5. Discussion:

Algorithm:

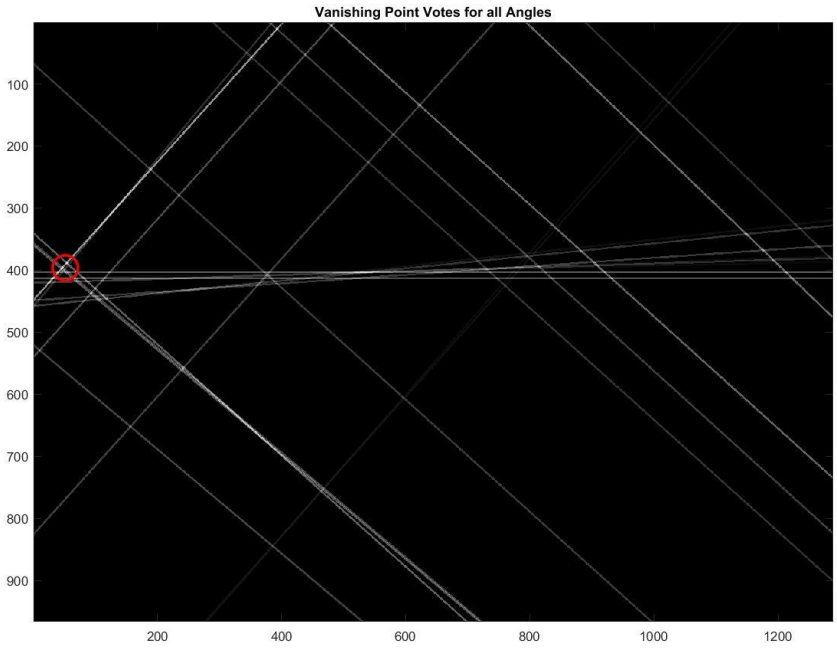
1) Read the image and smooth it out using Gaussian filter.
2) Consider only the green channel, since that works the best compared to the other channels.
3) Now, we run the sobel filter on the image to find the vertical and horizontal edges in the image. Using these edges, we calculate:
   a) The magnitude of the edges.
   b) The direction (angle) using the atan2() function. This angle is in radians.
4) Create a meshgrid with the x and y coordinates using the size of the image. Create another matrix which stores the votes. Initially this matrix is made of only zeros.
5) Now, we consider angles only at specific degrees. Since the images contain edges that are converging toward the center, I have considered edges that are in the range [-135,150].I use a step angle (delta) which I have considered as 45 degrees.
6) I now calculate the minimum and maximum angle which acts like a threshold for choosing the edges. Therefore, now I consider only the edges which are between [-157,157].
7) Now I set all the pixels in the image that are below the minimum and above the maximum angle threshold to zero. Thus, now our image will contain only the edges that fit our angle range. The rest of the unwanted edges are thrown away i.e. pixel value is set to 0 (black).
8) Now we form a matrix with only those pixels which have a value of 1.
9) Now, since we have to consider only the strongest edges. We consider only the top 5% of the edges.
   a) While tossing out the rest 95%. To do this, I sort the pixel values in the descending order using the sort() function. I learnt that the sort() function by default uses the ascending order , but since we have to consider the top 5%, I sort it in descending order and multiply it with (5/100).
10) Now, we form a matrix which contains only the strongest edges by setting all the other values to 0.
11) Now we choose points that are at the right angle and magnitude more than the top threshold value.
12) I now feed these edges into the hough() function which performs hough transform(). We feed the edge locations to this method.
    a) Using the output from the hough() method, I find peaks in the hough transform.

b) Next, I find the lines using the output from the hough() method and the peaks calculated.

