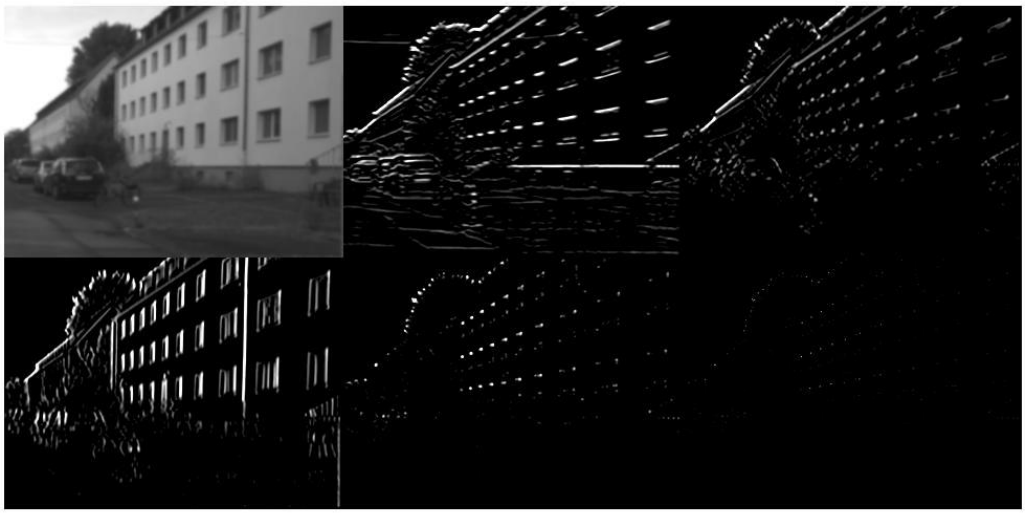
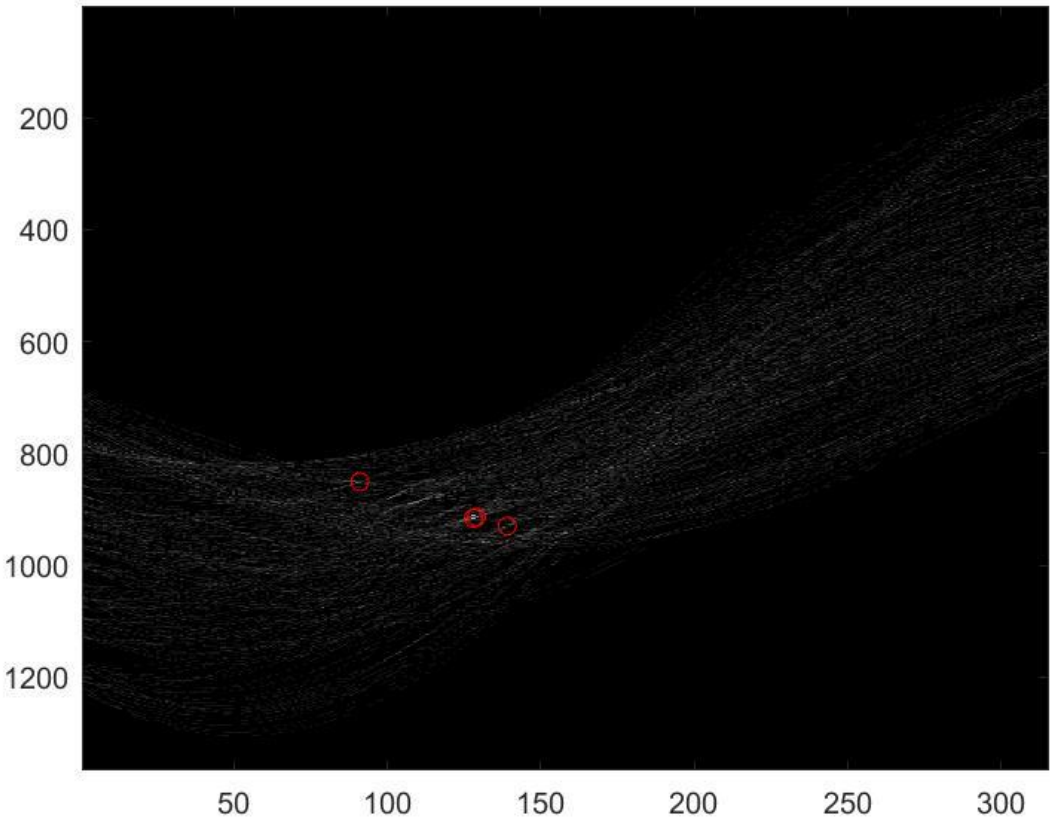
Jerry Chen

CS558

HW2 Report

The assignment is encoded in Matlab with a main script that calls onto multiple supporting functions for computations. Image pre-processing is done by first applying a gaussian filter with sigma 2, then applying the second derivative sobel filters which are acquired by convolving the respective sobel filters for xx, xy, and yy. Next the images are used to derive the hessian determinant which is then thresholded and put through non-maximum suppression in 3*3 neighborhoods to get the points. The ransac function takes in as parameters the hessian image, s(number of points), t(threshold), and p(probability) and returns the best line and inliers on that line as output. After getting the line and inliers they are plotted onto the original image with the inliers plotted as 3*3 squares and then deleted before the next iteration so that the same inliers will not be used again. After the loop ends, we get the four lines with strongest support from ransac. For the hough transformation, the function takes in the hessian image, rho bin and theta bin as parameters and returns as output H, rhos, and thetas. The highest values on H are found and plotted as circles and then reverted back into cartesian using the rhos and thetas and plotted onto the original image as the four lines with strongest support. There are two lines in the hough image that are really close to each other due to two nearby points in the hough space having high votes, this could be solved by deleting neighboring points as well instead of just the one point that was plotted.




200
400
600
800
1000
1200
50
100
150
200
250
300

```
I = imread('road.png');
I = im2double(I);
%Apply gaussian filter
sigma = 2;
GF = GaussianFilter(I,sigma);
%Apply sobel filters
ysobel = [1 0 -1; 2 0 -2; 1 0 -1];
xsobel = ysobel';
threshold = 0.1;
xx = conv2(xsobel,xsobel);
xy = conv2(xsobel,ysobel);
yy = conv2(ysobel,ysobel);
Gxx = myfilter(GF, xx);
Gxx = thresh(Gxx,threshold);
Gxy = myfilter(GF, xy);
Gxy = thresh(Gxy,threshold);
