CSCI-631: Foundations of Computer Vision

Submitted by: - Yash Jain (yj8359)

HOMEWORK – 11

Image chaining:

- Read the image and convert it to double.

```
im = im2double(imread( file_name ));
```

- Since we want stronger edges, we first smear the image using Gaussian filter. This will remove many of the unwanted edges.

```
% Smoothen image to remove unwanted noisy edges by applying gaussian filter
 gauss = fspecial('Gaussian',3,2);
 im_filtered = imfilter(im, gauss, 'same', 'repl');
```

- Find edges in the image using sobel filter. Compute edge magnitude and edge angle.

```
% Apply sobel filter to find the edges in the image
  sobel_vr_gradient = [-1 0 1;
                       -2 0 2;
                       -1 0 1] / 8;

   sobel_hr_gradient = sobel_vr_gradient.';

   % Finding Sobel horizontal and vertical edges
   im_horizontal_edges = imfilter(im_filtered, sobel_vr_gradient, 'same', 'repl');
   im_vertical_edges = imfilter(im_filtered, sobel_hr_gradient, 'same', 'repl');

   % Finding Sobel Edge Magnitude and angle
   edge_mag = ((im_horizontal_edges).^2 + (im_vertical_edges).^2).^(1/2);
   edge_angle = atan2(im_vertical_edges, im_horizontal_edges);
```

- Select the best color channel.

```
% I played with all 3 color channels in RGB space and found green as
% the best channel
    im_channel = edge_mag(:,:,2);
```

- For all angles specified in homework PDF document (-135, -45, 45, 135), compute brightest edges. I have computed top 8% edges. Also, we have to work with angle tolerance (-22 to 22) as mentioned in homework PDF. But since the angles that we use to obtain from the sobel are in radians so convert angle value in radians for the angles given (-135,-45,45,135) considering +22 and -22 tolerance.

```
% Considering -22 to +22 degree tolearance as disccussed in PDF
       low_tolerance_in_degree = angle_for_current_iteration - 22;
       high_tolerance_in_degree = angle_for_current_iteration + 22;

       % Since all the angle values we have obtained by sobel are in
       % radinans we need to convert the working angle in radians
       low_tolerance_in_radians = deg2rad(low_tolerance_in_degree);
       high_tolerance_in_radians = deg2rad(high_tolerance_in_degree);

       % consider only those angle values that are in range of low_tolerance to high_tolerance.
       % Rest all values are 0.
       % Do this in a desired color channel (I played with all 3 color channels of RGB and
         found green works the best)
       edge_angles_of_a_channel = edge_angle(:,:,2);
       edge_angles_of_a_channel(edge_angles_of_a_channel < low_tolerance_in_radians ) = 0;
       edge_angles_of_a_channel(edge_angles_of_a_channel > high_tolerance_in_radians) = 0;

       % Select only the edge angles that we are in range
       edge_angles_only_in_range = edge_angles_of_a_channel ~= 0;

% Computing strongest edge magnitude
edge_mag_corresponding_to_selected_angle = im_channel( edge_angles_only_in_range );

% sort all the edges
[ sorted_values ]   = sort( edge_mag_corresponding_to_selected_angle(:) );

% Find the brightest edge pixels from the selected edges in above step
% Taking top 8% of edges.(Best edges or strongest edges)
temp_brightest_pixels =  round( numel( edge_mag_corresponding_to_selected_angle ) * (1-0.08) );
```

```
% Find the threshold value
threshold_of_brightness    = sorted_values( temp_brightest_pixels );
best_edge_pixels      = im_channel;

% All values other than brightest pixels are set to 0
best_edge_pixels(im_channel < threshold_of_brightness) = 0;

brightest_edges = best_edge_pixels;
```

- Find the hough lines using hough, houghpeaks and houghlines. Description below:

Hough:
Note: Referred from MATLAB documentation
The function uses the parametric representation of a line: rho = x*cos(theta) + y*sin(theta). The function returns rho, which is the distance from the origin to the line along a vector perpendicular to the line, and theta, the angle in degrees(theta) between the x-axis and this vector. The function also returns the Standard Hough Transform, H, which is a parameter space matrix whose rows and columns correspond to rho and theta values respectively. Input parameter consist of strongest edges, Rhoresolution which tells us about the spacing of Hough transform bins along the rho axis and Theta will tell us the Theta value for the corresponding column of the output matrix.

HoughPeaks:
Note: Referred from MATLAB documentation.
Houghpeaks will find peaks in Hough transform matrix. The first parameter it takes is the H that we got from hough(). Second parameter will specify the maximum number of peaks to identify. By default, its value is 1. If you increase the value of second parameter means you will get more number of peaks and hence increase the number of lines detected by houghlines. Threshold parameter will specify the minimum value that should be taken as a peak. Third parameter is a size of suppression neighborhood which is the neighborhood around each peak that is set to zero after the peak is identified.

