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

Course Code	**19AI621**
Course Name	**Computer Vision**
Course Instructor	**Dr. Senthilkumar T**

Team Members	**Roll Number**	**Email**	**Contributions**
Abhishek Gopinath	CB.EN.P2AID20002	cb.en.p2aid20002@cb.students.amrita.edu	Dataset, Analytical/Analysis Questions, Block Diagram, Frequency Domain Filters, SURF, Deep Learning ConvLSTM, Literature Survey
Alan Henry	CB.EN.P2AID20010	cb.en.p2aid20010@cb.students.amrita.edu	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	cb.en.p2aid20024@cb.students.amrita.edu	Dataset, Analytical/Analysis Questions, Spatial Domain Filters, SIFT, Horn Schunck, Dense Optical Flow, Action Recognition, YOLOv3, Literature Survey

1. Abstract:

Lane detection is a multi-feature detection problem that is used as a part of an intelligent vehicle system. It helps as driver assistance, lane departure warning, and lane-keeping system, and also for self-driving cars. There is a high need for sophistication and accuracy is expected from lane detection systems to prevent road accidents and achieve high automation. Sensing for certain lane locations is still a hard task for the present solution. The approach used was canny edge detection along with Hough transform and CNN. TVT dataset contains 19383 image sequences for lane detection, and 39460 frames of them are labeled. These images are divided into two parts, a training dataset contains 9548 labeled images and is augmented by four times, and a test dataset is 1268 labeled images. The size of images in this dataset is 128*256. The CNN used is the UNet- ConvLSTM model which is a combination of encoder-decoder, convolutional, and LSTM models. The metrics used for the evaluation of the models are accuracy, F1-score, recall, and precision. The estimated value for accuracy is 98. The proposed work is novel as it can detect lanes in a single road with higher accuracy.

Keywords: Convolutional Neural Network, Canny Edge Detection, Long Short-Term Memory, Lane detection, Hough transform

2. 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.

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

a. 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.

b. 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. 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 objects, present in the data	Image or Video
Ridges	To identify the shape and outline the objects	Image or Video

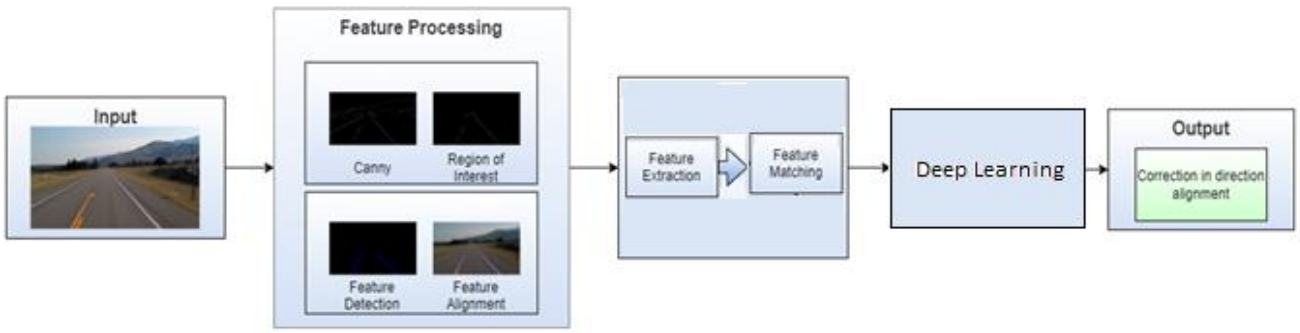
5. Introduction:

With the rapid development of high-precision optics sensors and electronic sensors, high-efficient, and high-effective computer vision, and machine learning algorithms, real-time driving scene understanding has become more and more realistic to us. Many companies have invested a batch of resources to develop advanced algorithms for driving scene understanding, targeting either an autonomous vehicle or a sophisticated driver assistance system (ADAS). Among various research topics of driving scene understanding, lane detection could also be a most simple one. Once lane positions are obtained, the vehicle will know where to travel and avoid the danger of running into other lanes.

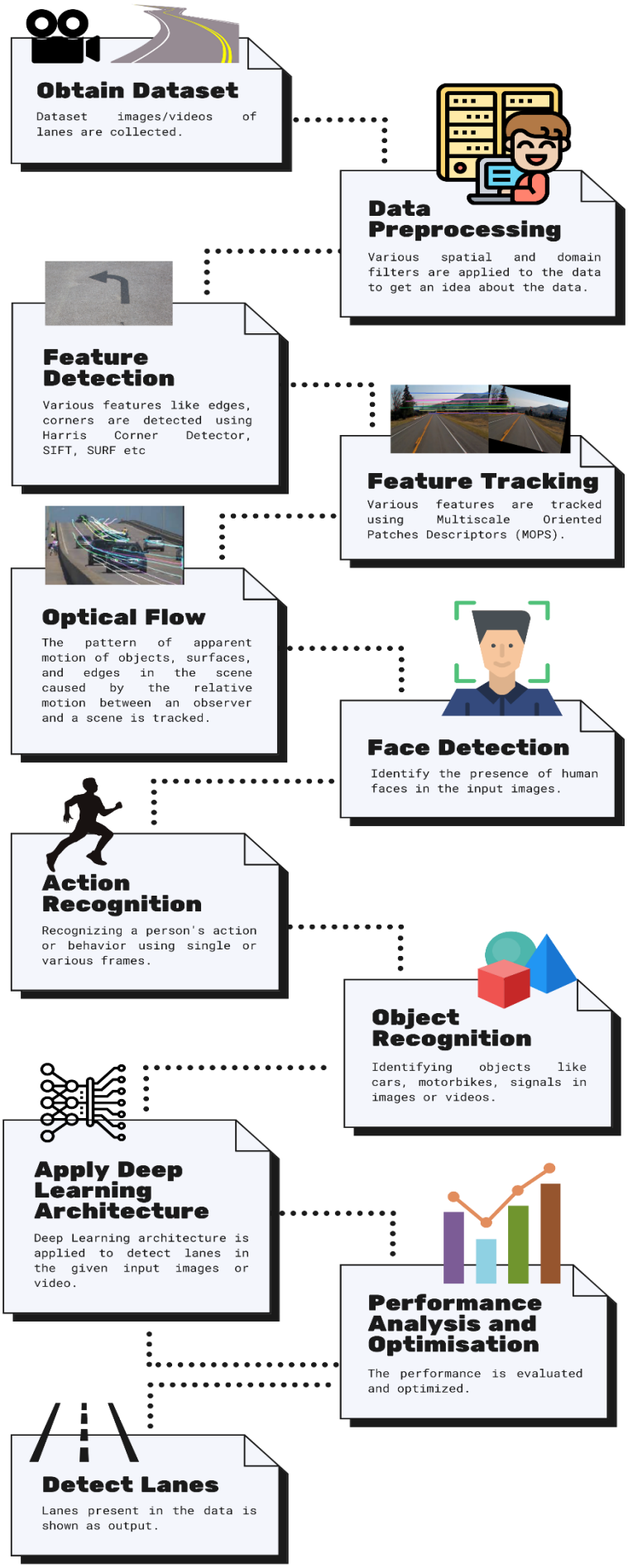
Several lane-detection methods have been proposed with sophisticated performance. However, most of these methods limit their solutions by detecting road lanes in one current frame of the driving scene, which would lead to low performance in handling challenging driving scenarios such as heavy shadows, severe road mark degradation, serious vehicle occlusion, as shown in the top three images. In these situations, the lane may be predicted with false direction, or may be detected in partial, or even cannot be detected at all. One main reason is that, the information provided by the current frame is not enough for accurate lane detection or prediction.

6. Architecture Diagrams:

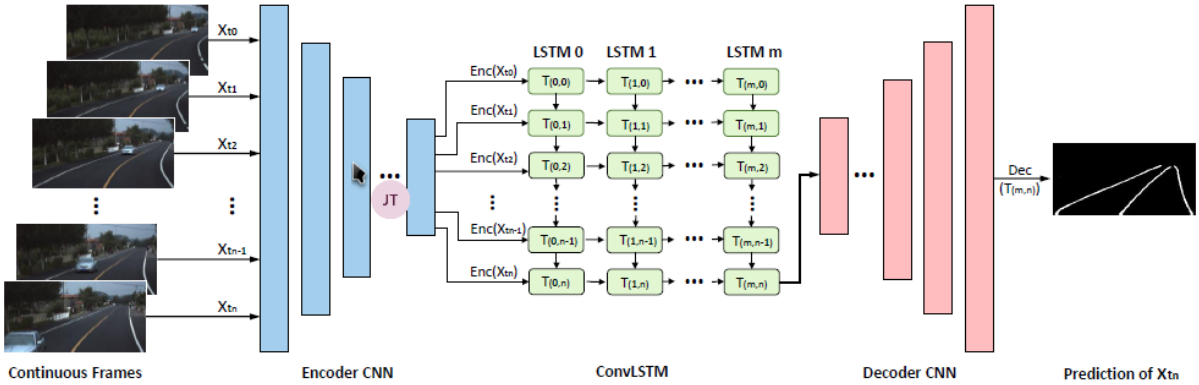
a. Block diagram:



b. Workflow:


Obtain Dataset
Dataset images/videos of lanes are collected.
Data Preprocessing
Various spatial and domain filters are applied to the data to get an idea about the data.
Feature Detection
Various features like edges, corners are detected using Harris Corner Detector, SIFT, SURF etc
Feature Tracking
Various features are tracked using Multiscale Oriented Patches Descriptors (MOPS).
Optical Flow
The pattern of apparent motion of objects, surfaces, and edges in the scene caused by the relative motion between an observer and a scene is tracked.
Face Detection
Identify the presence of human faces in the input images.
Action Recognition
Recognizing a person's action or behavior using single or various frames.
Object Recognition
Identifying objects like cars, motorbikes, signals in images or videos.
Apply Deep Learning Architecture
Deep Learning architecture is applied to detect lanes in the given input images or video.
Performance Analysis and Optimisation
The performance is evaluated and optimized.
Detect Lanes
Lanes present in the data is shown as output.

c. Architecture Diagram with Deep learning:



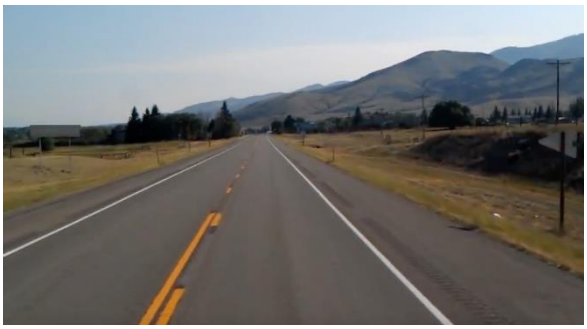
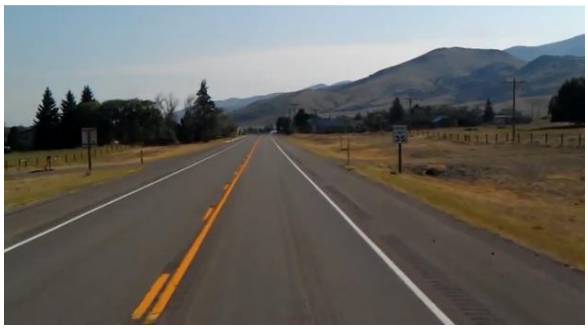
In our network, the input and output size of the ConvLSTM are equal to the size of the feature map produced by the encoder, namely, 8 x 16 for the UNet-ConvLSTM and 4 x 8 for the SegNet-ConvLSTM. The size of the convolutional kernel is 3 x 3. The ConvLSTM is equipped with 2 hidden layers, and each hidden layer has a dimension of 512.

7. 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	128 x 256.
Resolution	NA
Colour Model	RGB
Format	JPG, PNG, MP4
Data Acquisition	Web, Camera
No of classes	3
No of labeled frames	39460
No of Images	19383
URL	https://drive.google.com/drive/folders/1MI5gMDspzuV44lfwzpK6PX0vKuOHUbb_ https://www.youtube.com/watch?v=8XCz-FQgb40

Sample Images:




8. Image Preprocessing in Spatial Domain:

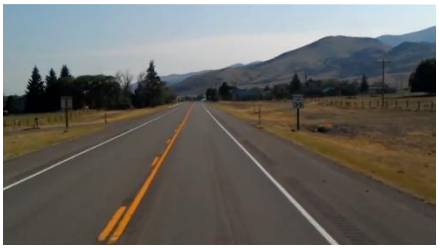
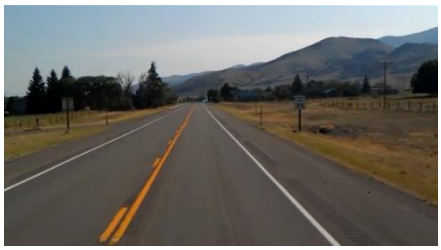
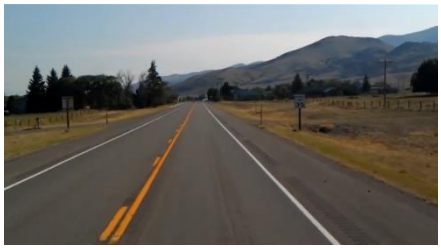
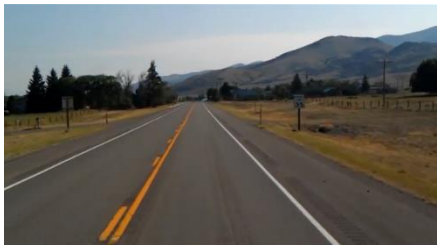
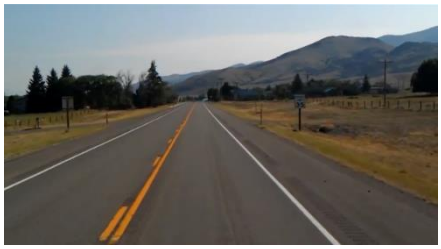
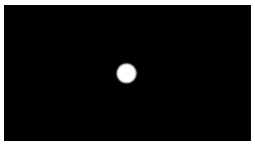
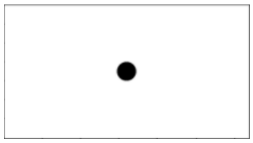
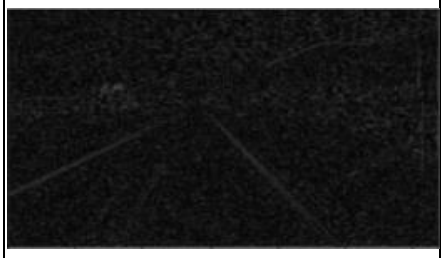
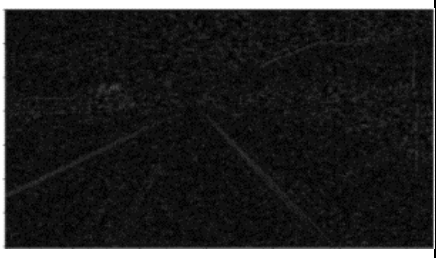
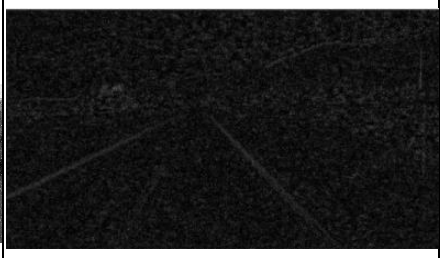
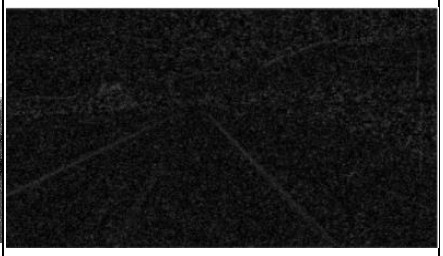
Domain	Filter	Input Image	Output Image	Principle
Spatial Domain	**Image Smoothing** ***Average Filter***			Reduce the amount of intensity variation between neighboring pixels. The average filter works by moving through the image pixel by pixel, replacing each value with the average value of neighboring 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.

9. Image Preprocessing in Frequency Domain:

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.

a. 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.

10. Performance Evaluation Metrics in Image Processing:

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.	$PSNR = 10\ log_{10}(\frac{R^2}{MSE})$ Where R^2 is the maximum possible pixel value of the image; MSE 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.	$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ Where MSE is the Mean Squared Error; n is the number of data points; y_i are the observed values; $\hat{y}_i$ 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.	$SSIM\ (x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x{}^2 + \mu_y{}^2 + c_1)(\sigma_x{}^2 + \sigma_y{}^2 + c_2)}$ Where $SSIM$ is the Structural Similarity Index; μ_x is the average of x; μ_y is the average of y; $\sigma_x{}^2$ is the variance of x; $\sigma_y{}^2$ is the variance of y; σ_{xy} is the covariance of x and y; c_1 and c_2 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.	$IoU = \frac{true\ positive}{true\ positive + false\ positive + false\ negative}$ Ideally, **IoU** > 0.5 is considered a good prediction.
Accuracy		It is the ratio of correctly predicted observation to the total observations	$Accuracy = \frac{true\ positive + true\ negative}{true\ positive + false\ positive + false\ negative + true\ negative}$ 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.	$Precision = \frac{true\ positive}{true\ positive + false\ positive}$ 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	$Recall = \frac{true\ positive}{true\ positive + false\ negative}$ An ideal value for it is closer to 100.
F1-Score		The weighted average of Precision and Recall is the F1 Score.	$F1\ Score = \frac{2 * (Recall * Precision)}{Recall + Precision}$ An ideal value for it is closer to 100.

11. Literature Survey:

a. Datasets relevant to application:

i. TuSimple Dataset:

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

Methodologies Used:

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

Dataset	# Frame	Train	Validation	Test	Resolution	Road type	# Lane
TuSimple	6408	3268	358	2782	1280×720	highway	<=5
CULane	133235	88880	9675	34680	1640×590	urban, rural and highway	<=4

Table 1. Basic information of two lane marker detection datasets.

Method	Accuracy(%)	FP	FN
SCNN[12]	96.53	0.0617	**0.0180**
LaneNet(+H-Net)[10]	96.40	0.0780	0.0244
EL-GAN[4]	96.39	0.0412	0.0336
PointLaneNet[2]	96.34	0.0467	0.0518
FastDraw[14]	95.2	0.0760	0.0450
ENet-SAD[5]	96.64	0.0602	0.0205
ERFNet-E2E[20]	96.02	0.0321	0.0428
PINet(4H)[6]	96.75	**0.0310**	0.0250
FOLOLane(ours)	**96.92**	0.0447	0.0228

Table 3. Performance of different methods on TuSimple testing set.

ii. CULane Dataset:

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

Experimental Results:

Table 3: Experimental results on spatial CNN at different positions, with $w = 9$.

Position	Output	Top hidden layer
F1 (0.3)	79.9	80.9
F1 (0.5)	68.8	71.6

Table 5: Comparison with other methods, with IoU threshold=0.5. For crossroad, only FP are shown.

Category	Baseline	ReNet	DenseCRF	MRFNet	ResNet-50	ResNet-101	Baseline+SCNN
Normal	83.1	83.3	81.3	86.3	87.4	90.2	**90.6**
Crowded	61.0	60.5	58.8	65.2	64.1	68.2	**69.7**
Night	56.9	56.3	54.2	61.3	60.6	65.9	**66.1**
No line	34.0	34.5	31.9	37.2	38.1	41.7	**43.4**
Shadow	54.7	55.0	56.3	59.3	60.7	64.6	**66.9**
Arrow	74.0	74.1	71.2	76.9	79.0	84.0	**84.1**
Dazzle light	49.9	48.2	46.2	53.7	54.1	**59.8**	58.5
Curve	61.0	59.9	57.8	62.3	59.8	**65.5**	64.4
Crossroad	2060	2296	2253	**1837**	2505	2183	1990
Total	63.2	62.9	61.0	67.0	66.7	70.8	**71.6**

iii. tvtLANE Dataset:

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.

iv. BDD100K:

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

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:

Training Set	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	mean IoU
Sem-Seg	94.3	63.0	84.9	**25.7**	**45.8**	**52.6**	56.2	54.1	**86.4**	45.1	95.3	62.4	22.1	90.2	50.5	68.3	0	35.5	**49.9**	56.9
Sem-Seg + Det	94.3	62.5	**85.2**	24.5	41.1	51.5	**63.1**	**57.9**	86.2	**47.4**	**95.5**	**64.6**	**28.1**	**90.8**	**52.9**	**70.7**	0	**43.4**	48.9	**58.3**
Sem-Seg + Lane + Driv	**94.8**	**65.8**	84.1	22.6	40.2	49.3	51.9	49.7	85.8	46.2	95.3	60.8	7.1	89.9	47.8	66.9	0	27.5	27.5	53.3

Table 8: Evaluation results for semantic segmentation. We explore segmentation joint-training with different tasks. Detection can improve the overall accuracy of segmentation, although their output structures are different. However, although Lane and Drivable area improve the segmentation of road and sidewalk, the overall accuracy drops.

b. Feature Extraction:

i. Robust lane detection & tracking based on novel feature extraction and lane categorization(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

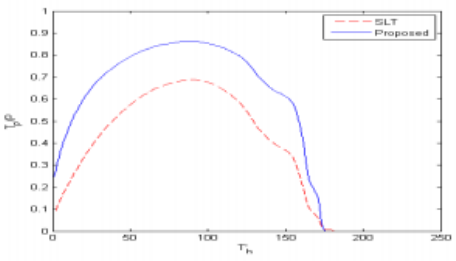


Fig. 2: T_p/P ratio for range of thresholds

Experimental Results:

Table 1: Detection results

Sequence	Total frames	Correct detection	Incorrect detection	Mis-Detectin
Highway	5677	5645	30	3
shadow	2090	2029	0	61
Night	2920	2606	314	0

ii. Lane detection algorithm based on local feature extraction(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

$$\hat{I}_{ROI}(x,y) = I_{ROI}(x,y) \otimes \hat{f}(x,y), \qquad \hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}}\{I_{ROI}(s,t)\}. \qquad I_{BW} = \begin{cases} 255 & I_{ROI} - \hat{I}_{ROI} \geq T \\ 0 & I_{ROI} - \hat{I}_{ROI} < T \end{cases}.$$

$$I_{BW} = \begin{cases} 255 & I_{ROI} - \hat{I}_{ROI} \geq T \\ 0 & I_{ROI} - \hat{I}_{ROI} < T \end{cases}. \qquad T = \sum_{k=0}^{L-1} u_k \times p(u_k),$$

- Interference Removed by Morphological Operation $I_{BW} \oplus B = \{z \mid (B)_z \cap I_{BW} \neq \varnothing\}.$
- Edge Refining $f_u(x) = \exp\left(-x^2/2\sigma_x^2\right),$
- Lane Detection using Hough transform

Experimental Results:

TABLE I. THE COMPARISON RESULT OF LANE DETECTION ALGORITHMS

Illumination type	Method	Correct rate	False rate	Missing rate
Night scene	Our method	94.06%	3.98%	1.96%
	Borkar's method[14]	92.09%	6.98%	0.93%
Rainy scene	Our method	98.67%	1.33%	0%
	Borkar's method[14]	86.67%	10.67%	2.67%
Fluctuating illumination scene	Our method	87.36%	5.75%	6.89%
	Borkar's method[14]	74.71%	10.34%	14.95%

(a) Night scene.

