**Case study:** [**Lane Detection for Autonomous Vehicles using Computer Vision Algorithm**](1.%09https:/github.com/overtunned/lane_detection/tree/main/Dataset)

|  |  |  |
| --- | --- | --- |
| **Course Code** | **19AI621** | |
| **Course Name** | **Computer Vision** | |
| **Course Instructor** | **Dr. Senthilkumar T** | |
|  | |  |
| **Team Members** | **Roll Number** | **Contributions** |
| Abhishek Gopinath | CB.EN.P2AID20002 | Dataset, Analytical/Analysis Questions, Block Diagram, Frequency Domain Filters, SURF, Deep Learning ConvLSTM, Literature Survey |
| Alan Henry | CB.EN.P2AID20010 | Dataset, Analytical/Analysis Questions, Harris Corner Detection, Viola-Jones Face Detection, MTCNN Face Detection, Lucas Kanade Optical Flow, Metrics, Literature Survey |
| Jiss Joseph Thomas | CB.EN.P2AID20024 | Dataset, Analytical/Analysis Questions, Spatial Domain Filters, SIFT, Horn Schunck, Dense Optical Flow, Action Recognition, YOLOv3, Literature Survey |

**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 the proper implementation of autonomous driving.

**2. Dataset Description:**

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

|  |  |
| --- | --- |
| **Type** | Colour Image |
| **Size** | Multiple sizes |
| **Row \* Column** | NA |
| **Resolution** | NA |
| **Colour Model** | RGB |
| **Format** | PNG |
| **Data Acquisition** | Web, Camera |
| **No of classes** | 3 |
| **No of Images** | 853 |
| **Annotation** | PASCAL VOC format. |

**Sample Images: **

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

**Analysis**

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**

1. Why vehicles are less in a specific area

2. Why more pedestrians are found at a certain point

3. Why the road is wet?

4. What type of road we are travelling on.

5. What time do most people drive

6. Do people dim their light while passing

7. How often a certain route is taken.

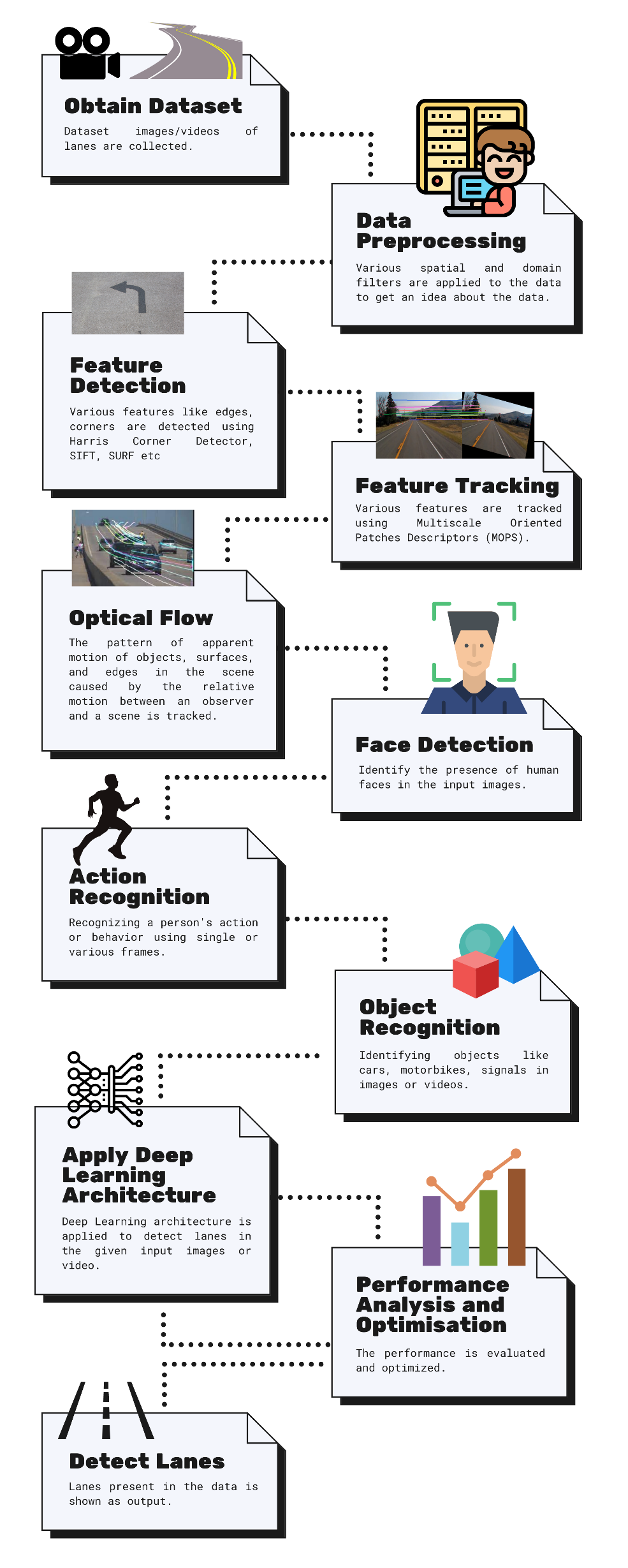
8. How many speed breakers were detected to analyze whether there is a school nearby.

9. Why there is no horn sign detected

10. Why vehicle in the front is slowing down

**4. Block Diagram:**

****



**5. Preprocessing:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Domain** | **Filter** | **Input Image** | **Output Image** | **Principle** |
| **Spatial**  **Domain** | **Image Smoothing**  ***Average Filter*** |  |  | Reduce the amount of intensity variation between neighboring pixels. The average filter works by moving through the image pixel by pixel, replacing each value with the average value of neighboring pixels, including itself. |
| ***Weighted Average Filter*** |  |  | Give more weight to the center value, due to which the contribution of the center becomes more than the rest of the values. Due to weighted average filtering, we can control the blurring of the image. |
| ***Gaussian Blurring*** |  |  | Blurring the image by a Gaussian function. Applying a Gaussian blur has the effect of reducing the image's high-frequency components. So, Gaussian blur is thus a low pass filter. |
| ***Median Filter*** |  |  | Non-linear digital filtering technique used for removing noise from an image or signal. It is done by sliding a window over the image. The filtered image is obtained by placing the median of the values in the input window, at the location of the center of that window, at the output image. |
| ***Image Sharpening*** |  |  | It highlights edges and fine details in the image. This becomes an effective high pass filter. |
| ***Roberts Filter*** |  |  | This filter highlights the edges of objects present in the image. |
| ***Sobel Filter*** |  |  | Edge detection filter that calculates the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction. |
| ***Gamma Transform*** |  |  | It controls the overall brightness of an image. Varying the amount of γ (Gamma) correction changes not only the brightness/ enhancement of the image but also the ratios of red to green to blue. |
| ***Log Transform*** |  | gamma = 0.8 | It replaces all pixel values, present in the image, with its logarithmic values. Log transformation is used for image enhancement as it expands dark pixels of the image as compared to higher pixel values. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Domain** | **Filter** | **Input Image**  **(with added noise)** | **Output Image** | **Principle** |
| **Frequency Domain** | ***Low Pass Filter*** | **Decentralized Image:** |  | The image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels. Using a low pass filter tends to retain the low-frequency information within an image while reducing the high-frequency information. |
| ***High Pass Filter*** | **Decentralized Image:** |  | It tends to retain the high-frequency information within an image while reducing the low-frequency information. The kernel of the high pass filter is designed to increase the brightness of the center pixel relative to neighboring pixels. |
| ***Ideal Low Pass Filter*** |  |  | It is used for image smoothing in the frequency domain. It removes high-frequency noise from a digital image and preserves low-frequency components. |
| ***Ideal High pass Filter*** |  |  | It is used for image sharpening in the frequency domain. It enhances the fine details and highlights the edges in the digital image. It removes low-frequency components from an image and preserves high-frequency components. |
| ***Butterworth Low Pass Filter*** |  |  | It is used for image smoothing in the frequency domain. It removes high-frequency noise from a digital image and preserves low-frequency components. It is commonly used for motion analysis. |
| ***Butterworth High Pass Filter*** |  |  | It enhances the fine details and highlights the edges in a digital image. It removes low-frequency components from an image and preserves high-frequency components. It is commonly used for motion analysis. |
| ***Gaussian Low Pass Filter*** |  |  | It is used for image smoothing in the frequency domain. It removes high-frequency noise from a digital image and preserves low-frequency components. It is commonly used for motion analysis. |
| ***Gaussian High Pass Filter*** |  |  | It enhances the fine details and highlights the edges in a digital image. It removes low-frequency components from an image and preserves high-frequency components. It is commonly used for motion analysis. |

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

**Spatial Domain:**

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

**Frequency Domain:**

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

* The spatial domain deals with the image plane itself whereas the Frequency domain deals with the rate of pixel change.
* The spatial domain works based on direct manipulation of pixels whereas the Frequency domain works based on modifying Fourier Transform.
* The spatial domain takes less time to compute whereas the Frequency domain takes more time to compute.

