Non-Intrusive Driving Behavior Characterization From Road-Side Cameras

Pavana Pradeep Kumar[1], Krishna Kant[1] and Amitangshu Pal[2]

Abstract—**In this paper, we demonstrate that Deep Learning (DL) and spatiotemporal reasoning can effectively identify driving behavior based on the videos captured by roadside cameras. The use of roadside infrastructure for such determination is twofold: (1) a global view of the vehicles and their interactions, and (2) no involvement or awareness of the vehicles or their drivers, so the determination is inexpensive, easy to deploy, and entirely non-intrusive. Furthermore, our method uses deep learning only for object detection and tracking and builds a flexible and explainable reasoning model to identify the driving behavior. The essential advantage of this approach is that we use deep learning only for tasks that can be accomplished efficiently and with high accuracy (i.e., object detection and tracking), which can be done in real-time. Although there are deep learning models for detecting complex activities (e.g., aggressive driving), they are much harder to train, require higher accuracy, and inferencing time may not satisfy real-time constraints. By using a setup with program-controlled robocars, we demonstrate that we can achieve accuracies of 98–99% for driving behavior characterization, and the mechanism can provide detection of 650 ms on a very dated desktop. The characterization can provide feedback to the driver (or the automated car) for improved traffic safety and roadway throughput.**

Keywords: Intelligent Transportation Systems, Reasoning, Event Logic, Smart Cities, Cyber-Physical Systems.

I. Introduction

Road accidents are responsible for ∼5 million severe injuries and ∼50K deaths annually in the USA [1]. Many of these accidents are caused by risky driving behavior [2]; therefore, monitoring driving behavior and providing feedback can significantly reduce the accidents or their severity. This paper aims to demonstrate that it is possible to accurately and non-intrusively characterize driving behavior from roadside camera views. We believe such techniques will be valuable even as Connected Automated Vehicles (CAVs) enter society. This is because the CAVs are likely to offer "personalization" to the users, allowing them to make the driving more or less aggressive [3], [4].

Vehicular and Driving Behavior Characterization: There is a tremendous amount of work on vehicular behavior characterization, mostly based on the in-vehicle monitoring of the (a) vehicular parameters (e.g., acceleration, braking, distance to the next car, speed and speed variability, lane change behavior, etc.) [5], and/or (b) driver (e.g., monitoring by various means whether the driver is drowsy, drunk, distracted, etc.) [6].

Extensive research has been conducted on *intrusive* methods of detecting driving behavior largely through smartphone video and other sensors. For example, a multi-sensor-smartphone-based mechanism is reported in [7]. A similar but vehicle-mounted acceleration measurement-based technique is discussed in [8]. The dashboard camera-based driver state identification is also explored extensively, e.g., yawning gestures [9], eye tracking [10], facial expressions) [11], etc.

The main problem with such methods is that they require initiatives on the part of the users to be monitored, and in many cases, even set up the equipment (e.g., smartphone) themselves to enable monitoring. Also, in most cases, the data would need to be streamed out of the car to a node for analysis, which has privacy issues and requires a stable internet connection which is not always available. Also, since each vehicle only has a local view, the behavior of a group of vehicles traveling together or nearby becomes interdependent, which makes it challenging to identify the cause-and-effect relationships among the vehicles. For example, a sudden slowing of a vehicle will cause the vehicle behind it also to slow suddenly. Thus, unless the movement data from multiple vehicles can be co-processed, in-vehicle monitoring cannot deal with such dependencies. Consequently, we do not believe such methods are practical, regardless of sophistication.

Our contributions: In this paper, we demonstrate that it is possible to characterize vehicular behavior very accurately using only the roadside cameras of the ITS system. To the best of our knowledge, this is the first such study; its key advantage is that it is entirely non-intrusive, inexpensive, and does not impact driving behavior in any way. Also, the global view can deduce the more obvious cause-and-effect relationships between adjacent vehicles (e.g., whether the car decelerated on its own or due to diminishing distance from the vehicle ahead). The proposed framework integrates (1) Deep Learning (DL) on camera images to recognize vehicles and estimate their kinematic parameters and (2) Spatiotemporal logic reasoning to characterize driving-related "micro-behaviors" (μBs) and their combination to identify driving behavior. We recognize three classes of driving behaviors, with an average accuracy of approximately 99%. Furthermore, we show that the error is primarily attributable to instances where a vehicle is not identified correctly in the image and thus can be improved by more robust tracking. The monitoring can provide feedback to the drivers (or automated vehicles) for improving both traffic safety and enhancing the roadway throughput.

Paper outline: The remainder of the paper is organized

[1] CIS Department, Temple University, Philadelphia, PA, USA. (e-mail: pavana.pradeep@temple.edu, kkant@temple.edu)

[2] CSE Department, Indian Institute of Technology Kanpur, Kanpur, India. (e-mail: amitangshu@cse.iitk.ac.in)

as follows. Section II presents our driving behavior characterization framework. Section III presents the experimental evaluation, and then Section IV concludes the paper.

II. Framework for Driving Behavior Prediction

A. Categorizing Driving behavior

Driving behavior is important from a safety perspective, but no standard way to characterize driving behavior exists. This paper adopts a 3-way classification, where driving is designated safe, aggressive, and distracted [12]. Please note that our "safe" category includes any behavior that is not considered aggressive or distracted; therefore, there is no undetermined behavior. For this categorization, we define a small set of "micro-behaviors" (μBs) and then characterize the driving based on the combination of those. Table I shows the μBs and abbreviations we use in the rest of the paper. For example, aggressive driving can be identified by a combination of μ_{wv}, μ_{os}, μ_{hb}, and μ_{tg}. In this paper, we formulate the driving behavior detection problem using spatiotemporal reasoning based on the location and movements of the vehicles determined from the YLLO-based analysis of video frames.

