1.

img = rgb2gray(img);

img = double(img);

[n,m] = size(filter);

[x,y] = size(img);

padded = padarray(img, [floor(n/2) floor(m/2)]); %zero padding the image to match the size of the filter

flipped = flip(flip(filter,1),2); %flipping filter for convolution

flipped\_array = reshape(flipped, 1, n\*m); %creating vector from the flipped filter to do dot product

for i = 1:(x)

for j= 1:(y)

patch = padded(i:i+(n-1), j:j+(m-1)); %taking a patch of the image

patch\_array = reshape(patch, 1, n\*m); %vectorizing patch

G(i,j)= dot(flipped\_array, patch\_array);

end

end

figure;imagesc(G);

(a)

(b)

It is possible to write a convolution in one pixel as a dot product between two vectors, you have to take a patch of the image that is the same size as the filter, vectorize them both (after flipping the filter on both axes) and then do dot product to get the pixel value (this is done in the code above). Reference: Lecture 2 page 43 & 45.

It is possible to write a full convolution between the image and the filter as a multiplication between a matrix and a vector. For an *n* x *n* image and an *m* x *m* filter, you would need to make *n* patches of the image with a size of *m* x *m.* If all these patches were vectorized and put into a matrix, you would have a  *n2*x *m* matrix. The vectorized filter would be a vector of size *m*. If you multiplied these two together, the resulting vector of size *n2* is the fully convolved image pixel values in a vector.

2.

(a)

Computational cost of the convolution: *n2m2*

If separable: 2*n2m*

Ref: Lecture 2 page 50

(b)

Ref: Lecture 2 page 57

(c)

function out = anisog(sigmax, sigmay)

k = 2\*(sigmax+sigmay)+mod(sigmax+sigmay,2);%The filter support

x = [-k:k];%for a [kxk] filter

y = x';

c = 1/(2\*pi\*sigmax\*sigmay);%The normalization constant

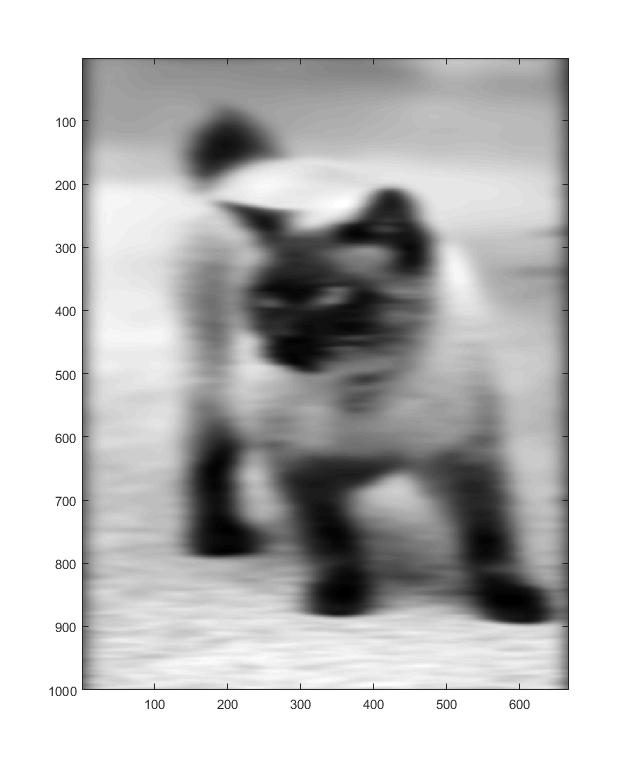
out = c\* exp(-0.5\*(((x.^2)/sigmax^2)+((y.^2)/sigmay^2)));

%equation from http://mathinfo.univ-reims.fr/IMG/pdf/Fast\_Anisotropic\_Gquss\_Filtering\_-\_GeusebroekECCV02.pdf

end

(d)

>>figure;imagesc(conv2(double(rgb2gray(imread('cat.jpg'))),anisog(15,2),'same'));colormap(gray)



(e)

Convolve on each axis separately, first horizontal and then vertical.

Vertical Filter =

Horizontal Filter =

Ref: Lecture 2 page 50

3.

