Jerry Chen

CS558

HW1 Report

My project is done in MATLAB and the only image-processing related built-in functions used are imread and imfilter . The script prompts for two inputs when ran, the filename which have to be enclosed in ‘ ‘ and the sigma value. I find that the value that works the best and produces the clearest result is 2 so that is what I have used in all of my examples. Before putting the image through any filters I convert it into a double image so that I could use the sqrt function later and get better results. The gaussian filter is created using for loops and the gaussian equation with the size of the filter being 6*sigma+1 as recommended in the slides. Then I use imfilter with the original image and the gaussian filter to get the gaussian image. Next, I defined a sobel filter in matrix form and filter it with the gaussian image twice, one with the inverse sobel, to get the horizontal and vertical gradients. I got the normal gradient image by using the sqrt(x^2+y^2) equation and also got the directions of the gradient with the inverse tangent equation. The gradient image is then thresholded using for loops and zeroing any pixels with intensity less than 0.2 which produced images that were close to the examples on the slides. Finally for the non-maximum suppression I classified each pixel as either vertical, right diagonal, horizontal, or left diagonal using the gradient direction matrix from earlier, the range was from -90 to 90 in the first and fourth quadrant of the plane if you visualize it. Depending on the classification I would compare it with its orthogonal neighbors and keep only the greater one to get the final edge image.

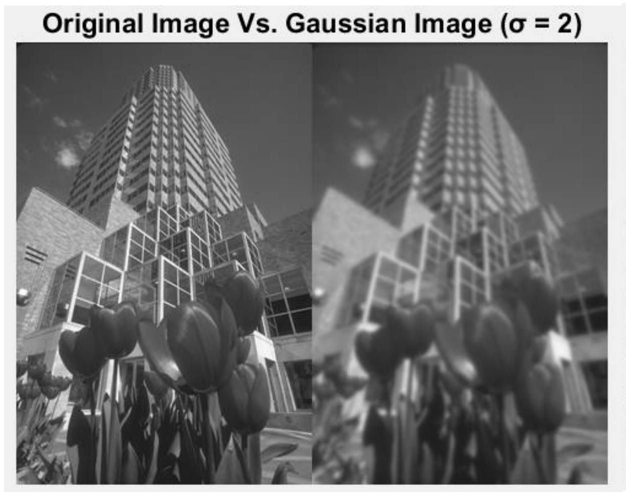
Source Code

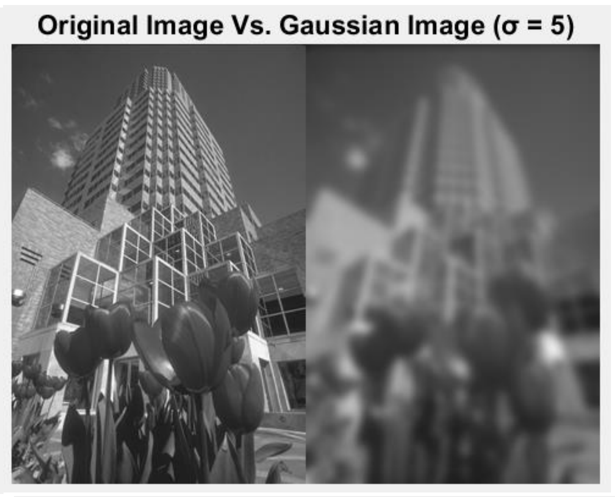
```
imname = input('Select image ');
I = imread(imname);
I = im2double(I);

%gaussian filter
sigma = input('Set sigma ');
sz = sigma*6 + 1;
g = zeros(sz,sz);
center = sigma*3 + 1;
for i = 1:sz
    for j = 1:sz
        g(i,j) = exp(-((i-center).^2+(j-
center).^2)/(2*sigma.^2))/(2*pi*sigma.^2);
    end
end
GF1 = imfilter(I,g,'replicate');

sobelmatrix = [1 0 -1; 2 0 -2; 1 0 -1];
Gx = imfilter(GF1, sobelmatrix', 'replicate');
Gy = imfilter(GF1, sobelmatrix, 'replicate');
Mag = sqrt(Gx.^2 + Gy.^2);
Norm = Mag;
Dir = atand(Gy./Gx);
%threshold gradient for 0.2
for i = 1:size(Mag,1)
    for j = 1:size(Mag,2)
```

