

DrivAid: Augmenting Driving Analytics with Multi-Modal Information

Bozhao Qi, Peng Liu, Tao Ji, Wei Zhao, Suman Banerjee

Department of Computer Sciences, University of Wisconsin-Madison, USA

{bqi2, tji24}@wisc.edu, {pengliu, wzhaoh, suman}@cs.wisc.edu

Abstract—The way people drive vehicles has a great impact on traffic safety, fuel consumption, and passenger experience. Many research and commercial efforts today have primarily leveraged the Inertial Measurement Unit (IMU) to characterize, profile, and understand how well people drive their vehicles. In this paper, we observe that such IMU data alone cannot always reveal a driver’s context and therefore does not provide a comprehensive understanding of a driver’s actions. We believe that an audio-visual infrastructure, with cameras and microphones, can be well leveraged to augment IMU data to reveal driver context and improve analytics. For instance, such an audio-visual system can easily discern whether a hard braking incident, as detected by an accelerometer, is the result of inattentive driving (e.g., a distracted driver) or evidence of alertness (e.g., a driver avoids a deer).

The focus of this work has been to design a relatively low-cost audio-visual infrastructure through which it is practical to gather such context information from various sensors and to develop a comprehensive understanding of why a particular driver may have taken different actions. In particular, we build a system called DrivAid, that collects and analyzes visual and audio signals in real time with computer vision techniques on a vehicle-based edge computing platform, to complement the signals from traditional motion sensors. Driver privacy is preserved since the audio-visual data is mainly processed locally. We implement DrivAid on a low-cost embedded computer with GPU and high-performance deep learning inference support. In total, we have collected more than 1550 miles of driving data from multiple vehicles to build and test our system. The evaluation results show that DrivAid is able to process video streams from 4 cameras at a rate of 10 frames per second. DrivAid can achieve an average of 90% event detection accuracy and provide reasonable evaluation feedbacks to users in real time. With the efficient design, for a single trip, only around 36% of audio-visual data needs to be analyzed on average.

I. INTRODUCTION

Due to its many implications, monitoring and evaluating driver’s behavior has always been a topic of great interest. Many commercial offerings (such as from Cambridge Mobile Telematics [1]) and research efforts have built systems that attempt to monitor and evaluate driving behaviors by analyzing information from On-Board Diagnostic (OBD) ports as well as motion sensor data (either from smartphones or other custom devices with Inertial Measurement Units (IMU) sensors). With the proper analysis, the data from these motion sensors can provide useful information about various activities within the drive itself, e.g., hard brakes, sudden lane changes, or fast acceleration [2]–[4].

While IMU sensors can provide accurate analytics on *what* happened during a drive, e.g., hard brakes, sudden lane

changes, etc., it should be apparent that such data alone does not answer *why* the driver acted in that manner. In particular, data from IMU sensors do not have sufficient contextual information to indicate whether such an action by the driver was good or bad. As shown in Figure 1, a hard brake could be due to driver distraction (a bad driving action), or it could be to avoid a deer that suddenly jumped to the front of the vehicle (a good driving action).

Introducing DrivAid: We believe that to better understand driving behaviors one can effectively leverage audio-visual cues, using a few vehicle-mounted cameras and microphones, to complement the existing use of IMU sensors. In particular, we build a low-cost and portable system called DrivAid that utilizes audio-visual sensors in the vehicle, processes such data feed locally and provides useful evaluation feedbacks to the driver both in real-time and offline.

Analyzing high resolution audio-visual data is often delegated to high-end GPU-enhanced compute clusters located in data centers. However, in our scenario, it is likely that the vehicles equipped with audio-visual sensors can easily generate a high volume of data rather quickly making the offload process either slow or expensive, or both. This implies that the audio-visual analytics should need to be performed locally and in real-time — if the analytics is any slower, then the data will continuously get backlogged and ultimately becoming stale at some point.

To address abovementioned challenges and support efficient audio-visual analytics, we experimented with in-vehicle GPU-enhanced computing platforms on which we deployed the analytics module. Compared to cloud computing, edge computing provides lower latency, greater responsiveness, and more efficient use of network bandwidth. *To avoid unnecessary resource-intensive image processing tasks and reduce computing workload of the whole system, we use smartphone motion sensors to detect different driving events and only conduct further analysis once an event is detected.* Figure 2 shows the workflow of DrivAid. DrivAid uses data from IMU and GPS sensors to detect driving events. When an event is detected, DrivAid fetches video and audio clips from the buffers to analyze. All the audio and video data are processed locally in the vehicle to preserve privacy. By leveraging various sensor data and implementing an analysis pipeline that leverages deep neural networks, we are able to offer useful insights of driving events and generate analysis results in real time. We believe the proposed architecture



Fig. 1: Possible causes of a hard brake.

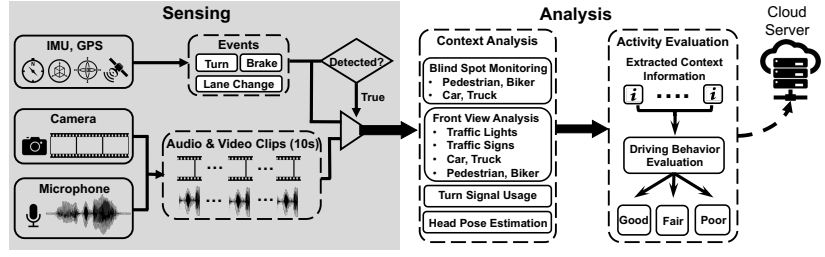


Fig. 2: A high level overview of DrivAid.

of DrivAid can be particularly useful in analyzing driver behaviors, and could be an essential part of a driver analytics subsystem.

