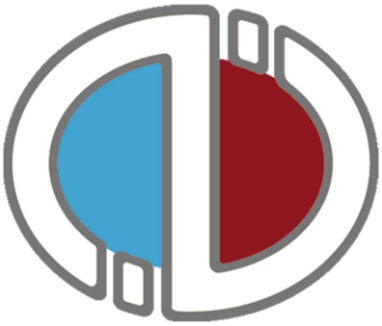
Report:
Vanishing point Detection

Jama Hussein MOHAMUD



Department of Electrical & Electronic Engineering

Anadolu University

Eskisehir, Turkey

Year: 2018

Introduction:

Vanishing point detection and estimation is a very interesting research in computer vision, because it is helpful for object detection, 3D reconstruction and camera calibration. Also, the vanishing point in road images can help to steer a robot.

Vanishing Points is a point where each set of parallel lines in the image intersect or meet. In geometric properties of projection, we know that, points project to points and lines project to lines in an image plane.

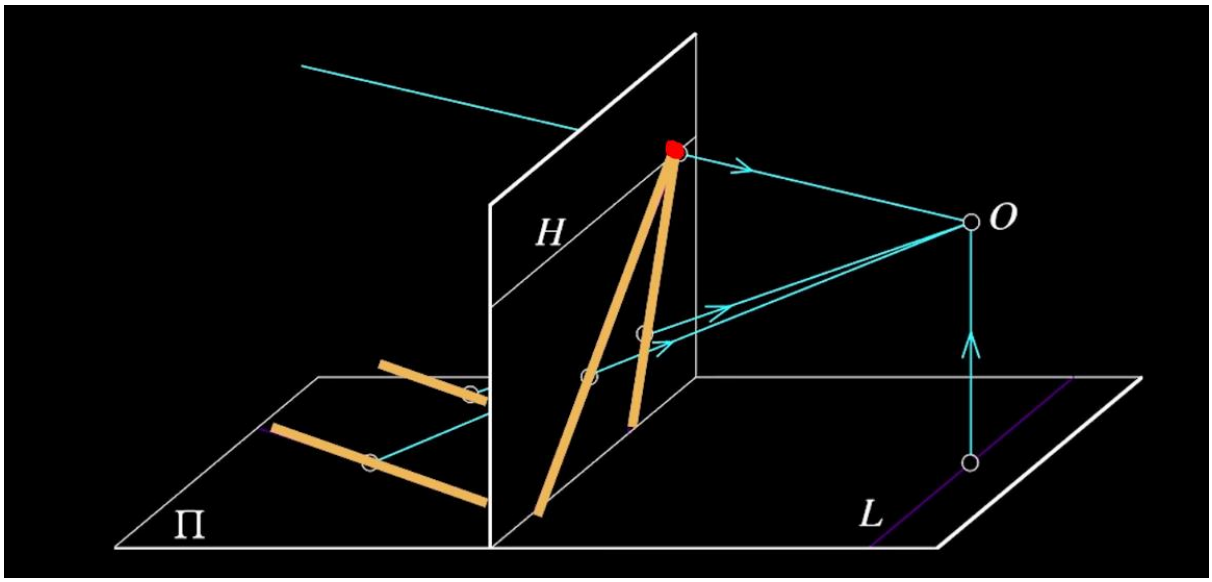


Fig 1: Parallel Lines converged to a single point in projection/image plane.