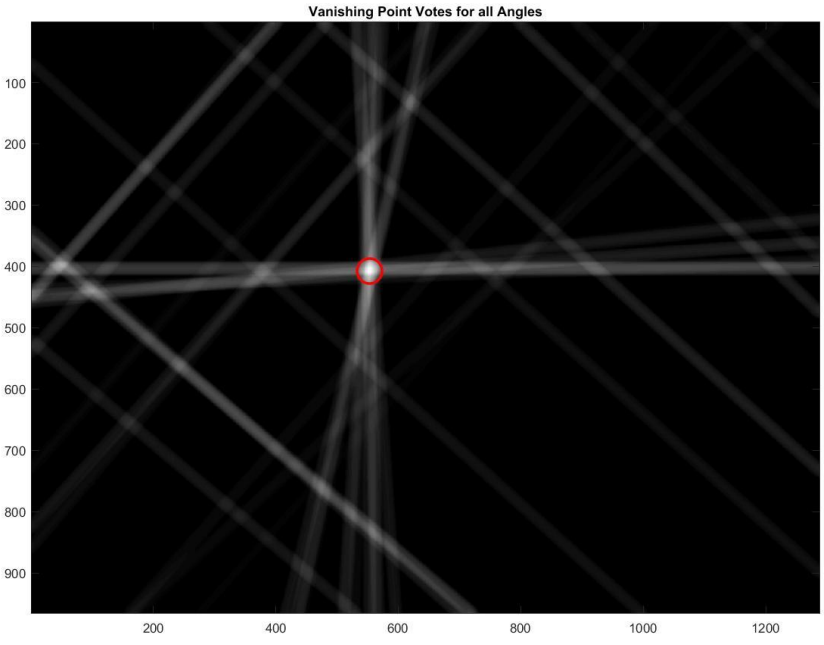
13) I now calculate the distance between the 2 points using the formula :
    a) Square root(square(dx)+square(dy)).
    b) Next, calculate the equation of the line in canonical form.
14) These are then used to calculate the distance from the line to all the other points in the image. Now, I form another matrix which will contain all the edges that have distance less than or equal to 1.5(randomly chosen factor). This matrix (b_vote_map) will contain the votes that are being given to the edges.
15) Now I form another matrix which for every value in b_vote_map will calculate a value (b_vote_map + distance between the 2 points) which is nothing but the length. I do this since to get the correct number of votes for the right vanishing point, only the longest lines must be favored. Therefore, instead of increasing the vote count by 1, I increase it by the length of the line.
16) Next, since the edges are extremely thin, they pass through different points. This may result in an error wherein the vanishing point will not get the maximum number of votes.Thus to sort this problem, we thicken the edges i.e. smear them out so they meet at a certain point which will infact be the actual vanishing point.
17) To do this, a filter of disk shape and size 11 is run on the edges. This will make the edges appear thicker and hence maximum number of edges will pass through a certain point**.**

Comparison with smeared edges and un-smeared edges: Input image: IMG_1325.jpg

a) Un-smeared edges: Wrongly detected vanishing point.



b) Smeared edges: Correctly detected vanishing point.



18) For all the pixels in the image, we now mind the point at which the maximum number of votes received. And set this pixel location as the vanishing point and mark it using a circle and 2 crosses.

RESULTS:

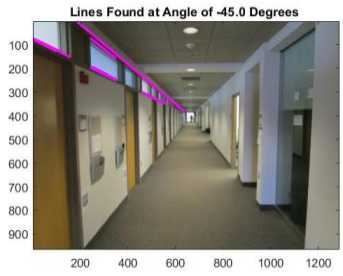
Original Image: IMG_1289.jpg

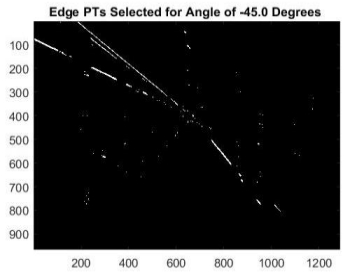
Top left: Image with magenta edges.

Top right: Image with pixels on the edges with our specified angle threshold.

Bottom left: Un-smeared edges with counted votes.

Image chain at -45 degree.