**6. List of Features:**

|  |  |  |
| --- | --- | --- |
| **Feature Name** | **Purpose** | **Category**  **[Image/Vision]** |
| Edges | To detect lanes | Image or Video |
| Blobs | To detect vehicles | Image or Video |
| Corner | To identify the type of vehicles o object, present in the data | Image or Video |
| Ridges | To identify the shape and outline the objects | Image or Video |

**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. Keypoint 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):***

|  |  |  |
| --- | --- | --- |
| **Multiscale Oriented Patches Descriptor (MOPS)** | ***Translation*** (***T = MT1)*** |  |
| ***Rotation*** (***T = MRMT1)*** |  |
| ***Affine Transformation*** |  |
| ***Perspective Transformation*** |  |
| ***Scaling*** |  |

#### **After Feature Detection and Tracking**:



**8. Scene:**

|  |  |  |
| --- | --- | --- |
| **Scene** | **Feature to be detected** | **Sample Image** |
| **Image of some cyclists on the road** | Faces (Pedestrian or Cyclists) |  |
| **Image of a road with lanes visible** | Lanes |  |
| **Image of a road with lanes and rarely visible cars** | Cars |  |

**9. List of objects in the scene and the features:**

|  |  |
| --- | --- |
| **Object Name** | **List of features for the object** |
| Lanes | Shape,  Colour,  Texture |
| Vehicles | Shape,  Colour,  Size |
| Pedestrians/Cyclists | Shape,  Colour,  Size |
| Signals and Signboards | Shape,  Colour,  Size |

**10. Face Detection**

**Viola-Jones Algorithm:**





|  |  |
| --- | --- |
| **Input Image** | **Output Image** |
|  |  |

**Number of faces present: 15**

**Number of faces detected: 9**

**Number of falsely detected faces: 1**

**False Positive Rate : 8/9**

**Detection Rate :7/15**

**11. Face detection algorithms and Deep learning architectures:**

|  |  |  |
| --- | --- | --- |
| Face Detection Algorithm | URL | 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 |
| Histogram of Oriented Gradients using Dlib | https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/ |  |
| Haar Cascade Classifiers using OpenCV | https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177 |  |
| CNN based face detector from dlib | https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c | dlib |
| MTCNN Face Detector for Keras | https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923 | MTCNN |

**12. Performance Evaluation Metrics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Category** | **Purpose** | **Formula** |
| Peak 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;  A PSNR value of 30 dB above is preferred and ideally higher it is that much better. |
| 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;  There is no correct value. The closer it is to zero better and if it's zero it means the model is perfect. |
| Structural Similarity Index (SSIM) | Spatial Domain | 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 a weak denominator;  Generally, it is ideal to get 1 which means high structural similarity and 0 if no structural similarity |
| Intersection Over Union (IoU) | Spatial Domain | It is essentially a method to quantify the percent overlap between the target mask and our prediction output. Used as object detection evaluation metrics. | Ideally, **IoU** > 0.5 is considered a good prediction. |
| Accuracy |  | It is the ratio of correctly predicted observation to the total observations | An ideal value for it is closer to 100. |
| Precision |  | The ratio of correctly predicted positive observations to the total predicted positive observations is the precision. | An ideal value for it is closer to 100. |
| Recall |  | Recall is the ratio of correctly predicted positive observations to all observations in actual class | An ideal value for it is closer to 100. |
| F1-Score |  | The weighted average of Precision and Recall is the F1 Score. | An ideal value for it is closer to 100. |

**13. Deep Learning Architectures:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Architecture Name | Category | Learning | Year | Applications |
| CNN (Convolutional neural networks) |  | Supervised Learning | 2006 | Image recognition, video analysis, and natural language processing |
| RNN (Recurrent neural networks) |  | Supervised Learning | 1980 | Speech recognition and handwriting recognition |
| SOM (Self-organizing Maps) |  | Unsupervised Learning | 1980 | Dimensionality reduction, clustering high-dimensional inputs to 2-dimensional output, radiant grade result, and cluster visualization |
| AE(Autoencoders) |  | Unsupervised Learning | 1980 | Dimensionality reduction, data interpolation, and data compression/decompression |
| LSTM (Long Short-Term Memory) | RNN | Supervised Learning | 1997 | Image and video captioning systems |
| GRU (Gated Recurrent Unit) | RNN | Supervised Learning | 2014 | Natural language text compression, handwriting recognition, speech recognition, gesture recognition, image captioning |
| RBM (Restricted Boltzmann Machines) | AE | Unsupervised Learning | 1986 | Dimensionality reduction and collaborative filtering |
| DBN (Deep Belief Networks) |  | Supervised Learning | 2006 | Image recognition, information retrieval, natural language understanding, and failure prediction |
| DSN (Deep Stacking Networks) |  | Supervised Learning | 2011 | Information retrieval and continuous speech recognition |
| LeNet | Spatial Exploitation | Supervised Learning | 1998 | Recognizing simple digit images,  recognition of handwritten zip code digits |
| AlexNet | Spatial Exploitation | Supervised Learning | 2012 | allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU |
| VGGNet | Spatial Exploitation | Supervised Learning | 2014 | Architecture included in the Keras library used for large scale image recognition |
| GoogleNet | Spatial Exploitation | Supervised Learning | 2014 | It achieves efficiency through reduction of the input image, whilst simultaneously retaining important spatial information. It is designed to be a powerhouse with increased computational efficiency compared to some of its predecessors or similar networks created at the time.  image classification and object detection |
| ResNet | Depth + Multi-Path | Supervised Learning | 2015 | Extremely *deep* networks can be trained using standard SGD (and a reasonable initialization function) through the use of residual modules.  builds on constructs known from pyramidal cells in the cerebral cortex |
| ResNext | Width | Supervised Learning | 2016 | ResNeXts is the adding of parallel towers/branches/paths within each module |
| DenseNet | Multi-Path | Supervised Learning | 2017 | Used to keep increasing the depth of deep convolutional networks. Require fewer parameters than an equivalent traditional CNN, as there is no need to learn redundant feature maps.  DenseNets layers are very narrow (e.g. 12 filters), and they just add a small set of new feature maps. |
| PolyNet | Width | Supervised Learning | 2017 | A Very **Deep PolyNet** is composed based on the module. Compared to Inception-ResNet-v2, **PolyNet** reduces the Top-5 validation error on single crops from 4.9% to 4.25%, and that on multi-crops from 3.7% to 3.45%. **PolyNet**, By using PolyInception module, better than Inception-ResNet-v2 |
| PyramidalNet | Width | Supervised Learning | 2017 | Used in Spectral-Spatial Hyperspectral Image Classification |
| YOLO (You Only Look Once) | Spatial Exploitation | Supervised Learning | 2015 | Object detection can help with image classification |
| SqueezeNet | Spatial Exploitation | Supervised Learning |  | It can obtain AlexNet-level accuracy (~57% rank-1 and ~80% rank-5) at only 4.9MB through the usage of “fire” modules that “squeeze” and “expand”. |
| SegNet | Spatial Exploitation | Supervised Learning |  | Autonomous driving, scene understanding |
| GAN (Generative Adversarial Network) | Spatial Exploitation | Supervised Learning |  | Generate Examples for Image Datasets. Generate Photographs of Human Faces. Generate Realistic Photographs. Generate Cartoon Characters |

**14. Training, Testing, and Validation**

* Training data is used to help our machine learning model make predictions. It’s the largest part of our dataset, forming at least 70-80% of the total data we’ll use to build our model.
* Validation data is primarily used to determine whether our model can correctly identify new data or if it’s overfitting to our original dataset.
* Testing data is used after both training and validation. It aims to test the accuracy of our final model against our targets.