Houghlines:
Note: Referred from MATLAB documentation.
Houghlines will find the line segments based on hough transform method. It takes as input the Theta and rho are values returned by the hough method. Also peaks is a matrix returned by the houghpeaks function. It has the row and column information of hough bins which we can use to searhc lines. It returns the lines in a structure array and the length of this array is equal to the number of merged line segments found in the image. 'Fillgap' : Fill the gaps found in the edges and 'MinLenght' takes the minimum length of a line.

```
% Referred from MATALB documentation
% Hough method will in detecting the hough lines. The function uses the parametric
 representation of a line: rho = x*cos(theta) + y*sin(theta).
% The function returns rho, which is the distance from the origin to the line along a vector
  perpendicular to the line,
% and theta, the angle in degrees(theta) between the x-axis and this vector. The function also
returns
        % the Standard Hough Transform, H, which is a parameter space matrix whose rows and
 columns correspond to rho
        % and theta values respectively.
        % Input parameter consist of strongest edges, Rhoresolution which tells us about the
 spacing of
        % Hough transform bins along the rho axis and Theta will tell us the Theta value for
 the
        % corresponding column of the output matrix H.
         [H,T,R] = hough( edges, 'RhoResolution', 1, 'Theta',  -90:2:89 );

        % Referred MATLAB documentaion
        % Houghpeaks will find peaks in Hough transform matrix.
        % The first parameter it takes is the H that we got from hough()
```

```
        % Second parameter will specify the maximum number of peaks to identify.  By default,
its value is 1.
        % If you increase the value of second parameter means you will get more number of peaks
and hence
        % increase the number of lines detected by houghlines.
        % Threshold parameter will specify the minimum value that should be taken as a peak.
        % Third parameter is a size of suppression neighborhood which is the neighborhood
around each
        % peak that is set to zero after the peak is identified.
        peaks    = houghpeaks(H,10,'threshold',ceil( 0.3 * max(H(:))),'NHoodSize',[5 5]);

        % Referred MATLAB documentaion
        % Houghlines will find the line segments based on hough transform method.
        % It takes as input the Theta and rho are values returned by the hough method. Also
peaks is a matrix
        % returned by the houghpeaks function. It has the  row and column
        % information of hough bins which we can use to searhc lines.
        % It returns the lines in a structure array and the length of this array is
        % equal to the number of merged line segments found in the image.
        % 'Fillgap' : Fill the gaps found in the edges and 'MinLenght' takes the minimum length
of a line.
        lines      = houghlines( edges, T, R, peaks,'FillGap', 130,'MinLength',320 )
```

- Then we compute distance of each edge pixel to the line. And add it to vote. Then we compute the maximum number of votes and that will be our vanishing point. I tried but having troubles applying the voting algo. The formula we can use to find it is:

$$\text{distance}(P_1, P_2, (x_0, y_0)) = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2 y_1 - y_2 x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}.$$

  Referred from Wikipedia (https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line#Line_defined_by_two_points)

- Then asked in question we can mark points in colors asked using markers.

```
% Showing the image and then plot the vanishing point in the image
        figure;
        imagesc( im );
        message = 'Vanishing Point';
        title( message, 'FontSize', 15 );

% Put hold and then plot the vanishing point(coordinates x and y) in
% the image displayed above
        hold on;
        plot( x, y, 'ro', 'LineWidth', 1, 'MarkerSize', 22 );
        plot( x, y, 'g+', 'LineWidth', 1, 'MarkerSize', 22 );
        plot( x, y, 'bx', 'LineWidth', 1, 'MarkerSize', 22 );
```

Algorithm to find Vanishing Points:

The overall logic of finding the vanishing point in the given image is using hough transformation method. Hough transformation is one of the way of fitting geometric figures into the model. In this homework, we have worked to fit lines in the model using hough.

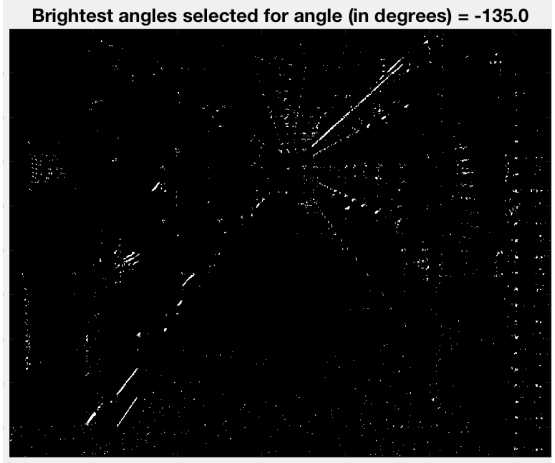
Hough transform is a voting algorithm where we work with strongest edges in an image. For every pixel present in representing strongest edges, we work with line of from r = x cos(theta) + y sin(theta). While working in this vector domain of line, we compute the maximum votes. For every line, we find the distance of all the other edge pixels in image and try to fit them in model. Then in the model, the point that gets maximum votes, will be our vanishing point.

