The Intro to Computer Vision labs will be run in Google Colaboratory, a Jupyter notebook environment that runs entirely in the cloud, you don't need to download anything. To run these labs, you must have a Google account.

Step 1: click on the assignment invite link -> **Accept this assignment**. Refresh page -> individual repo for the specific assignment is created automatically

Project 1: 3 weeks
https://classroom.github.com/a/Xao02nLg

Step 2: Navigate to http://colab.research.google.com/github -> Click the **Include Private Repos** checkbox -> **select the correct repo** (SistemeDeVedereArtificiala/assignment_name-student_name) -> Click on the jupyter notebook of the current assignment

Step 3: [GitHub sign-in window] In the popup window, sign-in to your Github account and authorize Colab to read the private files.

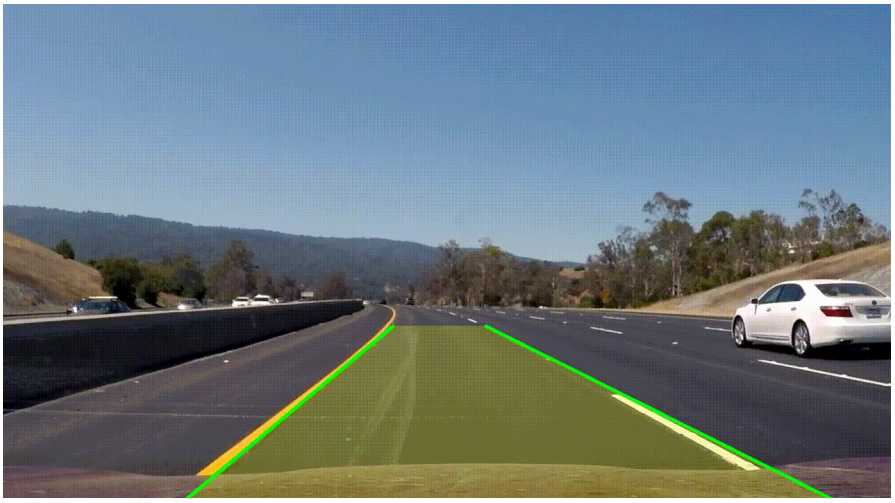
Step 4: [in colab] **File** -> **Save a copy to GitHub**. Select the correct repository for the SPECIFIC assignment -> Click the **Include Colab Link** -> Click **OK**

Step 5: [in colab] Navigate to the **Runtime** tab --> **Change runtime type**, under **Hardware accelerator** select **GPU/TPU** (tensor processing unit) according to your needs.

Read the suggestions and accomplish all tasks marked with **#TODO**.

!!! At the end of each laboratory **REPEAT step 4 in order to SAVE** the answers to your private repository (individual for each assignment)

Project 1: Lane Detection



Week 2: fitting lines using Hough Transform

Q: Wait, why aren't we done just by running Canny edge detection?

A: Because we only have edges, for the desired result we need to define the function which turns these edges into lines.

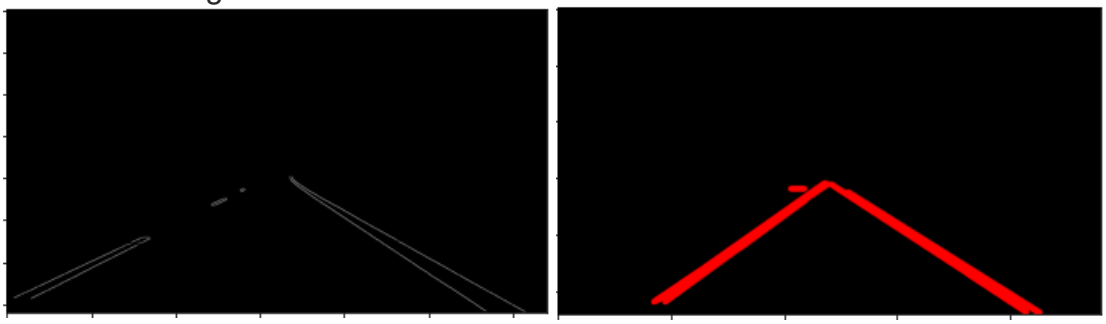


Figure 1: Edges (left) Hough lines (right)