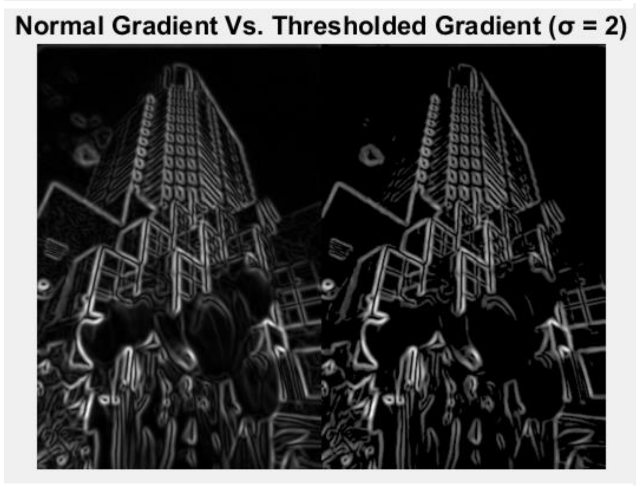
```
        if Mag(i,j) < 0.15
            Mag(i,j) = 0;
        end
    end
end

%non-maximum suppression
Thresh = Mag;
ht = size(Mag,1);
wd = size(Mag,2);
for i = 1:ht
    for j = 1:wd
        if Mag(i,j) ~=0
            degrees = Dir(i,j);
            neighbor1 = 0;
            neighbor2 = 0;
            if (degrees >= 67.5 || degrees < -67.5)
                %vertical
                if j < wd
                    neighbor1 = Mag(i,j+1);
                end
                if j > 1
                    neighbor2 = Mag(i,j-1);
                end
                if (Mag(i,j) >= neighbor1)
                    if neighbor1 ~= 0
                        Mag(i,j+1) = 0;
                    end
                end
                if (Mag(i,j) >= neighbor2)
                    if neighbor2 ~= 0
                        Mag(i,j-1) = 0;
                    end
                end
                if (Mag(i,j) < neighbor1 || Mag(i,j) < neighbor2)
                    Mag(i,j) = 0;
                end
            end
            if (degrees >= 22.5 && degrees < 67.5)
                %45diagonal
                if (i < ht && j < wd)
                    neighbor1 = Mag(i+1,j+1);
                end
                if (i > 1 && j > 1)
                    neighbor2 = Mag(i-1,j-1);
                end
                if (Mag(i,j) >= neighbor1)
                    if neighbor1 ~= 0
                        Mag(i+1,j+1) = 0;
                    end
                end
                if (Mag(i,j) >= neighbor2)
                    if neighbor2 ~= 0
                        Mag(i-1,j-1) = 0;
                    end
                end
                if (Mag(i,j) < neighbor1 || Mag(i,j) < neighbor2)
                    Mag(i,j) = 0;
                end
            end
            if (degrees >= -22.5 && degrees < 22.5)
                %horizontal
                if i < ht
                    neighbor1 = Mag(i+1,j);
                end
                if i > 1
                    neighbor2 = Mag(i-1,j);
                end
                if (Mag(i,j) >= neighbor1)
                    if neighbor1 ~= 0
                        Mag(i+1,j) = 0;
                    end
                end
                if (Mag(i,j) >= neighbor2)
                    if neighbor2 ~= 0
                        Mag(i-1,j) = 0;
                    end
                end
                if (Mag(i,j) < neighbor1 || Mag(i,j) < neighbor2)
                    Mag(i,j) = 0;
                end
            end
            if (degrees >= -67.5 && degrees < -22.5)
                %-45diagonal
                if (i > 1 && j < wd)
                    neighbor1 = Mag(i-1,j+1);
                end
                if (i < ht && j >1)
                    neighbor2 = Mag(i+1,j-1);
                end
                if (Mag(i,j) >= neighbor1)
                    if neighbor1 ~= 0
                        Mag(i-1,j+1) = 0;
                    end
                end
                if (Mag(i,j) >= neighbor2)
                    if neighbor2 ~= 0
                        Mag(i+1,j-1) = 0;
                    end
                end
                if (Mag(i,j) < neighbor1 || Mag(i,j) < neighbor2)
                    Mag(i,j) = 0;
                end
            end
```