(b) Rainy scene.

(c) Fluctuating illumination scene.

Fig. 8. Examples of correct detection result.

Fig. 9. Examples of false detection result.

iii. Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting(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.

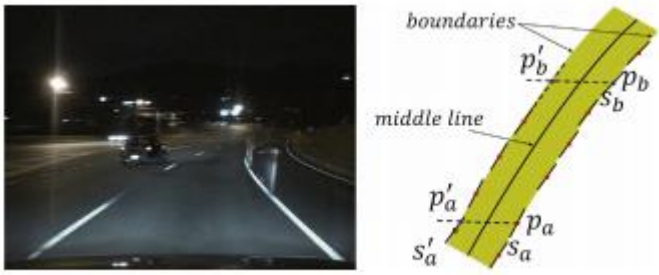


Fig. 1. A road image in night and our model for lane detection.

Dataset Used: Their 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:

Table 1
The performances of different detection methods under our dataset.

Scenario	Number	Wu's method [22]		LDTFE	
		AR (%)	FN (%)	AR (%)	FN (%)
Day-time	750	87.7	6.7	91.5	4.7
Night-time	750	92.2	4.1	95.9	1.9
Rain	750	93.5	3.4	94.8	2.1
Tunnel	750	89.6	3.6	92	2.3
Total	3000	90.8	4.45	93.6	2.8

Table 2
The performances of different detection methods under Caltech dataset.

Clip	Number	Aly's method [28]		LDTFE	
		AR (%)	FP (%)	AR (%)	FN (%)
Cordova1	250	97.2	3.0	92.2	5.4
Cordova2	406	96.2	38.4	97.7	1.8
Washington1	337	96.7	4.7	96.9	2.5
Washington2	232	95.1	2.2	98.5	1.7
Total	1225	96.3	11.6	96.5	2.7

c. Feature Tracking:

i. Effective Lane Detection and Tracking Method Using Statistical Modeling of Color and Lane Edge-Orientation (IEEE):

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

ii. A feature-based tracking algorithm for vehicles in intersections (IEEE):

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

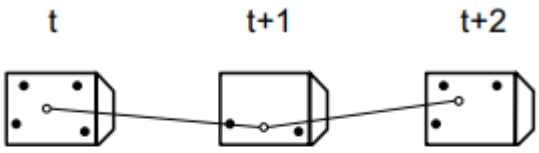


Figure 2: Illustration of the noise problem with methods that group features in each frame independently. The resulting trajectory of the centroid (the white dot) of the feature group (the black dots) shows a wrong vehicle movement.

Experimental Results:

Sequences	Length (frames)
Conflicts	5793
Karlsruhe	1050
Cambridge	1517

Table 1: Video sequences for evaluation, with their length (number of frames).

iii. A novel multi-lane detection and tracking system (IEEE):

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:

TABLE I: Final evaluation results

$\overline{FN}$	0.0249
$\overline{TP}$	0.9751
$\overline{FP}$	0.0116
Mean of the Lane Position Error	3.66px
Standard Deviation of the Lane Position Error	4.60px

iv. Robust Lane Detection and Tracking for Real-Time Applications (IEEE):

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:

TABLE I

COMPARISON OF THE PROPOSED ALGORITHM WITH OTHERS

Algorithms	Son [14]	Ding [12]	Proposed
ROI upper boundary	VP with HT	Fixed position	VP with Clustering and tracking scheme
ROI lower boundary	N/A	Fixed position	Layer information
Color lane detection	(Binary Intensity) OR (Binary Cb)	N/A	Intensity(V) + Cb+Cr → Binarization
Line segment detection	Pixel-based. Connected component clustering and fitting	Edge-based. Canny+HT, inner line selection	Edge-based. EDLines, Line clustering
Lane verification	Slope and Intercept	Slope	Scanline test (req.), Slope/ Lane width/ IP
Lane tracking	N/A	Kalman filter in A-ROI (triangular and fixed boundary)	Kalman filter in A-ROI (distorted trapezoidal and adaptive boundary)

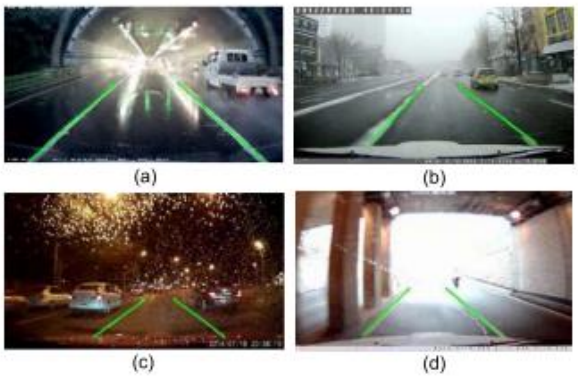


Fig. 6. Lane marking detection results under various driving conditions: (a) rainy day, (b) snowy day, (c) rainy night, (d) exiting tunnel.

d. Optical Flow:

i. Data fusion for overtaking vehicle detection based on radar and optical flow (IEEE):

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.

igure 3. Optical Flow widow highlighted in black.

Figure 7. Radar detection of a vehicle outside the field of view of the camera and optical flow positive detection.

Figure 9. Results for nightlight conditions. In this case back that shows the search zone is not very visible. The warning signal used to warn the driver in case of an overtaking vehicle is white.

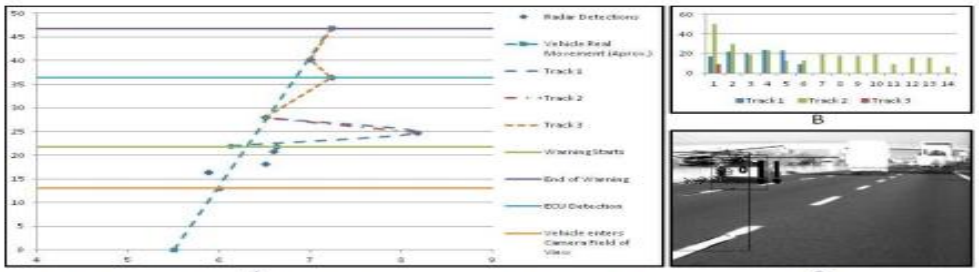


Figure 10. Vehicle overtaking in the left lane. A depicts the tracking behavior, the parts of the movement where the warning is triggered and when the ECU starts the tracking. B depicts the number of features per track in the several frames where they are detected. C is a capture of the image with the features highlighted.

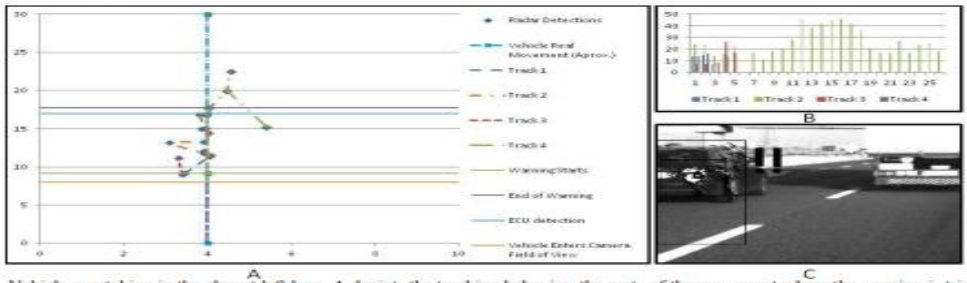


Figure 11. Vehicle overtaking in the closest left lane. A depicts the tracking behavior, the parts of the movement where the warning is triggered and when the ECU starts the tracking. B depicts the number of features per track in the several frames where they are detected. C is a capture of the image with the features highlighted

‘

ii. The use of optical flow for road navigation (IEEE):

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.

Inferences: Egomotion parameters can be computed with simpler procedures when clear landmarks, such as road markers, are present in the scene. The major requirement for the recovery of a reliable optical flow is the presence of enough texture in the images. Well-defined structures and high contrasts are not necessary. As a consequence, the optical flow can be computed in a variety of different scenarios and this versatility appears to be the major advantage of the proposed approach.

iii. Vehicle speed measurement based on gray constraint optical flow algorithm (ScienceDirect):

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.

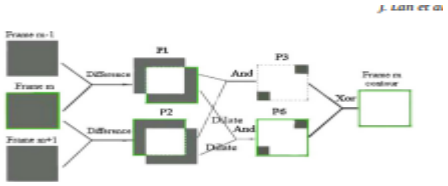


Fig. 2. Effect of improved three-frame difference.

Basic Optical Flow Equation:	Gray Constraint Equation:
$(I_x, I_y)(u, v)^T + I_t = 0$ where $I_x = \partial I/\partial x$, $I_y = \partial I/\partial y$, $I_t = \partial I/\partial t$.	$I(r + \Delta r) = M(r)I(r) + C(r)$ (13) where $I(r) = I(x,y,t)$, $M(r) = 1 + \Delta m(r)$, $\Delta m(r)$ is the deviation coefficient. $C(r)$ is interference.

Experimental Results:

Table 1
Vehicle move toward the camera.

Velocity range/km/h	The number of vehicle	Average error (%)
0–20	85	2.5
20–40	110	1.5
40–60	139	0.9
60–80	302	0.8
80–100	272	1.0
100–120	76	1.1
120–140	12	1.3
Above 140	4	1.5

Table 2
Vehicles move away from the camera.

Velocity range/km/h	The number of vehicle	Average error (%)
0–20	79	1.3
20–40	123	1.1
40–60	164	0.9
60–80	295	0.8
80–100	254	0.9
100–120	67	1.2
120–140	15	1.4
Above 140	3	1.7

Table 3
Average errors in different velocity range by the methods (%).

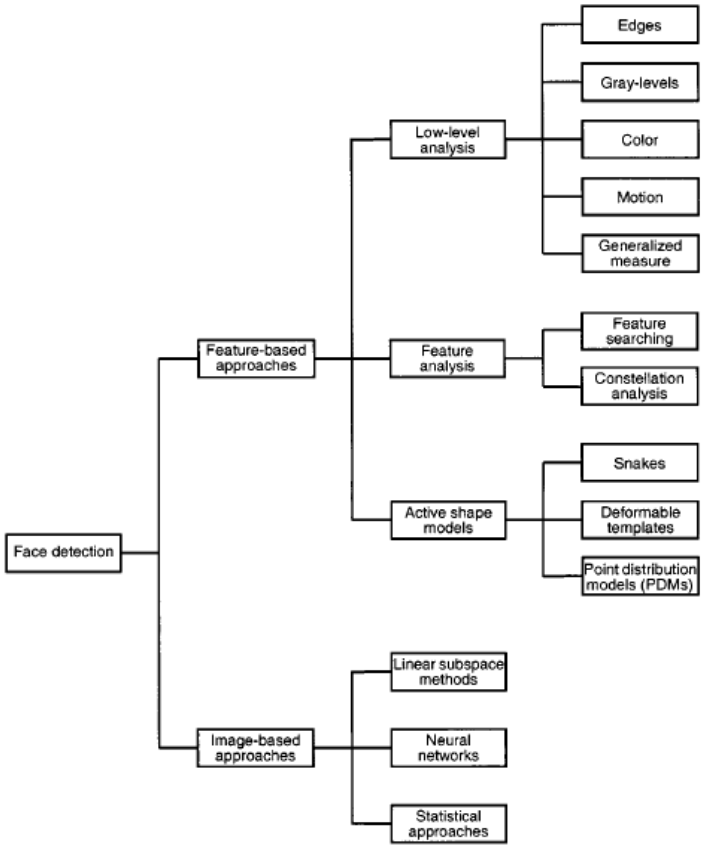
Velocity range/km/h	Method			
	Paper [1]'s method	Paper [2]'s method		Our method
		IFD	BS	
0–20		4.2	5.6	1.9
20–40	0.68	4.1	5.3	1.3
40–60	9.7	3.5	3.0	0.9
60–80	5.6	2.2	1.1	0.9
80–100	6.6	1.5	1.5	1.0
100–120	3.7	0.7	2.3	1.2
120–140		1.9	1.2	1.4
Above 140				1.6

e. Face Detection:

i. Face Detection: A Survey (ScienceDirect):

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.

