all/A_number;
    B_ave= B_all/B_number;
    T2=1/2*( A_ave+ B_ave );
end
T2*255
J=im2bw(I,T2);
axes(handles.axes2);
imshow(J);
handles.img=J;
guidata(hObject,handles);

%均匀性度量法
```

```
function Untitled_19_Callback(hObject, eventdata, handles)
I = getimage(gca);
I=rgb2gray(I);
minValue=min(min(I));
maxValue=max(max(I));
[row,col]=size(I);
Th=minValue+1;
perfactValue=10000000000;

for m=1:254
    k1=1;
    k2=1;
    for i=1:row
        for j=1:col
            if I(i,j)<m
                C1(1,k1)=I(i,j);k1=k1+1;
            else
                C2(1,k2)=I(i,j);k2=k2+1;
            end
        end
    end
    average1=mean(C1);
    variance1=0;
    for i=1:k1-1
        variance1=variance1+(C1(1,i)-average1)^2;
    end
    variance1 = variance1/(k1-1);
    p1=(k1-1)/(row*col);
    average2=mean(C2);
    variance2=0;
    for i=1:k2-1
        variance2=variance2+(C2(1,i)-average2)^2;
    end
    p2=(k2-1)/(row*col);
    variance2 = variance2/(k2-1);
    newValue=p1*variance1+p2*variance2;
    if (newValue<perfactValue)
        Th=m;
        perfactValue=newValue;
    end
end
for i=1:row
    for j=1:col
        if I(i,j) >= Th
            G(i,j)=255;
        else
            G(i,j)=0;
        end
    end
end
axes(handles.axes2);
imshow(G);
handles.img=G;
guidata(hObject,handles);

%聚类法分割
function Untitled_20_Callback(hObject, eventdata, handles)
im = getimage(gca);
    cform = makecform('srgb2lab');
    lab = applycform(im,cform);
    ab = double(lab(:,:,2:3));
    nrows = size(lab,1); ncols = size(lab,2);
    X = reshape(ab,nrows*ncols,2)';
    k = 5;
    max_iter = 100;
    [centroids, labels] = run_kmeans(X, k, max_iter);
    hold on; scatter(centroids(1,:),centroids(2,:), 60,'r','filled')
    hold on; scatter(centroids(1,:),centroids(2,:),30,'g','filled')
```

```
    box on; hold off;
    pixel_labels = reshape(labels,nrows,ncols);
    rgb_labels = label2rgb(pixel_labels);
    axes(handles.axes2);
imshow(rgb_labels);
handles.img=rgb_labels;
guidata(hObject,handles);
    function [centroids, labels] = run_kmeans(X, k, max_iter)
    centroids = X(:,1+round(rand*(size(X,2)-1)));
    labels = ones(1,size(X,2));
    for i = 2:k
        D = X-centroids(:,labels);
        D = cumsum(sqrt(dot(D,D,1)));
        if D(end) == 0, centroids(:,i:k) = X(:,ones(1,k-i+1)); return; end
        centroids(:,i) = X(:,find(rand < D/D(end),1));
        [~,labels] =
max(bsxfun(@minus,2*real(centroids'*X),dot(centroids,centroids,1).'));
      end
      for iter = 1:max_iter
        for i = 1:k, l = labels==i; centroids(:,i) = sum(X(:,l),2)/sum(l); end
        [~,labels] =
max(bsxfun(@minus,2*real(centroids'*X),dot(centroids,centroids,1).'),[],1);
    End

%大津法
function Untitled_21_Callback(hObject, eventdata, handles)
I = getimage(gca);
p=zeros(1,256);
for k=1:numel(I)
    p(I(k)+1)=p(I(k)+1)+1;
end
p=p/numel(I);
cump=zeros(1,256);
cump=cumsum(p);
temp=0:255;
temp=temp.*p;
m=cumsum(temp);
m_global=m(256);
var_b=zeros(1,256);
for k=1:256
    if p(k)~=0
        var_b(k)=(m_global*cump(k)-m(k)^2/(cump(k)*(1-cump(k))));%ÀàÍä·½²î
    end
end
tempk=0;
countk=0;
for k=1:256
    if var_b(k)==max(var_b)
        tempk=tempk+k;
        countk=countk+1;
    end
end
threshold=round(tempk/countk)-1;
seg_I=zeros(size(I));
for k=1:numel(I)
    if I(k)>=threshold
        seg_I(k)=1;
    end
end
axes(handles.axes2);
imshow(seg_I);
```