(a)

function out = magradient(im, sigma)

im = rgb2gray(im);

im = double(im);

k = 2\*sigma+mod(sigma, 2);%The filter support

x = [-k:k];%for a [kxk] filter

y = x';

c = 1/(sqrt(2\*pi)\*sigma);%The normalization constant

gx = c\* exp(-(x.^2)/(2\*sigma^2));

gy = c\* exp(-(y.^2)/(2\*sigma^2));

g = gy\*gx;

g\_x = (-x/(sigma^2)).\*g; %The partial derivative of g w.r.t. x

g\_y = (-y/(sigma^2)).\*g'; %The partial derivative of g w.r.t. x

grd\_x = conv2(im,g\_x,'same');

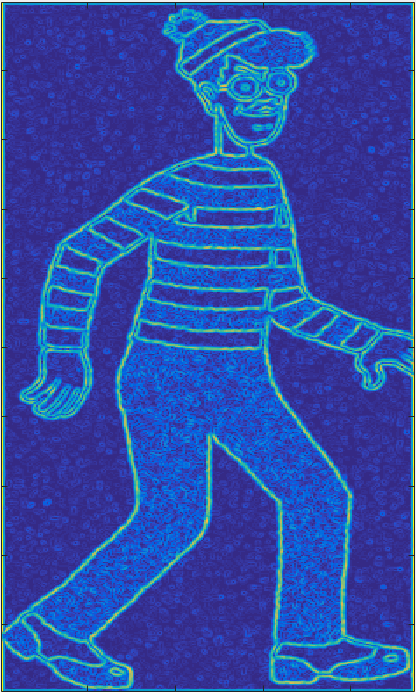
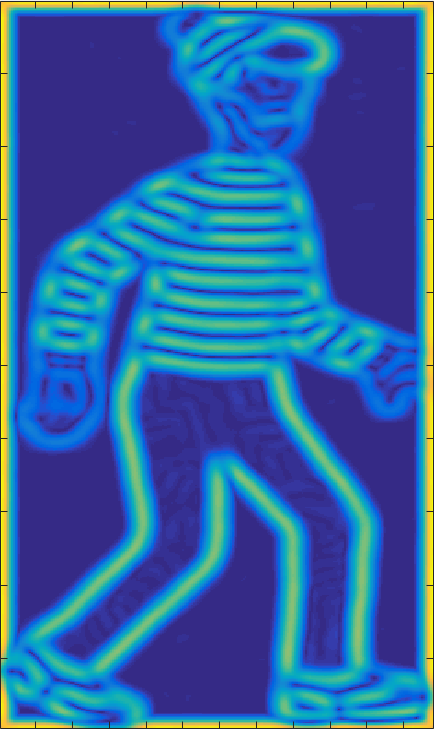
grd\_y = conv2(im,g\_y,'same');

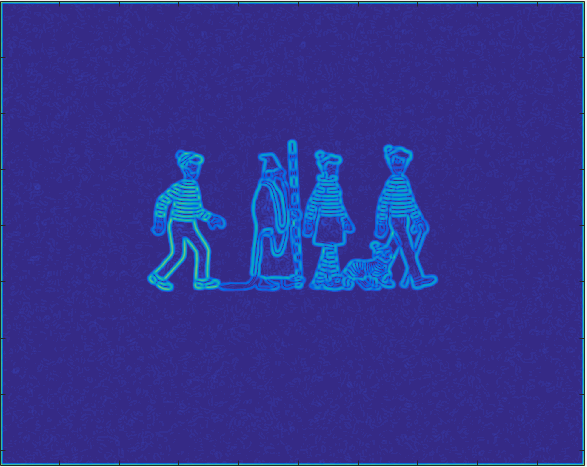
moe = sqrt((grd\_x.^2) + (grd\_y.^2));

out = moe;

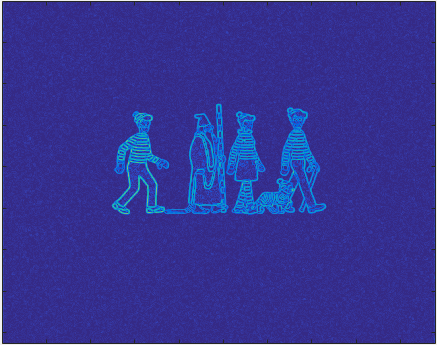
figure;

imagesc(moe);