Approach



Experimental Results:

FIG. 10. Face detection examples from Schneiderman and Kanade [170] (© 2000/2001 IEEE).

ii. Real-time human face detection and tracking (IEEE):

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.

Approach:

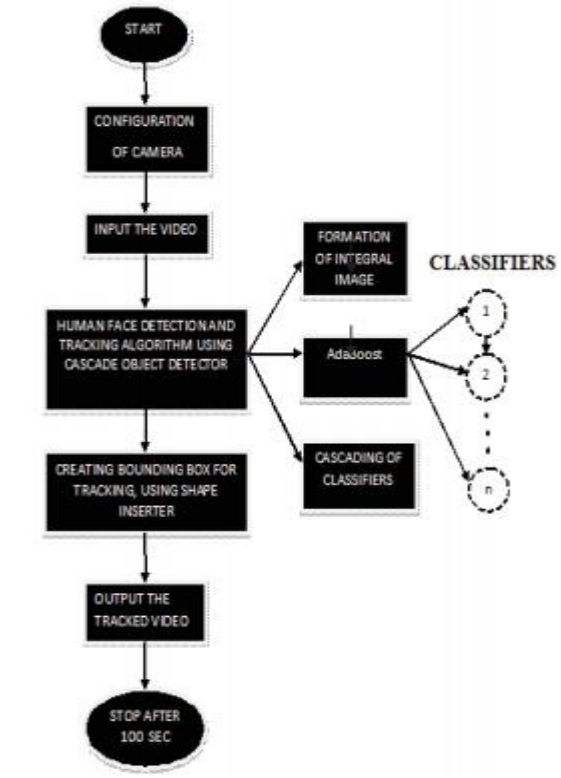


Fig 5: Flowchart for real time detection and tracking algorithm.

Experimental Results:

Figure 5: Real time face detection and tracking of one of the authors of this research paper

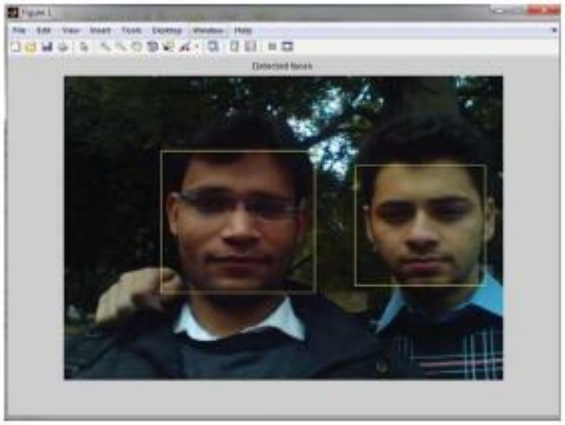


Figure 6: Multiple face detection and tracking.

iii. Face detection in color images(IEEE):

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.

TABLE 1
Major Face Detection Approaches

Authors	Year	Approach	Featurees Used	Head Pose	Test Databases	Minimal Face Size
Féraud et al. [11]	2001	Neural Networks	Motion; Color; Texture	Frontal and profile	Sussex; CMU; Web images	15×20
Maio et al. [24]	2000	Facial templates; Hough Transform	Texture; Directional images	Frontal	Static images	20×27
Garcia et al. [12]	1999	Statistical wavelet analysis	Color; wavelet coefficients	Frontal to near frontal	MPEG videos	80×48
Wu et al. [43]	1999	Fuzzy color models; Template matching	Color	Frontal to profile	Still color images	20×24
Rowley et al. [32], [33]	1998	Neural Networks	Texture	(Upright) frontal	CMU; FERET; Web images	20×20
Sung et al. [39]	1998	Learning	Texture	Frontal	Mug shots; CCD pictures; newspaper scans	19×19
Yang et al. [44]	1998	Multiscale segmentation; color model	Skin Color; intensity	Frontal	color pictures	NA
Colmenarez et al. [8]	1997	Learning	Markov processes	Frontal	FERET	11×11
Yow et al. [46]	1997	Feature; Belief networks	Geometrical facial features	Frontal to profile	CMU	60×60
Lew et al. [23]	1996	Markov Random Field; DFFS [27]	Most informative pixel	Frontal	MIT; CMU; Leiden	23×32

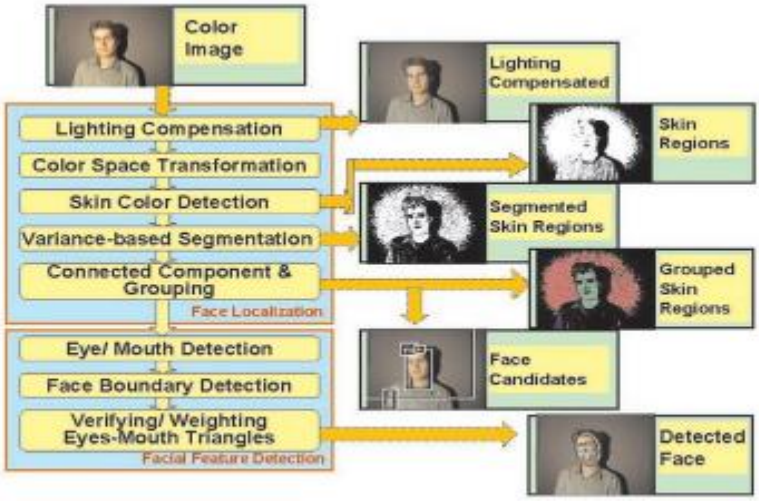


Fig. 1. Face detection algorithm.

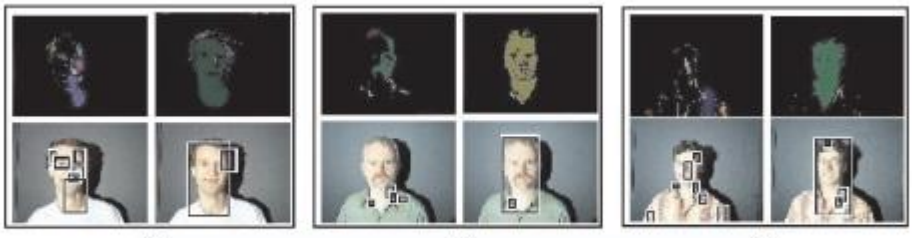


Fig. 6. Detection examples, with and without the transformation. For each of these examples, the images shown in (a) are skin regions and detections without the transformation, while those in (b) are results with the transformation.

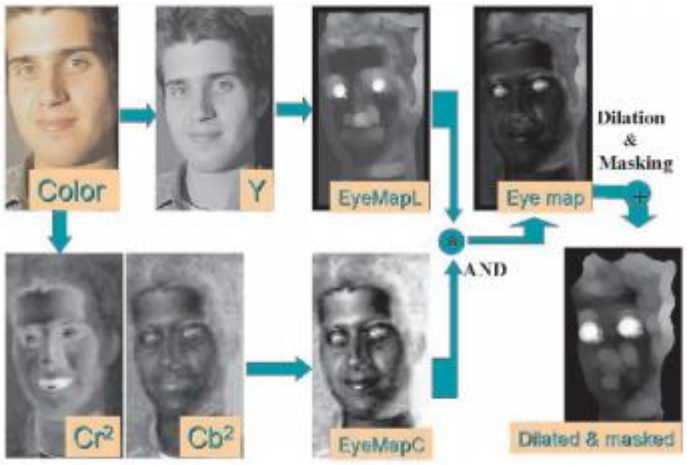


Fig. 8. Construction of eye maps.

Localization of facial features

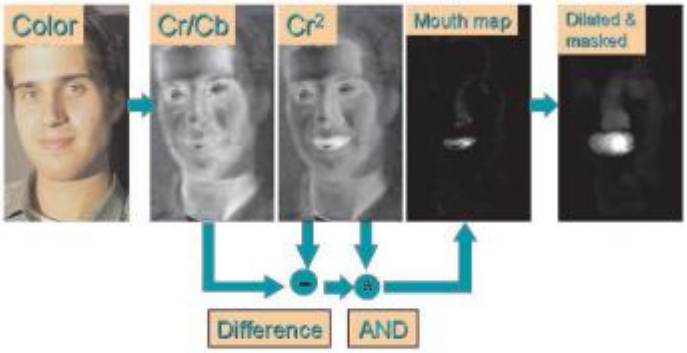
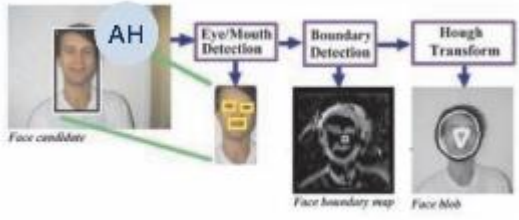


Fig. 9. Construction of the mouth map.

Eye Map

Formulae

$$EyeMapC = \frac{1}{3}\left\{(C_b^2) + (\tilde{C}_r)^2 + (C_b/C_r)\right\}, \qquad EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1},$$



ig. 10. Computation of face boundary.

Mouth Map

Formulae

$$MouthMap = C_r^2 \cdot \left(C_r^2 - \eta \cdot C_r/C_b\right)^2,$$

$$\eta = 0.95 \cdot \frac{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)^2}{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)/C_b(x,y)},$$

Face Boundary and face score

(a) (b)

Fig. 11. Face detection examples containing dark skin-tone faces. Each example contains an original image, grouped skin regions, and a lighting-compensated image overlaid with detected face and facial features.

Head Pose	Frontal	Near-Frontal	Half-Profile	Profile	Total
No. of images	66	54	75	11	206
Stage 1: Grouped skin regions					
No. of FP	3145	2203	3781	277	9406
DR (%)	95.45	98.15	96.00	100	96.60
Time (sec): average ± s. d.	1.56 ± 0.45				
Stage 2: Rectangle merge					
No. of FP	468	287	582	39	1376
DR (%)	95.45	98.15	96.00	100	96.60
Time (sec): average ± s. d.	0.18 ± 0.23				
Stage 3: Facial feature detection					
No. of FP	4	6	14	3	27
DR (%)	89.40	90.74	74.67	18.18	80.58
Time (sec): average ± s. d.	22.97 ± 17.35				

FP: False Positives, DR: Detection Rate.

TABLE 3
Detection Results on the Champion Database (Image Size ~ 150 × 220) on a 1.7GHz CPU

Stage	Grouped skin regions	Rectangle merge	Facial feature detection
No. of images	227		
No. of FP	5582	382	14
DR (%)	99.12	99.12	91.63
Time (sec): average ± s. d.	0.080 ± 0.036	0.012 ± 0.02	5.78 ± 4.98

FP: False Positives, DR: Detection Rate.

f. Action Recognition:

i. 3D Convolutional Neural Networks for Human Action Recognition (IEEE):

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.

Dataset Used: TRECVID Dataset, KTH Dataset

Experimental Results:

TABLE 4
Comparison of the Best Performance Achieved by Our Previous Methods in [26] (ICML Models) with the New Methods Proposed in This Paper (New Models)

	Precision		Recall		AUC	
FPR	0.1%	1%	0.1%	1%	0.1%	1%
New models	0.7824	0.6149	0.0340	0.1433	0.0201	0.8853
ICML models	0.7137	0.5572	0.0230	0.1132	0.0129	0.6752

TABLE 5
Action Recognition Accuracies in Percentage on the KTH Data

Method	Boxing	Handclapping	Handwaving	Jogging	Running	Walking	Average
3D CNN	90	94	97	84	79	97	90.2
Schüldt [13]	97.9	59.7	73.6	60.4	54.9	83.8	71.7
Dollár [14]	93	77	85	57	85	90	81.2
Niebles [56]	98	86	93	53	88	82	83.3
Jhuang [16]	92	98	92	85	87	96	91.7
Schindler [53]	–	–	–	–	–	–	92.7

ii. Pose Based Action Recognition of Vulnerable Road Users Using Recurrent Neural Networks (IEEE):

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.

iii. Multi-Task Deep Learning for Pedestrian Detection, Action Recognition and Time to Cross Prediction (IEEE):

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.

