Finding and Indexing Vehicle Maneuvers from Dashboard Camera Video

Stephen A. Zekany, Thomas F. Larsen, Ronald G. Dreslinski, and Thomas F. Wenisch
Department of Electrical Engineering and Computer Science, University of Michigan

***Abstract*—A key challenge for self-driving vehicle researchers is to curate massive instrumented vehicle datasets. A common task in their development workflow is to extract video segments that meet particular criteria, such as a particular road scenario or vehicle maneuver. We present a novel approach for detecting vehicle maneuvers from monocular dashboard camera video building upon a deep learning visual odometry model (DeepV2D) to estimate frame-accurate ego-vehicle movement. We leverage image classification and lane line estimation to extend our technique. We classify movement sequences against reference maneuvers using dynamic time warping. We describe, implement, and evaluate classifiers to recognize maneuvers such as turns, lane changes, and deceleration. We show that using deep learning visual odometry to estimate location is superior to consumer-grade high-resolution GPS for this application. We describe and implement a greedy approach to classify maneuvers and evaluate our approach on common road maneuvers. We find an overall AUROC value of 0.91 for turns and 0.84 for all maneuvers in our dataset.**

I. Introduction

State-of-the art systems for autonomous driving are built using massive datasets for training and evaluating vision and control algorithms [1]. A key challenge for self-driving vehicle researchers is to manage and curate these massive datasets. Video libraries associated with autonomous vehicles rapidly grow to enormous sizes; for example, the publicly available Berkeley DeepDrive dataset [34] comprises 100,000 video segments and over 1,100 hours of driving, while proprietary datasets can grow much larger [32].

A common task in the development workflow of autonomous systems is searching such a video archive to extract segments that meet particular criteria. Such searches might be used to find test scenarios to evaluate algorithm performance under unusual circumstances, for example, a segment where strong braking is needed to avoid a collision. Alternatively, when building a training dataset for deep learning algorithms, it may be desirable to oversample video segments for situations that arise rarely in practical driving. Existing data management systems are well suited to execute fast searches and queries over relational (tabular) data, but typically cannot do so for unstructured data like video. State-of-the-art video processing systems, like Scanner [24], include optimizations like frame skipping and efficient delta-frame decoding, but still largely resort to brute-force search over frames. Developments in deep learning, such as object detection and semantic segmentation, present new opportunities for video search systems to leverage at near-human accuracy [6] [8] [15].

In this paper, we develop a processing pipeline that generates an index, enabling searches for vehicle maneuvers from dashboard camera (dashcam) video. The pipeline classifies video frame sequences against a set of reference video clips, provided by the user, that demonstrate the vehicle maneuvers (e.g., right turn, U-turn, driving in reverse) to be indexed. Our system does not rely on GPS, accelerometry, or other metadata to determine the motion of the ego-vehicle. Instead, we use monocular dashcam video and existing deep learning models such as DeepV2D [27] and LaneNet [14] to derive a human-interpretable intermediate representation of the ego vehicle state, which includes features such as the trajectory and position of road markings. Additionally we present a technique to detect highway entrances and exits which leverages image classification.

Our system proceeds in three high-level phases. In the first phase, we use deep learning models to construct an internal representation of ego-vehicle state for each frame. In the second phase, for each possible maneuver, we search for and score possible matches by comparison to the state of known reference maneuvers. In this way, the constructed state acts as an index for the video library, enabling efficient search for vehicle maneuvers. In the third phase, we collate the individual maneuver predictions into a joint maneuver prediction. When multiple maneuvers coincide in time, we apply priority rules and a random forest classifier to resolve conflicting output from individual classifiers and produce the best labeling.

We evaluate our approach against human-labeled ground truth for common road maneuvers and find our performance exceeds a classification approach using measured GPS location. In addition to classifying sequences directly, we describe how the system can be tuned to eliminate as many uninteresting frames as possible while capturing *possibly relevant* video segments (i.e., tuning a particular recall threshold), thereby reducing workload for a human evaluator.

We describe, implement, and evaluate classifiers for these common road maneuvers. We find utilizing a “min pool” approach to allow each classifier to select the closest match between several reference maneuvers outperforms a single reference maneuver. We also build an overall classifier to operate the individual classifiers and prioritize detected maneuvers. We present descriptive statistics such as AUROC and F1 score over our dataset for the individual classifiers and the overall classifier.