Usefulness of DrivAid: The concept of using audio-visual cues in improved situational awareness around vehicles is most famously used in autonomous vehicle industries (along with a host of other sensors). However, unlike the high resolution and very high accuracy needs of autonomous vehicles in making driving decisions, the DrivAid system requires far lower resolution and accuracy, resulting in a significantly lower cost. Today’s autonomous vehicles install high-end sensors which aggregate a price of $\sim \$100,000$, while our system utilizes infrastructure with an aggregate cost of less than $\$500$. DrivAid delivers significant benefit for driving analytics at a lower cost which could benefit a wide range of applications. First, with proper incentives, insurance companies may ask their customers to install extra hardware (e.g. camera) on their vehicle, and evaluate their driving skills with a system like DrivAid. Such a camera-based system could offer more information when an accident happens. Second, many fleet operators, e.g., public transit systems, school buses, freight trucks, are often interested in understanding the behaviors of their drivers. DrivAid could identify unsafe habits and help fleet managers educate their drivers to take appropriate actions. Third, if certain drivers are known to be “experts,” then the annotated actions of such drivers could also be used to provide useful inputs to the design of autonomous vehicles. Moreover, with adjustments, DrivAid could be used to verify whether the autonomous vehicle can handle unusual or rare traffic events.

Contribution: The contributions of this work can be summarized as follow: i) We illustrate how a real-time low-cost sensing and analyzing system could be built that leverages audio-visual cues to augment driving behavior analysis based on IMU sensors in a holistic manner. ii) We build a lightweight, powerful system that can be easily deployed in regular vehicles by adapting and integrating existing algorithms to let them run efficiently in vehicular settings. iii) We evaluate our system with more than 1500 miles’ drive data. A prototype is deployed on a regular vehicle and evaluated through test drives of around 50 miles in real world environments. The evaluation results show that DrivAid can provide useful and reliable insights.

II. SYSTEM OVERVIEW

A. Design Consideration

In order to get a comprehensive understanding of a driver’s performance, a major challenge is to acquire enough context information, e.g. driver’s actions, and the surrounding traffic information. Current solutions, like XSense [5], V-Sense [2], can accurately detect various normal and abnormal events using data collected from motion sensors or OBD port, but fail to provide surrounding context information. To find the missing information, DrivAid buffers the most recent 10 seconds audio and video data from microphones and cameras for analysis. When an event of interest happens, DrivAid analyzes the data with computer vision techniques to get driving actions that cannot be detected by IMU and GPS sensors, including turn signal usage, mirror checking, blind spot checking, and so on. By combining information from various sources, DrivAid can provide more informative driver analytics and offer deeper insights into each detected events.

B. Our Solution

Figure 2 shows a high-level overview of the DrivAid design. DrivAid uses three types of sensors to collect data: i) IMU sensors and GPS of a phone, which can be used to detect driving events; ii) the rear camera of a phone to capture traffic ahead; iii) two cameras facing the blind spots on both sides, and one camera facing the driver. Timestamped sensor data, audio, and video are analyzed on an embedded computer deployed in the vehicle. We use an Android phone to record data from the accelerometer, gyroscope, magnetometer and stream them to the embedded computer for context analysis. Extracted context information is sent to an activity evaluation module, through which the detected event will be evaluated and categorized into different ratings.

IMU-based solutions can accurately detect various types of driving events with low power consumption. On the other hand, audio and video data analysis with deep neural networks are usually computationally expensive. Given the fact that a driver should have minimal operations when driving straight or there is not too much surrounding traffic. Hence, it is less important to monitor the driver during the whole trip, we only need to focus on how well the driver performs when a driving activity is detected. Following this principle, DrivAid uses motion sensors to continuously monitor various driving events, and analyze corresponding audio and video clips once an event is detected. Doing this can significantly reduce computing workload and power consumption of the whole system.

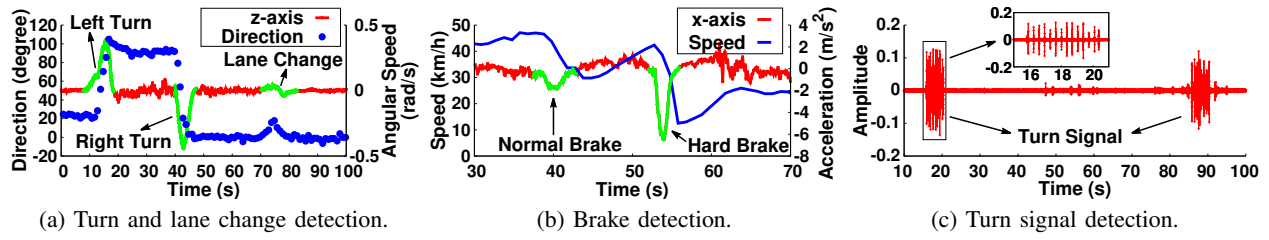


Fig. 3: The detection of driving activities.

