**Case study: Lane Detection for Autonomous Vehicles using Computer Vision Algorithm**

**Roll Name Name**

CB.EN.P2AID20002 Abhishek Gopinath

CB.EN.P2AID20010 Alan Henry

CB.EN.P2AID20024 Jiss Joseph Thomas

**1. Problem Statement/Objective:**

To detect lanes on the given dataset of video or images of roads or as real time, using computer vision algorithms which could be helpful in proper implementation of autonomous driving.

**2. Dataset Description**

The dataset given to our problem statement could be either as a set of images of as video format. There are more than 1000 images for dataset containing images and two or three videos for detecting the same.

**Sample images:**

****

****

****

**3. Analytical Questions/Statistical Questions/Prediction level Question**

**Analysis [10 questions]**

1.How many vehicles in the frame

2. what are the objects detected in it

3. How many pedestrians are detected in it

4. Detecting road signs.

5. Objects on the interest area.

6. Lines inside the interest area.

7. Traffic signals detected

8. Wet area inside the interest area.

9. Dividers inside the frame.

10. How far the lanes to be detected.

**Analytics [Historical data] [10 questions]**

1. Why vehicles are less in specific area

2. Why more pedestrians are found at certain point

3. Why the road is wet?

4. What type of road we are traveling on.

5. What time do most people drive

6. Do people dim their light while passing

7. How often a certain route is taken.

8. How many speed breakers detected to analyze whether there is school nearby.

9. Why there is no horn sign detected

10. Why vehicle in the front is slowing down

**4. Block Diagram**

**5. Preprocessing**

**5.a Image Processing in Spatial Domain (Filter Details)**

**Image Smoothing**

* ***Average Filter***



* ***Weighted Average Filter***

****

* ***Gaussian Blurring***



* ***Median Filter***

****

* ***Image Sharpening***



* ***Roberts Filter***

****

* ***Sobel Filter***

****

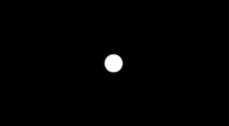
* ***Gamma Transform***

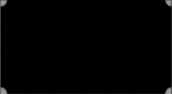


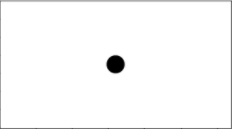
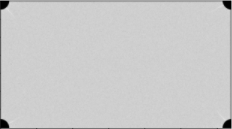
* ***Log Transform***

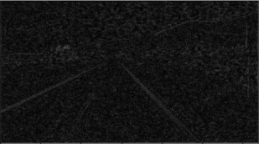


**5.b Image Processing in Frequency Domain (Filter Details)**

* **Image After Applying Noise:**
* **Low Pass Filter:**
* **Decentralized Image:**

****

* **Processed Image:**
* **High Pass Filter:**
* **Decentralize:**
* **Processed Image:**

****

* **Ideal Low Pass Filter**



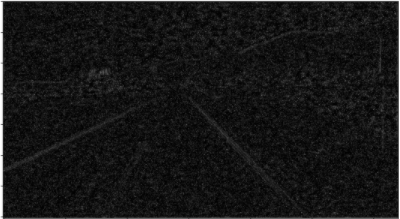
* **Butterworth Low Pass Filter:**

****

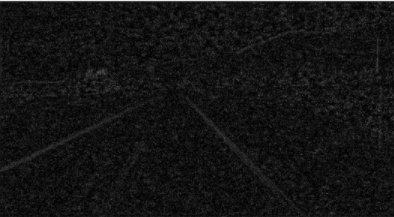
* **Gaussian Low Pass Filter**

****

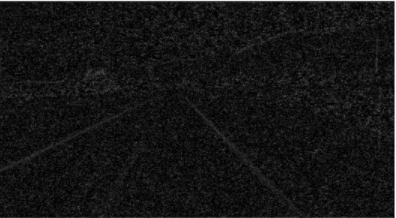
* **Ideal High pass Filter**

****

* **Butterworth High Pass Filter:**

****

* **Gaussian High Pass Filter**

****

**Observation after Comparison Between Spatial and Frequency Domain Filters:**

**Spatial Domain:**

**Input -> Image Processing -> Output**

**Frequency Domain:**

**Frequency + Distribution -> Image Processing -> Inverse Transformation -> Output**

* Spatial domain deals with image plane itself whereas Frequency domain deals with the rate of pixel change.
* Spatial domain works based on direct manipulation of pixels whereas Frequency domain works based on modifying fourier transform.
* Spatial domain takes less time to computer whereas Frequency domain takes more time to compute.