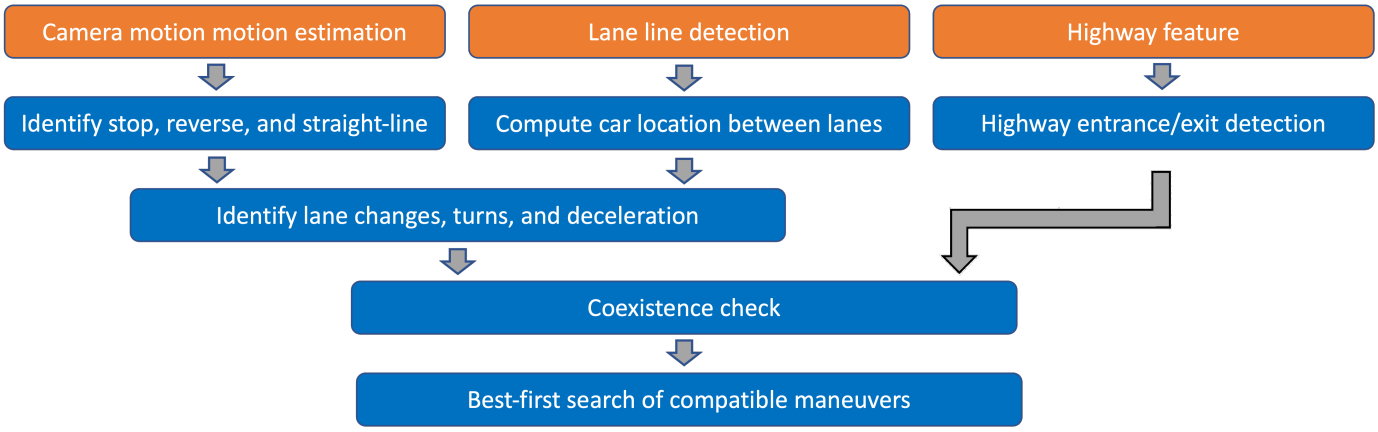

Fig. 1. Our software pipeline obtains output location estimates from DeepV2D, combines these into candidate trajectories, and uses a greedy approach to classify the best fit of a maneuver at any given time.

We make the following specific contributions:

- We use reconstructed X-Y position to search a video library for turn maneuvers using dynamic time warping (DTW).
- We show DTW is a useful algorithm to measure maneuver similarity beyond X-Y position by using projected lane position and estimated vehicle speed to detect lane changes and deceleration maneuvers, respectively.
- We implement and evaluate individual classifiers for these maneuvers.
- We describe, implement, and evaluate an overall classifier to compose results of individual maneuver classifiers and produce the best label for each time instant (video frame).

Overall we find estimations of vehicle maneuvers from monocular dashboard camera video is an effective technique to classify video frames.

II. Prior Work

In this section, we review two similar areas of work: aggressive driving detection and SLAM (simultaneous localization and mapping) from video.

A. Maneuver Classification

A number of studies have explored methods to evaluate *driving style* by analyzing vehicle characteristics and maneuvers. Driving style is typically a characterization of how "aggressive" a specific driver is relative to an aggregate set of reference drivers. This work is of particular interest to safety agencies and insurance companies, who wish to mitigate risk via measurement of individual or aggregate behavior. While our work does not focus on driving style or safety specifically, it is relevant because we utilize similar techniques for measuring vehicle maneuvers. Johnson et al. [11] describe a method to measure aggressiveness for turns and straight-line motion using dynamic time warping with a smartphone. Later work extended this approach with Bayesian classification [4], support vector machines, and random forest analysis [13]. Other work has focused on building a driver-centric model based on a series of maneuvers using a sensor array [3] or detection of aggressive driving using deep learning techniques [7] [23]. In contrast to our work, which requires only dashboard camera video with no additional metadata, this body of work typically utilizes smartphone sensors, such as an accelerometor, GPS, magnetometer, and gyroscope, which must be on-board the vehicle [33] [16].

Other work has been done with additional sensors, which allow for more fine-grained detection of maneuvers; for example, classification of lane changes (cut-ins and overtaking) using hidden Markov models (HMMs) with data collected from radar and LIDAR [17]. Work has also been done using HMMs to predict maneuvers [9].

B. Visual Odometry

Another significant body of work from the computer vision community has focused on the idea of *visual odometry* [21]. Borrowing ideas originating from SLAM, various techniques have focused on efficient ways to extract ego-vehicle motion from video, often for the purpose of mapping a route. ORB-SLAM is a modern monocular SLAM system that outputs camera motion [18]; our work builds on the contributions of this community. Specifically, DeepV2D [27], the deep learning algorithm we use to find camera motion, has operational similarity to ORB-SLAM.

III. Turn Detection Technique

Figure 1 shows an overview of our vehicle maneuver classification pipeline. We first use DeepV2D to extract translational and rotational camera movement at each frame to obtain a time series of X-Y coordinates corresponding to the ego-vehicle's trajectory. We can, optionally, use simple heuristics to label stop, reverse, and straight-line motion [35]. We then compare trajectory segments against reference maneuvers using dynamic time warping to compute a distance measure. Finally, we perform a best-first match of turn maneuvers using a greedy approach. We will describe each of these steps in detail in the following sections.