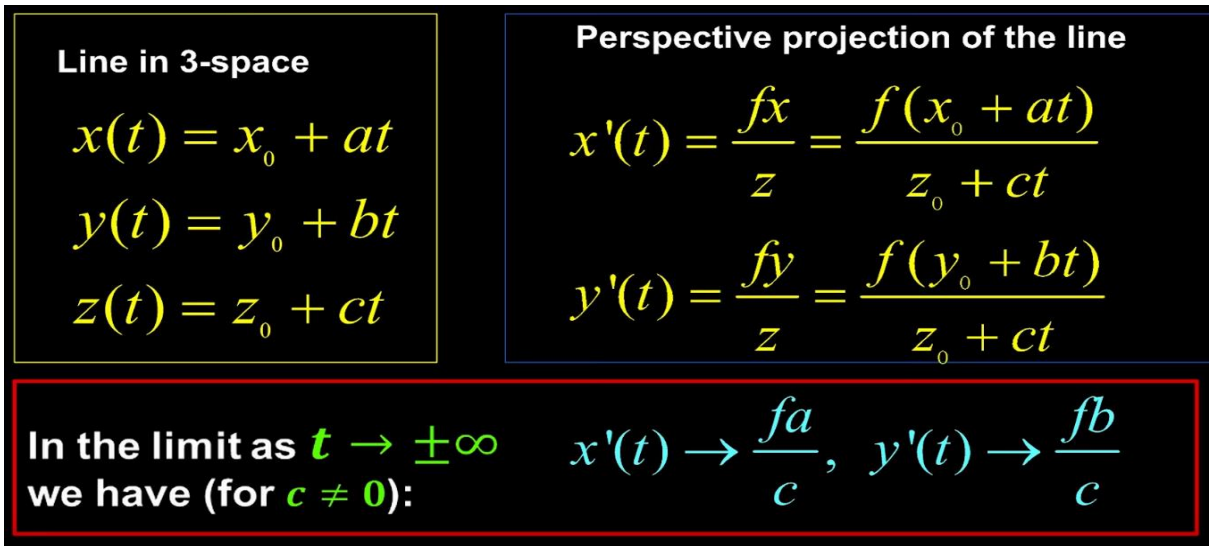


Fig 2: Showing that parallel lines in image converge too mathematically.

The two figures above proof that parallel lines in the world almost always converge to one point in image.

Hough Transform:
In this Home-work I used Hough Transform Algorithm to detect vanishing points of images. Although, Hough algorithm is a poor algorithm when it comes to vanishing point detection but I improved a lot and successfully achieved very good and optimal results. Before I go through the explanation of my code let me explain how Hough Transform Algorithm works.
Hough Transform Is a voting technique that can be used to answer these questions:

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which line?

But, the main point is

- Each edge point votes for compatible lines.
- Look for lines that get many votes.

So, how this works.
The intuition is that, lines in an image space correspond to point in to Hough space. Also, points in an image space correspond to lines in a Hough space. Let's look at the below two images.