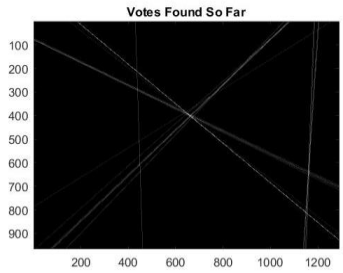
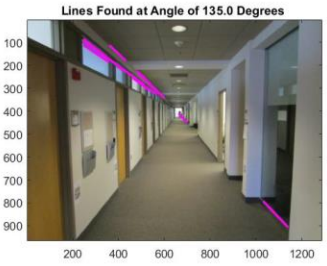
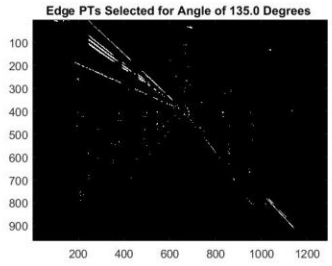


Image chain at 135 degrees:





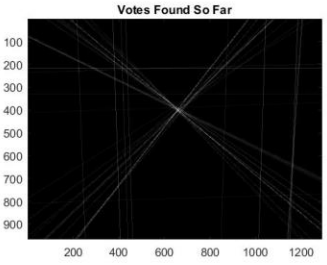
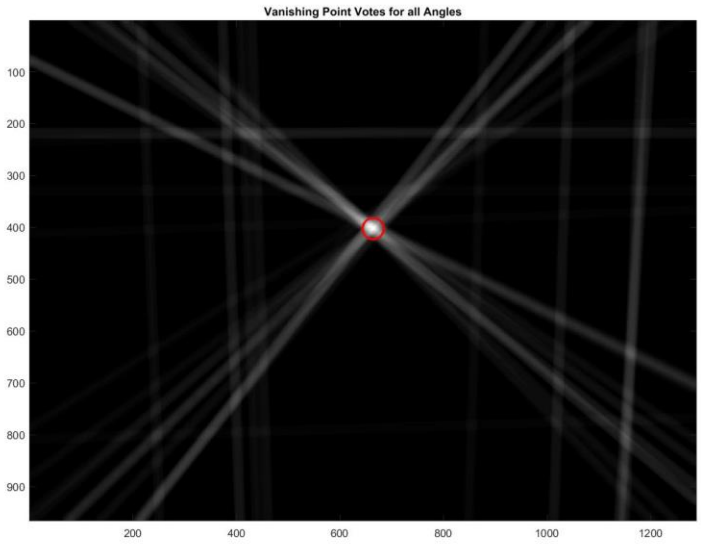


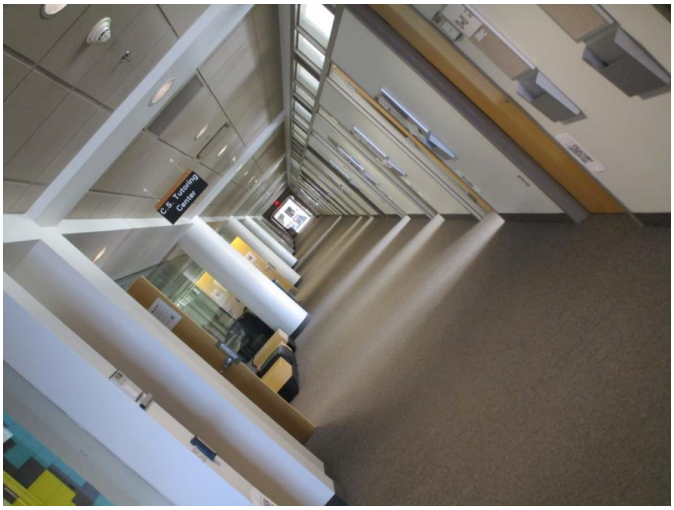
Image with smeared edges of the counted votes.



FINAL IMAGE: Image with marked vanishing point.

INPUT IMAGE: 'IMG_1325.jpg' Image rotated at 45 degrees.

Original image:



Final image: