**Project Based Learning Report**

on

**Real-time Lane Line Detection in Python**

Submitted in the partial fulfillment of the requirements

For the Project based learning in (**Essentials of Data Science**)

in

Electronics & Communication Engineering

By

**2014111113 Manvendra Vikram**

**2014111099 Aryan Verma**

**2014111089 Akhil Subhash**

Under the guidance of Course In-charge

Prof. Dnyanesh S.Lavhkare

Department of Electronics & Communication Engineering

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering,

Pune – 4110043

**Academic Year: 2021-22**

**Bharati Vidyapeeth**

**(Deemed to be University)**

**College of Engineering,**

**Pune – 411043**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

**CERTIFICATE**

Certified that the Project Based Learning report entitled, **“Real-time Lane Line Detection in python”** is work done by

**2014111113 Manvendra Vikram**

**2014111099 Aryan Verma**

**2014111089 Akhil Subhash**

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **Essentials of Data Science Course** of Bachelor of Technology Semester IV, Electronics & Communication Engineering.

**Date: 23 May 2022**

**Prof. Dnyanesh S.Lavhkare Dr. Tanuja S.Dhope**

**Course In-charge PBL Co-Ordinator**

**Dr. Arundhati A.Shinde**

**Professor & Head**

**ELECTRONICS & COMMUNICATION ENGINEERING**

|  |  |
| --- | --- |
| **Index: -** | |
| Page No. | Contents |
| 1-1 | Problem Statement with Solution |
| 2-3 | Description about project |
| 4 | Software Used |
| 5-9 | Results with Analysis |
| 10 | Conclusion & Outcome |

**Problem Statement :-**

What is Data Science? Why learn Data Science?

**Solution :-**

Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses complex machine learning algorithms to build predictive models. The data used for analysis can come from many different sources and presented in various formats.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data. Data science practitioners apply [machine learning](https://www.datarobot.com/wiki/machine-learning/) [algorithms](https://www.datarobot.com/wiki/algorithm/) to numbers, text, images, video, audio, and more to produce [artificial intelligence (AI)](https://www.datarobot.com/wiki/artificial-intelligence/) systems to perform tasks that ordinarily require human intelligence. In turn, these systems generate [insights](https://www.datarobot.com/wiki/insights/) which analysts and business users can translate into tangible business value.

Reasons to learn Data Science are: -

1. Learning about data science provides an opportunity for you to recreate yourself.
2. **We live in a digital world, everything is data-driven.** There is data science in **business, accounting, education, science, engineering, healthcare, technology, energy sector, government**, and so on.
3. **Data science is also a very promising field with lots of high paying job opportunities.**
4. **Basic data science skills are important for personal use.**
5. Great potential to branch out with different options.
6. Become a decision-maker, not every job opportunity will give you the power to make informed business decisions. For a data scientist, that is the core responsibility.
7. Less competitive because it is a highly analytical role, competition is less, but demand is not. With a limited talent pool, there is always a challenge for businesses to hire in these roles.

**1**

**Lane Line Detection**

Autonomous Driving Car is one of the most disruptive innovations in AI. Fueled by Deep Learning algorithms, they are continuously driving our society forward and creating new opportunities in the mobility sector. An autonomous car can go anywhere a traditional car can go and does everything that an experienced human driver does. But it’s very essential to train it properly. One of the many steps involved during the training of an autonomous driving car is lane detection, which is the preliminary step.

**Lane detection involves the following steps:**

Diagram

Description automatically generated

* **Capturing and decoding video file:**We will capture the video using VideoCapture object and after the capturing has been initialized every video frame is decoded (i.e., converting into a sequence of images).
* **Grayscale conversion of image:**The video frames are in RGB format, RGB is converted to grayscale because processing a single channel image is faster than processing a three-channel colored image.
* **Reduce noise:**Noise can create false edges, therefore before going further, it’s imperative to perform image smoothening. Gaussian filter is used to perform this process.
* **Canny Edge Detector:**It computes gradient in all directions of our blurred image and traces the edges with large changes in intensity.
* **Region of Interest:**This step is to consider only the region covered by the road lane. A mask is created here, which is of the same dimension as our road image. Furthermore, bitwise AND operation is performed between each pixel of our canny image and this mask. It ultimately masks the canny image and shows the region of interest traced by the polygonal contour of the mask.

**2**

* **Hough Line Transform:**The Hough Line Transform is a transform used to detect straight lines. The Probabilistic Hough Line Transform is used here, which gives output as the extremes of the detected lines

**Dataset:**The dataset consists of the video file of a road.

**Libraries Used-:**

* NumPy: It comes by default with anaconda
* Matplotlib: To install matplotlib, type – “pip install matplotlib” into your command line
* OpenCV: It can be installed in two ways, using anaconda, or using pip.   
  To install using anaconda, type- “conda install -c conda-forge OpenCV”, or to install using pip, type-   
  “pip install OpenCV-python” into your command line.

**2**

**3**

**Software Used: -**

**Visual Studio Code-**

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python, C++, and Fortran. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine.

