CS464- Introduction to Machine Learning Course Project Progress Report Predicting the Optimal Rotation Angle of a Steering Wheel

Section 1 – Group 14

Arda Sarp Yenicesu – Ayhan Okuyan – Emre Dönmez – Mert Erkul – Mert Sayar

Introduction

In this progress report, we will be focusing on the what we have achieved since the proposal, and what are we planning to do further on the project. As explained, we will be estimating the camera angles of vehicles based on the front camera image that we have acquired as data from Udemy's Self Driving Challenge 2. The data contains an image stream of a 40-minute drive parsed into 48000 images with 20 Hz frequency and its corresponding steering wheel labels are given as radians.

Preprocessing the Data

From the data that we are given, we will only be using the steering wheel label, hence we have disregarded the other columns of data labels such as coordinates. Then, we have started working with the images to resize them to disregard the upper 150 pixels rows of the 480 * 640 sized jpeg images. Then we have applied two separate edge detection algorithms in order to define the road edges and lines clearly. Edge detection basically is a filter to determine the sharp changes of intensity on an image. For that purpose, we have used Sobel and Canny algorithms and observed which one looks more suitable on a randomly chosen image data. The Python codes that we have written, and their respective outcomes are given below.

Sobel Edge Detection Algorithm

Sobel edge detection algorithm can be described as an approximate derivative of an image matrix. In order to achieve this, we define two 3x3 kernel matrices given as

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These correspond to a darker, gray and brighter column structure and row for the second one. Hence, when we convolve these with the image matrix, we get the gradient matrices of the image. Then, what we do is to take the magnitude of the x and y gradient matrices. We can express the mathematics as below. The letter "A" is used to denote the original image matrix.

$$G_x = K_x * A$$

$$G_y = K_y * A$$

$$|G| = \sqrt{G_x^2 + G_y^2}$$

The magnitude matrix corresponds to the image with the edges detected. The code and the outputs are below which take advantage of the Sobel function of the OpenCV library.

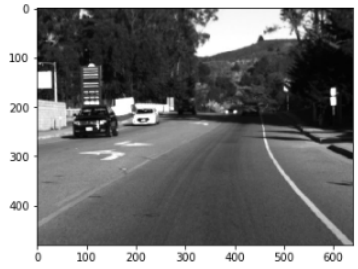
Sobel Edge Detection

Import the needed libraries

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

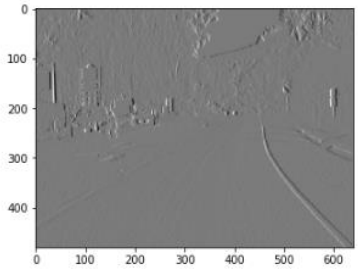
Read the image

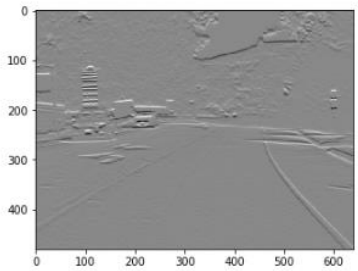
In [2]:
```
img = cv2.imread("1479425445683329266.jpg",0)
plt.imshow(img,cmap="gray")
plt.show()
```



Take the X and Y Gradients

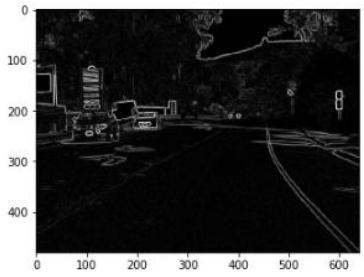
In [3]:
```
sobelx = cv2.Sobel(img,cv2.CV_32F,1 ,0, ksize=3)
sobely = cv2.Sobel(img,cv2.CV_32F,0 ,1, ksize=3)
plt.imshow(sobelx,cmap="gray")
plt.show()
plt.imshow(sobely,cmap="gray")
plt.show()
```