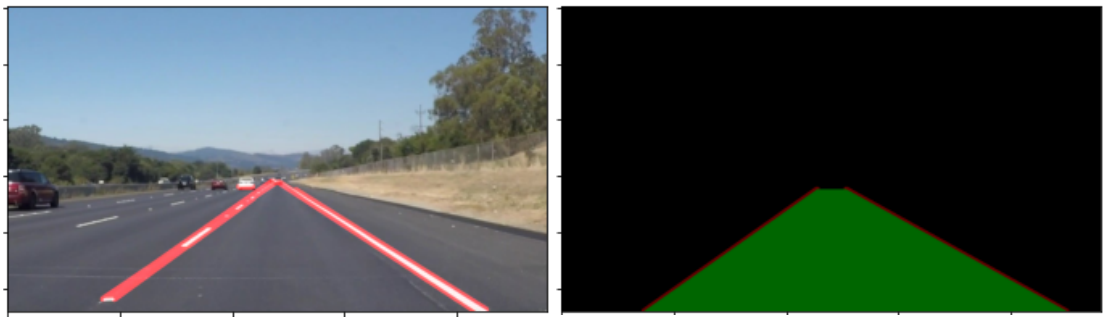
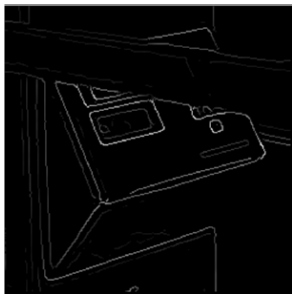
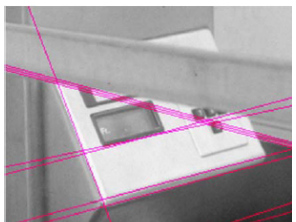


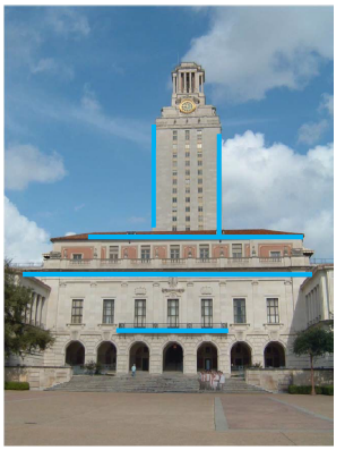
Figure 2: Hough lines over input (left), Mean lines (right)

<- Figure 3: Lane Detection

Figure 4 -> grayscale (top) edges (middle) Hough lines (bottom)







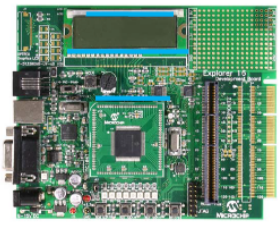


Figure 5: other objects characterized by presence of straight lines

Questions when fitting lines:

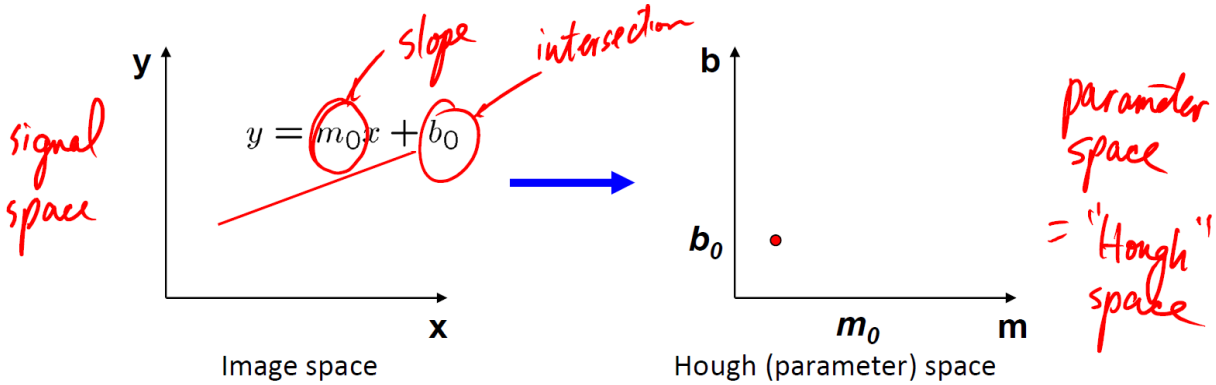
1. Given points that belong to a line, what is the line?
2. How many lines are there?
3. Which points belong to which lines?