Gyy = myfilter(GF, yy);
Gyy = thresh(Gyy,threshold);
%Hessian determinant thresholding
D = (Gxx.*Gyy)-((Gxy).^2);
D = thresh(D,threshold);
%Non-maximum suppression in 3*3 neighborhoods
```

```
S = suppression(D);
montage({GF,Gxx,Gxy,Gyy,D,S})

%ransac for 4 lines
Temp = S;
t = 1;
p = 0.95;
f = figure; imshow(I), hold on;
for n = 1:4
    [line,inliers] = ransac(Temp,2,t,p);
    %find extreme inliers and plot the line
    [~, miny] = min(inliers(:, 1));
    [~, maxy] = max(inliers(:, 1));
    figure(f), hold on;
    plot(inliers([miny maxy],1), inliers([miny maxy],2),
"LineWidth", 1), hold on;
    %plot inliers as 3*3 squares
    for i=1:length(inliers)
        t_idx = inliers(i,1);
        r_idx = inliers(i,2);
        [x2, y2] = meshgrid(t_idx-1:t_idx+1, r_idx-1:r_idx+1);
        hold on;
        scatter(x2(:), y2(:), 'square', 'y');
        %remove inliers after fit
        Temp(r_idx,t_idx) = 0;
    end
    hold off;
end

%hough transform for 4 lines
Temp = S;
rho_bin = 1;
theta_bin = 0.01;
[H,rho,theta] = hough_t(Temp,rho_bin,theta_bin);
f2 = figure; imagesc(H), colormap gray, hold on;
f3 = figure; imshow(I); hold on;
for i = 1:4
    %Find point with most votes and plot
    [m,idx] = max(H);
    [~,t_idx] = max(m);
    r_idx = idx(t_idx);
    figure(f2); scatter(t_idx,r_idx,'r');
    %Convert back to cartesian and plot
    r = rho(r_idx);
    t = theta(t_idx);
    x = 1:size(I,2);
    y = (r - x*cos(t))/sin(t);
```

```
    figure(f3); plot(x,y,'LineWidth',1);
    %Remove point after fit
    H(r_idx,t_idx) = 0;
end


function [H,rho_bin,theta_bin] = hough_t(img,rho_bin,theta_bin)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
[y,x] = find(img > 0);
feature_points = [x y];
total_points = length(feature_points);
max_rho = norm(size(img));
rho_bin = -max_rho:rho_bin:max_rho;
theta_bin = 0:theta_bin:pi;
H_ht = length(rho_bin);
H_wd = length(theta_bin);
H = zeros(H_ht,H_wd);

for i = 1:total_points
    x = feature_points(i,1);
    y = feature_points(i,2);
    for j = 1:H_wd
        theta = theta_bin(j);
        rho = x*cos(theta) + y*sin(theta);
        rho_idx = round(rho + H_ht/2);