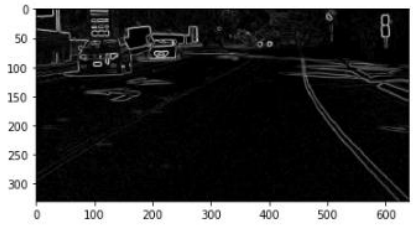
Take the magnitude of gradients

```
In [4]: sobel_mag, sobel_ang = cv2.cartToPolar(sobelx, sobely)
        #sobel_mag = sobelx + sobely
        plt.imshow(sobel_mag,cmap="gray")
        plt.show()
```



Crop the image

```
In [5]: crop_img = sobel_mag[150:, :]
        for i in range(330):
            for j in range(640):
                if crop_img[i][j] < 20:
                    crop_img[i][j] = 0
        plt.imshow(crop_img,cmap="gray")
        plt.show()
```

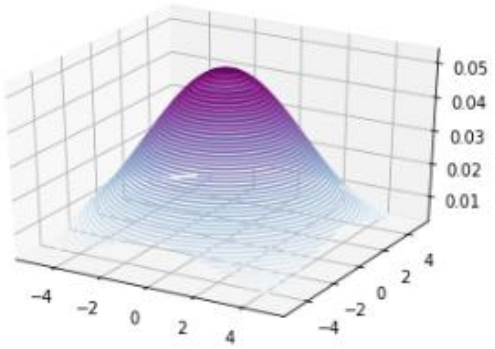


Canny Edge Detection

The other techniques that we have used is Canny Edge Detection which is a multistage detection algorithm. The first step consists of a noise reduction algorithm that uses a 5x5 Gaussian filter. The formula of a two-dimensional Gaussian function is given below.

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

An exemplary curve is given below with sigma equal to 3.



This formula is used to create a convolution matrix, in our case 5x5. By this, each pixel value is set to a weighted average of that pixel's neighborhood, with the original value having the greatest weight. Then we convolve this with the original image to blur the image and soften the details. Then, we apply the Sobel kernel that we have previously discussed to the blurred image.

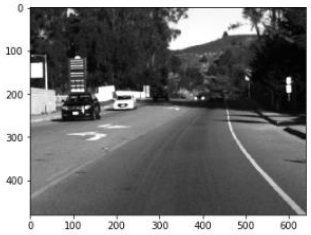
Canny Edge Detection

Import the needed libraries

In [1]:
```
import cv2
import numpy as np
import os
import glob
from matplotlib import pyplot as plt
from PIL import Image
```

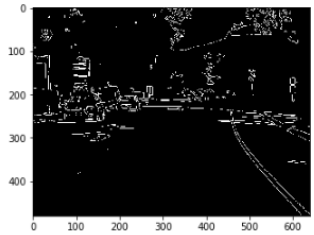
Read the image

In [2]:
```
new_img = 0
img_data = []
img = cv2.imread("147942544568329266.jpg",0)
plt.imshow(img,cmap="gray")
plt.show()
```



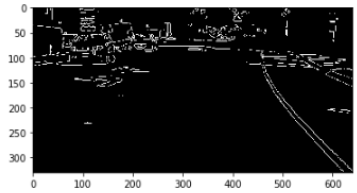
Apply Canny Edge Detection

In [3]:
```
canny = cv2.Canny(img,100,200)
img_data.append(canny)
plt.imshow(canny,cmap="gray")
plt.show()
```



Crop the Image

In [4]:
```
crop_img = canny[150:, :]
for i in range(330):
    for j in range(640):
        if crop_img[i][j] < 75:
            crop_img[i][j] = 0
plt.imshow(crop_img,cmap="gray")
plt.show()
```



Then, we have decided that using Canny edge detection is better since the road lines are clearer than the ordinary Sobel Detection. Then, we have implemented a code that would preprocess each image in the dataset and save them with their previous respective names. With this our file size has gotten substantially smaller and we have removed mostly irrelevant features.

Separation of the Training and Test Set

The data that we have accumulated contains 48000 images with their respective wheel angle labels. We have decided that from 48000 image data, 42000 of them to be training data and 6000 of them to be the test data. The possible segmentation of k-Cross Validation will be contemplated on in the future.