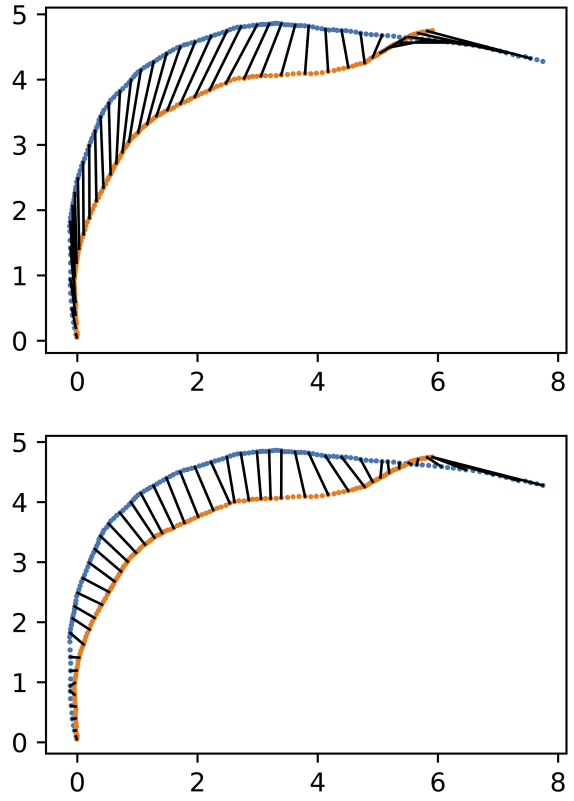

Fig. 2. An example of two right turns (in orange and blue) collected from our experiments. The location tracking is produced by DeepV2D and the maneuvers are automatically reoriented based on preceding motion. Rather than comparing maneuvers using Euclidean distance between discrete locations at a given time (top), we use Dynamic Time Warping (bottom), visualized here with lines connecting a sample of the match points.

A. DeepV2D

DeepV2D [27] is a deep learning network design for estimating camera motion and depth from video. DeepV2D consists of a Stereo Module, to perform stereo reconstruction from images, and a Motion Module, which uses depth to estimate camera motion. The motion module finds initial estimates for the sequence of frames using a pose regression network to estimate the transformation parameters between images. These initial estimates are then refined in an iterative process using a projective warping function on the differentiable transformation of the input image to produce a warped feature map. The estimated feature map is then compared to the original image feature map and the difference is used to update the pose estimate for the next frame.

We use a model of the DeepV2D architecture trained via RMSprop [29] and the Kitti dataset [5] (with ground truth motion estimated by ORB-SLAM2 [19]) to infer the motion of the ego-vehicle in our dataset. The input to DeepV2D is a series of five video frames and the output is the estimated depth map and motion between the third and fourth frames. We therefore process each series of five frames from the recorded video library to obtain a motion track at the same frame rate as the original video.

TABLE I
HEURISTICS TO ACCELERATE CLASSIFICATION

Maneuver	Heuristic
Left Turn	Forward and horizontal distance traveled must be ≥ 80% of reference maneuver and same direction
Right Turn	Forward and horizontal distance traveled must be ≥ 80% of reference maneuver and same direction
U-Turn	Horizontal distance traveled must be ≥ 80% of reference maneuver
K-Turn	Must contain at least half-second (15 frames) of reverse motion

B. Dynamic Time Warping

Dynamic time warping [26] (DTW) is a measure of the similarity of two signals that may differ in duration. DTW performs a sequential matching between the signals while ignoring time differences. In effect, it "stretches" ("warps") one signal (or parts of it) to match another and computes the difference between matched values as the distance measure. (Figure 2 shows an example of two right turns). Whereas the original DTW algorithm is of $O(n^2)$ computational complexity (where n is the number of matched points), implementations such as FastDTW can approximate DTW at $O(n)$ complexity [25]. The reduction in computation time, along with comparative simplicity relative to other methods of pattern recognition, has enabled DTW to be widely and efficiently used for applications like speech recognition [12], gesture recognition [28], and detection of vehicle maneuvers [11]. We use DTW to quantify the similarity of vehicle trajectories from DeepV2D against the reference maneuvers.

DeepV2D produces a three-dimensional rotation matrix and translation vector for each sequence of five frames. We found using DeepV2D to produce camera motion estimates at the original 30 frames-per-second of our test video produced the highest-quality motion estimates. We discard the Z component and keep a time series of ego-vehicle X-Y coordinates at each frame. Because of the high temporal resolution of the video, we can reconstruct camera motion estimates at equal or better resolution than is typically available from consumer-grade GPS devices.

