Ch1 Introduction

im\_input = imread('.\Pics\Ch01\lena.jpg');

figure, imshow(im\_input), title('original color image');

im\_gray = rgb2gray(im\_input);

figure, imshow(im\_gray), title('original color image');

imwrite(im\_gray,'gray.jpg');

imfinfo('.\Pics\Ch01\lena.jpg')

Ch2 Fundamentals

im\_inut = imread('.\Pics\Ch02\weixins.bmp');

im\_1 = imresize(im\_inut, 4,'nearest');

im\_2 = imresize(im\_inut, 4,'Bilinear');

im\_3 = imresize(im\_inut, 4,'Bicubic');

figure, imshow(im\_inut); title('image negatives method');

figure, imshow(im\_1); title('Nearest-neighbor interpolation');

figure, imshow(im\_2); title('Bilinear interpolation');

figure, imshow(im\_3); title('Bicubic interpolation');

Ch3 Spatial domain Image Enhancement

%%Image negative

im\_input = im2double(imread('.\Pics\Ch03\rose.jpg'));

im\_output = 1-im\_input;

figure, imshow([im\_input im\_output]), title('Gray-level Invert');

%%Log Transformations（对数变换）

im\_input = imread('.\Pics\Ch03\spectrum.tif');

im\_output = 3\*log(double(im\_input)+1);

figure, imshow(im\_input); title('input image');

figure, imshow(abs(im\_output),[]); title('image Log transform');

%%Gamma correction（伽马校正）

im\_input = im2double(imread('.\Pics\Ch03\MRI.jpg'));

im\_output1 = imadjust(im\_input,[0;1], [0;1], 1.2);

im\_output2 = imadjust(im\_input,[0;1], [0;1], 0.8);

figure, imshow(im\_input); title('input image');

figure, imshow([im\_output1 im\_output2]); title('Gamma correction');

%%Contrast stretching（对比度拉伸）

im\_input = im2double(imread('.\Pics\Ch03\pollen.tif'));

im\_output = imadjust(im\_input,[min(im\_input(:)) max(im\_input(:))], [0 1]);

figure, imshow(im\_input); title('input image');

figure, imshow(im\_output); title('Contrast stretch');

%%Histogram （直方图）

im\_input = im2double(imread('.\Pics\Ch03\pollen.tif'));

im\_output = imadjust(im\_input,[min(im\_input(:)) max(im\_input(:))], [0 1]);

figure, imshow(im\_input); title('input image');

figure, imshow(im\_output); title('Contrast stretch');

histogram(im\_input,64,'EdgeColor','r','FaceAlpha',1,'FaceColor','r');

hold on;

histogram(im\_output,64,'EdgeColor','k','FaceAlpha',0.2,'FaceColor','y');

hold off;

%%Histogram equalization（直方图均衡）

im\_input = im2double(imread('.\Pics\Ch03\pollen.tif'));

im\_output = imadjust(im\_input,[min(im\_input(:)) max(im\_input(:))], [0 1]);

im\_histeq = histeq(im\_input);

figure, imshow(im\_input); title('input image');

figure, imshow(im\_output); title('Contrast stretch');

figure, imshow(im\_histeq); title('Histgram equalization');

histogram(im\_input,64,'EdgeColor','r','FaceAlpha',0.5,'FaceColor','r');

histogram(im\_output,64,'EdgeColor','g','FaceAlpha',0.5,'FaceColor','g');

histogram(im\_histeq,64,'EdgeColor','b','FaceAlpha',0.5,'FaceColor','b');

%%Histogram specification（直方图规定化）

im\_input = im2double(imread('.\Pics\Ch03\pollen.tif'));

im\_ref = im2double(imread('.\Pics\Ch01\gray.jpg'));

im\_output = imhistmatch(im\_input,im\_ref,64);

figure, imshow(im\_input); title('input image');

histogram(im\_input,64,'EdgeColor','r','FaceAlpha',0.5,'FaceColor','r');

figure, imshow(im\_ref); title('Contrast stretch');

histogram(im\_ref,64,'EdgeColor','g','FaceAlpha',0.5,'FaceColor','g');

figure, imshow(im\_output); title('Histgram equalization');

histogram(im\_output,64,'EdgeColor','b','FaceAlpha',0.5,'FaceColor','b');

%%Adaptive histogram equalization（自适应直方图均衡）

im\_input = im2double(imread('.\Pics\Ch03\planet.png'));

figure, imshow(im\_input); title('input image');

histogram(im\_input,64,'EdgeColor','r','FaceAlpha',0.5,'FaceColor','r');

im\_output1 = histeq(im\_input);

figure, imshow(im\_output1); title('Histogram equalization result');

histogram(im\_output1,64,'EdgeColor','r','FaceAlpha',0.5,'FaceColor','r');

imwrite(im\_output1,'result1.jpg');

im\_output2 = adapthisteq(im\_input,'NumTiles',[25 25], 'ClipLimit', 0.05);

figure, imshow(im\_output2); title('Adaptive histogram equalization result');

histogram(im\_output2,64,'EdgeColor','g','FaceAlpha',0.5,'FaceColor','g');

imwrite(im\_output2,'result2.jpg');

%%空域滤波基础1

im\_input = im2double(imread('.\Pics\Ch03\test.tif'));

figure, imshow(im\_input); title('input image');

H1=[1 1 1;1 1 1;1 1 1]/9;

im\_output1 = filter2(H1,im\_input,'same');

figure, imshow(im\_output1); title('average filter1');

H2 = fspecial('average',35);

im\_output2 = imfilter(im\_input,H2,'symmetric','same','conv');

figure, imshow(im\_output2); title('average filter2');

%%空域滤波基础2

im\_input = im2double(imread('.\Pics\Ch03\test.tif'));

figure, imshow(im\_input); title('input image');

H1 = fspecial('average',3);

im\_output1 = imfilter(im\_input,H1);

figure, imshow(im\_output1); title('average filter');

H2 = fspecial('gaussian',[3 3],0.5);

im\_output2 = imfilter(im\_input,H2);

figure, imshow(im\_output2); title('Gaussian filter');

H3 = fspecial('log',[5 5], 0.5);

im\_output3 = imfilter(im\_input,H3);

figure, imshow(im\_output3); title('LoG filter');

%%空域滤波基础3

im\_input = im2double(imread('.\Pics\Ch03\test.tif'));

figure, imshow(im\_input); title('input image');

H1 = fspecial('average',3);

im\_output1 = imfilter(im\_input,H1);

figure, imshow(im\_output1); title('average filter1');

H2 = [1 2 1;2 4 2;1 2 1]/16;

im\_output2 = imfilter(im\_input,H2);

figure, imshow(im\_output2); title('average filter2');