B. System Architecture

We assume that the camera deployment is such that all vehicles on the roadway are visible without excessive image distortion. Given the increasing processing power in smart cameras, each camera can do the object detection and tracking tasks discussed here rather than sending the raw video feed to the next level, *Road-Side Units (RSUs)*. In our earlier work, we designed a lightweight object recognition and tracking algorithm called YLLO (You Look Less than Once) that can run in the cameras and avoid transmission of redundant frames to RSUs [13]. The RSUs receive video streams from multiple cameras along a road segment and use them to monitor driving behaviors and associated anomalies. Further processing, including perspective transformation and estimation of orientations and speeds of the objects, may be done by the camera itself or the RSUs. The RSU can then build a spatiotemporal logic model of the situation that includes all "facts" of different driving behaviors, the conditions leading to near-miss accidents, and the supporting "theories" (i.e., Newton's laws, arithmetic, etc.)

Table I: Microbehaviors for Driving Characterization

Microbehavior	Meaning
μ_{wv}: Weaving	Swerving between lanes
μ_{ss}: Sudden Steer	Abrupt heading changes on a straight road
μ_{hb}: Hard Braking	Braking deceleration is -8 m/sec
μ_{ld}: Lane Drift	Not keeping in the center of the lane
μ_{st}: Straddling	Alternatively hugging left & right side of lane
μ_{os}: Overspeeding	Driving 25% above the speed limit
μ_{tg}: Tailgating	Following a vehicle at distance $<$ 2m

Fig. 1 depicts our overall architecture, comprised of two different frameworks. One framework is based on Deep Learning (DL) and logical reasoning that runs on roadside cameras receiving traffic videos as input. The other is based on logical reasoning that runs on individual vehicles receiving in-vehicle parameters as inputs.

The initial stage of the roadside camera framework is a lightweight object detection and tracking model based on a Convolutional Neural Networks (CNNs) model called YLLO that we have developed for video analysis [13]. YLLO performs the processing of video sequences in a highly efficient manner. The second stage for each detected object (e.g., vehicles, pedestrians, etc.) is a spatiotemporal logic-based reasoning system that captures the relative movements of the objects in real time to identify various driving behaviors. An event recognition tool uses a formal specification of driving behavior as part of the reasoning framework that operates on individual vehicles.

C. YLLO-based Object Detection and Tracking

YLLO is a lightweight object detection technique based on YOLOv6 [14] and is optimized for continuous video streams by utilizing redundancy to identify the "only" essential frames. It is a three-stage process that begins with a scene change detection algorithm and progresses to object detection via YOLOv6. The Simple Online and Real-time Tracking with a Deep Association Metric (Deep-SORT) algorithm assign a tracker to each detected object or multiple objects. YLLO decouples classification and regression tasks to eliminate redundant objects between the frames. Additionally, before sending frames to object detection, for the scene change detection, it generates Color Difference Histograms (CDH) for edge orientations, where edge orientations are determined using the Laplacian-Gaussian edge detection framework.

D. Spatio-Temporal Reasoning

We formulate our reasoning as a Boolean satisfiability problem so that the popular SMT (Satisfiability Modulo Theory) [15] based tools can be used along with suitable theories. To reason about temporal aspects, we make use of a popular framework called *Event Calculus* (EC) that introduces the concept of Events, which are actions that occur at a specific point in time, and Fluents, which are entities whose state changes in response to the occurrence of an event or action. In particular, we have used an open-source prolog implementation termed "Event Calculus for Run-Time Reasoning" (RTEC) [16]. RTEC specifies temporal relationships using LTL (Linear Time Logic) but with integer time points which makes it possible to speak of real-time as well.

We can define μB in RTEC as rules that define the event instances using the predicates ``happens at" (hA) and ``happens for" (hF). The *fluents* that have time-varying properties and the effects of events on *fluents* are defined using the inA and tA predicates. The value of *fluents* at any time point is defined using the hoA and hoF predicates. If *F* is a variable ranging over *fluents*, the term *F*=*V* denotes that the variable *F* has a value *V*. Boolean fluents also exist with values true or false. Table II shows the predicates used in the RTEC tool.

Characterizing driving in terms of μBs brings up one important issue: some μBs are more important than others. Some might even be essential for the driving classification to be valid (e.g., hard braking for aggressive driving). We

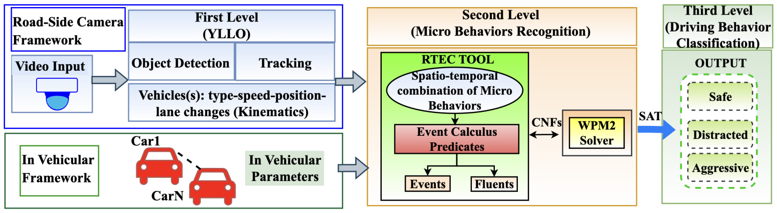

Road-Side Camera Framework
Video Input
First Level (YLLO)
Object Detection
Tracking
Vehicles(s): type-speed-position-lane changes (Kinematics)
In Vehicular Framework
Car1
CarN
In Vehicular Parameters
Second Level (Micro Behaviors Recognition)
RTEC TOOL
Spatio-temporal combination of Micro Behaviors
Event Calculus Predicates
Events
Fluents
CNFs
WPM2 Solver
SAT
Third Level (Driving Behavior Classification)
OUTPUT
Safe
Distracted
Aggressive

Fig. 1: Driving Behavior Characterization Framework.