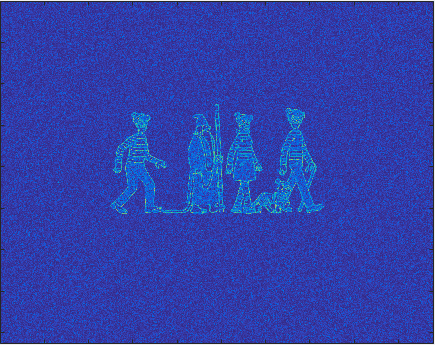
 Sigma = 1 Sigma = 3 Sigma = 5



Sigma = 5



Sigma = 3



Sigma = 1

(b) Additions to findWaldo.m highlighted.

function out = findWaldo(im, filter)

% returns the output of normalized cross-correlation between image im and

% filter

% by Sanja Fidler, UofT

% convert image (and filter) to grayscale

im\_input = im;

% get magnitude of gradients of the image and filter

im = magradient(im, 3);

filter = magradient(filter, 3);

filter = filter/sqrt(sum(sum(filter.^2)));

% normalized cross-correlation

out = normxcorr2(filter, im);

% plot the cross-correlation results

figure('position', [100,100,size(out,2),size(out,1)]);

subplot('position',[0,0,1,1]);

imagesc(out)

axis off;

axis equal;

% find the peak in response

[y,x] = find(out == max(out(:)));

y = y(1) - size(filter, 1) + 1;

x = x(1) - size(filter, 2) + 1;

% plot the detection's bounding box

figure('position', [300,100,size(im,2),size(im,1)]);

subplot('position',[0,0,1,1]);

imshow(im\_input)

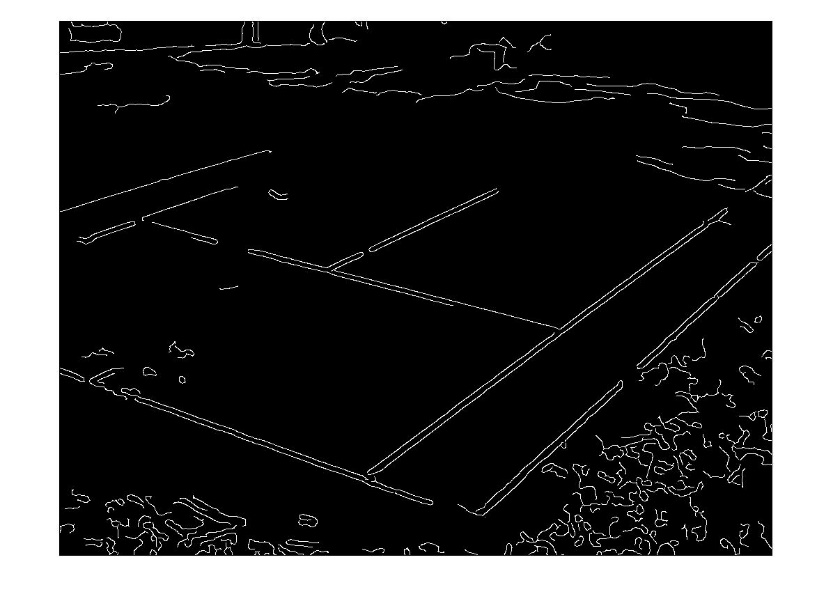
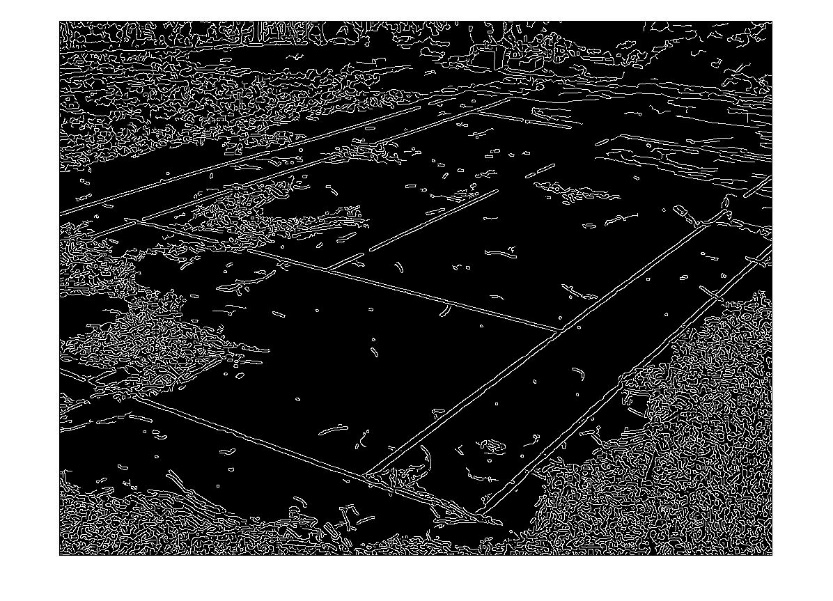
axis off;