III. EVENT DETECTION AND CONTEXT ANALYSIS

The DrivAid system detects three types of driving activities (turn, hard brake, and lane change) with the sensors on the smartphone. Once a driving activity is detected, DrivAid starts extracting context information from the video and audio clips collected by the IP cameras. Data from different sensors are not synchronized due to buffering (video and audio) and different system clocks on those devices. We leverage the recorded turn signal sounds to synchronize the data from multiple devices. A pattern matching algorithm is used to find the time difference between multiple signal traces.

A. Driving Event Detection

1) *Coordinate Alignment*: Smartphone-based mechanisms have been developed to detect various driving behaviors [2], [4], [6]. For instance, the vehicle's lateral dynamics can be obtained from the gyroscope sensor in the smartphone when a phone is aligned with the vehicle. DrivAid uses similar techniques as mentioned in [2], [5] for coordinate alignment.

2) *Turn and Lane Change Detection*: We detect turning and lane change events by using the gyroscope sensor and the GPS. Since the smartphone's coordinates are aligned with that of the car, the z-axis readings of the gyroscope sensor reflect lateral movements of the vehicle. Figure 3(a) illustrates the detection of left and right turns as well as a left lane change using the gyroscope sensor. Turning angles can be estimated by integrating the angular speed over time. A positive turning angle refers to a left turn while a negative turning angle represents a right turn. In addition to the gyroscope data, we monitor the vehicle heading changes with GPS coordinates. The vehicle turning events can be extracted by monitoring the changes in the traveling direction. This method could be a complementary method for turning detection. In Figure 3(a), the blue dots represent the traveling directions calculated from the GPS coordinates. Different angles stand for different directions. For instance, 0° corresponds to East, 90° indicates North, 180° denotes West, -90° represents South. From the figure, we can distinguish the turning events from the changes of direction angle. Using similar techniques, we can detect lane change events using both device orientation changes and direction angle changes. To differentiate between lane changes and driving on curvy roads, we implemented the same technique proposed in V-Sense [2].

3) *Brake & Stop Detection*: We use accelerometer readings as well as vehicle speed information to detect brakes and stops. Figure 3(b) illustrates the detection of normal and hard brakes. Brake events can be extracted from bumps in accelerometer

readings and decreasing in vehicle speed. A hard brake is defined as any condition when the vehicle decelerates faster than 7 mph (Miles per hour) per second. Stop events can be inferred from the vehicle speed.

B. Driver Behavior and Context Analysis

DrivAid analyzes the audio and video signal with signal processing and computer vision techniques to get driver's behaviors and also the context information.

1) *Driver Behavior Analysis*: The turn signal is a vital safety feature used to notify other drivers when making turns or switching lanes. DrivAid detects turn signal usages by analyzing turn signal sounds. We apply bandpass filters to filter out human speech as well as other background noises. Then a pattern matching technique is used to identify turn signals. Figure 3(c) shows examples of extracted turn signals from a camera's built-in microphone.

Drivers use mirrors to get a picture of their surrounding traffic, and check the blind spots when they need to make a turn or change the lane. DrivAid detects a driver's head pose to infer the driver's mirror and blind spot checking behaviors. We classify head poses into six categories, they are left and right wing mirror check, left and right blind spot check, rearview mirror check, and front view check. Figure 4d shows six different head poses of a driver. In Figure 4d (1), the driver is focusing on the front view. Figure 4d (2) and (5) show the driver is checking left and right wing mirrors correspondingly. The driver is checking the rearview mirror in Figure 4d (4). The driver is making left and right shoulder checks in Figure 4d (3) and (6). These six scenarios can be distinguished by checking head movements with respects to the yaw axis. The estimated rotation angles around the yaw axis of Figure 4d (1), (2), (4), (5) are -3.34° , -40.20° , 18.21° , 49.70° . As shown in Figure 4d (3) and (6), due to lack of facial features, it is hard to estimate head pose when the driver is checking right and left blind spots. To solve this issue, we track the rotation angle in a duration. If the angle keeps increasing and then disappears, then the driver is doing a right blind spot check, and vice versa. To calculate the head pose in 3D space, we need to know several locations of the face (eyes, nose, mouth, etc.) in 2D space, and also the 3D coordinates of those facial features in world coordinates. We assume the camera is calibrated so we are aware of the intrinsic parameters of the camera. DrivAid uses Dlib's facial landmark detector [7] to extract key points on a face from the captured image. And the solvePnP function of OpenCV [8] is used to get the head pose.

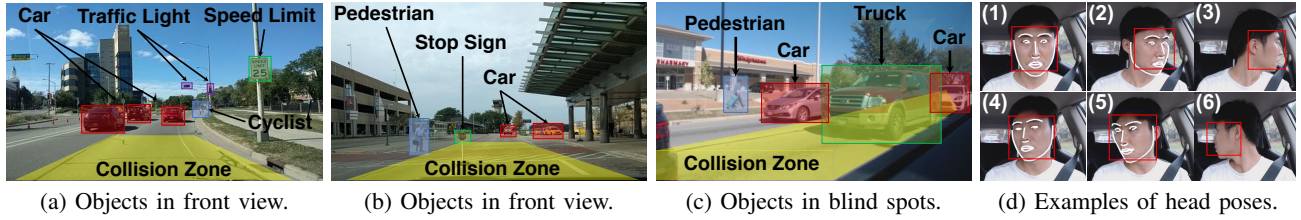


Fig. 4: Extracting context information using vision-based techniques.