**Rule of 10:**

It is a common rule of thumb that we need more than 10 times the data for the model than its degree of freedom. The degree of freedom can be anything it can be an attribute, a parameter, or a column in our data that affects the output of the model. The main aim is to compensate for most of the variability that our parameters may bring into the input of the model.

**15.** [**Face Detection using MTCNN**](https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/)

MTCNN or Multi-Task Cascaded Convolutional Neural Networks is a neural network that detects faces and facial landmarks on images. It was published in 2016 by Zhang et al. MTCNN output example. MTCNN is one of the most popular and most accurate face detection tools today.

Face Detection comparison result: https://datawow.io/blogs/face-detection-haar-cascade-vs-mtcnn

|  |  |  |
| --- | --- | --- |
|  | **Haar cascade** | **MTCNN** |
| Number of images in UTK Face | 24,111 | 24,111 |
| Number of cropped faces | 19,915 | 21,666 |
| Total number of extra faces from a single image | 947 | 428 |
| Recall | (19915 / 24111)\*100 = 82.60% | (21666 / 24111)\*100 = 89.85% |
| Precision | (18968 / 19915)\*100 = 95.24% | (21238/21666)\*100 = 98.02% |
| Face Detection in our sample image: |  |  |
| Number of faces present | 15 | 15 |
| Number of faces detected | 9 | 18 |
| Number of falsely detected faces | 1 | 3 |
| False Positive Rate | 1/9 | 3/18 |
| Detection Rate | 7/15 | 15/15 |

Multi-task Cascaded Convolutional Networks ([**MTCNN**](https://medium.com/@iselagradilla94/multi-task-cascaded-convolutional-networks-mtcnn-for-face-detection-and-facial-landmark-alignment-7c21e8007923)) is a framework developed as a solution for both face detection and face alignment. The process consists of three stages of convolutional networks that can recognize faces and landmark locations such as eyes, nose, and mouth.

Three stages: 1) The Proposal Network (P-Net), 2) The Refine Network (R-Net), 3) The Output Network

Three tasks: 1) Face Classification, 2) Bounding Box Regression, 3) Facial Landmark Localization

**16. Optical Flow**

* [**Sparse Optical Flow: Lucas-Kanade method**](https://docs.opencv.org/master/d4/d8b/group__datasets__ar.html)

All the neighboring pixels will have similar motions. Lucas-Kanade method takes a 3x3 patch around the point. So all the 9 points have the same motion. We can find () for these 9 points. So now our problem becomes solving 9 equations with two unknown variables which are over-determined. A better solution is obtained with the least square fit method. Below is the final solution which is two equation-two unknown problems and solves to get the solution.

****

* [**Horn–Schunck method**](https://github.com/lmiz100/Optical-flow-Horn-Schunck-method)

The Horn–Schunck method of estimating optical flow is a global method that introduces a global constraint of smoothness to solve the aperture problem. It assumes smoothness in the flow over the whole image. Thus, it tries to minimize distortions inflow and prefers solutions that show more smoothness.

|  |  |
| --- | --- |
|  |  |
| **C:\Users\Jiss\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\EF0E7D72.tmp** | |

* [**Dense Optical Flow**](https://docs.opencv.org/master/d4/d8b/group__datasets__ar.html)

Dense Optical Flow computes the optical flow for all the points in the frame. It is based on Gunner Farneback's algorithm which is explained in "Two-Frame Motion Estimation Based on Polynomial Expansion" by Gunner Farneback.



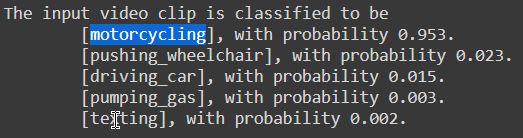
**17. Action Recognition**

Action recognition aims to recognize the actions and goals of one or more agents from a series of observations on the agents actions and the environmental conditions.

|  |  |  |
| --- | --- | --- |
| **Scene** | **Inference** | **Required Action** |
| A person walking through the side of the road/lane | Probability of him/her continue walking or cross the road | Slow down or stop the vehicle |
| The vehicle in the front is slowing down | Probability of heavy traffic in front or it is going to be parked or maybe turn the vehicle around | Slow down or stop the vehicle |
| Traffic signal turning to red/green |  | Stop/Proceed |
| Traffic signal turning to yellow |  | Slow down the vehicle |
| One vehicle hitting another vehicle | Probability of occurring an Accident | Verify manually and pass the message to Hospital and Police Station if required |

Sample images from the video:

Recognized actions from the input video:

****

**18. Object Recognition: YOLO (You Only Look Once)**

YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region.

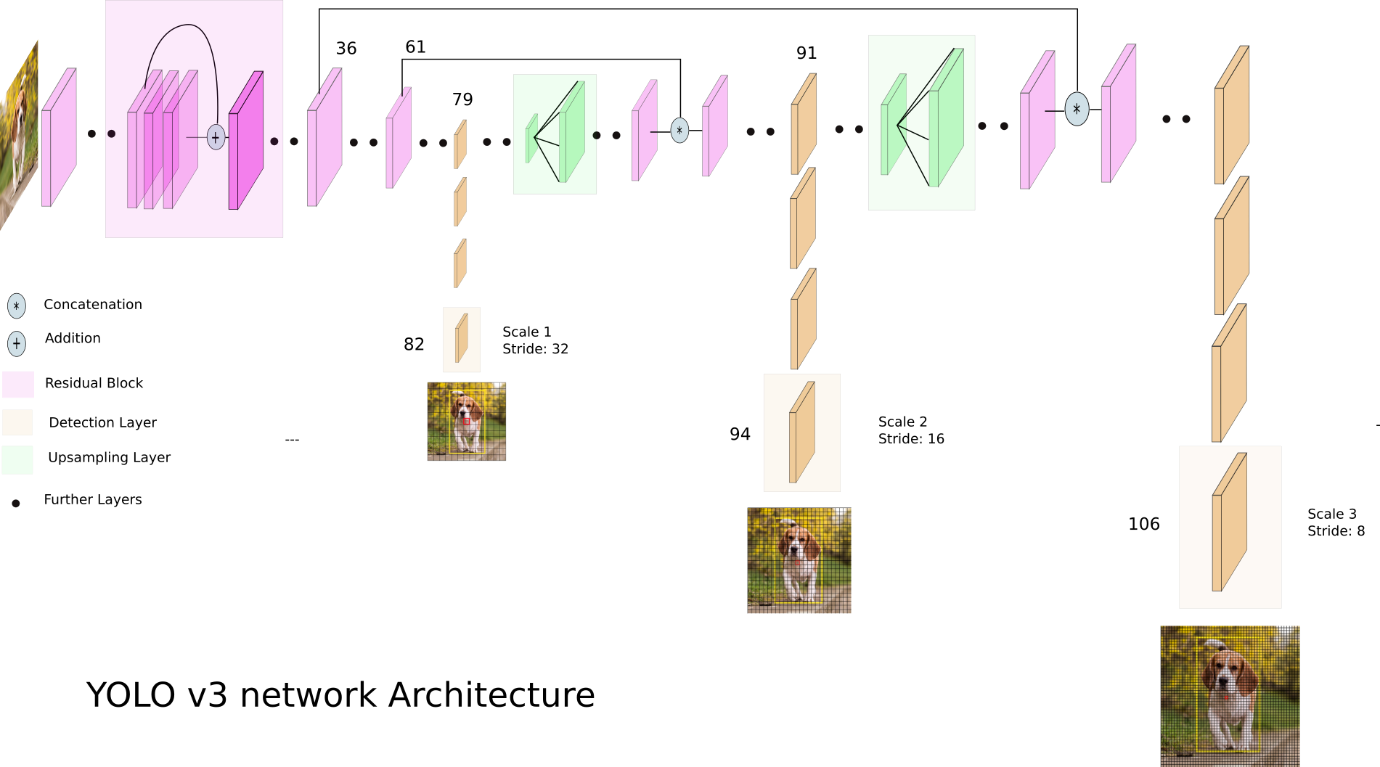
The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

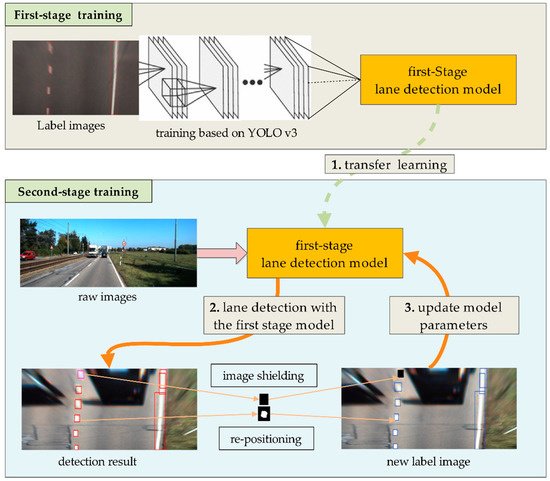
With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

YOLO learns generalizable representations of objects so that when trained on natural images and tested on the artwork, the algorithm outperforms other top detection methods.