Table II: Main predicates of RTEC

Predicate	Meaning
hA(E,T)	Event E *happens at* time T
hF(E,I)	Event E *happens for* I intervals
in(F=V)	*Initial* value of fluent F =V at time 0
hoA(F=V,T)	Value of fluent F *holds at* V at time T
hoF(F=V,I)	Value V of fluent F *holds for* I intervals
inA(F=V,T)	Fluent F with value V *initiated at* time T
tA(F=V,T)	Fluent F with value V *terminated at* time T

consider the corresponding μBs assertions *hard* in that they must hold. The others can be considered *soft* with a suitably assigned *weight* that reflects their importance. Let S_k denote the soft assertions for driving behavior k. We define weight as w_{ik} for each member i of the set S_k. Then, the driving behavior k will be recognized as (a) all hard assertions (μBs) holding and (b) $\sum_{i \in S_k} w_{ik} > W_k$ for some threshold W_k. Note that the weights need not be static but depend on various spatiotemporal factors and context (e.g., day vs. night time, roads with different speed limits, etc.) A weight change would require pausing all current condition evaluations, changing all weights that need to be changed simultaneously, and then restarting the evaluation. To handle hard/soft conditions, we can use an extension to the Boolean Satisfiability problem known as *Weighted Partial Maxsat* (WPM2). In WPM2, each clause is designated as either *hard* or *soft* with a given weight. We then have an optimization problem to find an assignment that satisfies all hard clauses and minimizes the total weight of soft clauses. The SAT core that the WPM2 solver returned confirms the existence of driving behavior.

E. Defining Events and Fluents in RTEC

The RTEC tool receives input as Event Calculus (EC) predicates representing time-stamped microbehaviors detected on individual video frames or the recorded in-vehicular parameters as shown in Fig 1. μBs such as acceleration, braking, lane change to the left or right, etc., are represented as events in EC that are defined along with their associated timestamps, which indicate the point in time at which the activity occurred. The hA predicate establishes this type of input. For instance, hA(hardBreaking(id6, 60) indicates that an object(id6) engaged in hard or sudden braking at video frame 60, which is determined by comparing the deceleration value going beyond a predefined threshold. Some of the microbehaviors are represented as fluents in the EC. We use the inA and tA predicates for expressing the conditions in which these fluents initiate and terminate a specific driving behavior described above.

The μBs represented as EC events are defined with hF predicate, which can also compute the associated intervals. For example, hF(laneDrifting(id3) = true, [(0, 60),(210, 280)] indicates that object3 was not keeping the center of the lane during the intervals (0, 60) and (210, 280). A few examples of events and fluents as follows :hF(proximityRight(v) = true) is a Boolean event that indicates that vehicle v is close to the right side of the lane, which is a combination of other related fluents and events, hoA(normalBraking(v)= true) is fluent that indicates that vehicle v executes normal braking) i.e., normal braking deceleration is denoted by a constant nbd of -3 m/s^2.

After defining events and fluents, we must define an initiation and termination map for each defined fluent in the system, indicating which events initiate and terminate which fluents. The next step is to specify the relation between fluents and events in the form of rules. For example, the initiation and termination map for fluent overSpeed(v) is shown in the following expression:

inA(overSpeed(v) = true, T) ← hoA(speed(v, S_v), T) ∧ hoA(atLane1(v), T) ∧ th(os, S_v >= os).
tA(overSpeed(v) = true, T) ← hoA(speed(v, S_v), T) ∧ hoA(lane1(v), T) ∧ th(os, S_v < os)

Here, overSpeed, speed are input events, and S_v denotes the momentary speed of vehicle v, and lane1 is the input fluent indicating the presence of the vehicle in lane1. th is a temporal predicate indicating the numerical threshold of driving parameters, and in this case, it represents the user-specified speeding threshold denoted by os. The ruleset defined states that overSpeed(v) is a Boolean fluent, which is invoked when a speed event is detected. Further, the vehicle is present in lane1, which is detected by fluent lane1, and the momentary speed of the vehicle, S_v is more than the user-specified speeding threshold (os). The event overSpeed(v) is terminated when the vehicle vs. speed is smaller than the speeding threshold. The exact definition applies when the vehicle is located in lane2, as indicated by fluent lane2 to detect the vehicle's speeding in lane2.

F. Characterizing Driving Behavior

As discussed in II-A, each category of driving behavior combines different types of μBs, wherein all μBs defining a driving behavior must be satisfied. For instance, the μBs that represent aggressive driving behavior as RTEC events or fluents include events such as suddenSteer, weaving, and fluents such as laneChange, atLane1, atLane2, overSpeed, tailgating, and hardBraking. The aggressive driving behavior is represented as a Boolean fluent as follows:

inA(aggressiveDriving(v) = true, T) ← hoA(atLane1(v), T) ∧ ¬ hoA(atLane2(v), T) ∧ hoA(hardBraking(v), T) ∧ hoA(laneChange(v), T) ∧ hoA(overSpeed(v), T) ∧ hA(weaving(v), T) ∧ hA(suddenSteer(v), T) ∧ ¬ hA(safeDriving(v), T) ∧ ¬ hA(distractedDriving(v), T)

Similarly, distracted driving behavior is a Boolean fluent defined as a conjunction of events like laneDrifting, straddling, and fluents like laneChange, atLane1, atLane2, slowSpeed, and normalBraking.