TABLE I: Object Detection

Vehicle & Person	Traffic Sign	Speed Limit
Car	Stop Sign	25 mph
Truck	Red Traffic Light	30 mph
Pedestrian	Yellow Traffic Light	40 mph
Biker	Green Traffic Light	60 mph

2) *Context Analysis*: We analyze the audio and video signal to get the context of driving events. Object detection is the backbone of DrivAid and the choice of a right detection model should balance the run-time efficiency and accuracy. We choose DetectNet [9] as the object detection model since it can deal with input image of varying sizes and provide reliable accuracies. And it can be easily accelerated by the optimized inference engine—Nvidia TensorRT [10].

Most intersections are controlled by traffic signs, and it is important to follow the traffic rules at an intersection. DrivAid can detect a list of objects appeared in video frames captured in the front view camera, with a special focus on common objects at intersections. The front view object detection model could recognize vehicles, pedestrians, traffic lights, stop signs and speed limits. We train a DetectNet model using two public traffic image datasets [11], [12]. Figure 4a and 4b show the detected objects at intersections with traffic lights and stop signs. We define a collision zone (yellow shaded region shown in figures) in each front view frame, objects in the collision zone will be considered as “threats” to the driver. In general, the area of the collision zone is fixed when the vehicle travels under a speed threshold. The size will increase if the vehicle travels faster than the speed threshold, the higher vehicle speed, the larger the collision zone. The speed threshold and the size of the collision zone are defined based on empirical experiments. If there is an object exists in the collision zone, the driver might not have enough time to avoid hitting that object. Note that traffic signs, such as stop signs, speed limits, will always be considered by the evaluation module.

Any time before changing or merging lanes, a driver needs to check if there is any object in blind spots. DrivAid monitors three most common types of objects that could appear in blind spot areas, they are vehicles (e.g. cars, trucks, etc.), pedestrians and bikers. As shown in Figure 4c, a pedestrian, a truck and two vehicles are detected in this frame. Similarly, we labeled a collision region to check whether there is anything in the blind spot area. The driver can make a safe lane change when the collision region is empty.

As mentioned in the previous section, DrivAid can detect four types of speed limits from the front view camera. DrivAid queries vehicle speeds from GPS every second. Once a speed

limit sign is detected from the video frame, it will be compared with the speed from GPS. When risky events are detected, we check whether those events are caused by speeding. Besides, we can also infer if the driver has the speeding habit.

IV. DRIVING ACTIVITY EVALUATION

By combining the context information with the activity detection results with IMU and GPS, we can infer whether the driver is doing right when events occur.

A. Data Fusion

It is always hard to reveal the whole picture on the road using information acquired from a single sensor. Hence, DrivAid combines information acquired from multiple sensors to develop a deeper understanding. For instance, once a turning or lane change event is detected, the system will check if the driver follows correct rules before making the turn or lane change, e.g., using turn signals, checking wing mirror and blind spot, etc. If the detected turn or lane change is accompanied by a turn signal and a blind spot check, then the driver behaved properly. Otherwise, further analysis may be required to judge the rationality of the driver’s behavior.

B. Comprehensive Driving Activity Evaluation

We implement a decision tree for driving behavior analysis. In this work, we mainly focus on evaluating three types of driving activities: turn, lane change, and hard brake. DrivAid caches the most recent 10 seconds of video and audio data. Once an activity is detected, DrivAid extracts context information from the 10-second data clips and use them to evaluate the driving activity. For each detected activity, we take different factors into consideration. Figure 5 shows the workflow of the analysis process. We use a decision tree to rate the driver’s performance during an activity. Items in the red dashed boxes are the factors we take into consideration for each event. These factors serve as the inputs of the decision tree. The outputs of the decision tree are the classified categories of each event. For example, an aggressive turn is confirmed if the driver is speeding when making the turn. A conservative lane change is determined only when the maneuver duration is too long while the other operations are well done. If a hard brake is caused by the sudden appearance of a pedestrian, it will be classified as a reasonable one. To build the classification decision tree, we start with listing out all the possible conditions and cases for each event. Then, find out all the possible nodes, and choose the most important node as the starting point. The classification decision tree eventually

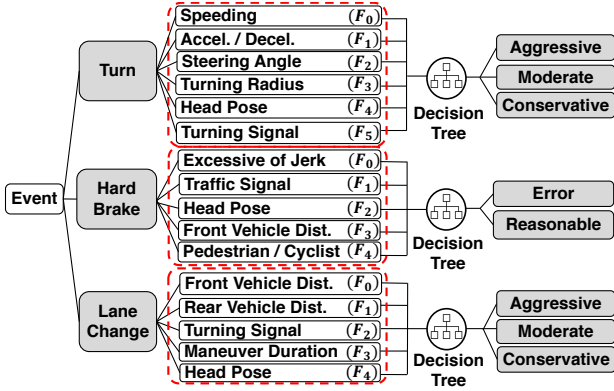


Fig. 5: The structure of decision tree evaluation model.

grows upon the start node. Lastly, we make sure every possible condition can be covered by the tree.

Here are the details of key nodes for each event. For the turning event, F_0 represents the difference between vehicle speed and speed limit. F_1 checks the smoothness when making the turn. F_2 and F_3 measure the level of the turn. F_4 and F_5 are binary values showing whether the driver does mirror and blind spot check, as well as the turn signal usage. For the hard brake event, F_0 describes the smoothness of the brake. F_1 to F_4 are all binary values. F_1 , F_3 , and F_4 represent the appearances of traffic signals, the vehicle in front collision region and pedestrians correspondingly. F_2 describes whether the driver's attention is distracted or not. For lane change events, F_0 and F_1 represent the front and rear vehicle distances in the next lane. F_3 describes how long does it take to finish the lane change. We divide the distance into three categories, far, moderate, and close. F_2 and F_4 are binary values which describe the usage of the turn signal and whether the driver checks wing mirrors and blind spots.