Methodologies Used:

- Train all pedestrian Bounding Boxes (BB) samples with the RetinaNet for pedestrian detection purposes
- Split the pedestrian Joint Attention for Autonomous Driving (JAAD) data set into four classes for pedestrian action functionality
- Train a Long Short-Term Memory (LSTM) model using only BB coordinates to estimate the time to cross of each pedestrian.

Experimental Results:

TABLE 2: The estimation of time to cross methods, independently of the detection-classification component. PPC:Pedestrian is Preparing to Cross the street. Real values: testing, independently the pedestrian time to cross estimation on the all real pedestrian samples; Detected Values: testing the detection component connected with the prediction component (Time to cross).

TABLE 1: The ... to Cross the street, PC= Pedestrian is Crossing the st

Learned on		Tested On	Past Time Steps	RMSE%	
				Real BB	Detected BB
Only PPC BB Coordinates	With Action Tag	All Samples	5	12.17	13.12
			14	9.36	11.72
			40	10.43	10.43
	Without Action Tag	All Samples	5	9.61	11.21
			14	13.38	13.34
			40	11.64	11.57
Only PPC BB Coordinates	with Action Tag	Only PPC BB Coordinates	5	5.87	8.03
			14	5.04	7.30
			40	4.76	4.88
	Without Action Tag	Only PPC BB Coordinates	5	5.75	7.14
			14	5.47	8.44
			40	5.86	8.26
All BB Coordinates	With Action Tag	All BB Coordinates	5	6.22	6.89
			14	5.57	8.71
			40	4.10	6.07
	Without Action Tag	All BB Coordinates	5	6.20	6.86
			14	5.36	6.32
			40	4.01	4.60

mAP
±CI
6.05%
±0.93
6.36%
±0.83

iv. Survey of pedestrian action recognition techniques for autonomous driving (IEEE):

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.

Methodologies used:

Pedestrian detection in autonomous driving:

- Traditional algorithms
  - HOG feature and SVM algorithm
  - DPM algorithm
- Deep learning algorithms
  - Two-stage method
  - One-stage algorithm

Action recognition:

- Traditional method
  - Optical flow characteristics
  - Gradient characteristics
  - Dense Trajectories (DT) algorithm
- Deep learning method based on two-stream network

Datasets used: KTH, the UCF series, Hollywood2, and Google AVA

Experimental Results:

Table 1 Human action recognition datasets.

Dataset name	Year	Number of classes	Number of actors	Number of videos	Attribute
KTH	2004	6	25	2391	Single action
UCF Sports	2008	9	Many	200	Single action
UCF 50[54]	2013	50	Many	6676	Single action
UCF101[55]	2012	101	Many	13 320	Single action
UCF YouTube[56]	2009	11	Many	⩾1100	Continuous action
Hollywood2	2009	12	Many	3669	Continuous action
Google AVA	2017	80	Many	⩾57 600	Continuous action

g. Object Recognition:

i. Practical object recognition in autonomous driving and beyond (IEEE):

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.

ii. You Only Look Once: Unified, Real-Time Object Detection (IEEE):

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.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

iii. Object Detection With Deep Learning: A Review (IEEE):

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.

COMPARISON OF TESTING CONSUMPTION ON VOC 07 TEST SET.

Methods	Trained on	mAP(%)	Test time(sec/img)	Rate(FPS)
SS+R-CNN [15]	07	66.0	32.84	0.03
SS+SPP-net [64]	07	63.1	2.3	0.44
SS+FRCN [16]	07+12	66.9	1.72	0.6
SDP+CRC [33]	07	68.9	0.47	2.1
SS+HyperNet* [101]	07+12	76.3	0.20	5
MR-CNN&S-CNN [110]	07+12	78.2	30	0.03
ION [95]	07+12+S	79.2	1.92	0.5
Faster R-CNN(VGG16) [18]	07+12	73.2	0.11	9.1
Faster R-CNN(ResNet101) [18]	07+12	**83.8**	2.24	0.4
YOLO [17]	07+12	63.4	**0.02**	**45**
SSD300 [71]	07+12	74.3	**0.02**	**46**
SSD512 [71]	07+12	76.8	0.05	19
R-FCN(ResNet101) [65]	07+12+coco	83.6	0.17	5.9
YOLOv2(544*544) [72]	07+12	78.6	0.03	40
DSSD321(ResNet101) [73]	07+12	78.6	0.07	13.6
DSOD300 [74]	07+12+coco	81.7	0.06	17.4
PVANET+ [116]	07+12+coco	**83.8**	0.05	21.7
PVANET+(compress) [116]	07+12+coco	82.9	0.03	31.3

h. Deep Learning Architecture:

i. Deep learning: Architectures, algorithms, applications (IEEE):

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.

ii. Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks (IEEE):

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.

Methods	Val_Accuracy (%)	Test_Acc (%)		Precision	Recall	F1-Measure
	Rural+Highway	Rural	Highway			
SegNet	96.78	98.16	96.93	0.796	0.962	0.871
UNet	96.46	97.65	96.54	0.790	**0.985**	0.877
SegNet_1 × 1	94.32	94.36	94.05	0.620	0.967	0.756
UNet_1 × 1	95.03	95.03	94.89	0.680	0.971	0.800
SegNet_FcLSTM	95.66	96.11	95.82	0.723	0.966	0.827
UNet_FcLSTM	96.42	96.71	96.33	0.782	0.979	0.869

iii. L-UNet: An LSTM Network for Remote Sensing Image Change Detection (IEEE):

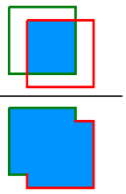
Since ConvLSTM shares, similar spatial characteristics with the convolutional layer, L-UNet, which substitutes partial convolution layers of UNet-to-Conv-LSTM and Atrous L-UNet (AL-UNet), which further using Atrous structure to multiscale spatial information is proposed. Experiments on two data sets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

i. Papers related to Performance Parameters:

i. A Survey on Performance Metrics for Object-Detection Algorithms (IEEE):

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.

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}},$$

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}.$$

The mean AP (mAP) is a metric used to measure the accuracy of object detectors over all classes in a specific database. The mAP is simply the average AP over all classes [15], [17], that is

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i, \quad (8)$$

with AP_i being the AP in the ith class and N is the total number of classes being evaluated.

ii. Computer vision algorithms and hardware implementations: A survey (ScienceDirect):

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:

Architecture	mAP (Pascal-VOC 2007)	mAP (MS-COCO)
R-CNN [54]	66.0%	–
SPPnet [56]	63.1%	–
Fast R-CNN [57]	70.0%	35.9%
Faster R-CNN [58]	73.2%	36.2%
Mask R-CNN [59]	–	39.8%
YOLO [60]	63.4%	–
SSD513 [61]	76.8%	31.2%
YOLOv2 (608) [62]	78.6%	21.6%
YOLOv3 (416) [63]	–	33.0%

iii. A Database and Evaluation Methodology for Optical Flow (IEEE):

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

$$\text{sqrt}[(u_0 - u_1)^2 + (v_0 - v_1)^2]$$

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.

$$\left[\sum_{(x,y)} \frac{(I(x,y) - I_{\text{GT}}(x,y))^2}{\|\nabla I_{\text{GT}}(x,y)\|^2 + \epsilon}\right]^{\frac{1}{2}}$$

j. Other related Papers:

i. Computer vision-based multiple-lane detection on a straight road and in a curve (IEEE):

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.

ii. Driving Lane Detection Based on Recognition of Road Boundary Situation (IEEE):

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.

12. 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

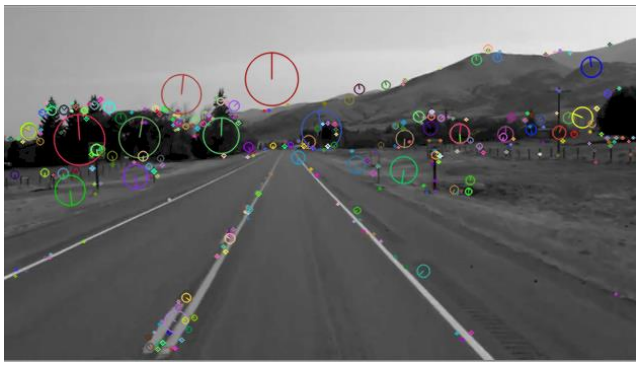
  4. For each pixel p in the grayscale image, consider a 3×3 window around it and compute the corner strength function. Call this its Harris value.

  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):***

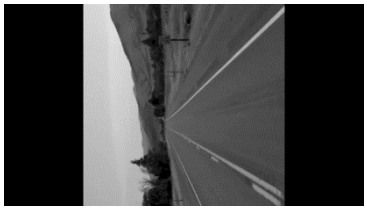
    1. Scale-space Extrema Detection (Feature point (also called key point) detection)
        - Potential location for finding features.
    2. Feature point localization
        - Accurately locating the feature key points.
    3. Orientation assignment
        - Assigning orientation to key points.
 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



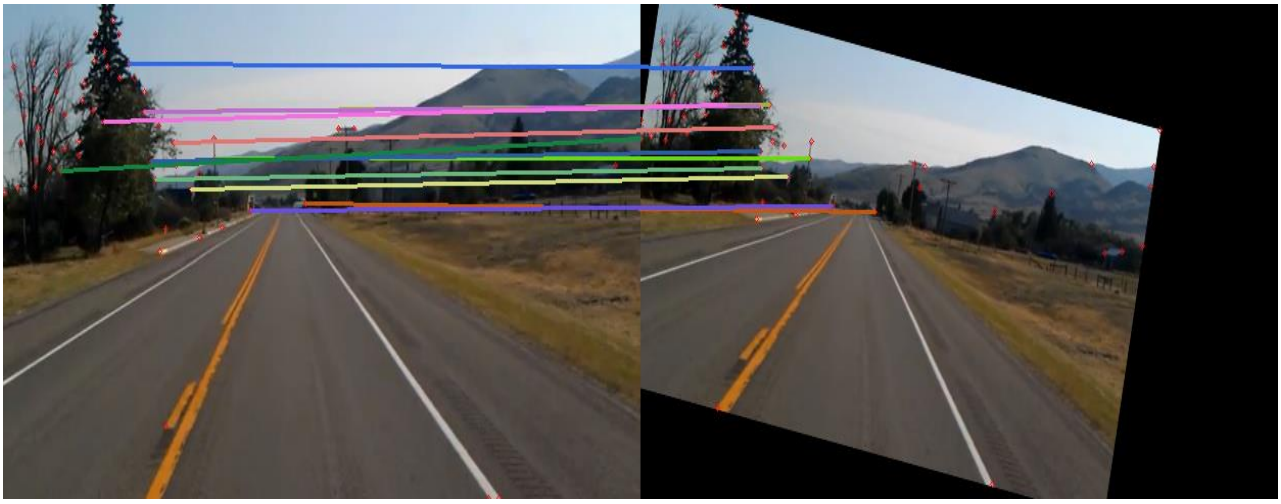
- ***SURF (Speeded-Up Robust Features):***