YOLOv3 uses a variant of Darknet, a framework to train neural networks, which originally has 53 layers. For the detection task, another 53 layers are stacked onto it, accumulating to a total of a 106-layer fully convolutional architecture. This explains the reduction in speed in comparison with the second version, which only has 30 layers.

* **YOLOv3 Archiitecture Diagram:**



[](https://www.mdpi.com/sensors/sensors-18-04308/article_deploy/html/images/sensors-18-04308-g001-550.jpg)YOLOv3 makes detections at three different scales. The detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network. The shape of the detection kernel is 1 x 1 x (B x (5 + C) ). Here B is the number of bounding boxes a cell on the feature map can predict, “5” is for the 4 bounding box attributes and one object confidence, and C is the number of classes. In YOLO v3 trained on COCO, B = 3 and C = 80, so the kernel size is 1 x 1 x 255. The feature map produced by this kernel has identical height and width to the previous feature map and has detection attributes along with the depth as described above.

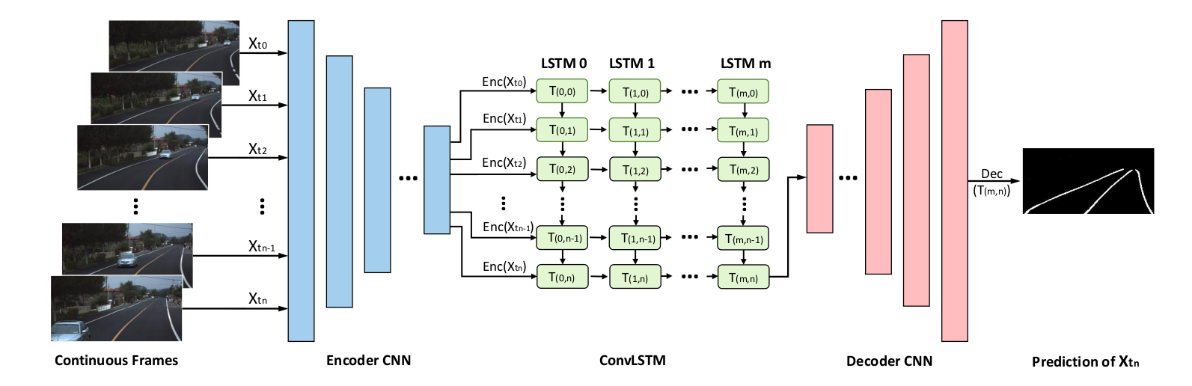
|  |  |
| --- | --- |
| **Sample frame from the input video** | **Sample frame from the output video** |
|  |  |
|  |  |
|  |  |
|  |  |

* [**YOLOv3 configuration parameters**](https://learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/)**:**

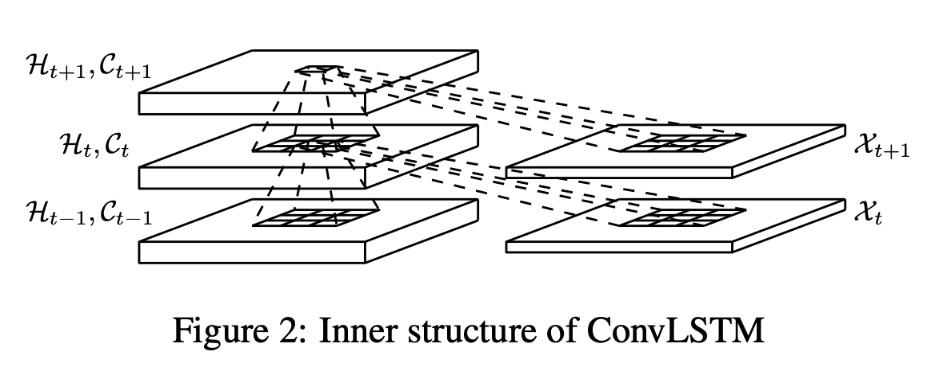
|  |  |
| --- | --- |
| Batch hyper-parameter: | batch=64  subdivisions=16 |
| Width, Height, Channels: | width=416  height=416  channels=3 |
| Momentum and Decay: | momentum=0.9  decay=0.0005 |
| Learning Rate, Steps, Scales, Burn In: | learning\_rate=0.001  policy=steps  steps=3800  scales=.1  burn\_in=400 |
| Data augmentation | angle=0  saturation = 1.5  exposure = 1.5  hue=.1 |
| Number of iterations | max\_batches=5200 |

**19. Deep Learning Architecture: LSTM (Long Short-Term Memory)**

Long Short-Term Memory (LSTM)s has the property of selectively remembering patterns for long durations of time.LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. It deals with the vanishing gradient problem encountered by traditional RNNs.

* [](https://raw.githubusercontent.com/qinnzou/Robust-Lane-Detection/master/LaneDetectionCode/save/result/network.png)**ConvLSTM Architecture Diagram:**

ConvLSTM is a type of recurrent neural network for Spatio-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. **ConvLSTM** is when we have the matrix multiplication calculation of the input with the **LSTM** cell replaced by the convolution operation. In contrast, **CNN**-**LSTM** is two different modules that are combined. The **CNN** is a regular **CNN** that acts as a spatial feature extractor.

If we view the states as the hidden representations of moving objects, a ConvLSTM with a larger transitional kernel should be able to capture faster motions while one with a smaller kernel can capture slower motions. **[](https://paperswithcode.com/method/convlstm)**

Inner Structure of ConvLSTM

|  |  |  |
| --- | --- | --- |
| **Sample frame from the input video** | **Ground Truth** | **Sample frame from the output video** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Literature Survey:**

* **Datasets relevant to application:**

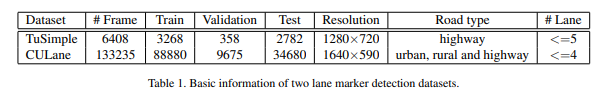
[**TuSimple Dataset:**](https://paperswithcode.com/dataset/tusimple)

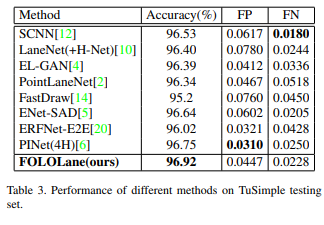
The TuSimple dataset consists of 6,408 road images on US highways. The resolution of the image is 1280×720. The dataset is composed of 3,626 for training, 358 for validation, and 2,782 for testing called the TuSimple test set of which the images are under different weather conditions.

Related Paper: [Focus on Local: Detecting Lane Marker from Bottom Up via Key Point](https://arxiv.org/abs/2105.13680v1)

Methodologies Used:

* Network for local geometry
  + Key point estimation
  + Local geometry construction
  + Network architecture
* Decoder for global geometry
  + Greedy decoder

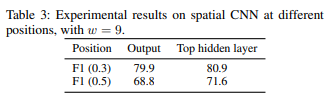
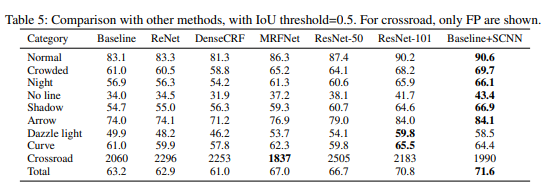




[**CULane Dataset**](https://xingangpan.github.io/projects/CULane.html)**:**

CULane is a large-scale challenging dataset for academic research on traffic lane detection. It is collected by cameras mounted on six different vehicles driven by different drivers in Beijing. More than 55 hours of videos were collected, and 133,235 frames were extracted. The dataset is divided into 88880 images for the training set, 9675 for the validation set, and 34680 for the test set. The test set is divided into normal and 8 challenging categories.