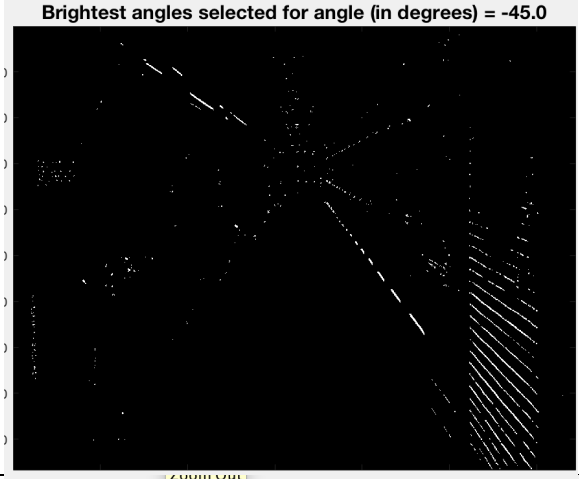
CSCI-631: Foundations of Computer Vision

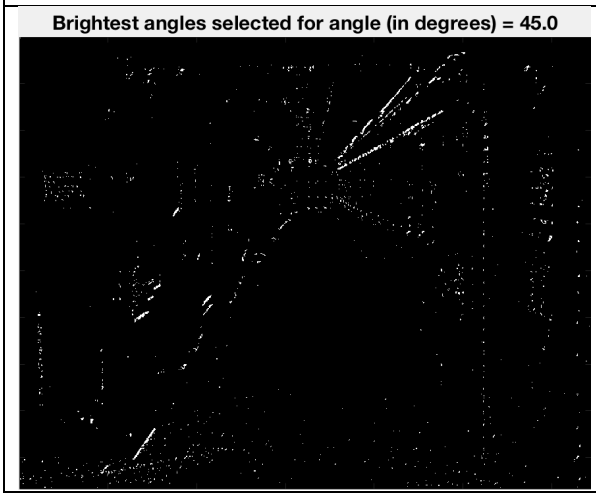
Submitted by: - Yash Jain (yj8359)

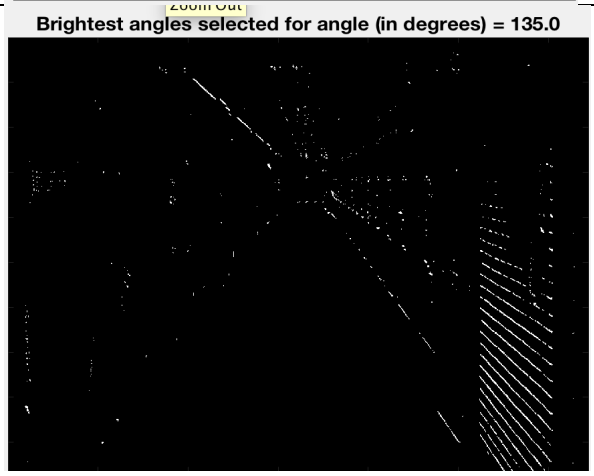
Output Images:

Edges:

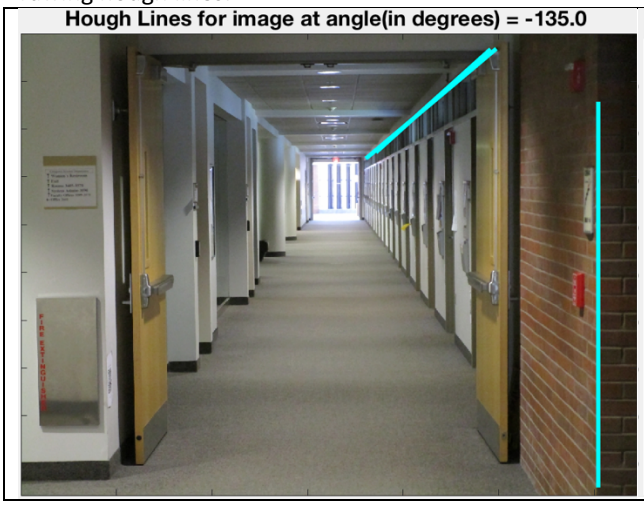


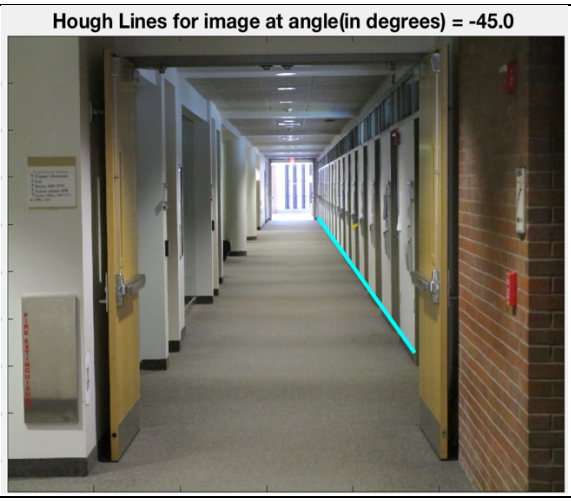






Drawing hough lines:





CSCI-631: Foundations of Computer Vision

Submitted by: - Yash Jain (yj8359)