%%Smoothing Filtering (平滑滤波)

im\_input = im2double(imread('.\Pics\Ch03\test.tif'));

H1 = fspecial('average',3);

im\_output1 = imfilter(im\_input,H1);

H2 = fspecial('average',5);

im\_output2 = imfilter(im\_input,H2);

H3 = fspecial('average',9);

im\_output3 = imfilter(im\_input,H3);

H4 = fspecial('average',15);

im\_output4 = imfilter(im\_input,H4);

H5 = fspecial('average',35);

im\_output5 = imfilter(im\_input,H5);

figure, imshow([im\_input im\_output1;im\_output2 im\_output3;im\_output4 im\_output5]);

title('smooting filter with different size of mask');

%%Median filter V.S Average Filter(平滑滤波)

im\_input = im2double(imread('.\Pics\Ch03\noisy1.tif'));

figure, imshow(im\_input); title('input image');

H1 = fspecial('average',3);

im\_output1 = imfilter(im\_input,H1);

im\_output2 = medfilt2(im\_input);

figure, imshow([im\_output1 im\_output2]);

title('smooting filter v.s median filter');

Median filter V.S Average Filter(平滑滤波)2

im\_input = im2double(imread('.\Pics\Ch03\noisy2.tif'));

figure, imshow(im\_input); title('input image');

H1 = fspecial('average',3);

im\_output1 = imfilter(im\_input,H1);

im\_output2 = medfilt2(im\_input,[5 5]);

figure, imshow([im\_output1 im\_output2]);

title('smooting filter v.s median filter');

%%Laplacian Operator

im\_input = im2double(imread('.\Pics\Ch03\moon.tif'));

figure, imshow(im\_input); title('input image');

H1 = [0 1 0; 1 -4 1; 0 1 0];

H2 = [1 1 1; 1 -8 1; 1 1 1];

im\_output1 = imfilter(im\_input,H1);

im\_output2 = imfilter(im\_input,H2);

im\_output3 = im\_input-im\_output2;

figure, imshow([im\_input im\_output3]); title('sharpen using Laplacian operator');

Unsharpen Operator and highboosting filtering

im\_input = im2double(imread('.\Pics\Ch03\moon.tif'));

figure, imshow(im\_input); title('input image');

filter\_mask = fspecial('gaussian');

im\_output1 = im\_input+1\*(im\_input - imfilter(im\_input,filter\_mask));

im\_output2 = im\_input+10\*(im\_input - imfilter(im\_input,filter\_mask));

figure, imshow(im\_output1); title('Unsharpen operator');

figure, imshow([im\_output1 im\_output2]); title('highboosting filter');

%%Gradient operator

im\_input = im2double(imread('.\Pics\Ch03\machine.png'));

figure, imshow(im\_input); title('input image');

H1 = [1,1,1;0,0,0;-1,-1,-1];

H2 = [1,2,1;0,0,0;-1,-2,-1];

im\_output1 = imfilter(im\_input,H1);

im\_output2 = imfilter(im\_input,H2);

figure, imshow(im\_output1); title('Prewitt operator');

figure, imshow(im\_output2); title('Sobel filter');

Ch4 Image enhancement in frequency domain

%Frequency filtering example 1

im\_input = im2double(imread('.\Pics\Ch04\rect.tif'));

figure,imshow(im\_input),title('原图');

F=fft2(im\_input);

S=abs(F);

figure, imshow(S,[]), title('幅度谱');

Fc = fftshift(F);

figure, imshow(abs(Fc),[]), title('中心变换后的幅度谱');

S2 = log(1+abs(Fc));

figure, imshow(S2 ,[]), title('对数变换');

%Frequency filtering example 2: without padding

im\_input = im2double(imread('.\Pics\Ch04\square.tif'));

figure,imshow(im\_input),title('原图');

[im\_h,im\_w] = size(im\_input);

F=fft2(im\_input);

% S=abs(F);

%Fc = fftshift(F);

H = lpfilter('gaussian',im\_h,im\_w,10);

G = H.\*F;

im\_output = real(ifft2(G));

figure, imshow(im\_output,[]),title('没有进行填充的滤波结果');

%Frequency filtering example 3 : with padding

im\_input = im2double(imread('.\Pics\Ch04\square.tif'));

figure,imshow(im\_input),title('原图');

PQ=paddedsize(size(im\_input));

Fp=fft2(im\_input, PQ(1), PQ(2));

Hp =lpfilter('gaussian', PQ(1),PQ(2), 20);

Gp = Hp.\*Fp;

gp = real(ifft2(Gp));

gpc = gp(1:size(im\_input,1),1:size(im\_input,2));

figure, imshow(gpc,[]), title('进行填充的滤波结果');

%Frequency filtering example 4: sobel operator

im\_input = im2double(imread('.\Pics\Ch04\house.tif'));

F=fft2(im\_input);

S=fftshift(log(1+abs(F)));

S=gscale(S);

figure, imshow(S); title('幅频图');

%h=fspecial('sobel')';

h=[ 1 0 -1;2 0 -2;1 0 -1];

freqz2(h);

PQ=paddedsize(size(im\_input));

H=freqz2(h,PQ(1),PQ(2));

H1=ifftshift(H);

figure,imshow(abs(H),[]),title('filter mask in spatial domain');

figure,imshow(abs(H1),[]),title('filter mask in frequency domain');

gs=imfilter(double(im\_input),h);

figure,imshow(gs,[]),title('Spatial domain filtering');% spatial domain filtering result

gf=dftfilt(im\_input,H1);

figure,imshow(gf,[]), title('frequency domain filtering');% frequency domain filtering result

%Ideal low-pass filtering

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=lpfilter('ideal',im\_h,im\_w,10);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1,[]),title('ILPF D0=10');

H2=lpfilter('ideal',im\_h,im\_w,30);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2,[]),title('ILPF D0=30');

H3=lpfilter('ideal',im\_h,im\_w,60);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3,[]),title('ILPF D0=60');

H4=lpfilter('ideal',im\_h,im\_w,160);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4,[]),title('ILPF D0=30');

%Butterworth low-pass filtering 1

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=lpfilter('btw',im\_h,im\_w,15,1.0);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1,[]),title('BLPF D0=15 n=1');

H2=lpfilter('btw',im\_h,im\_w,15,2.0);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2,[]),title('BLPF D0=15 n=2');

H3=lpfilter('btw',im\_h,im\_w,15,5.0);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3,[]),title('BLPF D0=15 n=5');

H4=lpfilter('btw',im\_h,im\_w,15,10.0);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4,[]),title('BLPF D0=15 n=20');

%Butterworth low-pass filtering 2

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=lpfilter('btw',im\_h,im\_w,5,2.0);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1,[]),title('BLPF D0=5');