V. SYSTEM IMPLEMENTATION

Supporting real-time data processing is a big challenge for DrivAid. To achieve this goal, we carefully choose the embedded computing platform and use the device-specific inference engine to execute the trained neural networks. The core component of DrivAid is the analysis module which runs in the Nvidia Jetson TX2 embedded computer. Jetson TX2 is an embedded system-on-module (SoM) with a hex-core ARMv8 64-bit CPU complex and a 256-core Pascal GPU [13]. We install JetPack 3.2 with L4T R28.2 on Jetson TX2. JetPack 3.2 bundles all the Jetson platform software, including TensorRT 3.0, CUDA 9.0, GStreamer, OpenCV and so on.

We implement DrivAid using GStreamer framework [14] in C and C++. GStreamer pipelines are used to manage each individual task, e.g., we create a pipeline for head pose estimation, and use another pipeline to monitor turn signal usage. A GStreamer element is a procedure through which the audio-video stream is processed. A GStreamer pipeline consists of a chain of elements. Audio-video streams flow through the pipelines and are processed in

a synchronized manner. To implement the context analysis module, we develop a GStreamer plugin including two GStreamer elements for head pose estimation and object detection. The head pose estimation element is implemented with Dlib. The object detection element loads a specific pre-trained DetectNet model and uses the model to recognize objects in input images. The object detection models are trained using the Nvidia Deep Learning GPU Training System (DIGITS) [15]. DIGITS is a wrapper for multiple deep learning frameworks including Caffe, Torch, and TensorFlow. We use Caffe to train our detection model, and copy the trained model snapshot to Jetson for inference applications. Our object detection models are trained on a Desktop with a Nvidia GeForce GTX 1060 GPU. Before using the trained model on Jetson TX2 for inference applications, we need to parse it and perform device-specific profiling and optimizations for the target deployment GPU. We use the TensorRT to perform such kind of optimization tasks.

In total, we have five pipelines to extract useful context information from audio and visual sensors. Here is a list of the pipelines we used:

Pipeline 1: Detecting vehicles, people, traffic signs and speed limits from the front view camera.

Pipeline 2 & 3: Detecting vehicles and people from left and right blind spot cameras.

Pipeline 4: Estimating head poses from the face camera.

Pipeline 5: Monitoring turn signal usage from audio streams. Pipeline 1 has two branches running for object detection in front view. These two branches load the Vehicle & Person and the Traffic Sign inference models correspondingly.

VI. EVALUATION

We deploy our hardware in vehicles and evaluate DrivAid under real-world conditions. In this section, we provide the details of our experiment settings, detection accuracies, and system resource consumption during the real-time process.

A. Experiment Setup

We deploy the hardware in two vehicles (Honda CRV and Nissan Rogue) and test DrivAid in real-world environments. We develop a custom Android application that runs on a Google Nexus 5X for IMU and GPS sensor data collection as well as video capturing in our prototype system. The data sampling rate is set to 10Hz for motion sensors and 1Hz for GPS. The phone captures video from the rear camera with a resolution of 640×480 @ 10 frames per second (FPS) 1 Mbps bitrate. The sensor and video data are streamed to the Jetson TX2 using TCP protocol through USB tethering service. We have three Logitech C920 video cameras capturing videos from the left and right blind spots and the driver. The video stream settings are the same as what we used for capturing video from the phone camera. Figure 6 shows the hardware components of DrivAid. Two cameras are mounted near wing mirrors facing the right and left blind spots. One camera facing the driver is placed on the dashboard behind the steering wheel. The phone is mounted on the center dashboard. Since

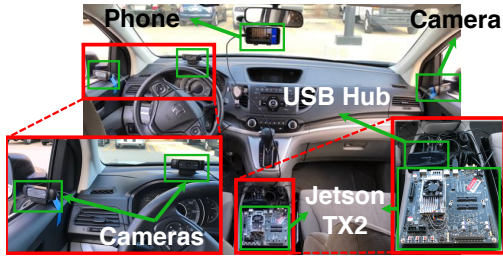


Fig. 6: The hardware components of DrivAid.

there is only one USB 3.0 port on Jetson TX2, we use a USB hub to connect cameras and phone. The Jetson TX2 (maximum power consumption is about 7.5 Watt) is powered by the cigarette lighter. In total, we have collected around 1500 miles of data from multiple drivers¹. Part of the data is collected by our partner [16]. We use the data to train and validate our models.

To evaluate DrivAid, we randomly select 10% of the original data. We mainly focus on analyzing the driving events and driver performance in urban areas since drivers tend to face more complex scenarios in urban environments than those on highways. To get the ground truth, we recruit three volunteers to find out all the events from the recorded data. For each detected event, we ask volunteers to identify driver's actions and surrounding traffic information from the corresponding video clips. Three volunteers should reach a consistent decision for each event, driver actions, and traffic information. For example, if volunteers recognize a lane change event from sensor data. Then, they will watch the corresponding video clip (captured from the front view camera) to double check this event. Further, they figure out driver's actions (e.g. mirror check, turn signal usage, etc.) and surrounding traffic information (e.g. any vehicle in blind spot area, the distance of the vehicle in front, etc.) from the video clips taken by all cameras. We created a separate thread to record the performance data of the hardware, including CPU/GPU usage and frequencies, as well as memory usage.