Related paper: [Spatial As Deep: Spatial CNN for Traffic Scene Understanding](https://arxiv.org/abs/1712.06080)

Experimental Results:

[**tvtLANE Dataset:**](https://drive.google.com/drive/folders/1MI5gMDspzuV44lfwzpK6PX0vKuOHUbb_?usp=sharing)

This dataset contains 19383 image sequences for lane detection, and 39460 frames of them are labeled. These images were divided into two parts, a training dataset contains 9548 labeled images and augmented by four times, and a test dataset has 1268 labeled images. The size of images in this dataset is 128\*256.

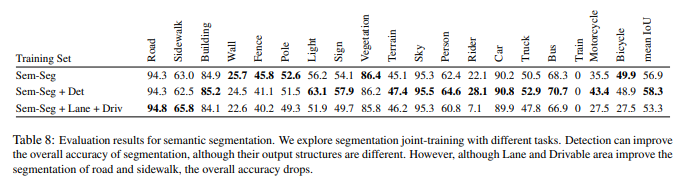
[**BDD100K**](https://bair.berkeley.edu/blog/2018/05/30/bdd/)**:**

BDD100K, the largest driving video dataset with 100K videos and 10 tasks to evaluate the exciting progress of image recognition algorithms on autonomous driving. The dataset possesses geographic, environmental, and weather diversity, which is useful for training models that are less likely to be surprised by new conditions. The dataset consists of 100,000 videos. Each video is about 40 seconds long, 720p, and 30 fps. The videos and their trajectories can be useful for imitation learning of driving policies, as in CVPR 2017 paper.

Related paper: [BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning](https://arxiv.org/abs/1805.04687)

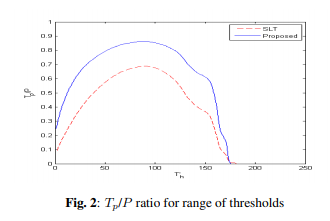
Methodologies Used:

* Image Tagging
* Object Detection
* Lane Marking
* Drivable Area
* Semantic Instance Segmentation
* Multiple Object Tracking
* Multiple Object Tracking and Segmentation
* Imitation Learning

Experimental Results:

* **Feature Extraction:**

[**Robust lane detection & tracking based on novel feature extraction and lane categorization**](https://ieeexplore.ieee.org/abstract/document/6855185)**(IEEE):**

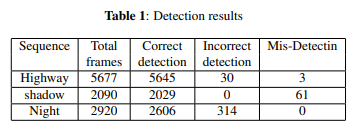
They introduce a robust lane detection and tracking algorithm to cope with complex scenarios and to decrease the effect of thresholds. For lane feature extraction, an extension to the symmetrical local threshold (SLT) is proposed to improve the feature map and obtain orientation information. Then, while creating a Hough accumulator, obtained orientation information is used to decrease computational complexity (≈ 60 times) and acquire a clearer accumulator. The left and right lanes are categorized by applying a mask on the Hough accumulator, which leads to low computational complexity and reduced sensitivity to thresholding.

Dataset Used: RoMa Datasets

Methodologies Used:

* Symmetrical Local Threshold
* Extension of Symmetrical Local Threshold
* Distance Transform
* Hough Transform
* Optimized Lane Categorization

Experimental Results:

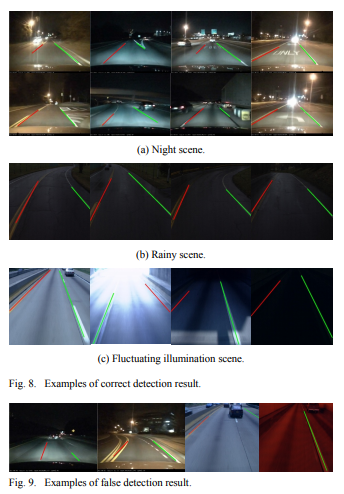


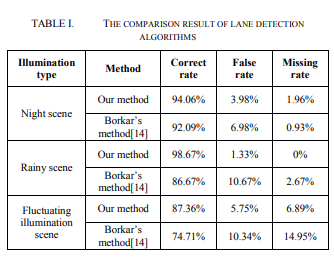
[**Lane detection algorithm based on local feature extraction**](https://ieeexplore.ieee.org/abstract/document/6775702)**(IEEE):**

An effective local feature extraction algorithm for lane detection is proposed in this paper. First, a lane region of interest (ROI) is determined by the location of the road surface that appeared in an image. Then, the light intensity and width of lane markings are taken as the local feature. A local threshold segmentation algorithm is utilized to extract lane-marking candidates followed by a morphological operation to obtain the accurate lane. An edge refining procedure is used to eliminate the interference and reduce computational cost. Finally, the lane marking is detected using Hough transform with some subsidiary conditions.

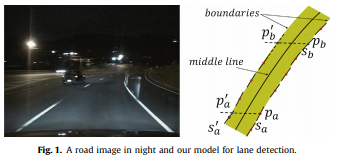
Methodologies Used:

* Region of Interest Selection
* Segmentation by the Median Local Threshold (MLT) Algorithm
* Interference Removed by Morphological Operation
* Edge Refining
* Lane Detection using Hough transform

Experimental Results:

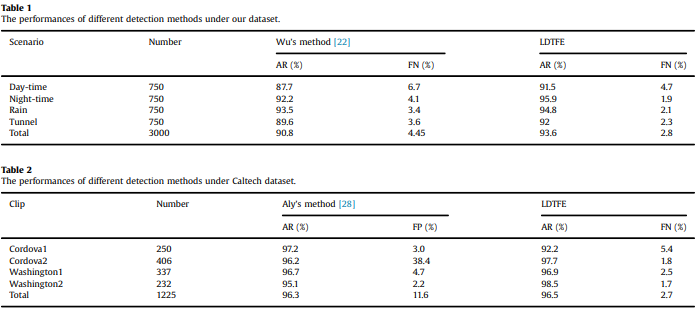


[**Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting**](https://www.sciencedirect.com/science/article/abs/pii/S0031320315004690)**(ScienceDirect):**

They proposed a novel lane detection method. Their method regards lane boundary as a collection of small line segments. They proposed a modified Hough Transform to detect small line segments. Small line segments are clustered based on our proposed similarity measurement. Removing interferential clusters depends on the balance of small line segments.

Dataset Used: Their own custom dataset, Caltech Dataset

Methodologies Used:

* Two-stage feature extraction
  + Small line segment extraction
  + Small line segment clusters
* Identification of lanes
  + Finding the candidate clusters
  + Identifying the final lanes

Experimental Results:

* **Feature Tracking:**

[**Effective Lane Detection and Tracking Method Using Statistical Modeling of Color and Lane Edge-Orientation (IEEE):**](https://ieeexplore.ieee.org/abstract/document/5369883)

This paper proposes an effective lane detection and tracking method using statistical modeling of lane color and edge orientation in the image sequence. At first, we will address some problems of classifying a pixel into two classes (lane or background) and detecting one exact lane. Generally, the probability of a pixel classification error conditioned on the distinctive feature vector can be decreased by selecting more distinctive features. A proposed pixel classifier model (Bayes decision rule for minimizing the probability of error) uses two distinctive features, lane color, and edge orientation, for classifying a lane pixel from a background image.

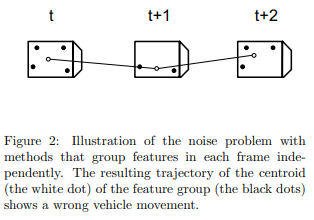
Methodologies Used:

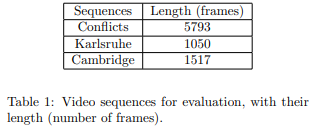
* Lane segmentation based on statistical Bayes decision rule
* Estimation of state-conditional densities for lane color and edge-orientation features
* Postprocessing
* Adaptive learning of estimated PDF

[**A feature-based tracking algorithm for vehicles in intersections (IEEE):**](https://ieeexplore.ieee.org/abstract/document/1640414)

Intelligent Transportation Systems need methods to automatically monitor road traffic and especially track vehicles. Most research has concentrated on highways. Traffic in intersections is more variable, with multiple entrances and exit regions. This paper describes an extension to intersections of the feature-tracking algorithm. Vehicle features are rarely tracked from their entrance in the field of view to their exit. Our algorithm can accommodate the problem caused by the disruption of feature tracks. It is evaluated on video sequences recorded on four different intersections.