H2=lpfilter('btw',im\_h,im\_w,15,2.0);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2,[]),title('BLPF D0=15');

H3=lpfilter('btw',im\_h,im\_w,30,2.0);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3,[]),title('BLPF D0=30');

H4=lpfilter('btw',im\_h,im\_w,80,2.0);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4,[]),title('BLPF D0=80');

%Gaussian low-pass filtering

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=lpfilter('gaussian',im\_h,im\_w,5);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1,[]),title('GLPF D0=5');

H2=lpfilter('gaussian',im\_h,im\_w,15);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2,[]),title('GLPF D0=15');

H3=lpfilter('gaussian',im\_h,im\_w,30);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3,[]),title('GLPF D0=30');

H4=lpfilter('gaussian',im\_h,im\_w,80);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4,[]),title('GLPF D0=80');

%Comparison of low-pass filtering

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=lpfilter('ideal',im\_h,im\_w,30);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1,[]),title('ILPF D0=15');

H2=lpfilter('btw',im\_h,im\_w,30,2.0);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2,[]),title('BLPF D0=15 n=2');

H3=lpfilter('gaussian',im\_h,im\_w,30);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3,[]),title('GLPF D0=15');

%Ideal high-pass filtering

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=hpfilter2('ideal',im\_h,im\_w,10);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1),title('IHPF D0=10');

H2=hpfilter2('ideal',im\_h,im\_w,30);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2),title('IHPF D0=30');

H3=hpfilter2('ideal',im\_h,im\_w,60);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3),title('IHPF D0=60');

H4=hpfilter2('ideal',im\_h,im\_w,160);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4),title('IHPF D0=30');

%Butterworth high-pass filtering

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=hpfilter2('btw',im\_h,im\_w,5,2.0);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1),title('BHPF D0=5');

H2=hpfilter2('btw',im\_h,im\_w,15,2.0);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2),title('BHPF D0=15');

H3=hpfilter2('btw',im\_h,im\_w,30,2.0);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3),title('BHPF D0=30');

H4=hpfilter2('btw',im\_h,im\_w,80,2.0);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4),title('BHPF D0=80');

%Gaussian high-pass filtering

im\_input = im2double(imread('.\Pics\Ch04\test.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w]=size(im\_input);

F=fft2(im\_input);

H1=hpfilter2('gaussian',im\_h,im\_w,5);

G=H1.\*F;

im\_output1=real(ifft2(G));

figure;imshow(im\_output1),title('GHPF D0=5');

H2=hpfilter2('gaussian',im\_h,im\_w,15);

G=H2.\*F;

im\_output2=real(ifft2(G));

figure;imshow(im\_output2),title('GHPF D0=15');

H3=hpfilter2('gaussian',im\_h,im\_w,30);

G=H3.\*F;

im\_output3=real(ifft2(G));

figure;imshow(im\_output3),title('GHPF D0=30');

H4=hpfilter2('gaussian',im\_h,im\_w,80);

G=H4.\*F;

im\_output4=real(ifft2(G));

figure;imshow(im\_output4),title('GHPF D0=80');

%Sharpen in frequency using Laplacian operator

im\_input = im2double(imread('.\Pics\Ch04\moon.tif'));

figure, imshow(im\_input); title('input image');

h=[ 1 1 1;1 -8 1;1 1 1];

freqz2(h);

PQ=paddedsize(size(im\_input));

H=freqz2(h,PQ(1),PQ(2));

H1=ifftshift(H);

figure,imshow(abs(H),[]),title('filter mask in spatial domain');

figure,imshow(abs(H1),[]),title('filter mask in frequency domain');

gf=dftfilt(im\_input,H1);

figure,imshow(gf,[]), title('Laplacian filering in frequency domain')

H2 = 1-H1;

gf2=dftfilt(im\_input,H2);

figure,imshow(gf2), title('frequency domain sharpen filtering')

highboosting filtering in frequency domain

im\_input = im2double(imread('.\Pics\Ch04\chest.tif'));

figure, imshow(im\_input); title('input image');

[im\_h,im\_w]=size(im\_input);

PQ = paddedsize(size(im\_input));

D0 = 0.05\*PQ(1);

HBW = hpfilter2('btw', PQ(1),PQ(2),D0,2);

gbw = dftfilt(im\_input,HBW);

gbw = gscale(gbw);

figure, imshow(gbw),title('high-pass filter result');

H = 0.5+2\*HBW;

ghf = dftfilt(im\_input,H);

ghf = gscale(ghf);

figure, imshow(ghf),title('highboosting filter result');

ghe = histeq(ghf, 256);

figure, imshow(ghe),title('final result after histogram equalization');

Homomorphic filtering

Img = imread('.\Pics\Ch04\Homomorphic.bmp');

Img =rgb2gray(Img);

Img =im2uint8(Img);

[M N]=size(Img); %得大小

figure;

imshow(Img); %显示图

Img=double(Img); %转换类型

lnImg = log(Img+1); %取对数

FImg = fft2(lnImg); %傅立叶变换

P =fftshift(FImg); %将频域原点移到图像中心

for i=1:M

for j=1:N

D(i,j)=((i-M/2)^2+(j-N/2)^2); %点（i,j）到频率平面原点的距离

end

end

c=0.8; %锐化参数，可调

Do=200 ; %一般是方差 (滤波器的高通截止频率)

H=(2.0-0.5)\*(1-exp(c\*(-D/(Do^2))))+0.5; %滤波器函数

hImg=FImg.\*H;

gImg=ifft2(hImg); %逆傅立叶变换

Y=exp(gImg); %取指数

G=real(Y);

figure,imshow(uint8((G)));

imwrite(G,'result.bmp');

Ch5 Image restoration

Noise models

im\_input = im2double(rgb2gray(imread('.\Pics\Ch05\check.bmp')));

figure,imshow(im\_input),title('原图');

[im\_h, im\_w] = size(im\_input);

im\_output1 = im\_input+imnoise2('gaussian',im\_h, im\_w);

figure, imshow(im\_output1,[]),title('高斯噪声');

im\_output2 = im\_input+imnoise2('rayleigh',im\_h, im\_w);

figure, imshow(im\_output2,[]),title('瑞利噪声');

,im\_h, im\_w

im\_output3 = im\_input+imnoise2('exponential',im\_h, im\_w);

figure, imshow(im\_output3,[]),title('指数噪声');

im\_output4 = im\_input+imnoise2('erlang',im\_h, im\_w);

figure, imshow(im\_output4,[]),title('伽马噪声（爱尔兰噪声）');

im\_output5 = im\_input+imnoise2('uniform',im\_h, im\_w);

figure, imshow(im\_output5,[]),title('均匀噪声');