B. Driving Event Detection

In order to avoid unnecessary computation tasks, our system is designed to detect various driving events mainly using smartphone motion sensors. DrivAid buffers audio and video streams in the most recent 10 seconds, and only conducts evaluation tasks once an event is detected.

First, we evaluate the detection accuracy of sensor-based algorithms. We use smartphone motion sensors to detect four different driving actions, namely turn, lane change, brake, and stop. Table II includes the detection accuracies of these four driving events. Brake and stop are relatively easy to detect and the precision and recall of these two events are 1 and around 0.99 correspondingly. The errors are mainly caused by vibrations of the vehicle (e.g. uneven road surface) and GPS signal lost (e.g. an underground parking lot). The precision and recall are more than 0.9 for turn detection, the misjudgments

TABLE II: The accuracy of driving event detection.

Event	# of TPs ¹	# of FPs ¹	# of GTs ¹	PR ²	RC ²
Turn	113	7	125	0.94	0.90
Lane Change	67	27	82	0.71	0.80
Brake	306	0	311	1	0.98
Stop	89	0	90	1	0.99

¹ The number of ground truth (GT), true and false positives (TP, FP).

² Precision (PR) and Recall (RC).

are mainly caused by unexpected vibrations of the vehicle. We have a slightly lower precision and recall for lane change detection. Some drivers tend to make gradual lane changes and it is relatively hard to capture all of them using sensor data fusion techniques. Due to these factors, we achieve a precision of 0.71 for lane change events. We implement pattern recognition techniques for detecting these events. In order to make sure every event can be detected successfully, we loosen the constraints of matching algorithm when conducting the real-time evaluation. In other words, our system can detect almost every event, but there exist more false positives. To be more specific, the total number of detected events is 489, and around 3825 in seconds (a single event lasts for 8 seconds on average). The total driving time is around 13650 seconds, which means DrivAid only needs to analyze around 36% of the collected data (we extract and evaluate 10-second video-audio clips for each event).

C. Head Pose Estimation

In this experiment, we recruited 5 volunteers to evaluate the head pose estimation accuracy. As mentioned in Section III-B1, we define six states of head poses. Figure 7a (a) to (e) shows how estimated head poses (measured in degrees) change over time when the driver does right and left wing mirror check, right and left blind spot check and rearview mirror check. When the driver focuses on the front view, the estimated head pose is between $\pm 10^\circ$. When the driver turns his head to the right, the estimated head pose is a positive number and vice versa. We ask each volunteer to sit in the vehicle, pretending he or she is driving it. Volunteers are asked to check mirrors and blind spots based on our instruction every 10 seconds. Each round lasts for five minutes. Table III shows the estimation accuracy. Our estimation module can achieve a precision of 0.83 and a recall of 0.77. Most misjudgments happen on rear mirror check since some people turn their heads very slightly when checking the rearview mirror. We use mean average precision (mAP) to measure the accuracy of object detectors. It is the average of the maximum precisions at different recall values. Figure 7b reports the mAPs of Pipeline 4 running at different frame rates. The higher the frame rate, the better the performance. With a higher frame rate, the pipeline can capture more head pose changes, thus, the classification result would be more accurate.

D. Front View & Blind Spot Monitoring

We prepared more than 1500 video frames from recorded videos in the real-world environment and labeled every object appears in each frame. First, we test our front view inference

¹We have received IRB approval for this research project.

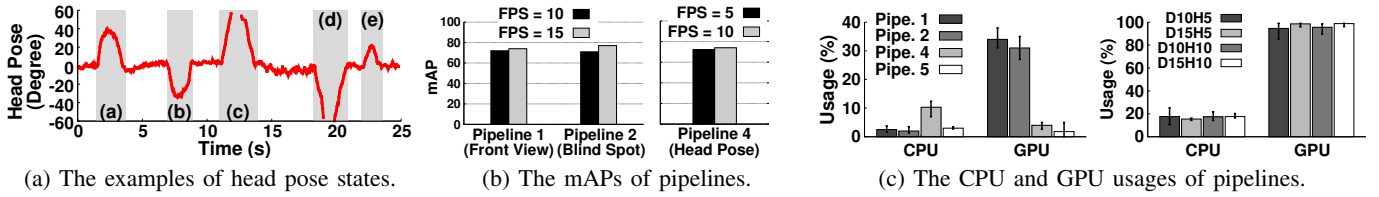


Fig. 7: Extracting context information using vision-based techniques.

TABLE III: The accuracy of object detection.

Event	# of TPs	# of FPs	# of GTs	PR	RC
Vehicle	5075	1904	7555	0.72	0.67
Pedestrian	1207	473	1942	0.71	0.62
Traffic Light	440	168	748	0.72	0.59
Speed Limit	427	138	721	0.75	0.59
Stop Sign	370	126	644	0.74	0.57
Turn Signal	162	17	173	0.91	0.93
Head Pose	116	23	150	0.83	0.77

model by comparing the inference results with ground truth. Table III reports the precision and recall of each object. We find the precision values for the vehicle, pedestrian, traffic light and speed limit are more than 0.7. Based on our observation, the light condition affects detection accuracy most. Although we have used a relatively large data set for training, improvements can be made by expanding image data set as well as changing the design of CNN. We leave this for future work. Next, we replay the recorded videos on a desktop, the video is displayed on a screen monitor. We start the front view pipeline and let the camera facing the screen. All the detected objects are plotted on the corresponding image frame and saved as a PNG file to a local disk. The total video length is around 20 minutes and we test the pipeline with frame rates of 10 and 15. The mAPs for front view pipeline (Pipeline 1) are shown in Figure 7b. We can see that the higher the frame rate, the higher the mAP value.

