附录

function [ retu ] = like( a,b )

% 列向量相似率计算函数

% retu a b 均是列向量

% 此处显示详细说明

in\_type = 255 ;% 像素值范围

long = size(a,1);

if long ~= size(b,1) % 两个列向量的长度应当等长

retu = 0;

disp('Error:Fun like,different long size.')

else

retu = 1 - sum((a-b)+(b-a))/long/in\_type ;

end

end

%% 载入数据，可视化，生成图像

%载入图像

for k = [ 2 3 5 ]

data(:,:,k) = xlsread('data.xls',k);

end

%处理

for k = [ 2 3 5 ]

data(:,:,k) = data(:,:,k)\*255/max(max(data(:,:,k)));

end

img = uint8( data ) ;

% 图像输出到文件

% for k = [ 2 3 5]

% name = sprintf( 'data%01i.png',k );

% imwrite(img(:,:,k),name)

% end

clear name k

%% 整理图像 180>>360

tast1 = img(1:256,:,2);

tast2 = img(257:512,:,2);

tast3 = tast2 ;

for k = 1:1:256

tast3(257-k,:)=tast2(k,:);

end

tast = [ tast1 tast3 ];

clear tast1 tast2 tast3 k

imshow(tast)

%% 测定角度

% 载入原始数据以校对角度

real = xlsread('data.xls',2);

% 生成模拟图

tast0

% close all

clc

real = real .\*255 ./ max(max(real));

real = uint8(real);

%%

xs = zeros(1,180) ;

for l = 1:180

a = real(:,l);

jilu = zeros(1,230) ;

for k = 1:230

b = img5(:,k);

jilu(k) = like(a,b);

end

% plot(1:230,jilu)

[a1,a2] = max(jilu);

xs(l) = a2 ;

end

plot(xs)

%% 解决第二题

map = xlsread('data.xls',4);

imgq2ct = xlsread('data.xls',3);

map = uint16( [(50-map(:,2))./0.2761+181 (map(:,1)-50)./0.2761+181 ] ) ;

imgq2 = iradon(imgq2ct,27:206);

figure(1);imshow(imgq2);title('iRadon');

imgmov = [ 28,38 ] ;

se = translate(strel(1),-imgmov); %大小不变，进行Y,X方向平移

imgq22 = imdilate( imgq2,se);

figure(2);imshow(imgq22);title('还原偏移量');

% 加点

% plot(map(:,1),map(:,2),'.','markersize',20);

clear se

imgq23 = imresize(imgq22,0.706);

imgq23 = (imgq23>0.1) .\* imgq23 ;

figure(3);imshow(imgq23);title('还原尺寸');

%%

xlswrite('problem2.xls',imgq23)

%% 解决第三题

map = xlsread('data.xls',4);

imgq3ct = xlsread('data.xls',5);

map = [ (50-map(:,2))./0.2761+181 (map(:,1)-50)./0.2761+181 ] ;

imgq3 = iradon(imgq3ct,27:206);

figure(1);imshow(imgq3);title('iRadon');

imgmov = [ 28,38 ] ;

se = translate(strel(1),-imgmov); %大小不变，进行Y,X方向平移

imgq32 = imdilate( imgq3,se);

figure(2);imshow(imgq32);title('还原偏移量');

clear se

imgq33 = imresize(imgq32,0.706);

% imgq33 = (imgq33>0.1) .\* imgq23 ;

figure(3);imshow(imgq33);title('还原尺寸');

%%

xlswrite('problem3.xls',imgq33)

%% 输入图片进行两次转换测试

%参数设定

imgmov = [ 28,38 ]; %旋转中心偏移量（ppi）

blc = 0.2822 ;% 每个像素的长度（mm/ppi）

set\_ppi = 40 ;% 边缘增加像素数目

p = 1:180 ; % 起始角度

cita = p ;clear p p1;

%载入原始截面图像

img = xlsread('data.xls',1);

% 按比例缩放图片

img1 = imresize(img,1.41744) ;

% 3\*3平滑滤波

img1 = filter2(fspecial('average',3),img1);

figure(1);imshow(img1);title('拉伸截面图');

%% 增大轮廓

img2 = zeros(363+2\*set\_ppi);

img2(set\_ppi+1:set\_ppi+363,set\_ppi+1:set\_ppi+363) = img1;

figure(2);imshow(img2);title('增大轮廓');

%% 修正偏移量

% 对原始像素图像进行旋转中心的偏移校正

se = translate(strel(1),imgmov); %大小不变，进行Y,X方向平移

img3= imdilate(img2,se);

for m = 1:1:size(img3,1)

for n = 1:1:size(img3,2)

if img3(m,n)<=0

img3(m,n) = 0 ;

end

end

end

clear m n

figure(3);imshow(img3);title('修正偏移量');

clear se

%% radon transform

img4 = radon(img3,cita);

figure(4);imshow(uint8(img4));title('CT原始数据');

lsln = size(img4,1) ;

%% 边缘化处理

% 对图像二进行边缘处理

I=xlsread('data.xls',2);

% imshow(uint8(I));

title('宿主图像');

BW(:,:,1)=~edge(I,'canny');

BW(:,:,2)=~edge(I,'log');

BW(:,:,3)=~edge(I,'sobel');

BW(:,:,4)=~edge(I,'Roberts');

BW(:,:,5)=~edge(I,'Prewitt');

figure;

subplot(1,5,1);imshow(BW(:,:,1));

title('canny算子');

subplot(1,5,2);imshow(BW(:,:,2));

title('log算子');

subplot(1,5,3);imshow(BW(:,:,3));

title('soble算子');

subplot(1,5,4);imshow(BW(:,:,4));

title('Roberts算子');

subplot(1,5,5);imshow(BW(:,:,5));

title('Prewitt算子');

clear I

%% 导出图像

imwrite(BW(:,:,1),'LINE\_canny.bmp');

imwrite(BW(:,:,2),'LINE\_log.bmp');

imwrite(BW(:,:,3),'LINE\_soble.bmp');

imwrite(BW(:,:,4),'LINE\_roberts.bmp');

imwrite(BW(:,:,5),'LINE\_prewitt.bmp');

%% 拉冬转换的测试

% im = xlsread('data.xls',1);

% imshow(im);

p = 1;

r = uint8(radon(im,p:1:p+179));

imshow(r)

% 连接tast1

I = r;

clear p r im

%% 拉冬转换逆运算

ladeng = iradon(r,1:1:180);

imshow(ladeng)

clear p r