Methodologies Used:

* Grouping features

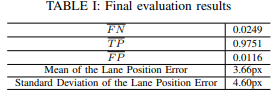
Experimental Results:

[**A novel multi-lane detection and tracking system (IEEE):**](https://ieeexplore.ieee.org/abstract/document/6232168)

In this paper, a novel spline-based multi-lane detection and tracking system is proposed. Reliable lane detection and tracking is an important component of lane departure warning systems, lane-keeping support systems, or lane change assistance systems. The major novelty of the proposed approach is the usage of the so-called Catmull-Rom spline in combination with the extended Kalman filter tracking. The new spline-based model enables accurate and flexible modeling of the lane markings. At the same time, the application of the extended Kalman filter contributes significantly to the system's robustness and stability.

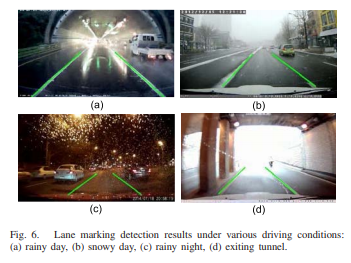
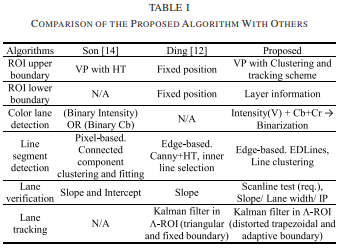
Methodologies Used:

* Using Catmull-Rom Splines for Lane Marking Representation
* State Vector
* Motion Model
* Measurement model
* Initialization of the Tracker
* Lane Merging Detection
* Clustering of Lane Markings

Experimental Results:

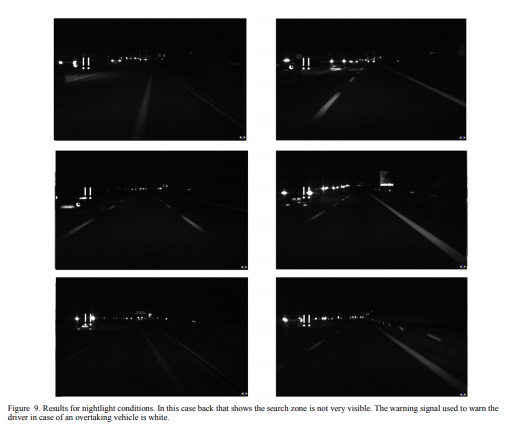
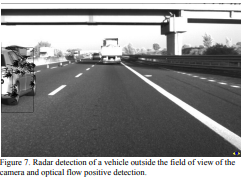
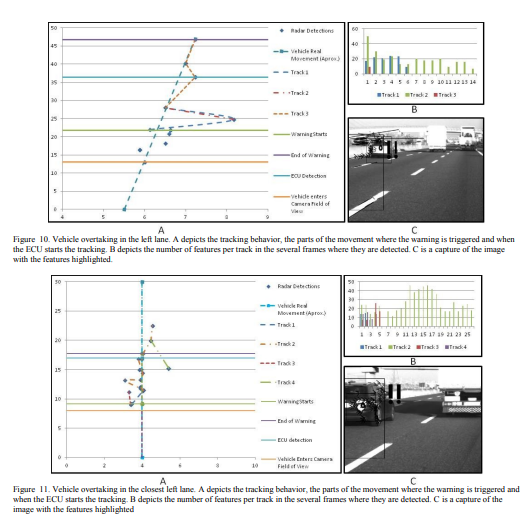
[**Robust Lane Detection and Tracking for Real-Time Applications (IEEE):**](https://ieeexplore.ieee.org/abstract/document/8303759)

An effective lane-detection algorithm is a fundamental component of an advanced driver assistant system, as it provides important information that supports driving safety. The challenges faced by the lane detection and tracking algorithm include the lack of clarity of lane markings, poor visibility due to bad weather, illumination and light reflection, shadows, and dense road-based instructions. In this paper, a robust and real-time vision-based lane detection algorithm with an efficient region of interest is proposed to reduce the high noise level and the calculation time.

Experimental Results:

* **Optical Flow:**

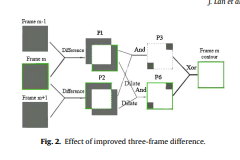
[**Data fusion for overtaking vehicle detection based on radar and optical flow (IEEE):**](https://ieeexplore.ieee.org/abstract/document/6232199)

In this paper an approach to a real application is presented, able to fulfill the requirements of such demanding applications. Most of the commercial sensors available nowadays are usually designed to detect front vehicles but cannot detect overtaking vehicles. The work presented here combines the information provided by two sensors, a Stop&Go radar, and a camera. Fusion is done by using the unprocessed information from the radar, and computer vision-based on optical flow. The basic capabilities of the commercial systems are upgraded giving the possibility to improve the front vehicles detection system, by detecting overtaking vehicles with a high positive rate.

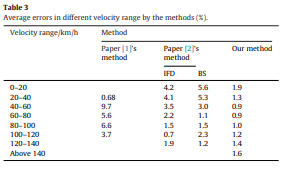
[**The use of optical flow for road navigation (IEEE):**](https://ieeexplore.ieee.org/abstract/document/660838)

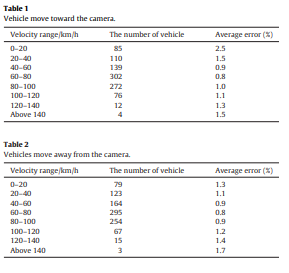
This paper describes procedures for obtaining a reliable and dense optical flow from image sequences taken by a TV camera mounted on a car moving in usual outdoor scenarios. By using correlation-based techniques and by correcting the optical flows for shocks and vibrations, useful sequences of optical flows can be obtained. When the car is moving along a flat road and the optical axis of the TV camera is parallel to the ground, the motion field is expected to be almost quadratic and have a specific structure.

[**Vehicle speed measurement based on gray constraint optical flow algorithm (ScienceDirect):**](https://www.sciencedirect.com/science/article/abs/pii/S0030402613009005)

****This paper presents a novel vehicle speed measurement method, which contains the improved three-frame difference algorithm and the proposed gray constraint optical flow algorithm. By the improved three-frame difference algorithm, the contour of moving vehicles can be detected exactly. Through the proposed gray constraint optical flow algorithm, the vehicle contour's optical flow value, which is the speed (pixels/s) of the vehicle in the image, can be computed accurately. Then, the velocity (km/h) of the vehicles is calculated by the optical flow value of the vehicle's contour and the corresponding ratio of the image pixels to the width of the road. The method can yield a better optical flow field by reducing the influence of changing lighting and shadow.

|  |  |
| --- | --- |
| Basic Optical Flow Equation: | Gray Constraint Equation: |
|  |  |

****Experimental Results:

****

* **Face Detection:**

[**Face Detection: A Survey (ScienceDirect):**](https://www.sciencedirect.com/science/article/abs/pii/S107731420190921X)

In this paper, they present a comprehensive and critical survey of face detection algorithms. Face detection is a necessary first step in face recognition systems, to localize and extract the face region from the background. It also has several applications in areas such as content-based image retrieval, video coding, video conferencing, crowd surveillance, and intelligent human-computer interfaces.

[**Real-time human face detection and tracking (IEEE):**](https://ieeexplore.ieee.org/abstract/document/6777046)

This paper describes the technique for real-time human face detection and tracking using a modified version of the algorithm suggested by Paul Viola and Michael Jones. The paper starts with the introduction to human face detection and tracking, followed by the apprehension of the Viola-Jones algorithm and then discussing the implementation in real video applications. Viola jones's algorithm was based on object detection by extracting some specific features from the image.

[**Face detection in color images(IEEE):**](https://ieeexplore.ieee.org/abstract/document/1000242)

Human face detection plays an important role in applications such as video surveillance, human-computer interface, face recognition, and face image database management. They propose a face detection algorithm for color images in the presence of varying lighting conditions as well as complex backgrounds.

* **Action Recognition:**

[**3D Convolutional Neural Networks for Human Action Recognition (IEEE):**](https://ieeexplore.ieee.org/abstract/document/6165309)

In this paper, they develop a novel 3D CNN model for action recognition. This model extracts features from both the spatial and the temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. The developed model generates multiple channels of information from the input frames, and the final feature representation combines information from all channels.

[**Pose Based Action Recognition of Vulnerable Road Users Using Recurrent Neural Networks (IEEE):**](https://ieeexplore.ieee.org/document/9308462)

Their proposed approach can classify basic movements after different and especially short observation periods. The classification will then be successively improved in case of a longer observation. This allows countermeasures, such as emergency braking, to be initiated early if necessary. The benefits of using 3D poses are evaluated by comparison with a method based solely on the head trajectory. We also investigate the effects of different observation periods.