We use the same DNN model as front view pipeline for blind spot monitoring (Pipeline 2). Different from Pipeline 1, Pipeline 2 only uses the Vehicle & Person inference model since traffic signs and speed limit signs are less likely to appear in blind spot area. Therefore, we achieve a similar detection accuracy to Pipeline 1.

E. Audio Monitoring

In this experiment, we extract the audios from recorded video streams to test the turning signal detection pipeline. Table III summaries the detection accuracy. Both the precision and recall are greater than 0.9. Most false positives and negatives happen on uneven roads since real turn signal pulses may be buried under vehicle vibration noises. Our detection algorithm works well in most cases and further improvements can be done to remove unexpected noises. Figure 7c (left) and Table IV report the system usage details of the turning signal detection pipeline (Pipeline 5). Pipeline 5 consumes the least system resources compared to other pipelines.

F. System Resource Usage

In this experiment, we evaluate the overall performance of DrivAid. As mentioned in previous sections, data from various

TABLE IV: The memory usages of pipelines.

	Pipe. 1	Pipe. 2	Pipe. 4	Pipe. 5
Usage (Mb)	972	964	138	9
	D10H5	D10H10	D15H5	D15H10
Usage (Mb)	1102	1103	1105	1108

sources, including phone, three webcams, a microphone, are pushed to five pipelines. We test DrivAid under different frame rate settings.

System Usage of a Single Pipeline: System usage of each pipeline is summarized in Figure 7c (left) and Table IV. We measure the CPU, GPU, and memory usage when running each individual pipeline at 10 FPS. The main task of front view pipeline (Pipe. 1) is to detect various objects, it has a very low CPU usage (less than 5%) and a relatively high GPU usage (more than 30%). Table IV shows the memory usage is around 970 Mb for Pipe. 1. The blind spot monitoring pipeline (Pipe. 2) consumes less resource than front view monitoring pipeline since it only has one branch. Our head pose estimation pipeline mainly uses CPU for computing and GStreamer uses a little GPU for video preparation. The memory usage is around 100 Mb, which is much lower than object detection pipelines. Based on our experiment result, a single pipeline can be run with the highest FPS of 30.

Overall System Usage: System usage details are summarized in Figure 7c (right) and Table IV. "D10H5" means the FPSs of object detection pipelines are set to 10 and 5 for head pose pipeline. From the results, we can see that the memory usages improve a little bit compared to running a single object detection pipeline. GPU is fully occupied by DrivAid at 15 FPS and CPU usage is around 20% at 10 FPS. Based on evaluation results, DrivAid can support a max of 10 FPS for the head pose pipeline and 15 FPS for object detection pipelines simultaneously. Currently, the head pose pipeline is running on a single core of the hex-core CPU, which is why the CPU is not fully occupied. Future improvements can be done to fully leverage the hex-core CPU.

G. Overall Performance in Real World Environments

We deploy DrivAid on a real vehicle and drive the vehicle on roads for around 50 miles. We perform normal actions during test drives, including turns, lane changes, brakes, and stops. There are two volunteers sitting in the vehicle and rate every event happens during the drive. The volunteers should come up with a consistent decision for each event. Their judgments serve as the ground truth. In general, DrivAid can achieve an accuracy higher than 90% for context extraction, and the outputs of decision tree analysis module are consistent with the ground truth. Table V summarizes

TABLE V: The accuracy of lane change event.

Event	TPs	FPs	GTs	PR	RC
Front Vehicle Dist.	25	9	31	0.74	0.71
Rear Vehicle Dist.	24	6	27	0.8	0.75
Turning Signal	31	2	34	0.94	0.91
Maneuver Duration	39	2	41	0.95	0.92
Head Pose	37	5	39	0.88	0.90

the details of lane change event analysis results. Due to the space limit, we omit the analysis details of turn and hard brake. Figure 8(a) illustrates how DrivAid develops a comprehensive understanding of a hard brake. To evaluate this hard brake, DrivAid extracts useful context information from video clips collected from front view, face view and blind spot cameras. For instance, there is enough space between the two vehicles when the vehicle starts to decelerate (Figure 8(b)), driver's attention is distracted before the hard brake happens (Figure 8(c)), and so on. Using these factors as inputs, the decision tree classifies the hard brake into "Error" category, which is consistent with the ground truth.

VII. RELATED WORK

Transportation-related analytics has been an active research area, such as transfer feasibility at train station [17], human mobility analytics [18], driving behavior analysis [19]–[21], and so on. Existing solutions have been proposed to detect and evaluate various kinds of driving behaviors using embedded IMU sensors and cameras in common mobile devices, such as lane-changes, turns, and so on [2], [4]–[6], [22]. Insurance companies use similar techniques to evaluate driver's behavior and offer discounts for drivers' with good driving habits [23], [24]. DrivAid aims to be a more holistic platform that provides real-time situational awareness of a driver and his actions in the context of the vehicle's surrounding environment. Other than evaluating driving behaviors, existing work also focuses on discovering useful context information in vehicular settings. PreDriveID [25] tries to identify drivers using the minimal dataset from in-vehicle sensors. AVR [26] broadens the vehicle's visual horizons by enabling it to share visual information with other nearby vehicles. Our work could serve as a light-weight platform that can provide useful context information, including driver actions and surrounding traffic, and bring benefits to other research work.