        H(rho_idx,j) = H(rho_idx,j) + 1;
    end
end
end


function [best_line, best_inliers] = ransac(img,s,t,p)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
[y,x] = find(img > 0);
feature_points = [x y];
total_points = length(feature_points);
N = Inf;
count = 0;
best_count = 0;
best_inliers = [];
best_line = [];
while N > count
    index1 = 0;
    index2 = 0;
    while (index1==index2)
```

```
        index1 = randi(total_points);
        index2 = randi(total_points);
    end
    p1 = feature_points(index1,:);
    p2 = feature_points(index2,:);

    a = p1(2)-p2(2);
    b = p2(1)-p1(1);
    d = p1(2)*p2(1)-p1(1)*p2(2);
    distance = zeros(total_points,1);
    for i = 1:total_points
        x = feature_points(i,1);
        y = feature_points(i,2);
        distance(i) = (abs(a*x+b*y-d))/(sqrt(a^2+b^2));
    end
    inliers = find(distance <= t);
    if (length(inliers) > best_count)
        best_count = length(inliers);
        best_inliers = feature_points(inliers,:);
        best_line = [a b d];
    end
    e = 1 - length(inliers)/total_points;
    N = log(1-p)/log(1-power((1-e),s));
    count = count + 1;
end
end
```

```
function result = myfilter(img,ft)
%UNTITLED Summary of this function goes here
%   filter with no padding
[ht,wd] = size(img);
result = zeros(ht,wd);
x = size(ft, 1) - 1;
for i = 1:(ht - x)
    for j = 1:(wd - x)
        tmp = img(i:i+x, j:j+x).*ft;
        result(i, j) = sum(tmp(:));
    end
end
end

function result = thresh(img,threshold)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
[ht,wd] = size(img);
```

```
result = zeros(ht,wd);
for i = 1:ht
    for j = 1:wd
        if img(i,j) < threshold
            result(i,j) = 0;
        else
            result(i,j) = img(i,j);
        end
    end
end
end
function result = suppression(img)
%UNTITLED2 Summary of this function goes here
%   non-maximum suppression in 3*3 neighborhoods
[ht,wd] = size(img);
result = zeros(ht,wd);
for i = 2:(ht - 1)
    for j = 2:(wd - 1)
        nb = img(i-1:i+1, j-1:j+1);
        if img(i,j) == max(nb(:))
            result(i,j) = img(i,j);
        end
    end
end
end
```