6.List of Features

|  |  |  |
| --- | --- | --- |
| Feature Name | Purpose | Category  [Image/Vision] |
| Corner |  |  |
| Lines |  |  |
| ROI |  |  |
|  |  |  |
|  |  |  |

**7.Feature Detection and Tracking**

* **Harris Corner Detection:**

1.Take the grayscale of the original image

2. Apply a Gaussian filter to smooth out any noise

3. Apply Sobel operator to find the x and y gradient values for every pixel in the grayscale image

4. For each pixel p in the grayscale image, consider a 3×3 window around it and compute the corner strength function. Call this its Harris value.

5. Find all pixels that exceed a certain threshold and are the local maxima within a certain window (to prevent redundant dupes of features)

6. For each pixel that meets the criteria in 5, compute a feature descriptor.

****

* **SIFT** **(Scale-Invariant Feature Transform):**

1. Scale-space Extrema Detection (Feature point (also called key point) detection)
   * Potential location for finding features.
2. Feature point localization
   * Accurately locating the feature key points.
3. Orientation assignment
   * Assigning orientation to key points.
4. Feature descriptor generation.
   * Describing the key points as a high dimensional vector.
5. Key point Matching

****

* **SURF (Speeded-Up Robust Features):**

1. Feature Extraction
   * Integral Images
   * Hessian matrix-based interest points
   * Scale-space representation
2. Feature Description
   * Orientation Assignment
   * Extract Descriptor Components

* **Multiscale Oriented Patches Descriptor (MOPS)**
* ***Translation*** ***(T = MT1)***



* ***Rotation*** ***(T = MRMT1)***

****

* ***Affine Transformation***
* ***Scaling***

#### **Affine Transformation**

****

#### **Perspective Transformation**

****

#### **Scaling**



8.Scene

[Place some sample images]

[Describe the scene with the image and list the objects to be identified]

[Make sure that human face also appears in the image]

9.List of Objects in scene and the features

**[Refer section 8 and present the table contents for section 9]**

|  |  |
| --- | --- |
| Object Name | List of features for the object |
| Lanes | Shape  Color |
| Vehicles | Shape  Color |
| Pedestrians | Shape  Color |
| Signals | Color |

10.Face Detection









11.Refer internet and fill the following related to face detection algorithms and Deep learning architecture

[A sample entry has been provided. Can also include algorithms related to violajones,adaboost classifier also]

|  |  |  |  |
| --- | --- | --- | --- |
| Face Detection Algorithm | URL | Dataset | Deep learning architecture |
| Facenet | <https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/#:~:text=FaceNet%20is%20a%20face%20recognition,of%20face%20recognition%20benchmark%20datasets.&text=About%20the%20FaceNet%20face%20recognition,implementations%20and%20pre%2Dtrained%20models>. |  | Facenet |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

12.

**Performance Metrics**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Category** | **Purpose** | **Formula** |
| Peek Signal to Noise Ratio (PSNR) | Spatial Domain | Gives the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. | Where is the maximum possible pixel value of the image; is the Mean Squared Error; |
| Mean Squared Error (MSE) | Spatial Domain | Measures the average squared difference between the estimated values and the actual value. It is a risk function, corresponding to the expected value of the squared error loss. | Where is the Mean Squared Error; is the number of data points; are the observed values; are the predicted values; |
| Structural Similarity Index (SSIM) |  | It is a perceptual metric that quantifies image quality degradation\* caused by processing such as data compression or by losses in data transmission. | Where is the Structural Similarity Index;  is the average of ; is the average of ; is the variance of ; is the variance of ; is the covariance of and; and are two variables to stabilize the division with weak denominator; |

13.Deep Learning Architectures

[Identify the applications where the deep learning architectures has been applied]

[refer DL\_1.pdf]

[refer : <https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-machine-learning-deep-learning-architectures/>]

[refer: https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Architecture Name | Category | Learning | Year of Design | Applications |
| LSTM | RNN | Supervised Learning |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

14.Training ,Testing and Validation

[Place the contents that is under :

<https://lionbridge.ai/training-data-guide/>]

[Provide your inference for 10-rule]

**References**

1. <https://github.com/overtunned/lane_detection/tree/main/Dataset>
2. [GitHub - rslim087a/road-video: Video required for finding lane lines](https://github.com/rslim087a/road-video)
3. [GitHub - rslim087a/road-image: image required for finding lane lines](https://github.com/rslim087a/road-image)
4. [CULane dataset](https://xingangpan.github.io/projects/CULane.html)
5. [Real-time detection of road lane-lines for autonomous driving](https://www.researchgate.net/publication/331478663_Real-Time_Detection_of_Road_Lane-Lines_for_Autonomous_Driving)
6. [Multi-Lane Detection and Tracking Using Vision for Traffic Situation Awareness](https://ieeexplore.ieee.org/document/9253415)
7. [Real-Time Tracking and Lane Line Detection Technique for an Autonomous Ground Vehicle System](https://link.springer.com/chapter/10.1007/978-981-15-0633-8_156#:~:text=Vehicle%20tracking%20has%20been%20performed,using%20adaptive%20background%20subtraction%20technique.)