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Icon

Description automatically generated

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

**4**

**Result with Analysis**

**Analysis of the code: -**

* First, we have imported three libraries – NumPy as np, matplotlib.pylab as plt, OpenCV as cv2.
* Secondly, we created a function “region\_of\_interest” – In which we use fillPoly function which fills the area bounded by several polygonal contours, & the function bitwise\_and which computes bitwise conjunction of the two arrays.
* In the function “drow\_the\_lines”- The function line draws the line segment between pt1 and pt2 points in the image. The line is clipped by the image boundaries. Thick lines are drawn with rounding endings.
* The function cvtColor converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.
* The function Canny finds edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges.
* The function HoughLinesP finds line segments in a binary image using the probabilistic Hough transform.
* Finally, we load the dataset- test video or image.

**Screenshots of code: -**

* **For a road image-**

*import* matplotlib.pylab *as* plt

*import* cv2

*import* numpy *as* np

**5**

def region\_of\_interest(img, vertices):

    mask = np.zeros\_like(img)

*#channel\_count = img.shape[2]*

    match\_mask\_color = 255

    cv2.fillPoly(mask, vertices, match\_mask\_color)

    masked\_image = cv2.bitwise\_and(img, mask)

*return* masked\_image

def drow\_the\_lines(img, lines):

    img = np.copy(img)

    blank\_image = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

*for* line *in* lines:

*for* x1, y1, x2, y2 *in* line:

            cv2.line(blank\_image, (x1,y1), (x2,y2), (0, 255, 0), thickness=5)

    img = cv2.addWeighted(img, 0.8, blank\_image, 1, 0.0)

*return* img

image = cv2.imread('road.jpg')

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

print(image.shape)

height = image.shape[0]

width = image.shape[1]

region\_of\_interest\_vertices = [

    (0, height),

    (width/2, height/2),

    (width, height)

]

gray\_image = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

canny\_image = cv2.Canny(gray\_image, 100, 200)

cropped\_image = region\_of\_interest(canny\_image,

                np.array([region\_of\_interest\_vertices], np.int32),)

lines = cv2.HoughLinesP(cropped\_image,

                        rho=6,

                        theta=np.pi/180,

                        threshold=160,

                        lines=np.array([]),

                        minLineLength=40,

                        maxLineGap=25)

image\_with\_lines = drow\_the\_lines(image, lines)

plt.imshow(image\_with\_lines)

plt.show()

**6**

**input image-**



**output image-**

**7**

**A picture containing text, scene, way, road

Description automatically generated**

**7**

* **For a road video-**

*import* matplotlib.pylab *as* plt

*import* cv2

*import* numpy *as* np

def region\_of\_interest(img, vertices):

    mask = np.zeros\_like(img)

*#channel\_count = img.shape[2]*

    match\_mask\_color = 255

    cv2.fillPoly(mask, vertices, match\_mask\_color)

    masked\_image = cv2.bitwise\_and(img, mask)

*return* masked\_image

def drow\_the\_lines(img, lines):

    img = np.copy(img)

    blank\_image = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

*for* line *in* lines:

*for* x1, y1, x2, y2 *in* line:

            cv2.line(blank\_image, (x1,y1), (x2,y2), (0, 255, 0), thickness=5)

    img = cv2.addWeighted(img, 0.8, blank\_image, 1, 0.0)

*return* img

*# = cv2.imread('road.jpg')*

*#image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)*

def process(image):

    print(image.shape)

    height = image.shape[0]

    width = image.shape[1]

    region\_of\_interest\_vertices = [

        (0, height),

        (width/2, height/2),

        (width, height)

    ]

    gray\_image = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

    canny\_image = cv2.Canny(gray\_image, 100, 120)

    cropped\_image = region\_of\_interest(canny\_image,

                    np.array([region\_of\_interest\_vertices], np.int32),)

    lines = cv2.HoughLinesP(cropped\_image,

                            rho=2,

                            theta=np.pi/180,

                            threshold=60,

                            lines=np.array([]),

                            minLineLength=40,

                            maxLineGap=50)

**8**

    image\_with\_lines = drow\_the\_lines(image, lines)

*return* image\_with\_lines

cap = cv2.VideoCapture('test.mp4')

*while* cap.isOpened():

    ret, frame = cap.read()

    frame = process(frame)

    cv2.imshow('frame', frame)

*if* cv2.waitKey(1) & 0xFF == ord('q'):

*break*

cap.release()

cv2.destroyAllWindows()

**12**

**Output Result-**  “git hub link”

**9**

**Project Outcome: -**

From this project, we learnt to describe a flow process for data science problems and classified data science problems into standard typology. We also learnt about correlating results to the solution approach followed and assessing the solution approach.

**Project Conclusion: -**

From this project, we gained the knowledge of software – Jupyter Notebook, Visual studios code. We learnt to analyse the datasets and afterwards, visualizing them. We learnt about **Grayscale conversion of image, Canny Edge Detector. Region of Interest, Hough Line Transform.**

**10**