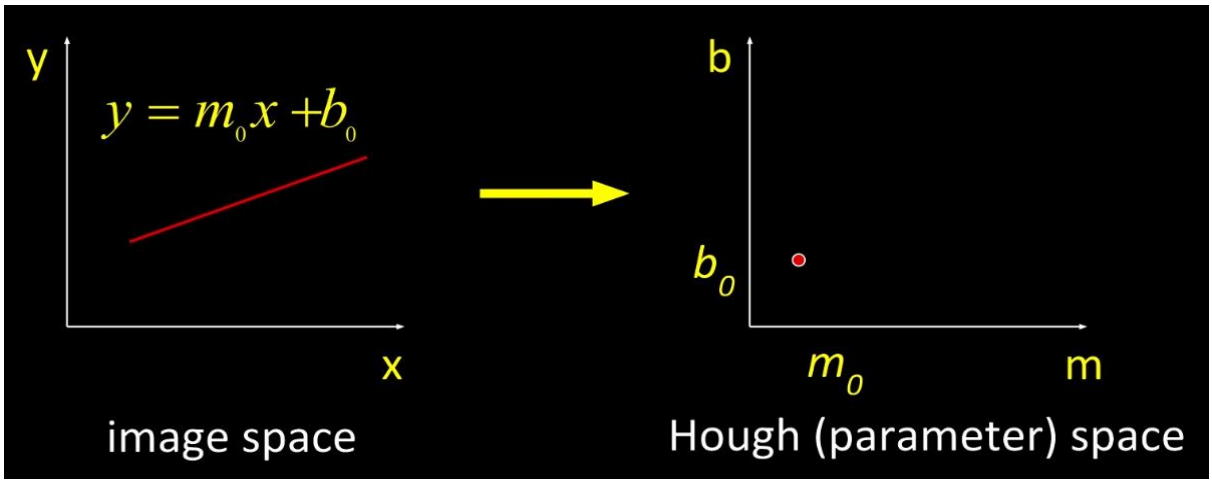


Fig 3: Line in image space corresponds to a point in a Hough space

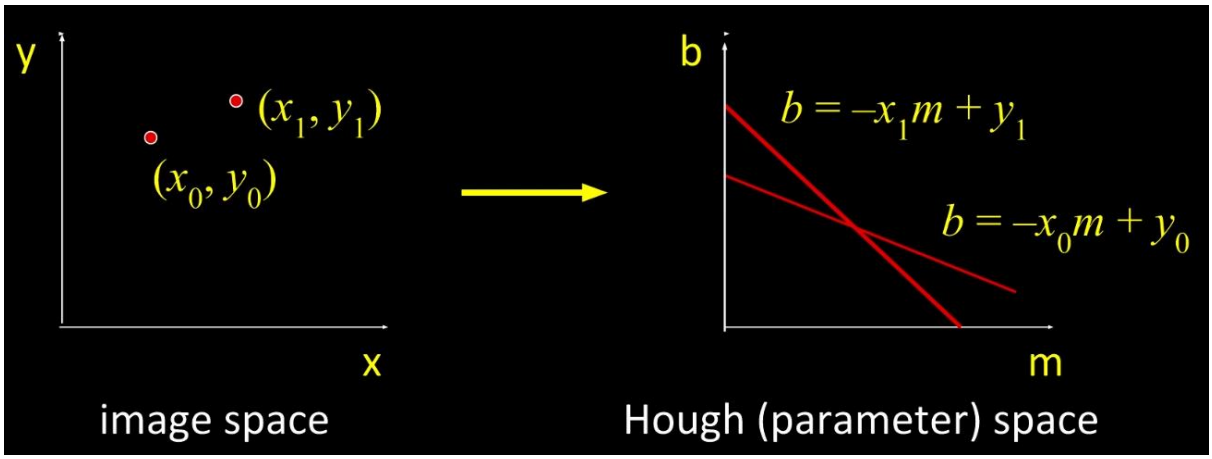


Fig 4: Point in image space correspond to a line in Hough space.

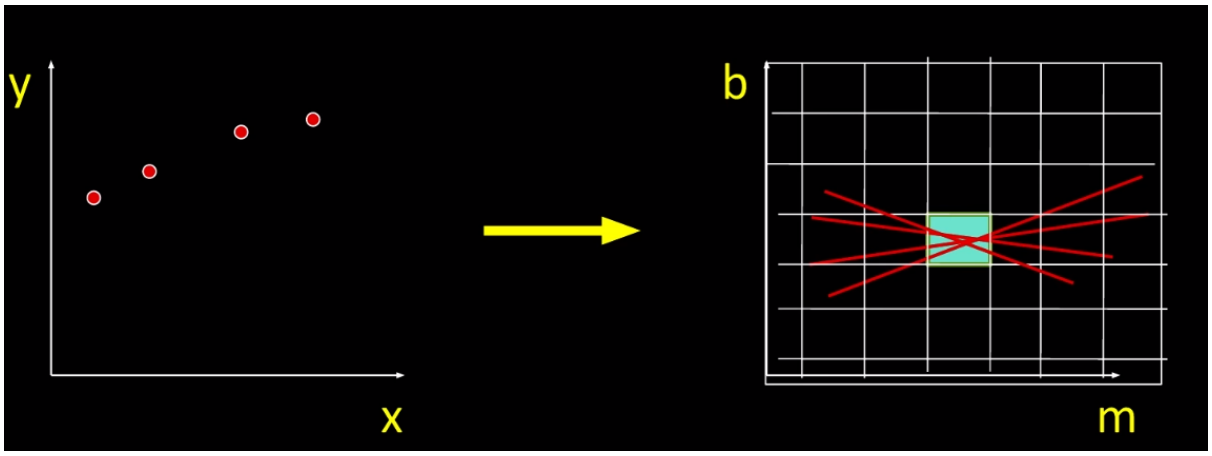


Fig 5: Grid view of the points that is voted for lines.

Figure 5, shows that each edge point in image space vote for a set of possible parameters in Hough space, and parameters with the most votes indicate line in image space.

However, because of the infinity of the equation of line (mx + b) we will use polar representation in a Hough transform.

Rho = $x \sin\theta + ycos\theta$ ---- Point in image space is now sinusoid segment in Hough space.

