# MAIN

%%read image

kangaroo = imread('kangaroo.pgm');

plane = imread('plane.pgm');

red = imread('red.pgm');

%% Gaussian Filter to 'kangaroo.pgm'

%process gaus filter to plane

for i = 1:10

subplot(2,5,i);

imshow(gaus\_filt(double(kangaroo),i));

title(['Gaussian filter with sigma' int2str(i)]);

end

%choose sigma = 1 and sigma = 9 to compare

kangaroo\_1 = gaus\_filt(double(kangaroo),1);

kangaroo\_2 = gaus\_filt(double(kangaroo),9);

pause;

%% Sobel Filter to 'plane.pgm'

%apply sobel filter to kangaroo\_1 and kangaroo\_2

%set threshold to be 80 and 40

kangaroo\_sobel\_1 = sobel\_filt(kangaroo\_1,80);

kangaroo\_sobel\_2 = sobel\_filt(kangaroo\_2,40);

%display the image

subplot(1,2,1);

imshow(kangaroo\_sobel\_1);

subplot(1,2,2);

imshow(kangaroo\_sobel\_2);

%choose kangaroo\_sobel\_2 to do futher process

pause;

%% Non-maxium supression to 'kangaroo.pgm'

%apply Non-maxium supression to kangaroo\_1

nms\_kangaroo = non\_max\_sup(kangaroo\_2,40);

%display the image

imshow(nms\_kangaroo);

pause;

%% Gaussian Filter to 'plane.pgm'

%process gaus filter to plane

for i = 1:10

subplot(2,5,i);

imshow(gaus\_filt(double(plane),i));

title(['Gaussian filter with sigma' int2str(i)]);

end

%choose sigma = 1 and sigma = 2 to

plane\_1 = gaus\_filt(double(plane),1);

plane\_2 = gaus\_filt(double(plane),2);

pause;

%% Sobel Filter to 'plane.pgm'

%apply sobel filter to plane\_1 and plane\_2

%set threshold to be 100

plane\_sobel\_1 = sobel\_filt(plane\_1,100);

plane\_sobel\_2 = sobel\_filt(plane\_2,100);

%display the image

subplot(1,2,1);

imshow(plane\_sobel\_1);

subplot(1,2,2);

imshow(plane\_sobel\_2);

%choose plane\_sobel\_1 to do futher process

pause;

%% Non-maxium supression to 'plane.pgm'

%apply Non-maxium supression to plane\_1

nms\_plane = non\_max\_sup(plane\_1,100);

%display the image

imshow(nms\_plane);

pause;

%% Gaussian Filter to 'red.pgm'

%process gaus filter to plane

for i = 1:10

subplot(2,5,i);

imshow(gaus\_filt(double(red),i));

title(['Gaussian filter with sigma' int2str(i)]);

end

%choose sigma = 2 and sigma = 4 to compare

red\_1 = gaus\_filt(double(red),2);

red\_2 = gaus\_filt(double(red),8);

pause;

%% Sobel Filter to 'red.pgm'

%apply sobel filter to red\_1 and red\_2

%set threshold to be 35

red\_sobel\_1 = sobel\_filt(red\_1,35);

red\_sobel\_2 = sobel\_filt(red\_2,35);

%display the image

subplot(1,2,1);

imshow(red\_sobel\_1);

subplot(1,2,2);

imshow(red\_sobel\_2);

%choose red\_sobel\_1 to do futher process

pause;

%% Non-maxium supression to 'red.pgm'

%apply Non-maxium supression to red\_1

nms\_red = non\_max\_sup(red\_1,35);

%display the image

imshow(nms\_red);

# GAUSSIAN FILTER

function gaus\_image = gaus\_filt(X, sigma)

%Initialization

[m,n] = size(X);

gaus\_image = zeros(m,n);

%set the gaus filter to be a 7\*7 matrix, so the half width is 3

halfwid = 3;

%initialize gaus filter

gaus\_filter = zeros(2\*halfwid + 1);

% ============================================

%creat gaus filter with sigma

[xx,yy] = meshgrid(-halfwid:halfwid,-halfwid:halfwid);

gaus\_filter = exp(-1/(2\*sigma^2) \* (xx.^2 + yy.^2));

gaus\_filter = gaus\_filter/sum(sum(gaus\_filter));

%duplicate the edge of image with half width of filter

X\_dup = edged\_dup(X,halfwid);

%mutiple the gaus filter with each 7\*7 matrix of the original image

for i = 1:m

for j = 1:n

temp\_X = X\_dup(i:i+2\*halfwid,j:j+2\*halfwid);

gaus\_image(i,j) = sum(sum(gaus\_filter .\* double(temp\_X)));

end

end

gaus\_image = uint8(gaus\_image);

end

# EDGE DUPLICATION

function X\_dup = edged\_dup(X,extra\_pixel)

%duplicate the edge of image with extra pixel

left\_most = X(:,1:extra\_pixel);

right\_most = X(:,end-extra\_pixel:end);

X\_dup = [left\_most,X,right\_most];

top = X\_dup(1:extra\_pixel,:);

bottom = X\_dup(end-extra\_pixel:end,:);

X\_dup = [top;X\_dup;bottom];

end

**SOBEL**

function [sobel\_image, sobel\_ori] = sobel\_filt(X,threshold)

%Initialization

[m,n] = size(X);

sobel\_image\_x = zeros(m,n);

sobel\_image\_y = zeros(m,n);

%set the sobel filter with x and y direction

sobel\_filter\_x = [-1,0,1;-2,0,2;-1,0,1];

sobel\_filter\_y = [-1,-2,-1;0,0,0;1,2,1];

%duplicate the edge

X\_dup = edged\_dup(X,1);

% ============================================

%iterate the sobel filter with image

for i = 1:m

for j = 1:n

temp\_X = X\_dup(i:i+2,j:j+2);

sobel\_image\_x(i,j) = sum(sum(sobel\_filter\_x .\* double(temp\_X)));

sobel\_image\_y(i,j) = sum(sum(sobel\_filter\_y .\* double(temp\_X)));

end

end

%calculate gradient magnitude

sobel\_image = sqrt(sobel\_image\_x.^2 + sobel\_image\_y.^2);

%calculate the gradient direction

sobel\_ori = atan((sobel\_image .\* sobel\_image\_y)./(sobel\_image .\* sobel\_image\_x));

sobel\_ori(isnan(sobel\_ori))=0;

%apply threshold to filted image

sobel\_image = sobel\_image > threshold;

sobel\_image = uint8(255\*sobel\_image);

end

**NON-MAXIMUM SUPRESSION**

function nms\_image = non\_max\_sup(X,threshold)

%Initialization

[m,n] = size(X);

nms\_image = zeros(m,n);

[X\_image, X\_ori] = sobel\_filt(X, threshold);

%convert non-nomorlized sobel image

X\_sobel = double(X) .\* double(X\_image/255);

% ============================================

%iterate the non-maxium supression with image

%split to four situation: horizontal, vertical, two diagonals

for i = 2:(m-1)

for j = 2:(n-1)

if X\_ori(i,j)==0

continue

elseif X\_ori(i,j) >= (-3/8 \*pi) & X\_ori(i,j) < (-1/8 \*pi)

nms\_image(i,j) = X\_sobel(i,j)\*(max([X\_sobel(i,j),X\_sobel(i-1,j+1), X\_sobel(i+1,j-1)])==X\_sobel(i,j));

elseif X\_ori(i,j) >= (-1/8\*pi) & X\_ori(i,j) < (1/8\*pi)

nms\_image(i,j) = X\_sobel(i,j)\*(max([X\_sobel(i,j),X\_sobel(i-1,j), X\_sobel(i+1,j)])==X\_sobel(i,j));

elseif X\_ori(i,j) >= (1/8 \*pi) & X\_ori(i,j) < (3/8 \*pi)

nms\_image(i,j) = X\_sobel(i,j)\*(max([X\_sobel(i,j),X\_sobel(i+1,j+1), X\_sobel(i-1,j-1)])==X\_sobel(i,j));

else

nms\_image(i,j) = X\_sobel(i,j)\*(max([X\_sobel(i,j),X\_sobel(i,j-1), X\_sobel(i,j+1)])==X\_sobel(i,j));

end

end

end

nms\_image = nms\_image./nms\_image \*255;

nms\_image(isnan(nms\_image)) = 0;

nms\_image = uint8(nms\_image);

end