To express the dependencies when modeling the driving behavior, we define derived events, i.e., the events that occur due to the change in state or value of another fluent and/or the occurrence of another event (e.g., the effect of normal or harsh braking on a vehicle's speed). These events indicate when specific actions occur in the traffic due to a combination of particular conditions. If **R** denotes the rules passed to the WPM2 solver before invoking the WPM2 solver, we find relevant rules corresponding to derived events based on current ongoing events. We identify the rules or relations **R'** $\subseteq$ **R** that lead to derived events based on the current events, either directly or indirectly. The dependency is expressed via a dependency graph G, where the vertices denote the rules/relations, and the (directed) edges denote the dependency between them. Finally, the rules **R'** expressed in Conjunctive Normal Form (CNF) are passed on to a WPM2 solver [17].

III. Experimental Evaluation

In this section, we evaluate our framework on our collected PiCarX dataset. The following metrics determine the effectiveness of our framework: (a) the accuracy and (b) the percentage of errors in driving behavior recognition. The experiments were performed on a computer with Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz, 32 GB RAM, and 1 TB SSD.

A. Modeling Vehicular Driving

The vehicular traffic on a roadway has been characterized by numerous models starting with the early 1950s. The models can be microscopic (i.e., model the behavior of each vehicle) or macroscopic (i.e., model the behavior of the traffic as a whole). Numerous microscopic (or "car-following") models exist, which are reviewed in a recent article [18] that also examines the incorporation of human factors into these models. Most models are continuous time, continuous space type, and for a single lane, express acceleration of a vehicle as a function of its current speed, distance to and speed of the next vehicle (and sometimes the previous vehicle), etc. Most models also introduce random slowdowns to model human behavior and break the strict vehicle-following behavior. Generally, lane change behavior is tacked on to the single-lane models with rules concerning when lane change can occur; however, such models quickly become very difficult to analyze mathematically.

In our implementation, we used the so-called Cellular Automaton (CA) model [19], simplifying the introduction of complex rules and simulation implementation by discretizing time and space. A roadway is seen as a sequence of fixed-sized cells in the CA model. At any point in time, a cell could be empty or occupied. That is, a cell can be occupied by only one vehicle, although it is possible to model large vehicles that occupy multiple consecutive cells. In each time step, a vehicle may move over some integral number of cells depending on its speed and the availability of the cells ahead. The CA model makes it easy to introduce complex rules that account for various situations, including the presence of signals or other traffic control mechanisms. Complex lane change rules can also be coded, such as a lane change decision based on the number of free cells ahead and behind in both the source and target lanes. It is typical to assume that a lane change always occurs in a single step, and the discrete model is well suited for this kind of discontinuous change. CA model makes it easy to provide various forms of driving personalities to a vehicle in terms of the vehicle following, lane change, safe/unsafe vehicle position in a cell, etc.

Table III: Statistics of Datasets Used

Data Collected	Behavior	#Samples
Road-Side Camera Video Data (PicarX)	Safe	410
	Distracted	340
	Aggressive	300
In-Vehicular Data (PicarX)	Safe	125
	Distracted	115
	Aggressive	110
TU-DAT Dataset	Safe	1200
	Distracted	720
	Aggressive	960

B. Experimental Setup and Dataset Collection

One significant difficulty in conducting this work was the scarcity of real datasets that provide the vehicular motion parameters and the external videos we can analyze.

This paper can be considered a proof of concept (PoC) of characterizing the vehicular behavior remotely and non-intrusively without any involvement of the vehicle/driver or deployment of any further instrumentation in the vehicles. In particular, this paper aims to study how accurately this can be done by comparing it against the ground truth. Unfortunately, conducting such a study with actual vehicles on the road is impossible because of safety concerns since we need the vehicles to perform unsafe maneuvers. Also, to compare the video monitoring results against the ground truth, we must tap into the Onboard Diagnosis (OBD) systems and obtain detailed vehicular information.

In order to get around these difficulties, we set up an entire infrastructure using the automated toy cars, known as "PiCarXs" [20] in an indoor basement environment. Each PiCarX carries a raspberry pi board for programmatically controlling the car. We assembled six such cars and used a logically centralized controller to independently control each car's behavior remotely over the WiFi link. We also set up a "roadside" external camera to record the videos of the cars independently and analyze those videos in real-time to determine and classify the behavior of each PiCarX. Each PiCarX used the 2-lane extension of the basic Cellular Automata (CA) car-following model [21] and further updated the model to introduce different driving behaviors. The software control of the vehicles can read their current speed and other parameters via appropriate sensors. This PicarX is equipped with speed sensors, acceleration sensors, heading sensors, proximity sensors, and a lane detection camera. These sensors are utilized to record ”only” the ground truth values necessary for validation with the proposed roadside camera framework. We analyze the driving behavior for both in-vehicle sensing units and roadside camera units using a mixture of CNN models and logic-based behavior analysis. Finally, we validate our claim of analyzing driving behavior "only" from the roadside units by correlating the in-vehicle data and the roadside data.

We captured High Definition Video Streams (HDVS) at 30 FPS with PicarX cars imitating different driving behavioral patterns on the road as discussed in section II-A. By varying speed limits, we also capture the driving traffic patterns on different road types, such as highways versus local roads. Along with capturing the video streams, we also record the eight in-vehicular driving parameters from all six robot cars, including vehicle orientation, acceleration, deceleration, braking, steering angle, lateral position, and lane change maneuvers to the left and right lanes.

Fig. 2(a) shows the side view of PicarX used in the experiments, respectively. The experimental setup is shown in Fig. 2(b) and shows two distinct lanes with a single-direction traffic flow and detection of Picars. We use a tripod and Sony FDR-AX33 Camcorder to record the simulation videos.

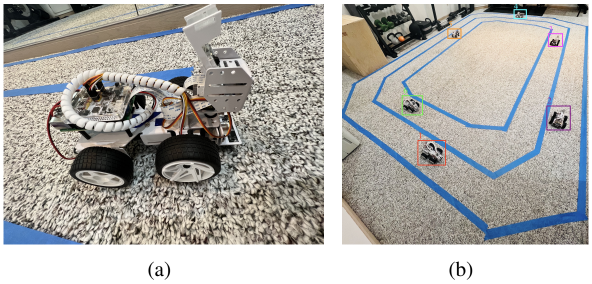

(a) (b)

Fig. 2: (a) Side View of PiCarX, (b) PicarX detection and tracking.

We conducted five video simulations for our controlled environment, with each video being around 2-3 minutes long and averaging around 27,000 video frames. In addition, to simulate the in-vehicle framework as discussed in section II, we have approximately 27000 seconds of time-series data for all required in-vehicle driving parameters as discussed above.

In addition to the simulated videos, we select approximately 48 real-world traffic videos from our other dataset, TU_DAT dataset, to evaluate the effectiveness of our proposed work. TU_DAT dataset was used in our previous work [22] to predict and resolve anomalous situations in CCTV traffic videos, which contain a diverse collection of accident videos collected in challenging environments. Table III shows the details of the data collected and the number of samples.

C. Implementation of Cellular Automata (CA) Model

We implement the CA car-following model in each PiCarX and then extend the model to implement different driving behaviors. A classical CA model is a uniform lattice of cells representing an identical finite automaton with a state and a transition function; each cell takes its state and the states of a set of neighboring cells defined by a time and space-invariant geometrical pattern. Starting with an initial condition, the CA evolves by activating all transition functions simultaneously and sequentially. We use a generic, multi-layered, and complete cellular automaton simulation engine in Python called "Cellular Automata General Environment" (CAGE) [23]. The transition rules in CAGE can be specified at different spatial levels and can change as a function of space and time, making it suitable for use alongside a logical reasoning tool.

In our two-lane experimental setup, we implement abstractions of the CA model, road topology, and neighborhood information in CAGE to represent the various driving behaviors and a two-lane car-following model. An address in CAGE is a tuple of one or more integers representing the location of a cell within a given topology. Topologies determine the arrangements of cells in a network, and neighborhoods encapsulate the translation of addresses to a list of their neighbors, considering the topology they are connected with. For our experiments, we implemented a line map topology with a radial neighborhood for each lane having a total of 35 cells.

D. PicarX Object Detection

As discussed in II-C, the roadside camera or RSUs use YLLO-based object detection and tracking framework and a logical-reasoning-based method to classify the driving behavior into one of three categories. The YLLO running on the cameras must be able to detect and classify the robot cars since we use robot cars to simulate road traffic of different driving behaviors. To train YLLO to recognize the robot cars, we constructed our training and testing set by selecting positive samples from the recorded videos. For negative samples, we have used the frames from our toy car experiment videos in our previous work [13]. Overall we have a total of 18,550 samples used to train the model. To reduce the workload of annotating the dataset, we have used an auto annotation tool [24], which is based on a semi-supervised architecture, where a model trained with a small amount of labeled data is used to produce the new labels for the rest of the dataset. Since YLLO is a CNN-based object detection model, it requires large amounts of data. Hence, we have used various augmentation techniques to reach sufficient data amounts, with most augmentations occurring at run-time using Keras built-in functions. Keras offers several techniques for online image augmentation, meaning that the augmentation is done as the network processes each image. Consequently, multiple augmentations can be performed without saving each image separately on the computer. The augmentations used were flipping, translation, shear, and rotation. The trained model has an overall identification accuracy of 90.11%. Fig. 2 (b) shows the PiCarX detection and tracking results of the trained YLLO model. Due to space limitations, the trained model's results are omitted from this paper.

We evaluate the performance of the proposed driving behavior detection system using both roadside cameras and an in-vehicle framework. We evaluate the classification performance of the driving behavior using standard performance metrics such as precision, recall, and accuracy.

E. Results and Discussion

Estimating various vehicular parameters, such as speed, vehicle orientation, etc., forms the basis of driving behavior analysis. For this, we first need to calibrate the camera so that it is possible to correct the inherent perspective distortion in the images. The perspective effect relates 3D points on the world coordinate system to 2D pixels on the image plane differently. This effect assigns distinct informational contents to different image pixels. Inverse Perspective Mapping (IPM) aims to invert the perspective effect, thereby imposing a uniform distribution of information across the image plane. To map the front-view image smoothly into a bird's-eye view for videos captured with PicarX cars, we employ the IPM technique.

Table IV: Performance Results using PicarX Dataset

Data Collected	Driving Behavior	Precision	Recall	Accuracy
Road-Side Camera	Safe	100	98.78	99.52
	Distracted	95.95	98.22	98.09
	Aggressive	97.98	97.0	98.57
In-Vehicular Data	Safe	100	99.2	99.71
	Distracted	98.28	99.13	99.14
	Aggressive	99.09	99.09	99.43

The performance of the proposed roadside camera-based driving behavior analysis framework is evaluated by first detecting a set of micro behaviors, which are then composed to classify the driving behavior into safe, distracted, and aggressive driving behaviors. We validate the proposed framework against the logic-based reasoning framework operating in individual vehicles, which provides the actual values of driving parameters such as speed, orientation, distance from the car in front, distance from the left or right side of the lane, etc. Table IV shows the results, and it can be seen that the roadside camera framework achieves an average precision and recall of 97.98% and 98.05%, respectively, and an average accuracy of 98.73% which are averaged over five runs of the experiments. The difference between these performance metrics values and those of the in-vehicle framework is small, ranging between 1-2 percent.

Table V: Performance Results using TU_DAT (without IPM)

Dataset	Driving Styles	Prec.	Recall	Accuracy
TU-DAT	Safe	95.0	95.0	95.83
	Distracted	83.5	84	91.7
	Aggressive	93.75	93.75	95.8

In addition, we analyzed the errors that may arise when characterizing driving behavior via roadside cameras. We observed that object detection and tracking errors are approximately 0.57%, and driving signal parameter estimation errors in comparison to the ground truth values are approximately 1.33%. Errors are nearly negligible, demonstrating the effectiveness of roadside camera-based driving behavior characterization. Based on our analysis, we conclude that the proposed framework can be used effectively to determine a vehicle's non-intrusive behavior without requiring the installation of additional sensors within the vehicles. We use land markings and standard formulas for the calculation to measure the speed and acceleration of PicarX vehicles from roadside cameras. In reality, these land markings could be located near the light poles where the roadside cameras are installed.

In addition to the recorded videos using PicarX, we have evaluated the proposed mechanism using the TU_DAT dataset. Since this dataset is comprised of CCTV traffic videos of anomalous situations from various parts of the world, these cameras' intrinsic and extrinsic parameters may be unknown or different from one another due to mounting setups and camera types. Therefore, individual calibration of the captured videos is not possible. Table V shows the performance results of the proposed roadside camera framework on TU_DAT. The average precision and recall values are 90.694%, and the accuracy is approximately 94.5%. *It is important to note that the worse accuracy here is primarily due to lack of perspective correction; if the camera position and angles were known, we believe that the accuracies here would be similar to those obtained using PiCarX.* The datasets used in this paper are made available for research use and can be found in GitHub[1].

1) Comparison with Machine Learning (ML) baseline models

We compare the performance of our proposed Roadside camera framework with existing Machine Learning models as shown in Fig. 3(a) where the x-axis shows the ML models used for comparison and the y-axis shows accuracy. We include different classifiers for comparisons like Decision Tree (DT), Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF), and Bagging Classifier (BC). Simple kinematic parameters like each vehicle's relative speed, acceleration, orientation angle, distance between vehicles, etc., and the micro behaviors collected for each frame are the inputs to these ML models. These collected data are sampled every five seconds and labeled into one of the three categories of driving behavior. We observe that the ensemble learning-based models are showing better results because they aggregate results of individual weak classifiers based on different strategies, and our logical reasoning-based method outperforms the ML baseline models.

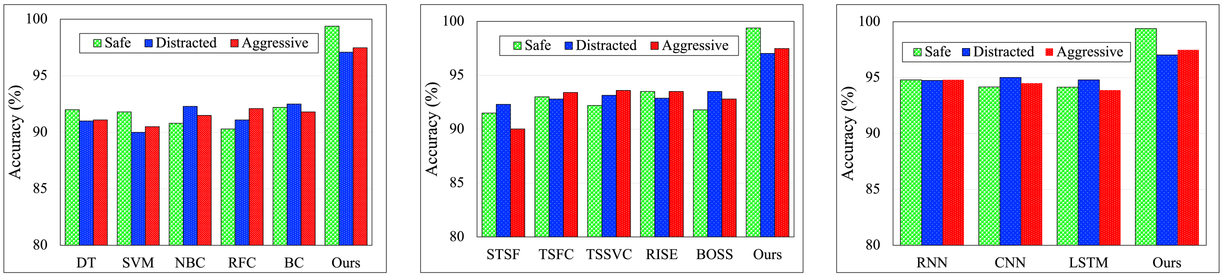
2) Comparison with time-series baseline models

To show the effectiveness of our proposed roadside camera and in-vehicular framework, we compare them with several baselines that work with multivariate time series classification using sktime library [25] as shown in Fig. 3(b). We train the following time-series classifier models, Supervised Time Series Forest Classifier (STSF), Time Series Forest Classifier (TSFC), Time Series Support Vector Classifier (TSSVC), Random Interval Spectral Ensemble (RISE), Ensemble of Bag of Symbolic Fourier Approximation Symbols (BOSS). As shown in Fig 3 (b), our proposed roadside camera framework achieves better accuracy than the above-mentioned time-series models.

3) Comparison with state-of-the-art Deep Learning (DL) baseline models

We compare our proposed framework with three other existing driving behavior classification models proposed in [26]–[28], and the performance results are shown in Fig 3(c). The authors in [26] have used a simple Recurrent Neural Network (RNN) to detect seven types of driving events, using data collected from an accelerometer sensor that can be found in an Android smartphone. Reference [28] is a 2D CNN model for analyzing driver behavior based on vehicle signals during driving. The recurrence plot technique converts the driving signals, including acceleration, gravity, throttle, speed, and RPM, to the images, which are then fed to a CNN to classify the image into five driving styles. In [27], the authors have

[1] https://github.com/pavana27/Driving-Behavior.git



(a) (b) (c)

Fig. 3: Comparison of Accuracy of proposed Roadside camera framework to (a) ML classifier models, (b) Time-series classifier models, and (c) State-of-art DL models.

classified driving behaviors through smartphone raw sensor data which are used as input to stacked Long Short Term Memory (LSTMs).