Extraction of Features

As explained earlier, our approach is to predict the steering wheel angle according to the images. The curvature of the preprocessed road is the feature in focus; hence we will extract these features relating to the images. The planned feature extraction method is SVR (Support Vector Regression) prior to applying the selected models overall. As the images that were preprocessed to find the lanes of the roads, the feature to be learned, curvature of the road, will be handled by averaging the pixels of the resized 330*640 images that have a sharp increase or decrease and append it to the vector so that the tendency of the road will be found about. The plan is to apply the model algorithms afterwards the feature extraction. Note that some of the models that are planned to integrate, can have hidden layers for feature extraction, thus this part might be implemented during the implementation of the algorithms.

Deciding on the Algorithms

In the project description, we are supposed to run and compare three different learning algorithms and discuss on the results. Hence, for this progress report we were supposed to get certain on what algorithms that we are going to use. We have selected to apply CNN (Convolutional Neural Network), Multilayer Perceptron and Transfer Learning.

Convolutional Neural Network

The first algorithm that we will implement is CNN. CNN algorithm consists of at least three layers, an input layer, an output layer and hidden layers. The convolution layers apply a convolution operation to the input and passes it to the next layer. The convolution corresponds to the response of an individual neuron to visual stimuli. Each convolution processes data for its own receptive field and the CNN algorithm hence reduces the number of free parameters used. The difference of CNN from a conventional multilayered perceptron is that in MLP each neuron takes input from each neuron on the previous layer however in CNN, the neuron receives input from only a subsection of the previous layer. These subsections are called as “receptive fields”.

The specifications of algorithm such as the number of convolutional and pooling layers, the structure of the fully connected layer will be decided on the course of implementation.

Multilayered Perceptron

MLP is a feed-forward artificial neural network which uses backpropagation for training. An input layer, an output layer and a hidden layer is necessary for construction. In MLP, each neuron has its own activation function. These activation functions can be both linear and nonlinear, nonlinear ones constructed to model the firing rates of biological neurons. The activation functions can be carried but mostly used ones are sigmoid functions.

Furthermore, MLPs are fully connected, meaning each neuron is connected to each neuron on the previous layer. After each data is processed, weight of that neuron changes, this is how the learning occurs and this is carried out through backpropagation. Error is represented for each neuron with the difference of target and current weight. Hence, the weights are adjusted to decrease the mean squared error for each node j, given as (where "e" is the error for one neuron),

$$E(n) = \frac{1}{2}\sum_{j} e_j^2\,(n)$$

For this purpose, we use gradient descent. The specifications for the MLP that will be constructed will be decided later on during implementation.

Transfer Learning

The transfer learning model will provide us to have accurate models without spending more time to construct a model from a scratch. In this project, we will decide on the steering wheel angle by examining the road images and this relates to an area of computer vision. The transfer learning for computer vision refers to use of pre-trained models such as VGG16, InceptionV3 and ResNet50. We will use the ResNet50 for this purpose as there are application based on this model so it will be easier to compare the accuracy of our implementation. This model provides us salient maps to extract salient features to estimate the steering wheel angle based on an image provided by the center camera of the Udacity self-driving car database.

Model Performance Measurements

The original challenge presented by Udacity expects the R^2 and RMS error calculations for applied models and has a leaderboard according to these model verification details. The proposed methodologies will be evaluated according to R^2 and RMS errors to find the best model for steering angle prediction. The optimal approach for such an application in real life will be decided on regarding the error percentage for R^2 and magnitude for RMS.

Work Load Assignment

From a broad perspective, there three separate tasks that will be done in order to successfully complete the project. Each learning algorithm will be applied by one or two members. The reports and remaining presentations will be constructed together, each person reporting his own contribution. The CNN model will be done by Mert Erkul and Ayhan Okuyan. The MLP model will be done by Arda Sarp Yenicesu and Mert Sayar and finally Transfer Learning the model will be done by Emre Dönmez and Ayhan Okuyan. The outputs of each algorithm will then be compared and discussed. The codes we will be written on Google Collab collaboratively, with distinct Python notebooks for the modules.

Conclusion

Up to this point, we have preprocessed our data by learning and using edge detection algorithms, resized and saved them. We have separated the data into training and test sets. Then we have researched and found three distinct learning algorithms that will be implemented being CNN, MLP and Transfer Learning. From onwards, we will be implementing these algorithms to find and detect the steering wheel angles.