Hough Transform is a voting technique that can be used to answer all of these.

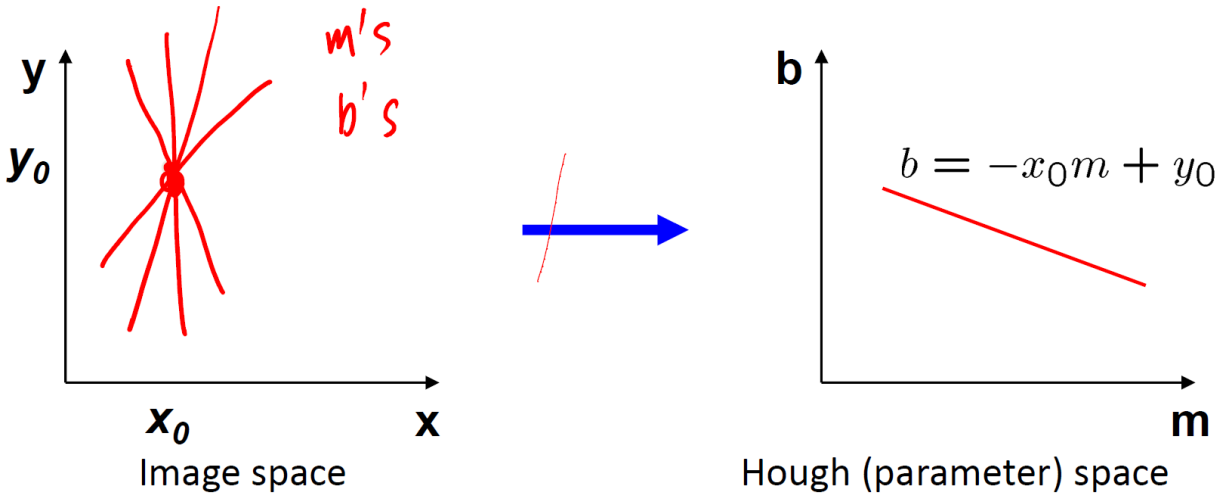
From Image space to Hough Space:

1. A line in the image corresponds to a point in Hough space.
2. To go from image space to Hough space:

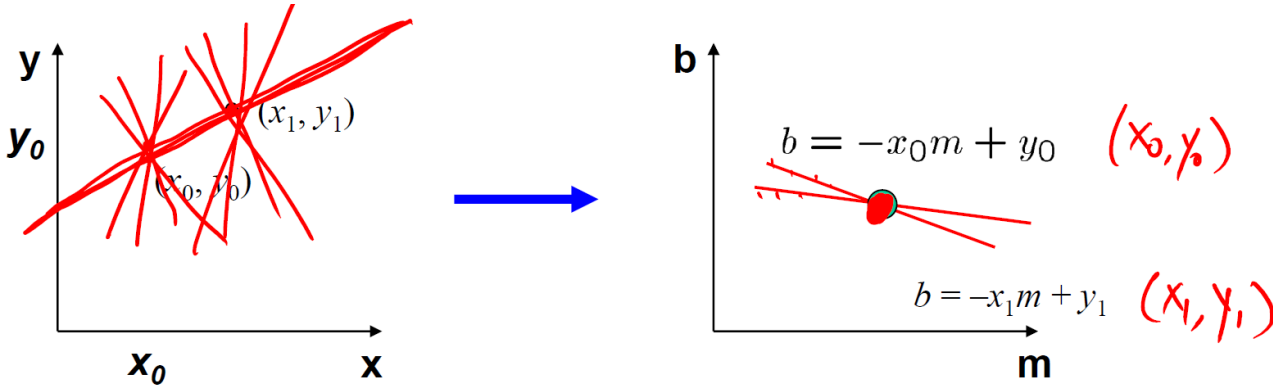
   Given a set of points (x,y), find all (m,b) such that y = mx + b

