Introduction to Computer Vision

Universitatea Tehnică "Gheorghe Asachi", Iași

Facultatea de Automatică și Calculatoare

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: 2 weeks

https://classroom.github.com/a/K6JIB4_U

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**.

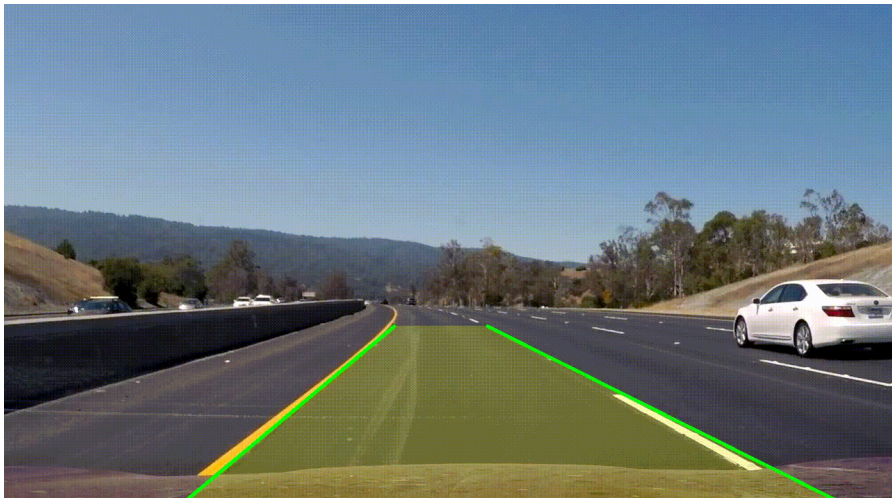
!!! At the end of each laboratory **REPEAT step 4 in order to SAVE** the answers to your private repository (individual for each assignment)

Project 1: Lane Detection

In the field of Computer Vision, extraction of geometric features from images is a common problem. Different approaches have been proven to be efficient in extracting particular features as part of the solution. Some solutions involve global examination of the input image, while others involve local examination at pixel level.

A basic feature of any self-driving system is **Lane Detection.**

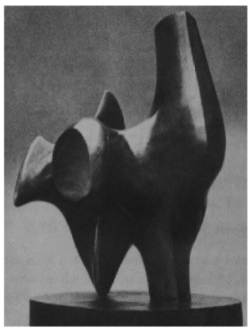
Desired solution



TESLA

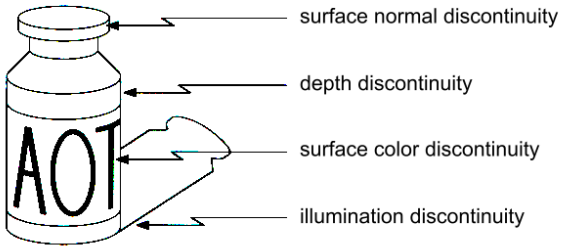
Week 1: edges

Edge detection



- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

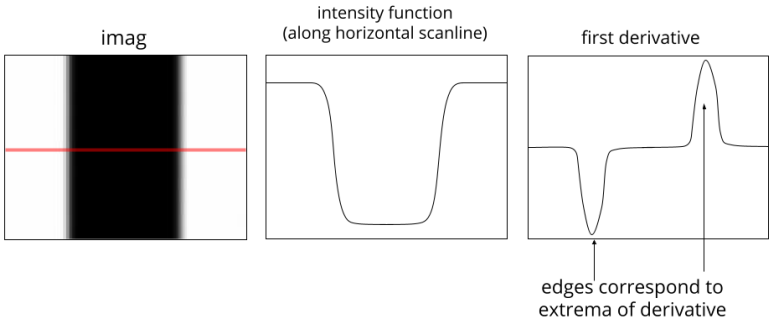
Origin of Edges

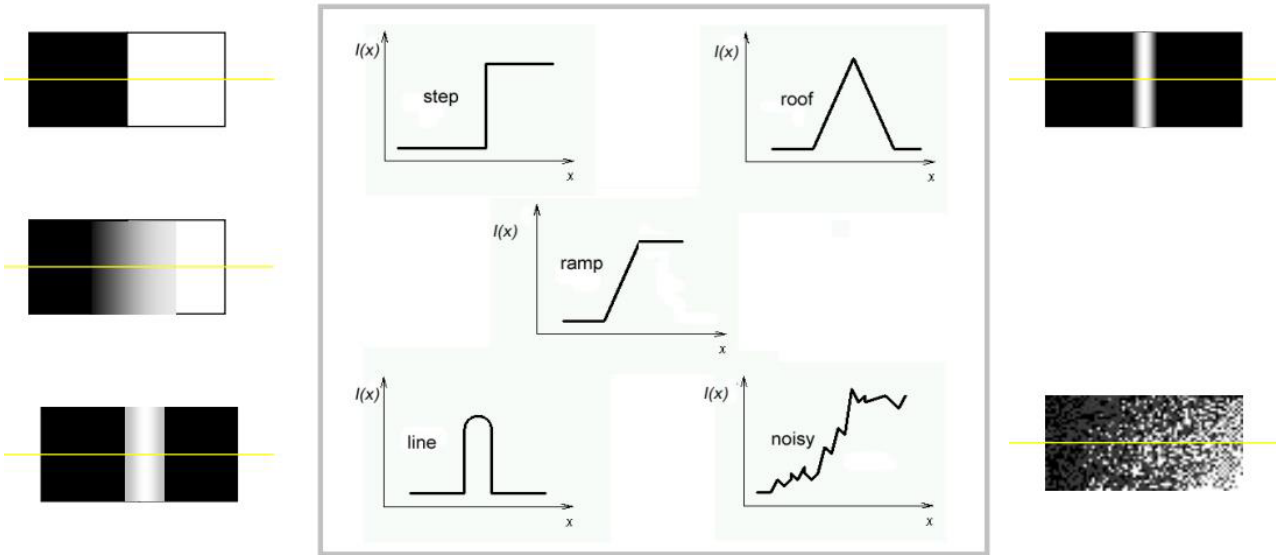


- Edges are caused by a variety of factors

Characterizing edges

- An edge is a place of *rapid change* in the image intensity function





Canny edge detector:

1. This is probably the most widely used edge detector in computer vision
2. Theoretical model: step-edges corrupted by additive Gaussian noise
3. Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization

Our first Computer Vision pipeline:

a. Filter image with derivative of Gaussian

```
kernel_size = 9 # Must be an odd number (3, 5, 7...)
smoothed_img = cv2.GaussianBlur(img, (kernel_size, kernel_size), 0)
```

b. Find magnitude and orientation of gradient
c. Non-maximum suppression
d. Linking and thresholding (hysteresis)
   i. define two thresholds: low and high
   ii. use the high threshold to start edge curves and the low threshold to

continue them

```
low_threshold = 180
high_threshold = 240
canny_img = cv2.Canny(smoothed_img, low_threshold,
high_threshold)
```

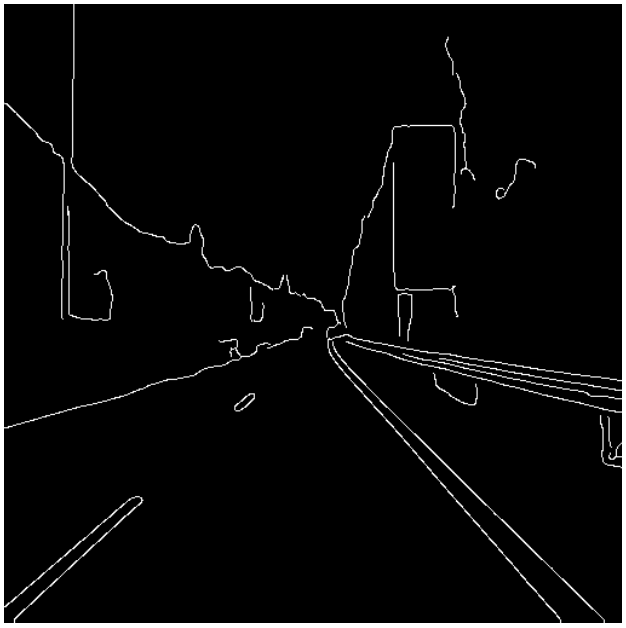
Demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Input image

Gaussian smoothing

Smoothed Gradient magnitude

Non-maximum suppression



Hysteresis threshold

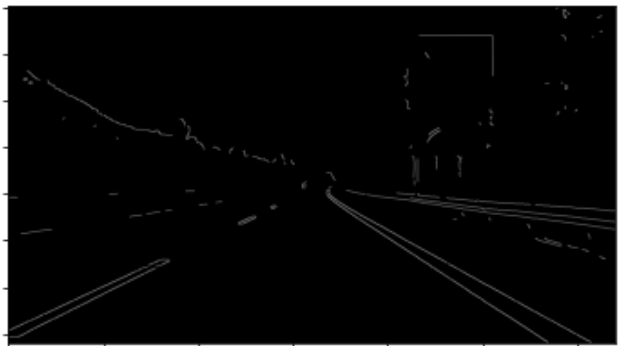
Intermediate result for Lane Detection: canny edges overlapped on input image

Region of interest:

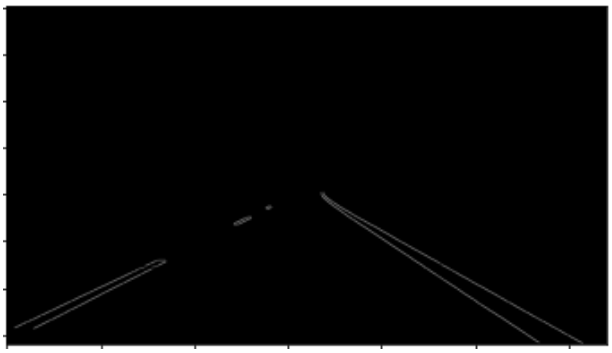
a. Define the mask for a lane
   i. Define Vertices (triangle adjusted to image size)
   ii. Mask

```
#defining a blank mask to start with
mask = np.zeros_like(img)
ignore_mask_color = 255
#filling pixels inside the polygon defined by "vertices" with the fill color
cv2.fillPoly(mask, vertices, ignore_mask_color)
```

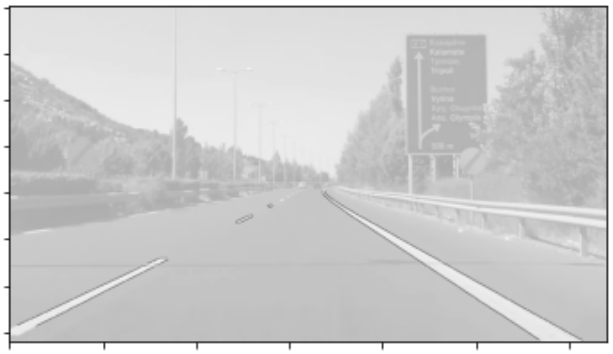
b. Masked image
   masked_image = cv2.bitwise_and(canny_img, mask)



Canny edges



Masked canny edges



Overlapped on grayscale

First step to Lane Detection is done. To be continued next week!

Further study on the topic at:

1. Introduction to Computer Vision (lecture 2), Cornell University
2. Computer Vision: Algorithms and Applications (chapter 4.2), Richard Szeliski