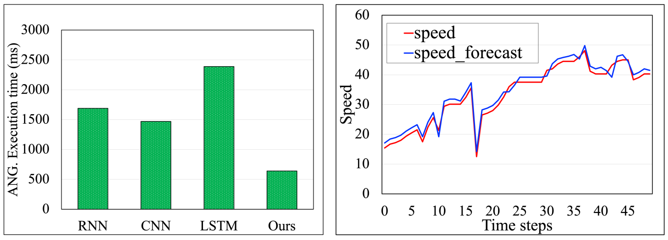
It can be observed from the results in Fig. 3(a) and (c) that DL-based approaches work better than classical ML approaches since they are capable of handling sequential data and capturing temporal dependencies existing in data. In contrast, our proposed model, which combines DL and a spatiotemporal logical reasoning-based approach, captures driving-related μBs with the highest accuracy compared to ML, DL, and time-series models. The accuracy of the in-vehicular framework is nearly identical to that of the roadside camera framework, as shown in Table IV compared to ML, DL, and time-series models. And therefore, these results are omitted from the paper.

4) Running Time Comparison

Fig. 4(a) compares the inference time of our framework against DL-based methods considered in section III-E3. For our framework, the reported time includes YLLO-based object detection, estimation of their kinematic parameters, and reasoning to determine μBs and final classification. For DL models, it is inference time for classification. It is seen that our inference time is the lowest. Though not reported, our mechanism would not incur additional training time beyond the image recognition tasks. The time required by DL models to classify driving behavior is proportional to the number of layers and the size of the model. Since the DL models must learn the μBs, they must be deeper. However, our proposed framework uses DL solely for object detection and does not need to learn the μBs, thereby maintaining a decent inference time to characterize the driving behavior. The average time taken by our framework depends on how complex the driving behavior is and to recognize the safe, distracted, and aggressive driving behavior is 580, 610, and 720 ms, respectively.

5) Prediction of Microbehaviors

As per Section II-A, the classification of driving behavior in this study is a combination of μBs that comprise road traffic parameters such as speed, distance to the car in front, braking acceleration, etc. In the context of the Advanced Traffic Management System (ATMS), precisely predicting these traffic parameters becomes increasingly crucial. Using forecasted data, such as a reduction in speed on the route ahead, travelers can reroute to save time. Many approaches adopt a black-box approach to traffic prediction, e.g., principal component analysis-based techniques and neural network-based techniques [29].



(a) (b)

Fig. 4: (a) Average Running Time Comparison, (b)VAR results - Forecast vs. Actual.

We utilize a classic statistical model, Vector Auto Regression (VAR) [30], to predict the aforementioned μBs and traffic parameters. The VAR model can capture dynamic interactions between multiple interrelated time series data and is used to predict the parameters based on their past values and the parameters collected along with them. In this study, the VAR is trained on the most recent measurements and then periodically retrained in the same manner, i,e., at any time *t*, the model is trained based on the past observations between *t - W* and *t - 1* where *W* is the training window length. *W* is determined heuristically through cross-validation using the training dataset, which is 30 seconds of the training window. This training window length minimizes the Mean-Absolute-Percentage-Error (MAPE) of point predictions, the cost function used to evaluate the predictors' estimation. Figure. 4 (b) shows the plot of actual vs. forecast speed values with a MAPE of 3.8% where the x-axis shows the time steps and the y-axis shows the speed values given in m/s. The trained VAR model provides a better estimate of future μBs and related traffic parameters; however, due to space constraints, the remaining results of the trained VAR model are omitted from this paper.

6) Contribution of Microbehaviors

Table VI: Impact μBs on Driving behavior

Micro Behaviors	**Aggr.**	**Distr.**
μ_{wv}: Weaving	0.8	0.71
μ_{ss}: Sudden Steer	0.71	N/A
μ_{hb}: Hard Braking	0.81	N/A
μ_{ld}: Lane Drift	0.72	0.75
μ_{st}: Straddling	0.85	0.72
μ_{os}: Overspeeding	0.76	N/A
μ_{tg}: Tailgating	0.71	0.84

Since the contribution of various μBs to the overall driving classification varies, their improvement will also have varying impacts. To evaluate this sensitivity, we considered a scenario where the drivers are asked to change a specific μB associated with aggressive and distracted driving, and they comply 50% of the time. Fig. VI reports the relative number of instances where the driving behavior was aggressive or distracted with and without the feedback. It is seen that keeping to the right or left side of the lane is the most significant μB for aggressive driving, but others are not far behind. For distracted driving, we see that driving too close to the next vehicle is the dominant behavior, but others are also significant.

Each row in this table represents the situation where the driver is informed about its undesired μB, and as a result, the driver reduces that behavior by 50%. (This is done one at a time for each row, not cumulatively). The columns show the aggressive and distracted driving occurrences, including the μB. The reported quantity is the ratio of the instances after and before the feedback and thus shows the impact of the feedback. The Not Applicable (N/A) reference indicates that the corresponding μB is not considered when classifying a driving behavior as aggressive or distracted. The table demonstrates that the most effective μB improvements for aggressive and distracted driving are speed control and maintaining the center of the lane. Thus, such an analysis can tell us the most effective feedback for manual or automated vehicles.