    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)***		The translation is a transformation that slides a figure on the coordinate plane without changing its shape, size, or orientation. It is a special case of an affine transformation.
	Rotation* (*T = MRMT1)		Rotation is the process of turning or rotating the image by a certain angle, for eg: 90 degree.
	Affine Transformation		Affine Transformation helps to modify the geometric structure of the image, preserving parallelism of lines but not the lengths and angles. It preserves collinearity and ratios of distances. This technique is also used to correct Geometric Distortions and Deformations that occur with non-ideal camera angles.
	Perspective Transformation		It means the change in perspective of an image from one view to another, for eg: changing the perspective view of a box from a different angle.
	Scaling		Resizing of a digital image

After Feature Detection and Tracking:



Scene:

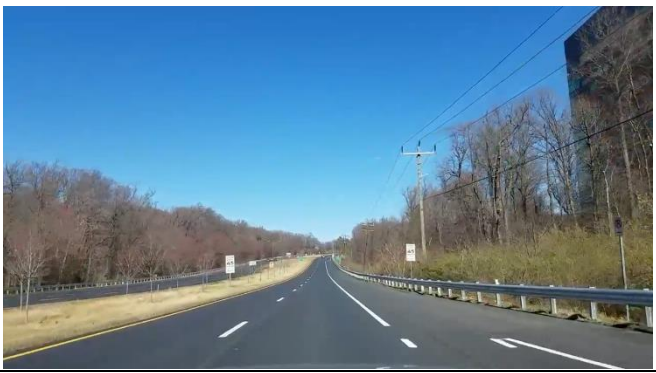
Scene	Feature to be detected	Sample Image
Image of some cyclists on the road	Faces (Pedestrian or Cyclists)	
Image of a road with lanes visible	Lanes	

Image of a road with lanes and rarely visible cars	Cars	

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

13. Face Detection

a. Viola-Jones Algorithm:

Algorithm 1 Computing a 24×24 image's Haar-like feature vector

Input: a 24×24 image with zero mean and unit variance
Output: a $d \times 1$ scalar vector with its feature index f ranging from 1 to d
Set the feature index $f \leftarrow 0$
Compute feature type (a)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ **do**
  for all (w, h) such that $i + h - 1 \le 24$ and $j + 2w - 1 \le 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    record this feature parametrized by $(1, i, j, w, h)$: $S_1 - S_2$
    $f \leftarrow f + 1$
  end for
end for
Compute feature type (b)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ **do**
  for all (w, h) such that $i + h - 1 \le 24$ and $j + 3w - 1 \le 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + 2w, j + 3w - 1]$
    record this feature parametrized by $(2, i, j, w, h)$: $S_1 - S_2 + S_3$
    $f \leftarrow f + 1$
  end for
end for
Compute feature type (c)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ **do**
  for all (w, h) such that $i + 2h - 1 \le 24$ and $j + w - 1 \le 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    record this feature parametrized by $(3, i, j, w, h)$: $S_1 - S_2$
    $f \leftarrow f + 1$
  end for
end for
Compute feature type (d)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ **do**
  for all (w, h) such that $i + 3h - 1 \le 24$ and $j + w - 1 \le 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    compute the sum S_3 of the pixels in $[i + 2h, i + 3h - 1] \times [j, j + w - 1]$
    record this feature parametrized by $(4, i, j, w, h)$: $S_1 - S_2 + S_3$
    $f \leftarrow f + 1$
  end for
end for
Compute feature type (e)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ **do**
  for all (w, h) such that $i + 2h - 1 \le 24$ and $j + 2w - 1 \le 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    compute the sum S_4 of the pixels in $[i + h, i + 2h - 1] \times [j + w, j + 2w - 1]$
    record this feature parametrized by $(5, i, j, w, h)$: $S_1 - S_2 - S_3 + S_4$
    $f \leftarrow f + 1$
  end for
end for

Algorithm 2 Integral Image

Input: an image I of size $N \times M$.
Output: its integral image II of the same size.
Set $\mathrm{II}(1,1) = I(1,1)$.
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    $\mathrm{II}(i,j) = I(i,j) + \mathrm{II}(i,j-1) + \mathrm{II}(i-1,j) - \mathrm{II}(i-1,j-1)$ and II is defined to be zero whenever its argument (i,j) ventures out of I's domain.
  end for
end for

Algorithm 3 Feature Scaling

Input: an $e \times e$ image with zero mean and unit variance ($e \geq 24$)
Parameter: a Haar-like feature type and its parameter (i, j, w, h) as defined in Algorithm 1
Output: the feature value
if feature type (a) **then**
  set the original feature support size $a \leftarrow 2wh$
  $i \leftarrow \lfloor ie/24 \mathrm{K} \rfloor$, $j \leftarrow \lfloor je/24 \mathrm{K} \rfloor$, $h \leftarrow \lfloor he/24 \mathrm{K} \rfloor$ where $\lfloor z \mathrm{K} \rfloor$ defines the nearest integer to $z \in \mathbb{R}_+$
  $w \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq \lfloor 1 + 2we/24 \mathrm{K}/2 \rfloor,\ 2\kappa \leq e - j + 1\}$
  compute the sum S_1 of the pixels in $[i, i+h-1] \times [j, j+w-1]$
  compute the sum S_2 of the pixels in $[i, i+h-1] \times [j+w, j+2w-1]$
  return the scaled feature $\frac{(S_1 - S_2)a}{2wh}$
end if
if feature type (b) **then**
  set the original feature support size $a \leftarrow 3wh$
  $i \leftarrow \lfloor ie/24 \mathrm{K} \rfloor$, $j \leftarrow \lfloor je/24 \mathrm{K} \rfloor$, $h \leftarrow \lfloor he/24 \mathrm{K} \rfloor$
  $w \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq \lfloor 1 + 3we/24 \mathrm{K}/3 \rfloor,\ 3\kappa \leq e - j + 1\}$
  compute the sum S_1 of the pixels in $[i, i+h-1] \times [j, j+w-1]$
  compute the sum S_2 of the pixels in $[i, i+h-1] \times [j+w, j+2w-1]$
  compute the sum S_3 of the pixels in $[i, i+h-1] \times [j+2w, j+3w-1]$
  return the scaled feature $\frac{(S_1 - S_2 + S_3)a}{3wh}$
end if
if feature type (c) **then**
  set the original feature support size $a \leftarrow 2wh$
  $i \leftarrow \lfloor ie/24 \mathrm{K} \rfloor$, $j \leftarrow \lfloor je/24 \mathrm{K} \rfloor$, $w \leftarrow \lfloor we/24 \mathrm{K} \rfloor$
  $h \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq \lfloor 1 + 2he/24 \mathrm{K}/2 \rfloor,\ 2\kappa \leq e - i + 1\}$
  compute the sum S_1 of the pixels in $[i, i+h-1] \times [j, j+w-1]$
  compute the sum S_2 of the pixels in $[i+h, i+2h-1] \times [j, j+w-1]$
  return the scaled feature $\frac{(S_1 - S_2)a}{2wh}$
end if
if feature type (d) **then**
  set the original feature support size $a \leftarrow 3wh$
  $i \leftarrow \lfloor ie/24 \mathrm{K} \rfloor$, $j \leftarrow \lfloor je/24 \mathrm{K} \rfloor$, $w \leftarrow \lfloor we/24 \mathrm{K} \rfloor$
  $h \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq \lfloor 1 + 3he/24 \mathrm{K}/3 \rfloor,\ 3\kappa \leq e - i + 1\}$
  compute the sum S_1 of the pixels in $[i, i+h-1] \times [j, j+w-1]$
  compute the sum S_2 of the pixels in $[i+h, i+2h-1] \times [j, j+w-1]$
  compute the sum S_3 of the pixels in $[i+2h, i+3h-1] \times [j, j+w-1]$
  return the scaled feature $\frac{(S_1 - S_2 + S_3)a}{3wh}$
end if
if feature type (e) **then**
  set the original feature support size $a \leftarrow 4wh$
  $i \leftarrow \lfloor ie/24 \mathrm{K} \rfloor$, $j \leftarrow \lfloor je/24 \mathrm{K} \rfloor$
  $w \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq \lfloor 1 + 2we/24 \mathrm{K}/2 \rfloor,\ 2\kappa \leq e - j + 1\}$
  $h \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq \lfloor 1 + 2he/24 \mathrm{K}/2 \rfloor,\ 2\kappa \leq e - i + 1\}$
  compute the sum S_1 of the pixels in $[i, i+h-1] \times [j, j+w-1]$
  compute the sum S_2 of the pixels in $[i+h, i+2h-1] \times [j, j+w-1]$
  compute the sum S_3 of the pixels in $[i, i+h-1] \times [j+w, j+2w-1]$
  compute the sum S_4 of the pixels in $[i+h, i+2h-1] \times [j+w, j+2w-1]$
  return the scaled feature $\frac{(S_1 - S_2 - S_3 + S_4)a}{4wh}$
end if

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**

b. Face Detection using MTCNN

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**) 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

c. 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

14. 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.

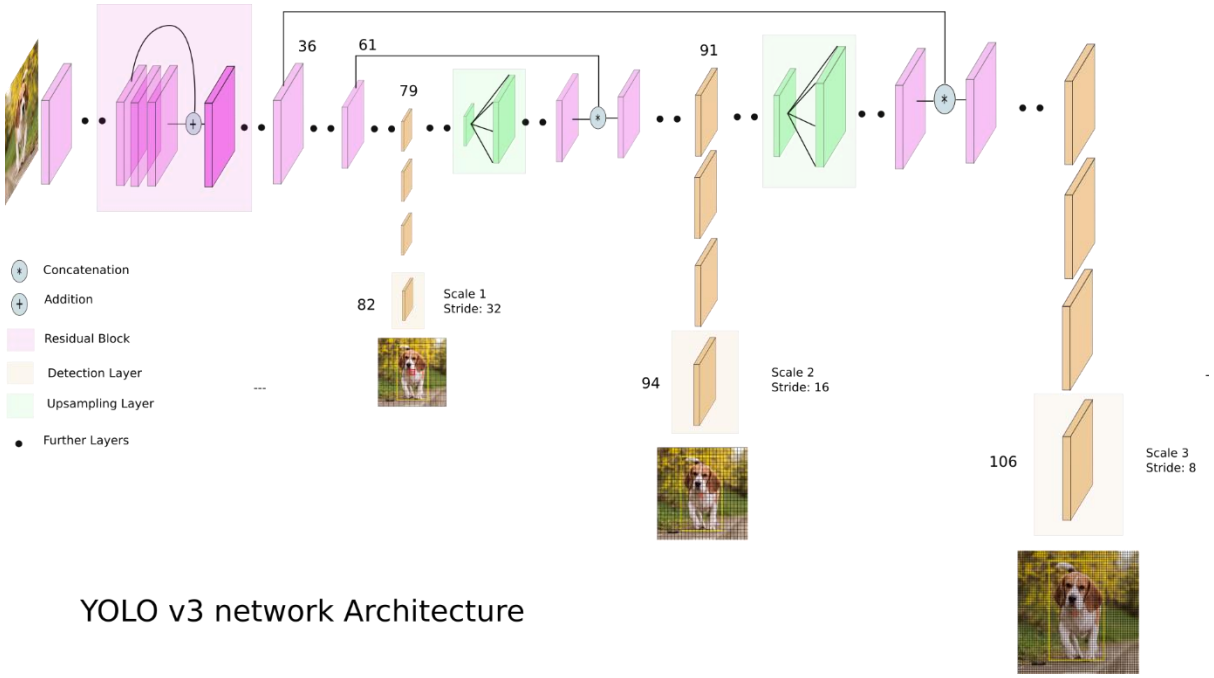
The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together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

