CS 558: Homework Assignment 2 -- Line Detection
Program Language: MATLAB
Name: Lan Chang
Date: 03/22/17

1. Pre-processing

First, I apply Gaussian filter using the function in Homework1. Next, I use the Sobel filter as derivative operator to calculate the determinant of Hessian and extract features. Then, I use a threshold (=2) on the determinant. Finally, I apply non-maximum suppression and get the features.

Result:

2. RANSAC

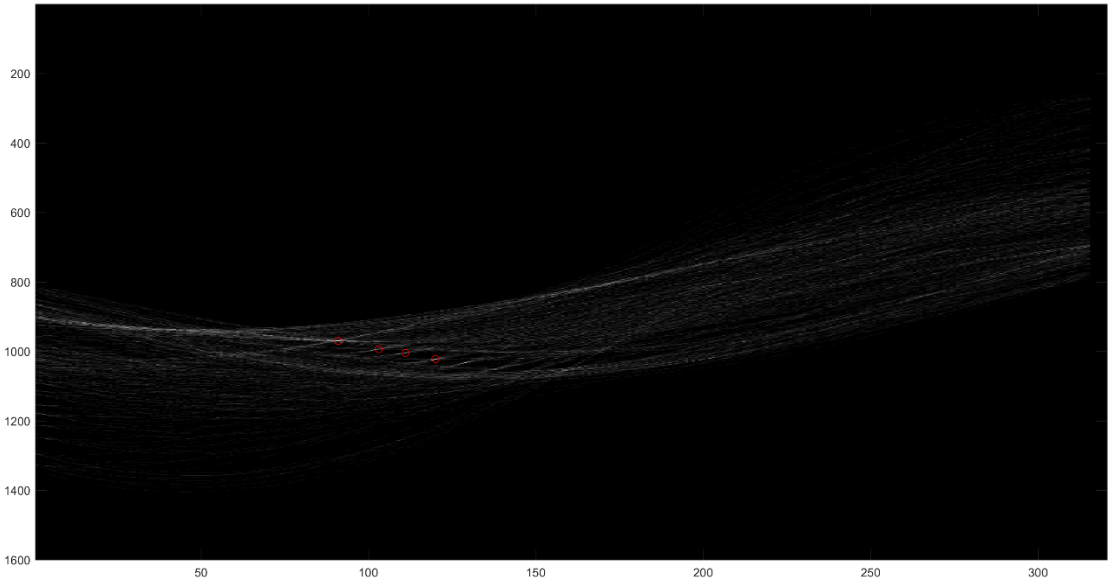
After pre-processing, I apply RANSAC on these points. For detecting each line, I first randomly select two points, hypothesize a model like ax + by = d and calculate a, b and d, and calculate the distance between each point and this line. Next, if the distance is smaller than the distance threshold (=1), I will select this point as inlier and count the number of these points. Then, if this model has the largest inliers, I will choose this model as the best select the best model. Finally, I draw the line and the inliers on the image, and delete these inliers.

Result:

3. Hough Transform

First, I initialize an accumulator, the bin size of rho is 1 and the bin size of theta is 0.01. Next, for each points, I calculate the rho for each theta, and fill the accumulator with this pair of rho and theta. Then, I select the bin with maximum number and get the pair of rho and theta. Finally, I draw the line using this pair of rho and theta, and clear this bin.

Accumulator with four bins:



Result:

4. MATLAB Code

(1) homework2.m

```
clear all;

% input the image
image=imread('road.png');
height=size(image,1);
width=size(image,2);

% Problem 1: pre-processing
% apply Gaussian filter
sigma=1;
size=sigma*6+1;
half=(size-1)/2;
[x,y]=meshgrid(-half:half,-half:half);
g_filter=exp(-(x.^2+y.^2)/(2*sigma^2))/(2*pi*sigma^2);
g_filter=g_filter./sum(g_filter(:));
image1=im2double(image);
image2=filtering(image1,g_filter);

% use the Sobel filters as derivative operators
s_filter_x=[-1,0,1;-2,0,2;-1,0,1];
s_filter_y=[1,2,1;0,0,0;-1,-2,-1];
i_x=filtering(image2,s_filter_x);
i_y=filtering(image2,s_filter_y);
i_xx=filtering(i_x,s_filter_x);
i_yy=filtering(i_y,s_filter_y);
i_xy=filtering(i_x,s_filter_y);

% hessian
hessian_threshold=2;
image3=i_xx.*i_yy-i_xy.*i_xy;
image3(image3<hessian_threshold)=0;

% apply non-maximum suppression
image4=zeros(height,width);
for i=2:height-1
    for j=2:width-1
        temp=image3(i-1:i+1,j-1:j+1);
        if max(temp(:))==image3(i,j)
            image4(i,j)=image3(i,j);
        end
    end
end
```

```
% show the result
figure,imshow(image4);

lines_num=4;

% Problem 2: RANSAC
% all points
[y,x]=find(image4>0);
points=[x y];

% parameters for RANSAC
t=1;       % distance threshold
s=2;       % initial number of points
p=0.95; % probability for inlier

% show the result
f1=figure;imshow(image),hold on;
for n=1:lines_num
      points_num=length(points);
      N=inf;
      count=0;
      inliers_num_best=0;
      inliers_idx_best=[];

      while N>count
            % randomly select 2 points
            p1_idx=0;
            p2_idx=0;
            while (p1_idx==p2_idx||p1_idx==0||p2_idx==0)
                  p1_idx=round(rand*points_num);
                  p2_idx=round(rand*points_num);
            end
            p1=points(p1_idx,:);
            p2=points(p2_idx,:);

            % hypothesize a model(ax+by=d)
            a=p1(2)-p2(2);
            b=p2(1)-p1(1);
            d=p1(2)*p2(1)-p1(1)*p2(2);

            % compute error function(calculate the distance)
            dis=inf(points_num,1);
            for i=1:points_num
```

```
            dis(i)=abs(a*points(i,1)+b*points(i,2)-d)/sqrt(a*a+b*b);
        end

        % select points consistent with model
        inliers_idx=find(dis<=t);
        inliers_num=length(inliers_idx);

        % select the best model
        if inliers_num>inliers_num_best
            inliers_num_best=inliers_num;
            inliers_idx_best=inliers_idx;
            a_best=a;
            b_best=b;
            d_best=d;
        end

        e=1-inliers_num/points_num;
        N=log(1-p)/log(1-power((1-e),s));
        count=count+1;
    end

    % all inliers
    inliers=points(inliers_idx_best,:);

    % draw the line
    [~,idx1]=min(inliers(:,1));
    [~,idx2]=max(inliers(:,1));
    figure(f1);
    line_x=inliers(idx1,1):inliers(idx2,1);
    line_y=(d_best-a_best.*line_x)./b_best;
    plot(line_x,line_y,'b');

    % draw the point
    for i=1:inliers_num_best
        point_x=inliers(i,1);
        point_y=inliers(i,2);
        rectangle('Position',[point_x-1,point_y-1,3,3],'Curvature',[0
0],'FaceColor','r','edgecolor','r')
    end

    % delete used points
    points(inliers_idx_best,:)=[];
end
```

```
% Problem 3: Hough Transform
% all points
[y,x]=find(image4>0);
points=[x y];
points_num=length(points);

% parameters for accumulator
t_bin=0.01;
r_bin=1;

% initialize accumulator
h_height=1600;
h_width=320;
H=zeros(h_height,h_width);

% fill the accumulator
for i=1:points_num
    point_x=points(i,1);
    point_y=points(i,2);
    for t=0:t_bin:pi
        r=point_x*cos(t)+point_y*sin(t);
        i1=round(r+h_height/2);
        i2=round(t*100+1);
        H(i1,i2)=H(i1,i2)+1;
    end
end

% show the accumulator
f2=figure;imagesc(H),colormap('gray'),hold on;

% show the result
f3=figure;imshow(image),hold on;
for n=1:lines_num
    [~,idx]=max(H(:));

    % find the rho and theta
    r_idx=mod(idx,h_height);
    t_idx=(idx-r_idx)/h_height+1;
    r=r_idx-h_height/2;
    t=(t_idx-1)/100.0;
    figure(f2);
    scatter(t_idx,r_idx,'r');

    % draw the line
```

```
        line_x=1:width;
        line_y=(r-line_x.*cos(t))/sin(t);
        figure(f3);
        plot(line_x,line_y,'b');

        % delete used points
        i3=r_idx-1:r_idx+1;
        i4=t_idx-1:t_idx+1;
        H(i3,i4)=0;
end
```

(2) filtering.m

```
function image_fil=filtering(image_ori,filter)
% width and height of original image
width_i=size(image_ori,2);
height_i=size(image_ori,1);

% width and height of filter
width_f=size(filter,2);
height_f=size(filter,1);
half_wf=(width_f-1)/2;
half_hf=(height_f-1)/2;

% extend
image_temp1=zeros(height_i+half_hf*2,width_i+half_wf*2);
for i=1:height_i
    for j=1:width_i
        image_temp1(i+half_hf,j+half_wf)=image_ori(i,j);
    end
end

% replicate boundary
for i=1:height_i+half_hf*2
    for j=1:width_i+half_wf*2
        if j<=half_wf&&i>=half_hf+1&&i<=height_i+half_hf
            image_temp1(i,j)=image_temp1(i,half_wf+1);
        elseif j>=width_i+half_wf+1&&i>=half_hf+1&&i<=height_i+half_hf
            image_temp1(i,j)=image_temp1(i,width_i+half_wf);
        elseif i<=half_hf&&j>=half_wf+1&&j<=width_i+half_wf
            image_temp1(i,j)=image_temp1(half_hf+1,j);
        elseif i>=height_i+half_hf+1&&j>=half_wf+1&&j<=width_i+half_wf
            image_temp1(i,j)=image_temp1(height_i+half_hf,j);
        elseif j<=half_wf&&i<=half_hf
            image_temp1(i,j)=image_temp1(half_hf+1,half_wf+1);
        elseif j<=half_wf&&i>=height_i+half_hf+1
            image_temp1(i,j)=image_temp1(height_i+half_hf,half_wf+1);
        elseif i<=half_hf&&j>=width_i+half_wf+1
            image_temp1(i,j)=image_temp1(half_hf+1,width_i+half_wf);
        elseif j>=width_i+half_wf+1&&i>=height_i+half_hf+1
            image_temp1(i,j)=image_temp1(height_i+half_hf,width_i+half_wf);
        end
    end
end

% filtering
```

```
image_temp2=image_temp1;
for i=1+half_hf:height_i+half_hf
      for j=1+half_wf:width_i+half_wf
            image_temp2(i,j)=sum(sum(filter.*image_temp1(i-half_hf:i+half_hf,j-
half_wf:j+half_wf)));
      end
end

% cut
image_fil=zeros(height_i,width_i);
for i=1:height_i
      for j=1:width_i
            image_fil(i,j)=image_temp2(i+half_hf,j+half_wf);
      end
end

end
```