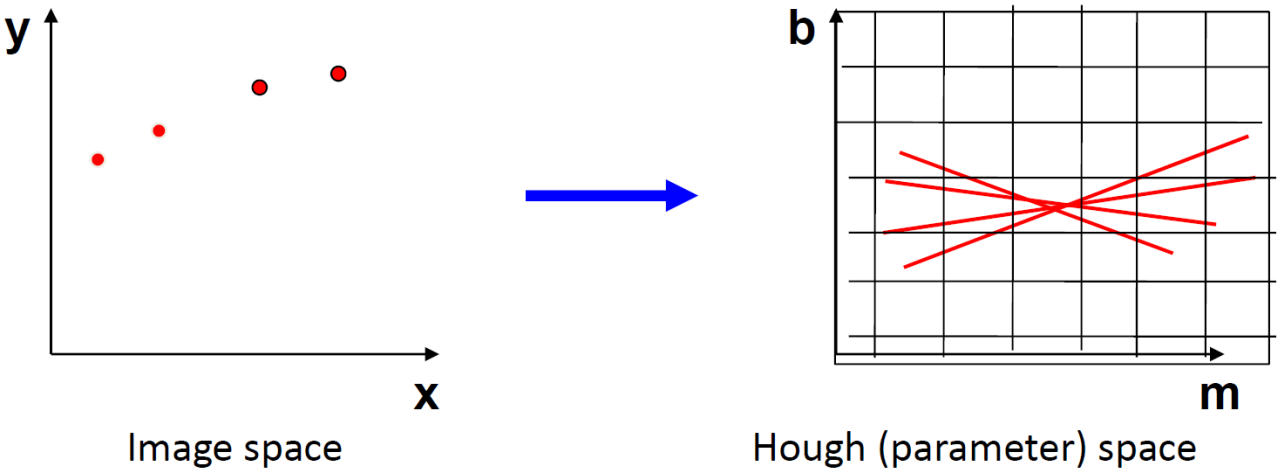


3. Q: What does a point (x0, y0) in the image space map to?
   A: the solutions of b = -x0m + y0. This is a line in Hough space.



4. Q: What are the line parameters for the line that contains both (x0, y0) and (x1, y1)?
   A: It is the intersection of the lines b = –x0m + y0 and b = –x1m + y1





Q: How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

A: 1. Let each edge point in image space vote for a set of possible parameters in Hough space

2. Accumulate votes in a discrete set of bins; parameters with the most votes indicate line in image space.

Hough Transform a voting technique:

1. Record all possible lines on which each edge point lies.
2. Look for lines that get many votes.

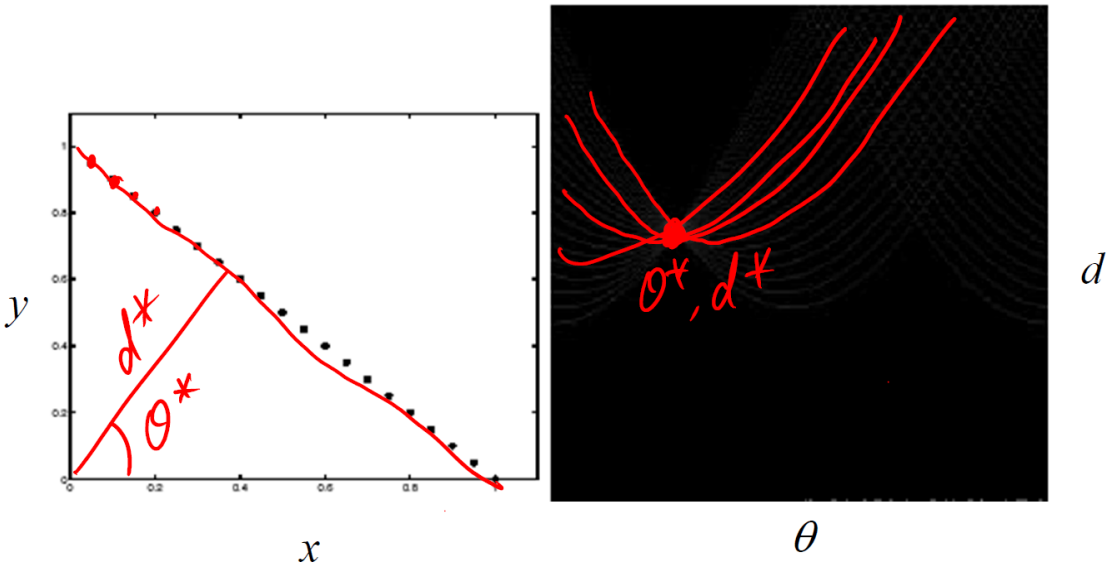
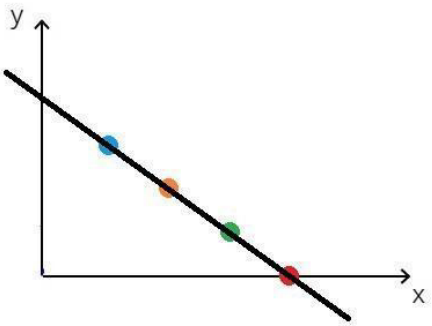