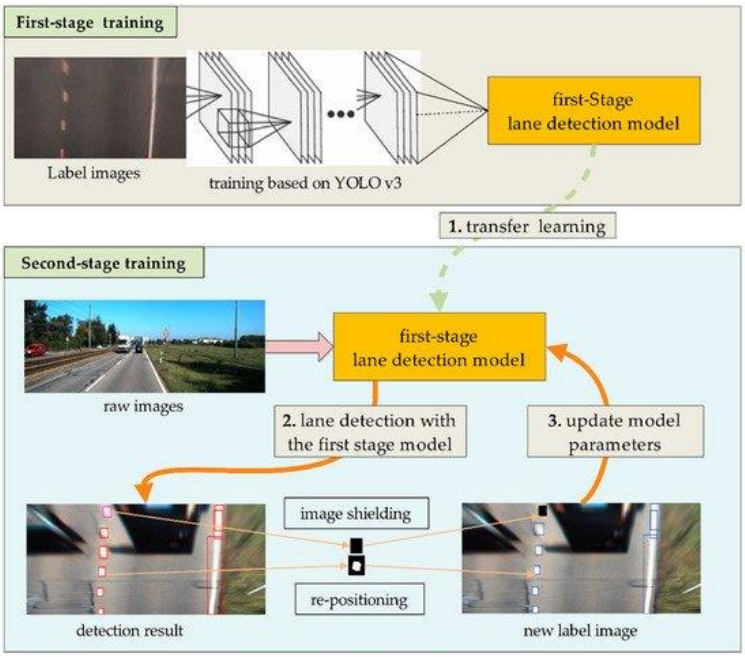
YOLOv3 uses a variant of Darknet, a framework to train neural networks, which originally has 53 layers. For the detection task, another 53 layers are stacked onto it, accumulating to a total of a 106-layer fully convolutional architecture. This explains the reduction in speed in comparison with the second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

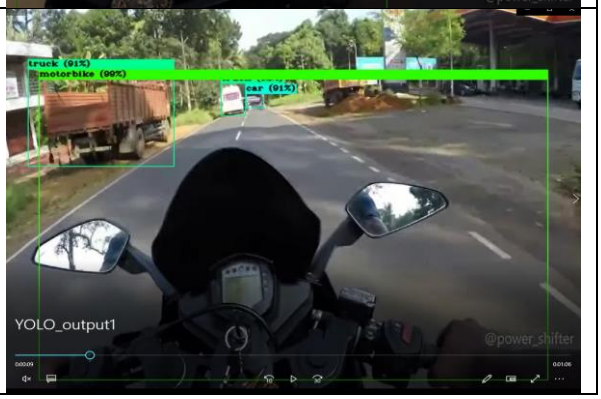
a. YOLOv3 Architecture Diagram:



YOLO v3 network Architecture

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
	
	

b. YOLOv3 configuration parameters:

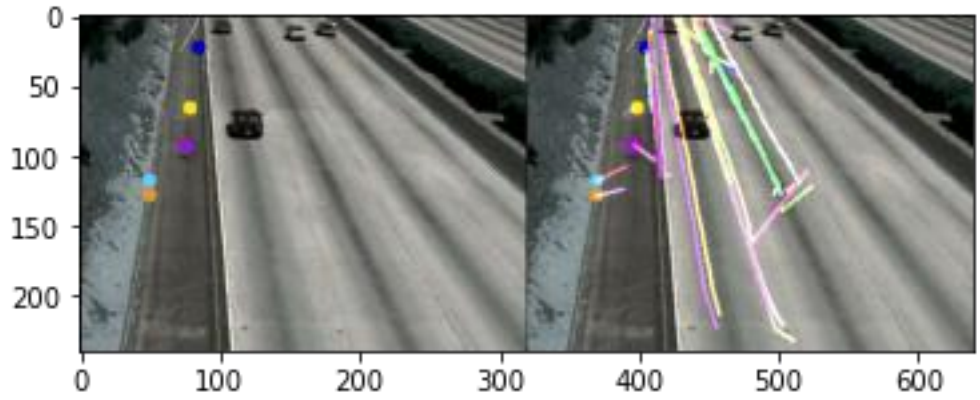
Batch hyper-parameter:	batch=64 subdivisions=16
Width, Height, Channels:	width=416 height=416 channels=3
Momentum and Decay:	momentum=0.9 decay=0.0005
Learning Rate, Steps, Scales, Burn In:	learning_rate=0.001 policy=steps steps=3800 scales=.1 burn_in=400
Data augmentation	angle=0 saturation = 1.5 exposure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

15. Optical Flow

a. Sparse Optical Flow: Lucas-Kanade method

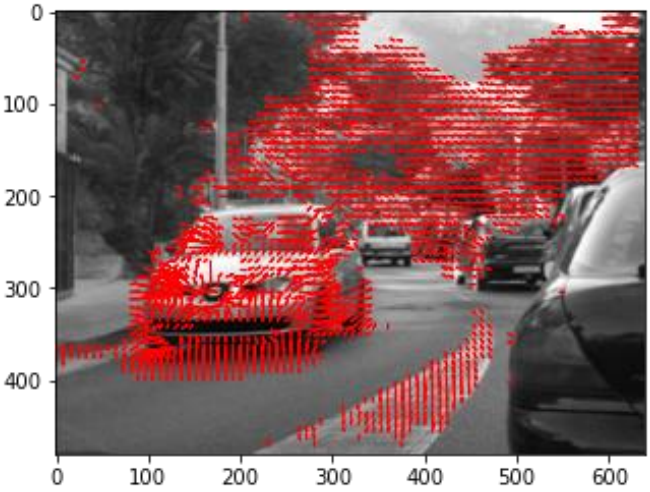
All the neighboring pixels will have similar motions. Lucas-Kanade method takes a 3x3 patch around the point. So all the 9 points have the same motion. We can find (f_x, f_y, f_t) for these 9 points. So now our problem becomes solving 9 equations with two unknown variables which are over-determined. 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.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum if_{x_i}{}^2 & \sum if_{x_i y_i} \\ \sum if_{x_i y_i} & \sum if_{y_i}{}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum if_{x_i} f_{t_i} \\ -\sum if_{y_i} f_{t_i} \end{bmatrix}$$



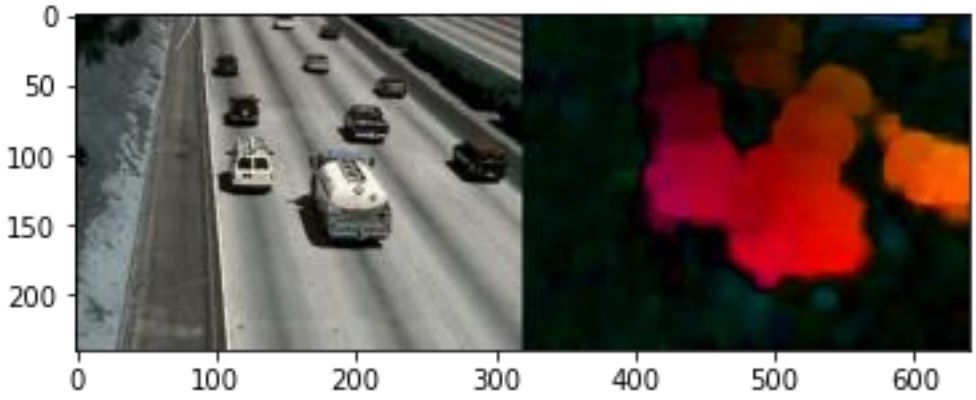
b. Horn–Schunck method

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. Dense Optical Flow

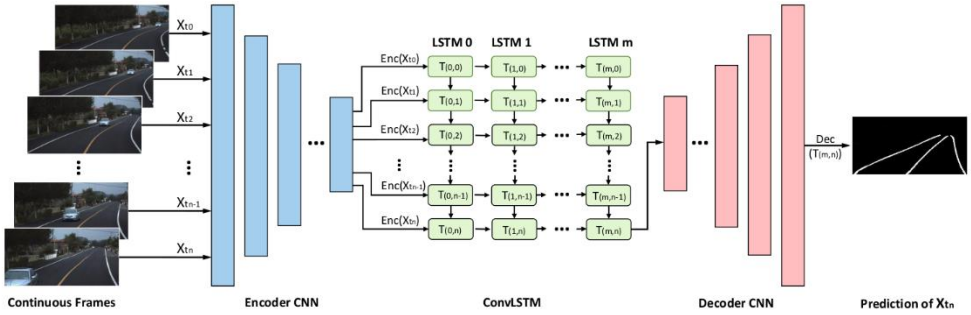
Dense Optical Flow computes the optical flow for all the points in the frame. It is based on Gunner Farneback's algorithm which is explained in "Two-Frame Motion Estimation Based on Polynomial Expansion" by Gunner Farneback.



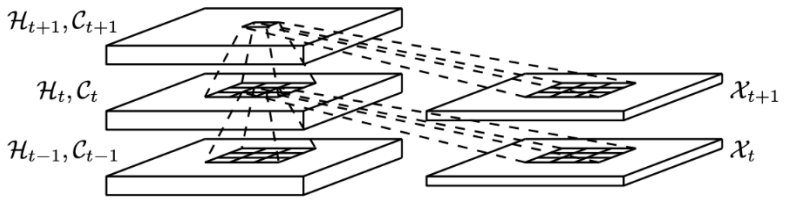
16. Deep Learning Architecture: LSTM (Long Short-Term Memory):

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.

- **ConvLSTM Architecture Diagram:**

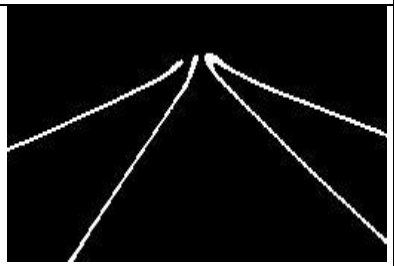
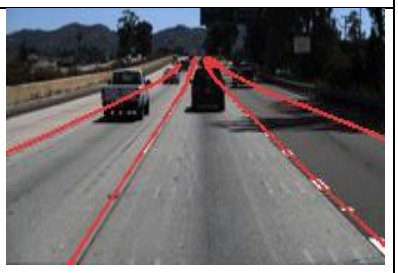
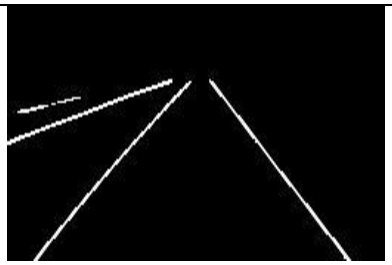
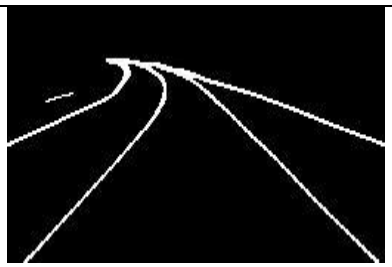
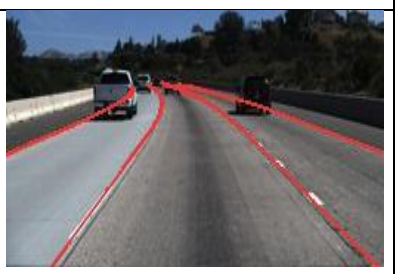
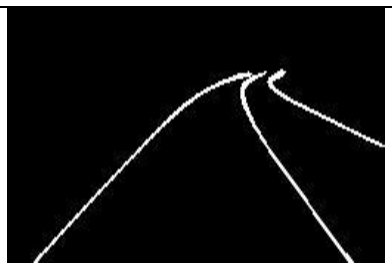


ConvLSTM is a type of recurrent neural network for Spatio-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. **ConvLSTM** 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.



Inner Structure of ConvLSTM

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.

Sample frame from the input video	Ground Truth	Sample frame from the output video
		
		
		
		

- **Dataset Description: tvtLANE Dataset:**

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.