[**Multi-Task Deep Learning for Pedestrian Detection, Action Recognition and Time to Cross Prediction (IEEE):**](https://ieeexplore.ieee.org/document/8854084)

They propose 1) a pedestrian detection and action recognition component-based, on RetinaNet; 2) an estimation of the time to cross the street for multiple pedestrians using a recurrent neural network. For each pedestrian, the recurrent network estimates the pedestrian's action intention to predict the time to cross the street. Based on their experiments on the JAAD dataset, and show that integrating multiple pedestrian action tags for the detection part when a merge with a recurrent neural network (LSTM) allows a significant performance improvement.

[**Survey of pedestrian action recognition techniques for autonomous driving (IEEE):**](https://ieeexplore.ieee.org/abstract/document/8954864)

In this survey, they present a detailed description of the architecture for pedestrian action recognition in autonomous driving and compare the existing mainstream pedestrian action recognition techniques. They also introduce several commonly used datasets used in pedestrian motion recognition. Finally, they present several suggestions for future research directions.

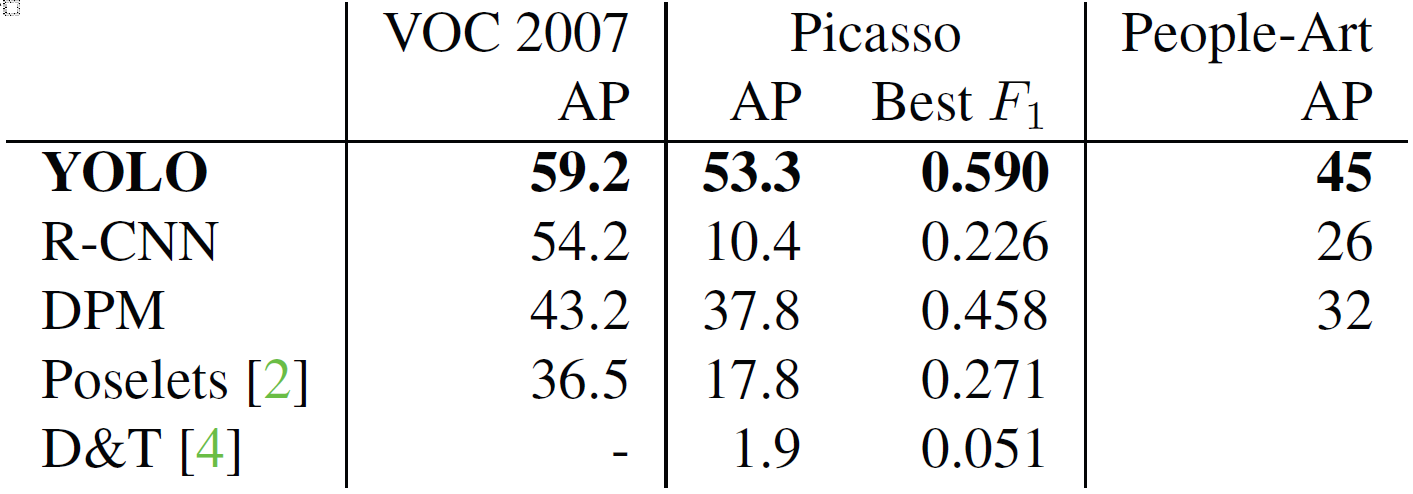
* **Object Recognition:**

[**Practical object recognition in autonomous driving and beyond (IEEE):**](https://ieeexplore.ieee.org/abstract/document/6301978)

This paper is meant as an overview of the recent object recognition work done on Stanford's autonomous vehicle and the primary challenges along this particular path. The eventual goal is to provide practical object recognition systems that will enable new robotic applications such as autonomous taxis that recognize hailing pedestrians, personal robots that can learn about specific objects in your home, and automated farming equipment that is trained on-site to recognize the plants and materials that it must interact with.

[**You Only Look Once: Unified, Real-Time Object Detection (IEEE):**](https://ieeexplore.ieee.org/document/7780460)

YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, they frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Their unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives in the background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

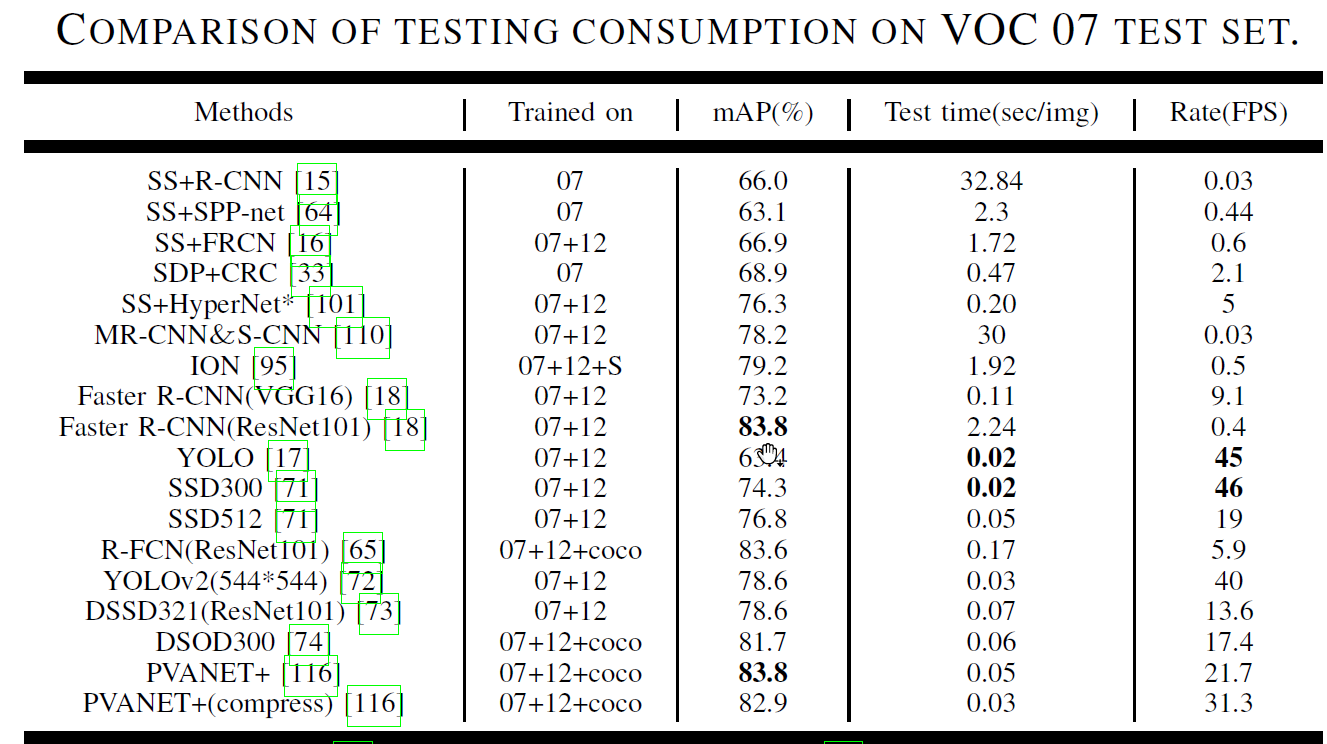


[**Autonomous Driving Technology through Image Classification and Object Recognition based on CNN (ACM):**](https://dl.acm.org/doi/abs/10.1145/3448823.3448841)

Using CNN-based YOLOv3 implements object recognition and image classification, which are the core technologies of autonomous driving technology and Based on the data, it is implemented through the AI autonomous vehicle kit, so that it is possible to make suggestions or ideas for the development of Autonomous driving technology.

[**Object Detection With Deep Learning: A Review (IEEE):**](https://ieeexplore.ieee.org/document/8627998)

Their review begins with a brief introduction to the history of deep learning and its representative tool, namely, the convolutional neural network. Then, we focus on typical generic object detection architectures along with some modifications and useful tricks to improve detection performance further. As distinct specific detection tasks exhibit different characteristics, we also briefly survey several specific tasks, including salient object detection, face detection, and pedestrian detection. Experimental analyses are also provided to compare various methods and draw some meaningful conclusions. Finally, several promising directions and tasks are provided to serve as guidelines for future work in both object detection and relevant neural network-based learning systems.