Noise models: Gaussian noise

im\_input = im2double(rgb2gray(imread('.\Pics\Ch05\check.bmp')));

figure,imshow(im\_input),title('原图');

im\_output1 = imnoise(im\_input, 'gaussian',0,0.01);

figure, imshow(im\_output1),title('高斯噪声 var=0.01');

im\_output2 = imnoise(im\_input, 'gaussian',0,0.1);

figure, imshow(im\_output2),title('高斯噪声 var=0.1');

im\_output3 = imnoise(im\_input, 'gaussian',0,0.5);

figure, imshow(im\_output3),title('高斯噪声 var=0.5');

Noise models: Salt-pepper noise

im\_input = im2double(rgb2gray(imread('.\Pics\Ch05\check.bmp')));

figure,imshow(im\_input),title('原图');

im\_output1 = imnoise(im\_input, 'salt & pepper',0.01);

figure, imshow(im\_output1),title('椒盐噪声 density=0.01');

im\_output2 = imnoise(im\_input, 'salt & pepper',0.1);

figure, imshow(im\_output2),title('椒盐噪声 density=0.1');

im\_output3 = imnoise(im\_input, 'salt & pepper',0.5);

figure, imshow(im\_output3),title('椒盐噪声 density=0.5');

Noise removal using average filter 1

im\_input = im2double(imread('.\Pics\Ch05\board.tif'));

%figure,imshow(im\_input, []),title('原图');

im\_noise = imnoise(im\_input, 'gaussian',0,0.05);

figure, imshow([im\_input im\_noise]),title('高斯噪声 var=0.1');

im\_output1 = spfilt(im\_noise,'amean');

%figure, imshow(im\_output1),title('算术均值滤波结果');

im\_output2 = spfilt(im\_noise,'gmean');

figure, imshow([im\_output1 im\_output2]),title('算术均值滤波 v.s. 几何均值滤波');

Noise removal using average filter 2

im\_input\_pepper = im2double(imread('.\Pics\Ch05\pepper\_only.tif'));

%figure,imshow(im\_input\_pepper),title('含有椒噪声原图');

im\_input\_salt = im2double(imread('.\Pics\Ch05\salt\_only.tif'));

%figure,imshow(im\_input\_salt),title('含有盐噪声原图');

im\_output1 = spfilt(im\_input\_pepper,'chmean',3,3,1.5);

figure, imshow([im\_input\_pepper im\_output1]),title('逆谐波均值滤波去除椒噪声 Q=1.5');

im\_output2 = spfilt(im\_input\_salt,'chmean',3,3,-1.5);

figure, imshow([im\_input\_salt im\_output2]),title('逆谐波均值滤波去除盐噪声 Q=-1.5');

im\_output3 = spfilt(im\_input\_pepper,'chmean',3,3,-1.5);

figure, imshow([im\_input\_pepper im\_output3]),title('逆谐波均值滤波去除椒噪声 Q=-1.5');

im\_output4 = spfilt(im\_input\_salt,'chmean',3,3,1.5);

figure, imshow([im\_input\_salt im\_output4]),title('逆谐波均值滤波去除盐噪声 Q=1.5');

Noise removal using statistic order filter 1

im\_input = im2double(imread('.\Pics\Ch05\board.tif'));

%figure,imshow(im\_input\_pepper),title('含有椒噪声原图');

im\_noise = imnoise(im\_input,'salt & pepper',0.4);

figure,imshow([im\_input im\_noise]),title('原图和含噪图像');

im\_output1 = spfilt(im\_noise,'median',3,3);

im\_output2 = spfilt(im\_output1,'median',3,3);

im\_output3 = spfilt(im\_output2,'median',3,3);

figure, imshow([im\_output1 im\_output2 im\_output3]),title('中值滤波结果');

Noise removal using statistic order filter 2

im\_input = im2double(imread('.\Pics\Ch05\board.tif'));

%figure,imshow(im\_input\_pepper),title('含有椒噪声原图');

im\_noise = imnoise(im\_input,'salt & pepper',0.2);

figure,imshow([im\_input im\_noise]),title('原图和含噪图像');

im\_output1 = spfilt(im\_noise,'median',3,3);

im\_output2 = spfilt(im\_noise,'max',3,3);

im\_output3 = spfilt(im\_noise,'min',3,3);

figure, imshow([im\_output1 im\_output2 im\_output3]),title('median v.s max v.s min');

im\_input\_pepper = im2double(imread('.\Pics\Ch05\pepper\_only.tif'));

im\_output4 = spfilt(im\_input\_pepper,'max',3,3);

figure,imshow([im\_input\_pepper im\_output4]),title('最大值滤波器滤除椒噪声');

im\_input\_salt = im2double(imread('.\Pics\Ch05\salt\_only.tif'));

im\_output5 = spfilt(im\_input\_salt,'min',3,3);

figure,imshow([im\_input\_salt im\_output5]),title('最小值滤波器滤除盐噪声');

Noise removal: comparison

im\_input = im2double(imread('.\Pics\Ch05\board.tif'));

im\_noise1 = imnoise(im\_input,'gaussian',0,0.02);

im\_noise2 = imnoise(im\_noise1,'salt & pepper',0.2);

figure, imshow([im\_input im\_noise1 im\_noise2]); title('原图 高斯噪声 高斯噪声+椒盐噪声');

im\_output1 = spfilt(im\_noise2,'amean',5,5);

im\_output2 = spfilt(im\_noise2,'gmean',5,5);

im\_output3 = spfilt(im\_noise2,'median',5,5);

im\_output4 = spfilt(im\_noise2,'atrimmed',5,5,24);

figure, imshow([im\_output1 im\_output2]); title('算术均值滤波 几何均值滤波');

figure, imshow([im\_output3 im\_output4]); title('中值滤波 修正的阿尔法均值滤波');

Degradation function estimatation: motion blur

im\_input = im2double(rgb2gray(imread('.\Pics\Ch05\cover.bmp')));

%figure, imshow(im\_input),title('input image');

PSF=fspecial('motion', 50,135);

im\_output = imfilter(im\_input, PSF, 'circular');

figure, imshow([im\_input im\_output]),title('degratation of mition blur');

Invert filter

im\_input = im2double(rgb2gray(imread('.\Pics\Ch05\cover.bmp')));

%figure, imshow(im\_input),title('input image');

PSF=fspecial('motion', 50,135);

im\_degratation1 = imfilter(im\_input, PSF, 'circular');

%figure, imshow([im\_input im\_output1]),title('degratation of mition blur');

im\_noise = imnoise2('Gaussian',size(im\_input,1),size(im\_input,2), 0, 0.01);