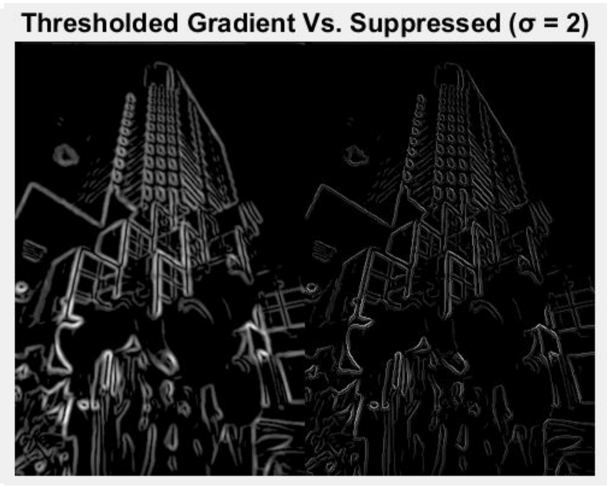
```
        end
    end
end
montage({I,GF1,Gx,Gy,Norm,Thresh,Mag})
```

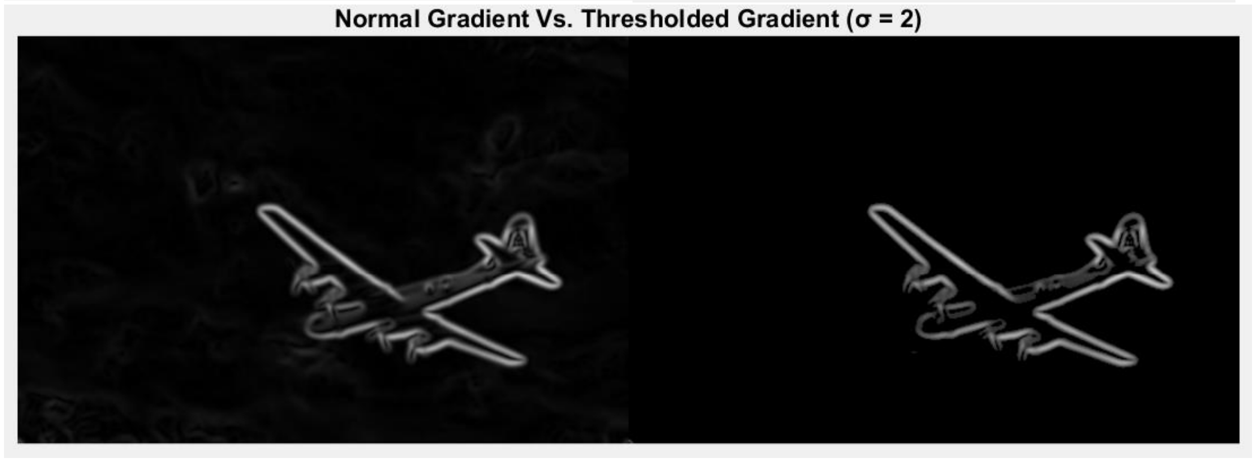
Images











Normal Gradient Vs. Thresholded Gradient (σ = 2)