IV. Conclusions and Discussion

In this paper, we demonstrate a proof of concept for characterizing vehicular behavior using only the roadside cameras of the ITS system. The essential advantage of this method is that it can be implemented in the roadside infrastructure transparently and inexpensively and can have a global view of each vehicle's behavior without any involvement of or awareness by the individual vehicles or driving. The monitoring can be used for advising or alerting the driver and for improving the signage on the road that warns the drivers or is used the change the requirements (e.g., speed limit, lane following requirements, traffic control options, etc.).

The work presented here can be extended in three ways: (a) Fusing views from multiple cameras to cover areas that cannot be covered well by a single camera and tracking each vehicle individually through its features (color, shape, size, etc.), (b) Determining what feedback to provide to drivers, how, and how often to avoid distraction and enhance compliance, and (c) Considering more general environments, such as those with more than two lanes, bi-directional traffic, traffic signals, sharp curves, poor road conditions, etc.

References

[1] "Car crash statistics," Website, https://www.bankrate.com/insurance/car/car-crash-statistics/.

[2] T. Stewart, "Overview of motor vehicle crashes in 2020," white paper, 2022.

[3] Y. Brück *et al.*, "Investigation of personality traits and driving styles for individualization of autonomous vehicles," in *IHSI*. Springer, 2021.

[4] R. Phinnemore *et al.*, "Happy driver: Investigating effect of mood on preferred style of driving in self-driving cars," in *HAI*, 2021, pp. 139–147.

[5] C. M. Martinez *et al.*, "Driving style recognition for IVs control and ADAS: A survey," *IEEE T-ITS*, vol. 19, no. 3, pp. 666–676, 2017.

[6] M. Ramzan *et al.*, "A survey on state-of-the-art drowsiness detection techniques," *IEEE Access*, vol. 7, pp. 61 904–61 919, 2019.

[7] D. A. Johnson *et al.*, "Driving style recognition using a smartphone as a sensor platform," in *IEEE-ITS*, 2011, pp. 1609–1615.

[8] S. K. Kwon *et al.*, "Driving behavior classification and sharing system using cnn-lstm and v2x comm." *Applied Sciences*, vol. 11, no. 21, 2021.

[9] W. Rongben *et al.*, "Monitoring mouth movement for driver fatigue or distraction with one camera," in *IEEE-ITS*, 2004, pp. 314–319.

[10] M. S. Devi *et al.*, "Driver fatigue detection based on eye tracking," in *ICETET*, 2008, pp. 649–652.

[11] P. R. Tabrizi *et al.*, "Drowsiness detection based on brightness and numeral features of eye image," in *IIH-MSP*, 2009, pp. 1310–1313.

[12] "The visual detection of dwi motorists." https://www.nhtsa.gov/staticfiles/nti/pdf/808677.pdf.

[13] P. Pradeep *et al.*, "Resource efficient edge computing infrastructure for video surveillance," *IEEE TSC*, March 2021.

[14] C. Li *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[15] L. De Moura *et al.*, "Satisfiability modulo theories: Introduction and applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, 2011.

[16] A. Artikis *et al.*, "An event calculus for event recognition," *IEEE TKDE*, vol. 27, no. 4, pp. 895–908, 2014.

[17] C. Ansótegui *et al.*, "A new algorithm for weighted partial maxsat," in *Proceedings of the AAAI Conference on AI*, vol. 24, no. 1, 2010.

[18] M. Saifuzzaman *et al.*, "Incorporating human-factors in car-following models: a review," *TR-C*, vol. 48, pp. 379–403, 2014.

[19] S. Maerivoet *et al.*, "Cellular automata models of road traffic," *Physics reports*, vol. 419, no. 1, pp. 1–64, 2005.

[20] "Picarx AI Driven Car," https://docs.sunfounder.com/projects/picar-x/en/stable/, Accessed: 01 October 2021.

[21] S. Wolfram *et al.*, "A new kind of science," *Appl. Mech. Rev.*, vol. 56, no. 2, pp. B18–B19, 2003.

[22] P. P. Kumar *et al.*, "C-FAR: A compositional framework for anomaly resolution in intelligent transportation systems," *IEEE T-ITS*, 2022.

[23] I. Blecic *et al.*, "A generalized rapid development environment for CA based simulations," in *ICCA*. Springer, 2004, pp. 851–860.

[24] "Auto annotation tool," https://github.com/AlvaroCavalcante/auto_annotate, accessed: 20 August 2022.

[25] M. Löning *et al.*, "sktime: A unified interface for machine learning with time series," *arXiv preprint arXiv:1909.07872*, 2019.

[26] E. Carvalho *et al.*, "Exploiting the use of rnns for driver behavior profiling," in *IJCNN*, 2017, pp. 3016–3021.

[27] K. Saleh *et al.*, "Driving behavior classification based on sensor data fusion using lstms," in *IEEE ITSC*, 2017, pp. 1–6.

[28] M. Shahverdy *et al.*, "Driver behavior detection and classification using deep cnns," *Expert Systems with Applications*, vol. 149, p. 113240, 2020.

[29] P. Duan *et al.*, "A unified spatio-temporal model for short-term traffic flow prediction," *IEEE Transactions ITS*, vol. 20, no. 9, pp. 3212–3223, 2018.

[30] J. H. Stock *et al.*, "Vector autoregressions," *Journal of Economic perspectives*, vol. 15, no. 4, pp. 101–115, 2001.