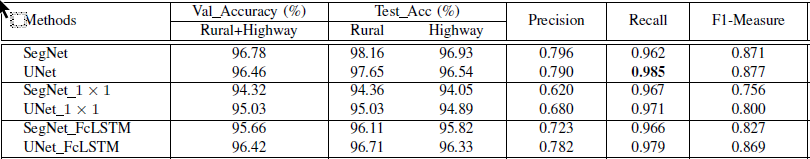
* **Deep Learning Architecture:**

[**Deep learning: Architectures, algorithms, applications (IEEE):**](https://ieeexplore.ieee.org/document/7477319)

This article consists of a collection of slides from the author's conference presentation. Some of the topics covered include Machine learning 101: Neural nets, backdrop, RNNs; Applications; Structured prediction; Unsupervised learning; "Neural Programs"; Architecture exploration; Towards hardware-friendlier DL; and Software.

[**Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks (IEEE):**](https://ieeexplore.ieee.org/document/8883072)

They investigate lane detection by using multiple frames of a continuous driving scene and propose a hybrid deep architecture by combining the convolutional neural network (CNN) and the recurrent neural network (RNN). Specifically, information of each frame is abstracted by a CNN block, and the CNN features of multiple continuous frames, holding the property of time-series, are then fed into the RNN block for feature learning and lane prediction. Extensive experiments on two large-scale datasets demonstrate that the proposed method outperforms the competing methods in lane detection, especially in handling difficult situations.

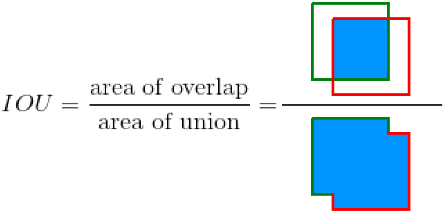
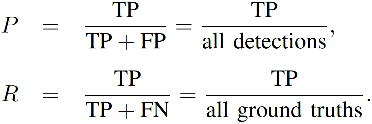
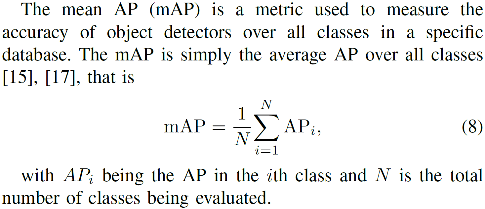


[**L-UNet: An LSTM Network for Remote Sensing Image Change Detection (IEEE):**](https://ieeexplore.ieee.org/document/9301184)

Since ConvLSTM shares, similar spatial characteristics with the convolutional layer, L-UNet, which substitutes partial convolution layers of UNet-to-Conv-LSTM and Atrous L-UNet (AL-UNet), which further using Atrous structure to multiscale spatial information is proposed. Experiments on two data sets are conducted and the proposed methods show the advantages both in quantity and quality when compared with some other methods.

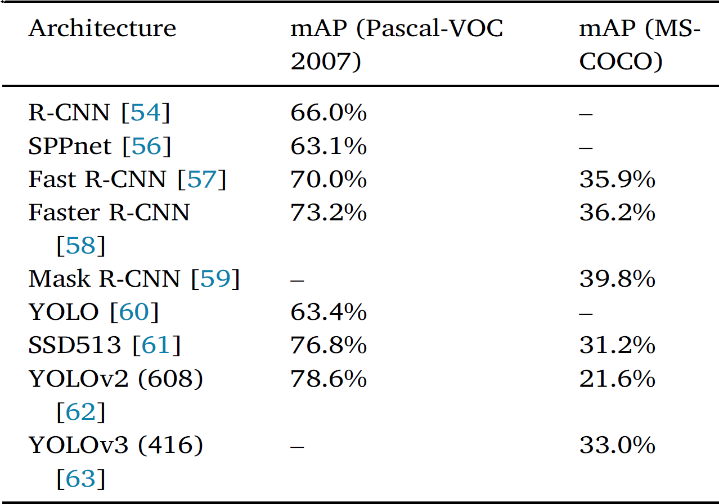
* **Performance Parameters:**

[**A Survey on Performance Metrics for Object-Detection Algorithms (IEEE):**](https://ieeexplore.ieee.org/document/9145130)

This work explores and compares the plethora of metrics for the performance evaluation of object-detection algorithms. Average precision (AP), for instance, is a popular metric for evaluating the accuracy of object detectors by estimating the area under the curve (AUC) of the precision × recall relationship. Depending on the point interpolation used in the plot, two different AP variants can be defined and, therefore, different results are generated. AP has six additional variants increasing the possibilities of benchmarking.

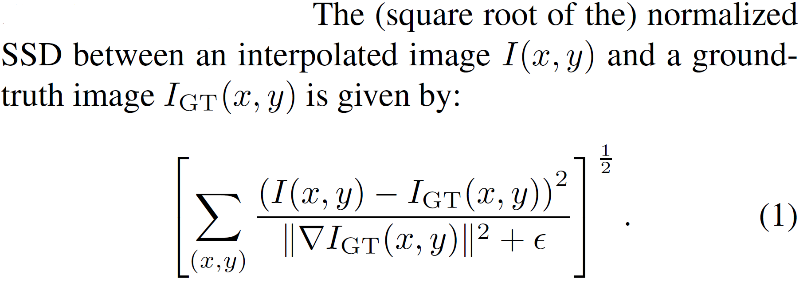
[**Computer vision algorithms and hardware implementations: A survey (ScienceDirect):**](https://www.sciencedirect.com/science/article/pii/S0167926019301762)

This paper aims at providing a comprehensive survey of the recent progress on computer vision algorithms and their corresponding hardware implementations. The prominent achievements in computer vision tasks such as image classification, object detection, and image segmentation brought by deep learning techniques are highlighted. On the other hand, a review of techniques for implementing and optimizing deep-learning-based computer vision algorithms on GPU, FPGA, and other new generations of hardware accelerators are presented to facilitate real-time and/or energy-efficient operations. Finally, several promising directions for future research are presented to motivate further development in the field.

Experimental Results:

[**A Database and Evaluation Methodology for Optical Flow (IEEE):**](https://ieeexplore.ieee.org/document/4408903)

The most commonly used measure of performance for optical flow is an angular error (AE). The AE between two flows (u0, v0) and (u1, v1) is the angle in 3D space between (u0, v0, 1.0) and (u1, v1, 1.0). The AE is usually computed by normalizing the vectors, taking the dot product, and then and then taking the inverse cosine of their dot product. Although the AE is prevalent, it is unclear why errors in a region of smooth non-zero motion should be penalized less than errors in regions of zero motion. Hence, we also compute an absolute error, the error in flow endpoint (EP) defined by

****For image interpolation, they use the SSD between the ground-truth image and the estimated interpolated image. They also include a gradient-normalized SSD. The (square root of the) normalized SSD between an interpolated image I(x, y) and a ground truth image.

* **Other related Papers:**

[**Computer vision-based multiple-lane detection on a straight road and in a curve (IEEE):**](https://ieeexplore.ieee.org/document/5476151)

For straight road, the central lane was detected in the original image using Hough transformation and a simplified perspective transformation was designed to make estimations. In the case of the curved path, a complete perspective transformation was performed and the central lane was detected by scanning at each row in the top view image. The system was able to detect lane marks that were not distinct or even obstructed by other vehicles.

[**Driving Lane Detection Based on Recognition of Road Boundary Situation (IEEE):**](https://ieeexplore.ieee.org/document/8615784)

This paper presents the method that recognizes the road boundary situation from a single image and detects a driving lane based on the recognition result. Driving lane detection is important for lateral motion control of the vehicle and it is usually realized based on lane mark detection. However, there are some roads where lane marks such as white lines are not drawn. Also, when the road is covered with snow, lane marks cannot be seen. In these cases, it's necessary to detect the boundary line between the roadside object and the road surfaces. Since traffic lanes are divided by various roadside objects, such as curbs, grass, walls, and so on, it's difficult to detect all kinds of road boundary including lane marks by a single algorithm.

**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. [CULane dataset](https://xingangpan.github.io/projects/CULane.html):

https://drive.google.com/drive/folders/1mSLgwVTiaUMAb4AVOWwlCD5JcWdrwpvu

1. TUsimple dataset: https://github.com/TuSimple/tusimple-benchmark/issues/3
2. [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)
3. [Multi-Lane Detection and Tracking Using Vision for Traffic Situation Awareness](https://ieeexplore.ieee.org/document/9253415)
4. [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.)
5. https://tryolabs.com/resources/introductory-guide-computer-vision