% im\_degratation2 = imnoise(im\_degratation1, 'gaussian',0,0.01);

im\_degratation2 = im\_degratation1+im\_noise;

figure, imshow([im\_input im\_degratation1 im\_degratation2]),title('degratation of mition blur');

% imwrite(im\_degratation1,'1.jpg');

% imwrite(im\_degratation2,'2.jpg');

im\_output1 = deconvwnr(im\_degratation1, PSF);

im\_output2 = deconvwnr(im\_degratation2, PSF);

figure, imshow(im\_output1),title('invert filter results');

figure, imshow(im\_output2,[]),title('invert filter results');

% imwrite(im\_output1,'3.jpg');

% imwrite(im\_output2,'4.jpg');

Wiener filter v.s Invert filter

im\_input = im2double(rgb2gray(imread('.\Pics\Ch05\cover.bmp')));

%figure, imshow(im\_input),title('input image');

PSF=fspecial('motion', 50,135);

im\_degratation1 = imfilter(im\_input, PSF, 'circular');

%figure, imshow([im\_input im\_output1]),title('degratation of mition blur');

im\_noise = imnoise2('Gaussian',size(im\_input,1),size(im\_input,2), 0, 0.01);

im\_degratation2 = im\_degratation1+im\_noise;

figure, imshow([im\_input im\_degratation2]),title('degratation of mition blur and noise');

% restoration by invert filter

im\_output1 = deconvwnr(im\_degratation2, PSF);

imwrite(im\_output1,'1.jpg');

% 基于信噪比是常数的维纳滤波

Sn =abs(fft2(im\_noise)).^2;

nA = sum(Sn(:))/numel(im\_noise);

Sf = abs(fft2(im\_input)).^2;

fA=sum(Sf(:))/numel(im\_input);

R= nA/fA;

im\_output2 = deconvwnr(im\_degratation2, PSF,R);

imwrite(im\_output2,'2.jpg');

% restoration by Wiener filter with NCORR and NCORR

NCORR = fftshift(real(ifft2(Sn))); % 噪声自相关系数

ICORR = fftshift(real(ifft2(Sf))); % 信号自相关系数

% 基于自相关系数的维纳滤波

im\_output3 = deconvwnr(im\_degratation2, PSF, NCORR, ICORR);

figure, imshow([im\_output1 im\_output2 im\_output3]),title('Wiener filter results');

imwrite(im\_output3,'3.jpg');

Ch6 Color image processing

RGB color space

rgbcube(1,1,1)

im\_input = im2double(imread('.\Pics\Ch06\Chalk.jpg'));

figure,imshow(im\_input),title('原图');

im\_input\_r = im\_input(:,:,1);

im\_input\_g = im\_input(:,:,2);

im\_input\_b = im\_input(:,:,3);

figure, imshow(im\_input\_r,[]), title('R通道图');

figure, imshow(im\_input\_g,[]), title('G通道图');

figure, imshow(im\_input\_b,[]), title('B通道图');

im\_input\_r1 = 1-im\_input\_r;

im\_input\_g1 = 1-im\_input\_g;

im\_input\_b1 = 1-im\_input\_b;

im\_output = cat(3,im\_input\_r1,im\_input\_g1,im\_input\_b1);

figure, imshow([im\_input im\_output]), title('彩色原图和反色结果图');

CMYK color space

im\_input = im2double(imread('.\Pics\Ch06\photo.jpg'));

im\_output = imcomplement(im\_input);

figure,imshow([im\_input im\_output]),title('原图');

HSI color space

im\_input = im2double(imread('.\Pics\Ch06\scene.png'));

im\_hsi = rgb2hsi(im\_input);

im\_hsi\_h = im\_hsi(:,:,1);

im\_hsi\_s = im\_hsi(:,:,2);

im\_hsi\_i = im\_hsi(:,:,3);

figure,imshow(im\_input),title('原图');

figure,colormap('hsv'),imshow(im\_hsi\_h),title('H分量：色调'); % 加上颜色

figure,imshow(im\_hsi\_s),title('S分量：饱和度');

figure,imshow(im\_hsi\_i),title('I分量：强度');

HSI color space 2

im\_input = im2double(imread('.\Pics\Ch06\color\_bars.tif'));

im\_hsi = rgb2hsi(im\_input);

im\_hsi\_h = im\_hsi(:,:,1);

im\_hsi\_s = im\_hsi(:,:,2);

im\_hsi\_i = im\_hsi(:,:,3);

figure,imshow(im\_input),title('原图');

figure,imshow(im\_hsi\_h),title('H分量：色调'); % 加上颜色

figure,imshow(im\_hsi\_s),title('S分量：饱和度');

figure,imshow(im\_hsi\_i),title('I分量：强度');

Pseudocolor Processing : Intensity Slicing 1

im\_input = im2double(imread('.\Pics\Ch06\weld.tif'));

figure, imshow(im\_input),title('input image');

[im\_h,im\_w] = size(im\_input);

im\_bw = im2bw(im\_input,254/255);

im\_output = cat(3,im\_bw,im\_bw,1-im\_bw);

figure;imshow(im\_output),title('Intensity slice result');

Pseudocolor Processing : Intensity Slicing 2

im\_input = im2double(imread('.\Pics\Ch06\picker\_phantom.tif'));

im\_input = imadjust(im\_input);

figure, imshow(im\_input),title('input image');

[im\_h,im\_w] = size(im\_input);

im\_output = grayslice(im\_input,8);

%colormap

mymap = [0,0,0;0,0,1;1,0,1;0,1,1;0,1,0;1,1,0;1,0,0;1,1,1];

figure;imshow(im\_output,mymap),title('Intensity slice result');

Pseudocolor Processing : Intensity Slicing 3

im\_input = rgb2gray(imread('.\Pics\Ch06\world\_light.jpg'));

im\_input = imadjust(im\_input);

figure, imshow(im\_input),title('input image');

[im\_h,im\_w] = size(im\_input);

im\_output = grayslice(im\_input,256);% colormap

figure;imshow(im\_output,jet(256)),title('Intensity slice result');

colorbar

Pseudocolor Processing : Gray-level to Color Transformation

im\_input1 = imread('.\Pics\Ch06\WashingtonDC Band\_R.tif');

im\_input2 = imread('.\Pics\Ch06\WashingtonDC Band\_B.tif');

im\_input3 = imread('.\Pics\Ch06\WashingtonDC Band\_G.tif');

im\_input4 = imread('.\Pics\Ch06\WashingtonDC Band4.tif');

im\_color1 = cat(3,im\_input1,im\_input2,im\_input3);

im\_color2 = cat(3,im\_input4,im\_input2,im\_input3);

figure;imshow([im\_color1 im\_color2]),title('Gray-level to Color Transformation result');