VIII. CONCLUSION

This paper introduces DrivAid, a multi-modal and light-weight real-time sensing system that leverages audio-visual cues to augment driving behavior analysis. Our system overcomes the limitations of existing IMU-based solutions and can provide fruitful contextual information for driver behavior profiling. DrivAid leverages mobile sensing and computer vision techniques to extract various context information of the driver and the surrounding environments. Further driving behavior evaluation is conducted using the extracted information. We build a prototype of DrivAid on a low-cost embedded computer with deep learning inference accelerator. The prototype is deployed to a regular vehicle and tested in

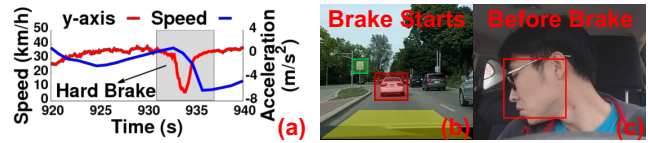


Fig. 8: Detailed analysis of a hard brake.

real-world environments. The evaluation results show that our system can process data in real time and provide a good understanding of each driving behavior. We believe such a real-time sensing and analysis system can enable a wide range of applications.

IX. ACKNOWLEDGMENTS

We would like to acknowledge the anonymous reviewers. This work is supported in part by the Wisconsin Alumni Research Foundation and the US National Science Foundation through awards CNS-1345293, CNS-14055667, CNS-1525586, CNS-1555426, CNS1629833, CNS-1647152, CNS-1719336, and CNS-1838733.

REFERENCES

- [1] DriveWell, <https://www.cmtelematics.com/>, 2017.
- [2] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible sensing of vehicle steering with smartphones," in *ACM MobiSys*, 2015, pp. 1–13.
- [3] G. Castignani, R. Frank, and T. Engel, "Driver behavior profiling using smartphones," in *ITSC*. IEEE, 2013, pp. 552–557.
- [4] C. Karatas, L. Liu, H. Li, J. Liu, Y. Wang *et al.*, "Leveraging wearables for steering and driver tracking," in *IEEE INFOCOM*, 2016.
- [5] L. Kang and S. Banerjee, "Practical driving analytics with smartphone sensors," in *VNC, 2017 IEEE*. IEEE, 2017, pp. 303–310.
- [6] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca *et al.*, "Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones," in *ACM MobiSys*, 2013.
- [7] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.
- [8] OpenCV, <https://opencv.org/>, 2018.
- [9] A. Tao, J. Barker, and S. Sarathy, "Detectnet: Deep neural network for object detection in digits," *Parallel Forall*, 2016.
- [10] NVIDIA, <https://developer.nvidia.com/tensorrt>, 2018.
- [11] LISA, <http://cvrr.ucsd.edu/LISA/datasets.html>, 2017.
- [12] Udacity. (2017) Annotated driving dataset. [Online]. Available: <https://github.com/udacity/self-driving-car/tree/master/annotations>
- [13] Nvidia, <https://developer.nvidia.com/embedded/buy/jetson-tx2>, 2018.
- [14] GStreamer, <https://gstreamer.freedesktop.org/>, 2017.
- [15] Nvidia, <https://developer.nvidia.com/digits>, 2018.
- [16] C. A. E. R. Institute, <http://www.caei.com.cn/>, 2018.
- [17] W. Zhao, L. Xu, Z. S. Dong, B. Qi, L. Qin *et al.*, "Improving transfer feasibility for older travelers inside high-speed train station," *Transportation Research Part A*, vol. 113, pp. 302–317, 2018.
- [18] B. Qi, L. Kang *et al.*, "A vehicle-based edge computing platform for transit and human mobility analytics," in *ACM/IEEE SEC*, 2017.
- [19] L. Liu, H. Li, J. Liu, C. Karatas *et al.*, "Bigroad: Scaling road data acquisition for dependable self-driving," in *ACM MobiSys*, 2017.
- [20] L. Kang, W. Zhao, B. Qi *et al.*, "Augmenting self-driving with remote control: Challenges and directions," in *ACM HotMobile*, 2018.
- [21] R. Meng, S. Nelakuditi, S. Wang, and R. R. Choudhury, "Omniview: A mobile collaborative system for assisting drivers with a map of surrounding traffic," in *ICNC*. IEEE, 2015, pp. 760–765.
- [22] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Smartphone-based adaptive driving maneuver detection: A large-scale evaluation study," *IEEE ITS*, vol. 18, no. 9, pp. 2330–2339, 2017.
- [23] P. Snapshot, <https://www.progressive.com/auto>, 2017.
- [24] StateFarm, <https://www.statefarm.com/insurance/auto/discounts>, 2017.
- [25] G. Kar, S. Jain, M. Gruteser, J. Chen *et al.*, "Predriveid: pre-trip driver identification from in-vehicle data," in *ACM/IEEE SEC*, 2017.
- [26] H. Qiu, F. Ahmad, R. Govindan *et al.*, "Augmented vehicular reality: Enabling extended vision for future vehicles," in *ACM HotMobile*, 2017.