Hough Transform Algorithm:

-Initialize H[rho, theta] = 0
-For each edge point in E(x,y) in the image
  -For theta = 0 to 180
    - $rho = x \sin\theta + ycos\theta$
    - H[rho, theta] += 1
-Find the values of (rho, theta) where H[rho, theta)] is maximum
-The detected line in the image is given by rho = $x \sin\theta + ycos\theta$

My approach:

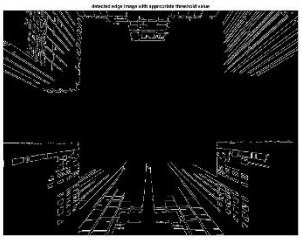
NB: before you go through the code, I will highly recommend you to read the report in order to get intuition of how my code works. And also check the results folder where you get to see all the images and there vanishing points.

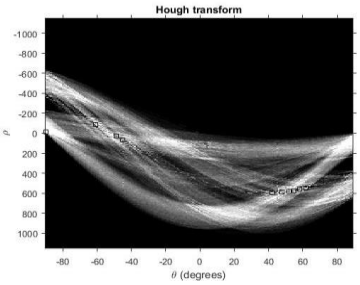
Step 1: read the image
Step 2: Reduce the image size by adding extra pixels around the image.
For some images we don't need this step and we will set to 0. However, there are other images that are too big and difficult to see where the lines converge. So, this step will help us easily visualize where the vanishing point is, by adding extra pixels around the image.
Step 3: Converting image to grayscale

Step 4: Thresholding is the simplest method of image segmentation. From grayscale image, thresholding can be used to create binary images. We can say that it's filtering method, because in this case we are using thresholding to remove unwanted lines. Some of the sample images have different thresholding value, while most of them have the same value.
Step 5: Applying edge detection. Edges are very important in Hough transform. Is an image processing technique for finding the boundaries of objects within images.
Step 6: Applying Hough Transform.
Step 7: Finding Hough Peaks
Step 8: Obtaining Lines from Hough Lines
Step 9: plotting the vanishing point which is where the lines intersect.

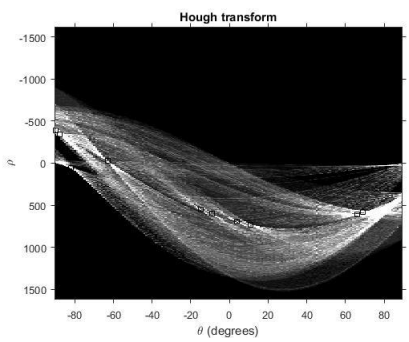
Results:

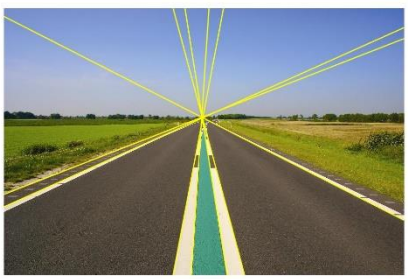
Sample 1- No pixels added, threshold = 80, Edge threshold = 0.8, Houghlines 'MinLength' = 50





Sample 2 - No pixels added, threshold = 145, Edge threshold = 0.9, Houghlines 'MinLength' = 90





As you can see from above two sample images, we have obtained the vanishing point. More sample images with their vanishing point can be found in the results folder.

The below table shows us the values that needs to be changed when testing with different images. For example, as we mentioned, some images need to be reduced in size in order to see where lines converge or intersect.

	Number of pixels added (L)	threshold	Edge threshold	Houghlines 'MinLength'
Sample 1	No	80	0.8	50
Sample 2	300	80	0.8	80
Sample 3	300	90	0.8	120
Sample 4	No	145	0.9	90
Sample 5	No	200	0.6	120
Sample 6	No	80	0.9	80
Sample 7	300	80	0.9	90

Conclusion and Future Work:
I have finalized vanishing point detection using edge detection and Hough transform algorithm. We have explained how Hough transform and the code works. We obtained a good estimate of vanishing points. In my future work, I am planning to add tracking method using filters like Kalman filter to track the vanishing point in videos.

References:
[1]. https://pdfs.semanticscholar.org/f5a2/eb9c8d15f919c897fac80a392bdd34913ce7.pdf
[2]. https://classroom.udacity.com/courses/ud810/lessons/2673608575/Concepts/35179090140923
[3].https://classroom.udacity.com/courses/ud810/lessons/2870588566/concepts/34824487900923