Ch7 Binary image morphological processing

Strcuture element

SE2 = strel('square',3)

SE3 = strel('diamond',2)

SE4 = strel('disk',3)

SE5 = strel('sphere',3)

SE6 = strel('line',5,45)

% MN=[1,1,1,1;1,1,1,1;1,1,1,1];

% SE1 = strel('rectangle',MN)

Erosion operator 1

im\_input = imread('.\Pics\Ch07\wirebond\_mask.tif');

figure, imshow(im\_input),title('输入图像');

se1 = strel('diamond', 2);

im\_output1 = imerode(im\_input, se1);

se2 = [0 0 1 0 0; 0 1 1 1 0; 1 1 1 1 1;0 1 1 1 0;0 0 1 0 0];

im\_output2 = imerode(im\_input, se2);

figure, imshow([im\_output1,im\_output2]),title('腐蚀结果 ');

imoutput3 = xor(im\_output1,im\_output2);

figure, imshow(imoutput3),title('腐蚀结果对比 ');

Erosion operator 2

im\_input = imread('.\Pics\Ch07\wirebond\_mask.tif');

figure, imshow(im\_input),title('输入图像');

se1 = strel('square', 11);

im\_output1 = imerode(im\_input, se1);

se2 = strel('square', 15);

im\_output2 = imerode(im\_input, se2);

se3 = strel('square', 45);

im\_output3 = imerode(im\_input, se3);

figure, imshow([im\_output1,im\_output2 im\_output3]),title('腐蚀结果对比 ');

Dilation operator 1

im\_input = imread('.\Pics\Ch07\broken\_text.tif');

figure, imshow(im\_input),title('输入图像');

se1 = strel('disk', 1);

im\_output1 = imdilate(im\_input, se1);

figure, imshow(im\_output1),title('膨胀结果 ');

Dilation operator 2

im\_input = imread('.\Pics\Ch07\small\_squares.tif');

figure, imshow(im\_input),title('输入图像');

se1 = strel('square', 13);

im\_output1 = imerode(im\_input, se1);

figure, imshow(im\_output1),title('腐蚀结果 ');

im\_output2 = imdilate(im\_output1, se1);

figure, imshow(im\_output2),title('膨胀结果 ');

Open and Close 1

im\_input = imread('.\Pics\Ch07\shapes.tif');

figure, imshow(im\_input),title('输入图像');

se1 = strel('square', 20);

im\_output1 = imopen(im\_input, se1);

figure, imshow(im\_output1),title('开运算结果 ');

im\_output2 = imclose(im\_input, se1);

figure, imshow(im\_output2),title('闭运算结果 ');

im\_output3 = imclose(im\_output1, se1);

figure, imshow(im\_output3),title('先开后闭运算结果 ');

Open and Close 2

im\_input = imread('.\Pics\Ch07\noisy\_fingerprint.tif');

figure, imshow(im\_input),title('输入图像');

se1 = strel('square', 3);

im\_output1 = imopen(im\_input, se1);

figure, imshow(im\_output1),title('开运算结果 ');

im\_output2 = imclose(im\_output1, se1);

figure, imshow(im\_output2),title('再进行闭运算结果 ');

Hit and missing transform

im\_input = imread('.\Pics\Ch07\small\_squares.tif');

figure, imshow(im\_input),title('输入图像');

B1 = strel([0 0 0; 0 1 1; 0 1 0]);

B2 = strel([1 1 1; 1 0 0; 1 0 0]);

im\_output = bwhitmiss(im\_input, B1, B2);

figure, imshow(im\_output), title('HMT result');

boundary extraction

im\_input = imread('.\Pics\Ch07\man.jpg');

im\_input = im2bw(im\_input,0.5);

figure, imshow(im\_input),title('输入图像');

SE1 = strel('disk',1);

SE2 = strel('square',5);

im\_output1 = imerode(im\_input, SE1);

im\_output2 = imerode(im\_input, SE2);

im\_output1 = im\_input-im\_output1;

im\_output2 = im\_input-im\_output2;

im\_output3 = bwmorph(im\_input,'remove');

figure, imshow([im\_output1 im\_output2 im\_output3]), title('boundary extraction result');

Hole filling

im\_input = im2bw(rgb2gray(imread('.\Pics\Ch07\ten\_objects2.jpg')),0.5);

%figure, imshow(im\_input),title('输入图像');

im\_output1 = imfill(im\_input, 'holes');

figure, imshow([im\_input im\_output1]), title('hole filling result');

Connected component extraction

im\_input = imread('.\Pics\Ch07\ten\_objects.tif');

figure, imshow(im\_input),title('输入图像');

im\_output1 = bwlabel(im\_input,8);

im\_output1 = im2double(im\_output1/max(im\_output1(:)));

figure, imshow(im\_output1), title('Connected component extraction result');

Thin

im\_input = imread('.\Pics\Ch07\bone.tif');

figure, imshow(im\_input),title('输入图像');

im\_output1 = bwmorph(im\_input,'thin',Inf);

figure, imshow(im\_output1), title('Connected component extraction result');

Thicken

im\_input = imread('.\Pics\Ch07\bone.tif');

figure, imshow(im\_input),title('输入图像');

im\_output1 = bwmorph(im\_input,'thin',Inf);

im\_output2 = bwmorph(im\_output1,'thicken',10);

figure, imshow([im\_output1 im\_output2]), title('Connected component extraction result');

Skeleton

im\_input = imread('.\Pics\Ch07\bone.tif');

figure, imshow(im\_input),title('输入图像');

im\_output1 = bwmorph(im\_input,'thin',Inf);

im\_output2 = bwmorph(im\_input,'spur',Inf);

figure, imshow([im\_output1 im\_output2]), title('thin v.s skeleton');

Pruning

im\_input = imread('.\Pics\Ch07\bone.tif');

figure, imshow(im\_input),title('输入图像');

im\_output1 = bwmorph(im\_input,'skel',Inf);

im\_output2 = bwmorph(im\_output1,'spur',5);

im\_output3 = bwmorph(im\_output1,'spur',10);

figure, imshow([im\_output1 im\_output2 im\_output3]), title('pruning');

Morphological dilation reconstruction

im\_mask = im2bw(imread('.\Pics\Ch07\recon\_mask.tif'), 0.5);

% figure, imshow(im\_mask),title('MASK图像');MASK图像

im\_marker = im2bw(imread('.\Pics\Ch07\recon\_marker.tif'),0.8);

figure, imshow([im\_mask im\_marker]),title('MASK图像 &　Marker图像');

im\_output = imreconstruct(im\_marker,im\_mask);

figure, imshow(im\_output), title('Morphological dilation reconstruction result');

Morphological open reconstruction

im\_mask = imread('.\Pics\Ch07\book\_text.tif');

im\_erode = imerode(im\_mask, ones(51,1));

im\_open = imopen(im\_mask, ones(51,1));

im\_output = imreconstruct(im\_erode, im\_mask);

figure, imshow([im\_mask im\_erode; im\_open im\_output]), title('Morphologicalopen reconstruction result');

Ch8 Image segmentation

Global threshold 1

im\_input = rgb2gray(imread('.\Pics\Ch10\noisy\_fingerprint.jpg'));

figure,imshow(im\_input),title('原图');

figure, imhist(im\_input),title('原图的直方图');

count = 0;

T = mean2(im\_input);

done = false;

while ~done

count = count + 1;

g = im\_input > T;

Tnext = 0.5\*(mean(im\_input(g)) + mean(im\_input(~g)));

done = abs(T - Tnext) < 0.5;

T = Tnext;

end

count

T

im\_output = im2bw(im\_input, T/255);

figure,imshow(im\_output);title('阈值化结果');

imwrite(im\_output, '1.jpg');

Global threshold 2

im\_input = imread('.\Pics\Ch10\polymersomes.tif');

figure,imshow(im\_input),title('原图');

figure, imhist(im\_input),title('原图的直方图');

count = 0;

T = mean2(im\_input);

done = false;

while ~done

count = count + 1;

g = im\_input > T;

Tnext = 0.5\*(mean(im\_input(g)) + mean(im\_input(~g)));

done = abs(T - Tnext) < 0.5;

T = Tnext;

end

count

T

im\_output = im2bw(im\_input, T/255);

figure,imshow(im\_output);title('阈值化结果');

% imwrite(im\_output, '1.jpg');

Otsu method

im\_input = imread('.\Pics\Ch10\polymersomes.tif');

figure,imshow(im\_input),title('原图');

T = graythresh(im\_input);

T\*255

im\_output = im2bw(im\_input, T);

figure,imshow(im\_output);title('Otsu阈值化结果');

Otsu method with smooting filter 1

im\_input = imread('.\Pics\Ch10\septagon.tif');

figure,imshow(im\_input),title('原图');

% figure, imhist(im\_input),title('原图的直方图');

T1 = graythresh(im\_input);

T1\*255

im\_output1 = im2bw(im\_input, T1);

figure,imshow(im\_output1);title('Otsu阈值化结果：原始图像');

im\_noise = imnoise(im\_input,'gaussian',0,0.05);

figure,imshow(im\_noise),title('含噪图像');

figure, imhist(im\_noise),title('含噪图像的直方图');

T2 = graythresh(im\_noise);

T2\*255

im\_output2 = im2bw(im\_noise, T2);

figure,imshow(im\_output2);title('Otsu阈值化结果：含噪图像');

h = fspecial('average',[5 5]);

im\_smooting = imfilter(im\_noise,h);

figure,imshow(im\_smooting),title('含噪图像平滑处理');

figure, imhist(im\_smooting),title('含噪图像平滑处理后的直方图');

T3 = graythresh(im\_smooting);

T3\*255

im\_output3 = im2bw(im\_smooting, T3);

figure,imshow(im\_output3);title('Otsu阈值化结果：含噪图像平滑处理');

Otsu method with smooting filter

im\_input = imread('.\Pics\Ch10\septagon\_noise.tif');

figure,imshow(im\_input),title('原图');

figure, imhist(im\_input),title('原图的直方图');

T1 = graythresh(im\_input);

T1\*255

im\_output1 = im2bw(im\_input, T1);

figure,imshow(im\_output1);title('Otsu阈值化结果：原始图像');

h = fspecial('average',[5 5]);

im\_smooting = imfilter(im\_input,h);

figure,imshow(im\_smooting),title('平滑处理图像');

figure, imhist(im\_smooting),title('平滑处理后的直方图');

T2 = graythresh(im\_smooting);

T2\*255

im\_output2 = im2bw(im\_smooting, T2);

figure,imshow(im\_output2);title('Otsu阈值化结果：平滑处理');

Threshold-based segmentation using OTSU method, smooting filter and gradient 1

f = im2double(imread('.\Pics\Ch10\septagon\_noise.tif'));

figure,imshow(f),title('原图');

figure, imhist(f),title('原图的直方图');

sx = fspecial('sobel');

sy = sx';

gx = imfilter(f,sx,'replicate');

gy = imfilter(f,sy,'replicate');

grad = sqrt(gx.\*gx + gy.\*gy);

grad = grad/max(grad(:));

h = imhist(grad);

Q = percentile2i(h, 0.999);

markerImage = grad > Q;

figure, imshow(markerImage),title('二值模板图');

fp = f.\*markerImage;

figure, imshow(fp),title('处理图像');

hp = imhist(fp);

hp(1) = 0; % 为了显示方便

figure, bar(hp),title('处理后的直方图');

T = graythresh(hp);

g = im2bw(f, T);

figure, imshow(g) ,title('最终二值化结果');

Threshold-based segmentation using OTSU method, smooting filter and gradient 2

f = im2double(imread('.\Pics\Ch10\yeast.tif'));

figure,imshow(f),title('原图');

figure, imhist(f),title('原图的直方图');

[Tf SMf] = graythresh(f);

gf = im2bw(f, Tf);

figure, imshow(gf),title('OTSU分割结果');

w = [-1 -1 -1; -1 8 -1; -1 -1 -1];

lap = abs(imfilter(f, w, 'replicate'));

lap = lap/max(lap(:));

h = imhist(lap);

Q = percentile2i(h, 0.995);

markerImage = lap > Q;

fp = f.\*markerImage;

figure, imshow(fp) ,title('标记图像');

hp = imhist(fp);

hp(1) = 0; % 为了显示方便

figure, bar(hp),title('处理后的直方图');

T = graythresh(hp);

g = im2bw(f, T);

figure, imshow(g),title('最终二值化结果');

Variable threshold method 1

f = im2double(imread('.\Pics\Ch10\yeast.tif'));

figure,imshow(f),title('原图');

TGlobal = multithresh(f,2);

gGlobal = imquantize(f,TGlobal);

imshow(gGlobal,[]), title('双阈值分割结果');

g = localthresh(tofloat(f), ones(3), 30, 1.5, 'global');

SIG = stdfilt(f, ones(3));

figure, imshow(SIG, [ ]), title('标准差图');

figure, imshow(g), title('可变阈值分割结果');

Variable threshold method 2

im\_input = im2double(imread('.\Pics\Ch10\spot\_shaded\_text.tif'));

figure, imshow(im\_input),title('input image');