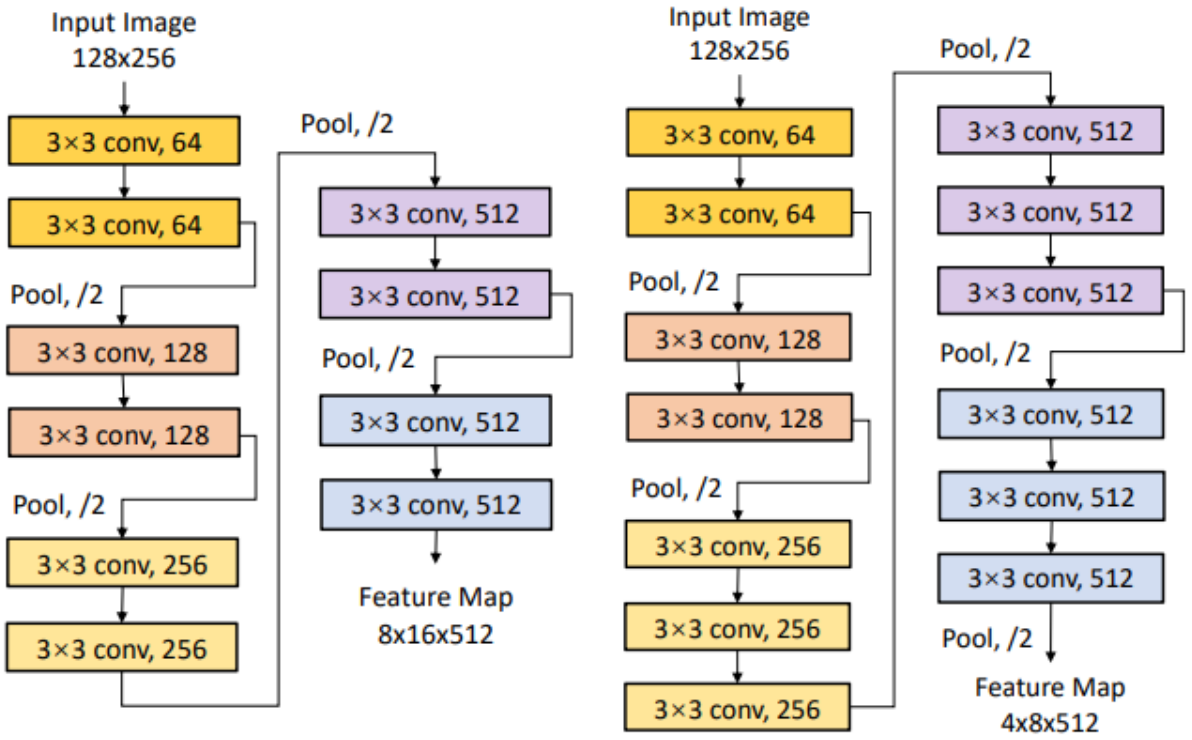
- **Training, Testing, and Validation**
  - Training data is used to help our machine learning model make predictions. It's the largest part of our dataset, forming at least 70-80% of the total data we'll use to build our model.
  - Validation data is primarily used to determine whether our model can 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.

- **Model Working:**

A block of the encoder network includes two convolution layers with twice the number of convolution kernels as comparing to the last block and a pooling layer is used for downsampling the feature map. In the decoder CNN, the size and number of feature maps should be the same as their counterparts in the encoder CNN.

In U-Net, feature-map appending is performed in a straightforward way between the corresponding sub-blocks of the encoder and decoder. While in SegNet, the deconvolution in the decoder is performed by using the indices recorded in the encoder.



UNet **SegNet**

Hyperparameter	**Value**
Resolution	256 x 128
Batch Size	16
Number of Epochs	100
Number of Conv. layers	SegNet: 13
	UNet: 10
Number of MaxPooling layers	SegNet: 5
	UNet: 4
Size of kernel	3 x 3

Performance Analysis			
Method	**Precision**	**Recall**	**F1-Measure**
SegNet_ConvLSTM	0.852	0.964	0.901
UNet_ConvLSTM	0.857	0.958	0.904

- **Deep Learning Architectures Comparison:**

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

- **Features list:**

	Scene Classification														
	Lanes						Object				Inter Class		Intra Class		
F. No	Colors	Shapes	Edges	Blobs	Corner	Ridges	Text	Vehicles	Pedestrians	Signals	Lane	No Lane	Straight Lane	Curved Lane	Scene
I1	15	3	12	1	15	16	4	6	2	1	0	1	0	0	Pedestrian crossing
I2	8	1	11	10	5	5	7	18	9	0	1	0	1	0	Vehicle overtaking
I3	8	20	16	6	17	17	14	8	15	0	1	0	1	0	Signal turning green
I4	19	6	5	18	12	7	10	6	13	0	0	1	0	0	Vehicle passing by
I5	16	3	6	11	1	4	14	9	10	0	1	0	0	1	People standing next to road

17. Action Recognition:

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

Classification	Type	Specification
Road Based on Location	Village /Rural Roads	Greenery and buildings could be seen consecutively
	City /District Roads	Lots of buildings could be seen consecutively
	Remote Area Roads /State Highway /National Highway	Lots of greenery could be seen consecutively
Road based on Material	Bituminous /Concrete Road	Mostly Black / grey in color
	Wet /Muddy Road	Mostly High contrast black / Orange in color
Road Based on Traffic Volume	Light Traffic Roads	<400 vehicles per day
	Medium Traffic Roads	400 - 1000 vehicles per day
	High Traffic Roads	>1000 vehicles per day
Road Based on Geometry	Straight Road	Lines on both sides would be straight
	Curvy Road	Lines on both sides would be curvy
Road Based on Lane Marking Visibility	Visible Lane Marking	White/Yellow line could be identified on the road
	Unvisible Lane Marking	White/Yellow line could not be identified on the road

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

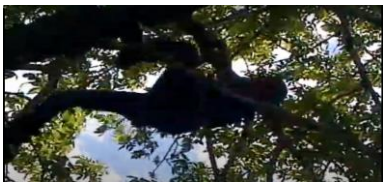
Sample images from the video:

- **Dataset Details:**

The dataset contains 400 human action classes, with at least 400 video clips for each action. Each clip lasts around 10s and is taken from a different YouTube video. The actions are human focussed and cover a broad range of classes including human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands. We describe the statistics of the dataset, how it was collected, and give some baseline performance figures for neural network architectures trained and tested for human action classification on this dataset. We also carry out a preliminary analysis of whether an imbalance in the dataset leads to bias in the classifiers.

- **Description of the Scene:**

Details of the scene with images and the type of actions:

Action Name	Description	Sample Image	Output with probability
Riding motorcycle	The video obtained from the camera fixed o the helmet of a person riding a motorcycle.		[motorcycling] 0.953. [pushing wheelchair] 0.023. [driving car] 0.015. [pumping gas] 0.003. [texting] 0.002.
Climbing a tree	The video of a man is climbing a tree by stepping on its branches.		[climbing tree] 0.994. [trimming trees] 0.004.
Riding a bicycle	The video of a kid riding a bicycle on a road		[motorcycling] 0.809. [riding a bike] 0.145. [riding unicycle] 0.012.
Playing the piano	The video of a person playing the piano.		[playing recorder] 0.658. [shuffling cards] 0.046. [playing trumpet] 0.030.
Javelin Throw	The video of a person throwing a javelin		[javelin throw] 0.998. [pole-vault] 0.001.
Riding a scooter	The video of a small kid riding a toy scooter		[riding scooter] 0.895. [hoverboarding] 0.105.

- **Deep Learning Architecture:**

stage	*Slow* pathway	*Fast* pathway	output sizes $T \times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	*Slow* : 4×224^2 *Fast* : 32×224^2
conv_1	1×7^2, 64 stride 1, 2^2	5×7^2, 8 stride 1, 2^2	*Slow* : 4×112^2 *Fast* : 32×112^2
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : 4×56^2 *Fast* : 32×56^2
res_2	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 3$	*Slow* : 4×56^2 *Fast* : 32×56^2
res_3	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	*Slow* : 4×28^2 *Fast* : 32×28^2
res_4	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 6$	*Slow* : 4×14^2 *Fast* : 32×14^2
res_5	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	*Slow* : 4×7^2 *Fast* : 32×7^2
global average pool, concate, fc			# classes

SlowFast networks for video recognition. This model involves

(i) A Slow pathway, operating at a low frame rate, to capture spatial semantics, and a Fast pathway, operating at a high frame rate, to capture motion at fine temporal resolution. The Fast pathway can be made very lightweight by reducing its channel capacity, yet can learn useful temporal information for video recognition.

- **Inference:**

From the results, we can infer that the actions are being recognized on the abstract level.

SlowFast model gives a mAP of 42.5 for the kinetic400 dataset. The accuracy for each sample image has been shown in the table above.

18. Future enhancements:

Enhance the lane detection system by adding lane fitting into the proposed framework. In this way, the detected lanes will be smoother and have better integrity. Besides, when strong interferences exist in dim environment, SegNet-ConvLSTM was found to function better than UNet-ConvLSTM, which needs more investigations.

19. References:

1. https://github.com/overtunned/lane_detection/tree/main/Dataset
2. GitHub - rslim087a/road-video: Video required for finding lane lines
3. CULane dataset:

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

4. TUsimple dataset: https://github.com/TuSimple/tusimple-benchmark/issues/3
5. Real-time detection of road lane-lines for autonomous driving
6. Multi-Lane Detection and Tracking Using Vision for Traffic Situation Awareness
7. Real-Time Tracking and Lane Line Detection Technique for an Autonomous Ground Vehicle System
8. https://tryolabs.com/resources/introductory-guide-computer-vision
9. https://paperswithcode.com/dataset/tusimple

10. https://arxiv.org/abs/2105.13680v1
11. https://xingangpan.github.io/projects/CULane.html
12. https://arxiv.org/abs/1712.06080
13. https://drive.google.com/drive/folders/1MI5gMDspzuV44lfwzpK6PX0vKuOHUbb_?usp=sharing
14. https://bair.berkeley.edu/blog/2018/05/30/bdd/
15. https://arxiv.org/abs/1805.04687
16. https://ieeexplore.ieee.org/abstract/document/6855185
17. https://ieeexplore.ieee.org/abstract/document/6775702
18. https://www.sciencedirect.com/science/article/abs/pii/S0031320315004690
19. https://ieeexplore.ieee.org/abstract/document/5369883
20. https://ieeexplore.ieee.org/abstract/document/1640414
21. https://ieeexplore.ieee.org/abstract/document/6232168
22. https://ieeexplore.ieee.org/abstract/document/8303759
23. https://ieeexplore.ieee.org/abstract/document/6232199
24. https://ieeexplore.ieee.org/abstract/document/660838
25. https://www.sciencedirect.com/science/article/abs/pii/S0030402613009005
26. https://www.sciencedirect.com/science/article/abs/pii/S107731420190921X
27. https://ieeexplore.ieee.org/abstract/document/6777046
28. https://ieeexplore.ieee.org/abstract/document/1000242
29. https://ieeexplore.ieee.org/abstract/document/6165309
30. https://ieeexplore.ieee.org/document/9308462
31. https://ieeexplore.ieee.org/document/8854084
32. https://ieeexplore.ieee.org/abstract/document/8954864
33. https://ieeexplore.ieee.org/abstract/document/6301978
34. https://ieeexplore.ieee.org/document/7780460
35. https://ieeexplore.ieee.org/document/8627998
36. https://ieeexplore.ieee.org/document/7477319
37. https://ieeexplore.ieee.org/document/8883072
38. https://ieeexplore.ieee.org/document/9301184
39. https://ieeexplore.ieee.org/document/9145130
40. https://www.sciencedirect.com/science/article/pii/S0167926019301762
41. https://ieeexplore.ieee.org/document/4408903
42. https://ieeexplore.ieee.org/document/5476151
43. https://ieeexplore.ieee.org/document/8615784