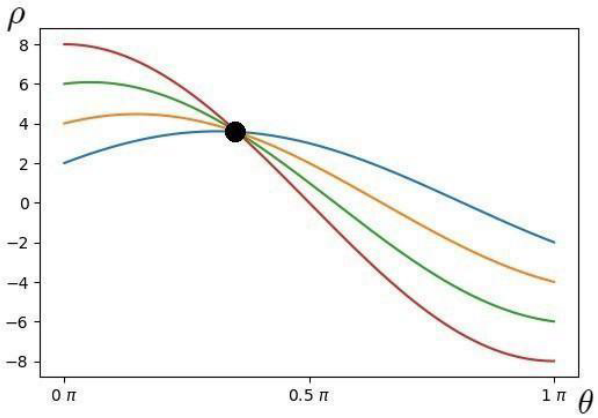
Image space
edge coordinates

Votes

Bright value = high vote count
Black = no votes



Points which form a line



Bunch of sinusoids intersecting at one point

Figure 6:
input image (left)

Hough space (right)

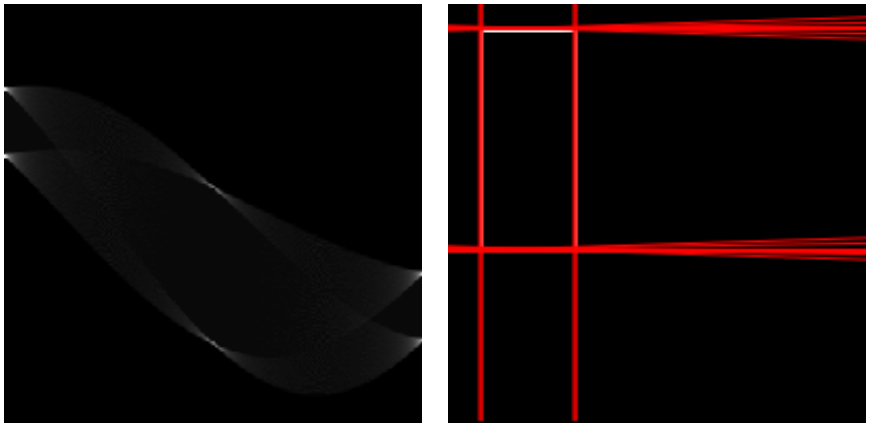


Figure 7:
Hough Transform (left)
Hough Lines (right)

Key takeaways for project 1:

1. Use Gaussian Blur to remove all noise from the image
2. Use canny edge detection to isolate edges in the image
3. Use Bitwise And function to isolate edges which correspond to the lane lines
4. Use Hough Transform to turn the edges into lines

Hough Transform DEMO 1 (here)
Hough Transform DEMO 2 (here)

Further study on the topic at:

1. Stanford Vision Lab, Lecture 4: Finding lines, Professor Fei-Fei Li
2. Computer Vision: Algorithms and Applications (chapter 4.3), Richard Szeliski