T = graythresh(im\_input);

g1 = im2bw(im\_input, T);

figure, imshow(g1),title('Otsu法分割结果');

g2 = movingthresh(im\_input, 20, 0.5);

figure, imshow(g2), title('移动平均法分割结果');

Variable threshold method 3

im\_input = im2double(imread('.\Pics\Ch10\sine\_shaded\_text.tif'));

figure, imshow(im\_input),title('input image');

T = graythresh(im\_input);

g1 = im2bw(im\_input, T);

figure, imshow(g1),title('Otsu法分割结果');

g2 = movingthresh(im\_input, 20, 0.5);

figure, imshow(g2), title('移动平均法分割结果');

Point detection

im\_input = im2double(imread('.\Pics\Ch10\wirebond.tif'));

figure, imshow(im\_input),title('input image');

H = [-1 -1 -1; -1 8 -1; -1 -1 -1];

im\_filter = abs(imfilter(im\_input,H));

im\_output = imbinarize(im\_filter,0.9\*max(im\_filter(:)));

figure;imshow(im\_output),title('point detection result');

Line detection

im\_input = im2double(imread('.\Pics\Ch10\wirebond.tif'));

figure, imshow(im\_input),title('input image');

H=[-1 -1 -1; -1 8 -1; -1 -1 -1];

im\_output = imfilter(im\_input, H);

im\_bw = (im\_output >=0);

figure;imshow(im\_output,[]),title('标定');

figure;imshow(abs(im\_output)),title('绝对值');

figure;imshow(im\_output.\*im\_bw),title('只取正值');

Sobel operator

im\_input = im2double(imread('.\Pics\Ch10\building.tif'));

im\_input = imresize(im\_input,0.5);

figure, imshow(im\_input); title('input image');

H1=[ -1 -2 -1;0 0 0;1 2 1];

H2=H1';

im\_output = im\_input;

% H = fspecial('average', [5 5]);

% im\_output = imfilter(im\_input, H);

% figure, imshow(im\_output); title('after smooting');

im\_grad\_x = imfilter(im\_output,H1);

im\_grad\_y = imfilter(im\_output,H2);

im\_grad = abs(im\_grad\_x)+abs(im\_grad\_x);

im\_edge1 = im\_grad > 0.2\*max(im\_grad(:));

im\_edge2 = im\_grad > 0.33\*max(im\_grad(:));

figure;imshow(abs(im\_grad\_x),[]),title('|gx|');

figure;imshow(abs(im\_grad\_y),[]),title('|gy|');

figure;imshow(im\_grad,[]),title('|gx|+|gy|');

figure;imshow(im\_edge1),title('edge map: using low threshold');

figure;imshow(im\_edge2),title('edge map: using high threshold');

Edge detection using gradient operator: comparison

im\_input = im2double(imread('.\Pics\Ch10\building.tif'));

im\_input = imresize(im\_input,0.5);

figure, imshow(im\_input); title('input image');

im\_edge\_roberts = edge(im\_input,'Roberts');

im\_edge\_prewitt = edge(im\_input,'Prewitt');

im\_edge\_sobel = edge(im\_input,'Sobel');

figure;imshow(im\_edge\_roberts),title('edge map: using roberts operator');

figure;imshow(im\_edge\_prewitt),title('edge map: using prewitt operator');

figure;imshow(im\_edge\_sobel),title('edge map: using sobel operator');

LoG operator

im\_input = im2double(imread('.\Pics\Ch10\building.tif'));

% im\_input = imresize(im\_input,0.5);

figure, imshow(im\_input); title('input image');

G = fspecial('gaussian', 25,4);

im\_gaussian = imfilter(im\_input, G);

H1 = [1 1 1; 1 -8 1; 1 1 1];

im\_log = imfilter(im\_gaussian, H1);

im\_edge1 = edge(im\_log,'zerocross', 0);

thre1 = 0.005\*max(im\_log(:))

im\_edge2 = edge(im\_log,'zerocross',thre1);

im\_edge3 = edge(im\_input,'log');

% figure;imshow(im\_edge\_roberts),title('edge map: using roberts operator');

figure;imshow(im\_log,[]),title('LoG image');

figure;imshow(im\_edge1),title('LoG+zerocross with th=0');

figure;imshow(im\_edge2),title('LoG+zerocross with th=0.005\*max');

figure;imshow(im\_edge3),title('edge map: using LoG with system');

imwrite(im\_edge1,'log.jpg');

Hough transform 1

f = zeros(101, 101);

f(1, 1) = 1; f(101, 1) = 1; f(1, 101) = 1;

f(101, 101) = 1; f(51, 51) = 1;

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

imshow(H, [], 'XData', theta, 'YData', rho ,'InitialMagnification', 'fit')

axis on, axis normal

xlabel('\theta'), ylabel('\rho')

Hough transform 2

im\_input = im2double(imread('.\Pics\Ch10\airport.tif'));

figure, imshow(im\_input); title('input image');

f=edge(im\_input,'canny');

[H, theta, rho] = hough(f, 'ThetaResolution', 0.2);

imshow(H, [], 'XData', theta, 'YData', rho, 'InitialMagnification', 'fit')

axis on, axis normal

xlabel('\theta'), ylabel('\rho')

peaks = houghpeaks(H, 2);

hold on

plot(theta(peaks(:, 2)), rho(peaks(:, 1)), 'linestyle', 'none', 'marker', 's', 'color', 'y') ;

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

figure, imshow(f), hold on

for k = 1:length(lines)

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

plot(xy(:,1), xy(:,2), 'LineWidth', 4, 'Color', [1 1 0]);

end

Region growing

im\_input = im2double(imread('.\Pics\Ch10\weld.tif'));

figure, imshow(im\_input); title('input image');

[im\_output, NR, SI, TI] = regiongrow(im\_input, 1, 0.26);

figure, imshow(im\_output); title('region growing image');

figure, imshow(SI,[]); title('region growing image');

figure, imshow(TI); title('region growing image');

Region split and merge

im\_input = imread('.\Pics\Ch10\cygnusloop.tif');

figure, imshow(im\_input); title('input image');

im\_output1 = splitmerge(im\_input, 32, @predicate);

im\_output2 = splitmerge(im\_input, 16, @predicate);

im\_output3 = splitmerge(im\_input, 8, @predicate);

im\_output4 = splitmerge(im\_input, 4, @predicate);

im\_output5 = splitmerge(im\_input, 2, @predicate);

figure, imshow(im\_output1,[]); title('segmentation result with dim=32');

figure, imshow(im\_output2,[]); title('segmentation result with dim=16');

figure, imshow(im\_output3,[]); title('segmentation result with dim=8');

figure, imshow(im\_output4,[]); title('segmentation result with dim=4');

figure, imshow(im\_output5,[]); title('segmentation result with dim=2');