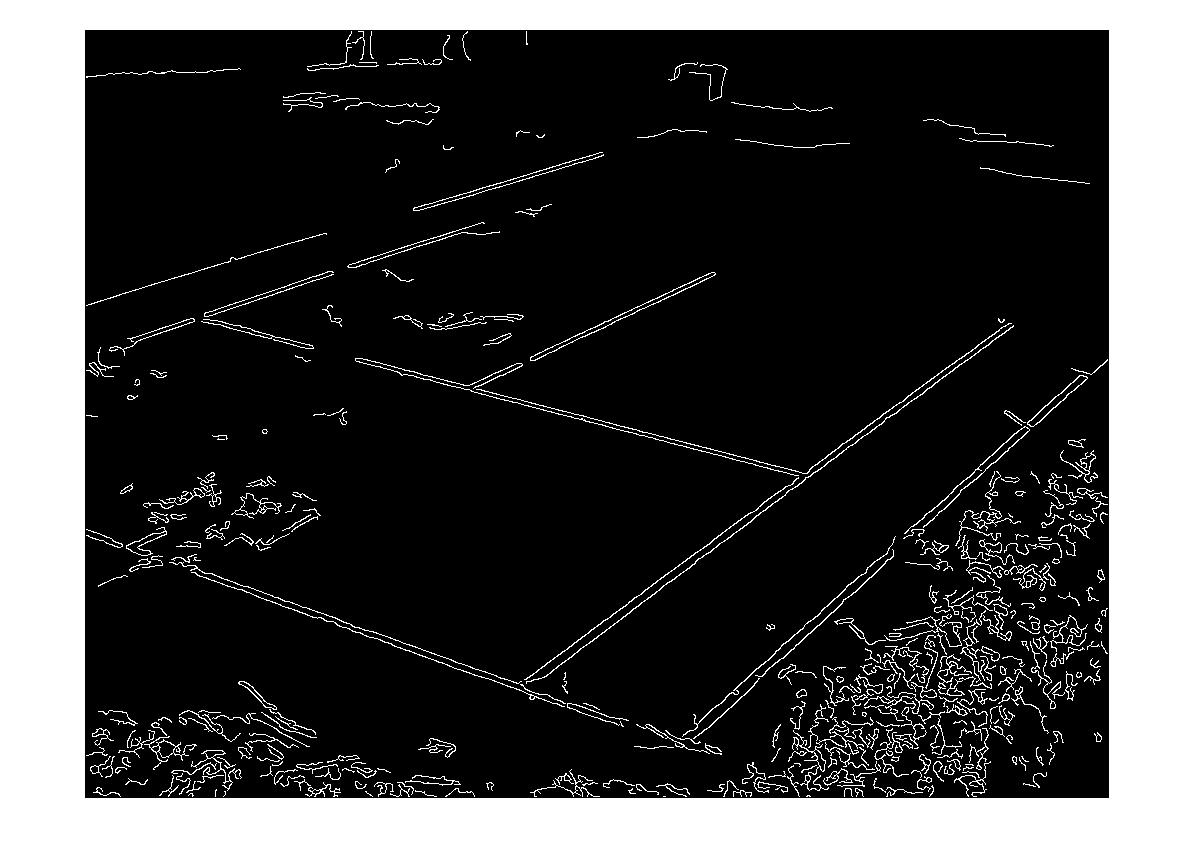
axis equal;

rectangle('position', [x,y,size(filter,2),size(filter,1)], 'edgecolor', [0.1,0.2,1], 'linewidth', 3.5);

end

4.

(a) Default Threshold: 0.5, Sigma: 3



Threshold: 0.5, Sigma 1.4

(b)

Seeing as they are the only long straight lines in this image, isolating