C. Endpoint Detection and Candidate Maneuvers

Since we have no information about when a given maneuver may start or end, we perform automatic and simple endpoint detection for each DTW sequence. As the DTW matching process has no a priori information – knowing nothing about the positional track in advance – it must consider the possibility of a maneuver starting at every time step (video frame). We assume the length of a candidate maneuver can be 50% to 150% the length of the reference maneuver in 10% increments, which allows for normal variation in ego-vehicle velocity. We allow the start and end time to be any even-numbered frame in the recording, as we find this has no effect on accuracy. While the processing time to calculate the DTW matching score for an individual reference maneuver is relatively short, performing compar-

TABLE II
CLASSIFICATION ACCURACY WITH AND WITHOUT HEURISTICS

	AUROC Value with Heuristic	**AUROC Value without Heuristic**
Left Turn	0.90	0.92
Right Turn	0.89	0.88
U-Turn	0.75	0.50
K-Turn	1.0	0.96
Processing time (seconds)	1098	7816

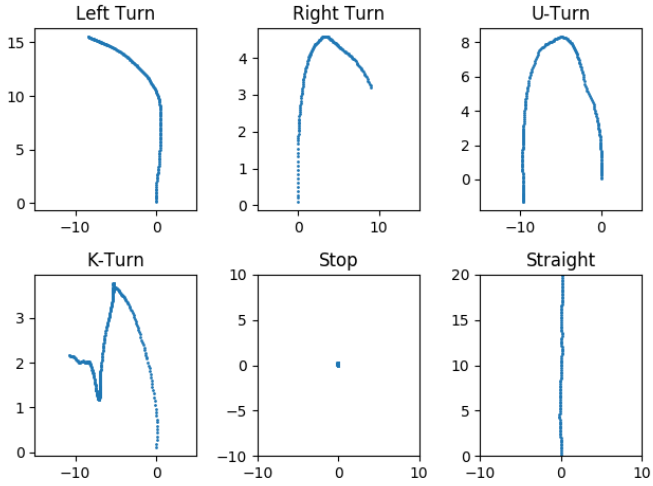

Fig. 3. Sample vehicle maneuver trajectories using location estimates from DeepV2D. X and Y scales are nominally in meters, though DeepV2D overestimates distance in some cases.

isons of all possible candidate maneuvers is computationally expensive. As a strategy to reduce processing time, we enforce a few simple filtering heuristics on the four types of turns processed with DTW (summarized in Table I). For left and right turns, the overall motion must match 80% of the candidate maneuver's movement along the x- and y-axes. U-turns must have sufficient x-axis movement, and K-turns (three point turns) require at least a half second (15 frames) of reverse motion. We conducted a preliminary study on our preliminary dataset (five video recordings for which we had associated GPS data). We find these simple heuristics slightly improve maneuver classification accuracy while reducing computational time by an order of magnitude (shown in Table II).

D. Reference Maneuvers and Best-First Search

We define four turn categories: left turn, right turn, U-turn, and K-turn (examples are shown in Figure 3). (Note that the K-Turn looks odd because DeepV2D incorrectly estimates rotational speed in reverse, simply because the model isn't trained on reverse motion. However the error is consistent across maneuvers, so classification is still possible.) Each turn category is effectively an independent classifier on the camera-motion data produced by DeepV2D. DeepV2D motion estimation is expensive but need only be done once; a user can re-run subsequent analysis with a new set of reference maneuvers.

At least one reference maneuver must be defined for each turn classifier. For a reference maneuver, the user is responsible for defining the start and end time and selecting the baseline length for candidate maneuvers (which will then be scaled as appropriate). We compute DTW distances and select non-overlapping maneuvers (with an optional enforced "dead time" between maneuvers) with the lowest distance measures first. This means each time in a video will be scored with the lowest-distance match to the reference maneuver, and to find the true maneuvers we can evaluate different choices of thresholds for distance measure to capture a high number of true positives while eliminating as many false negatives as possible.

We believe this video search technique functions best as an analogy to a *Bloom filter* [2] (a data structure to determine whether a value is possibly in a set or definitely not in a set). Like a Bloom filter, our turn search techniques function best as an *augmentation* of human searching: we wish to eliminate as many uninteresting frames as possible without eliminating any maneuvers that we seek.

E. Multiple Reference Maneuvers with Min Pool

We desire a distance measure that quantifies the similarity of two maneuvers. Since every maneuver is slightly different there is no perfect reference maneuver that defines a left turn – instead there is a range of what can be considered a left turn. For example, left turns can be sharp, shallow, or simply perpendicular. We improve our model using multiple reference maneuvers and calculating the minimum distance measure to any of them using a min pool function which selects the minimum from a set of input values. Let R be the set of reference maneuvers, c be a candidate maneuver, and $\mathrm{dtw}(c, r)$ be the dynamic time warping distance between c and r. Our new distance is:

$$d(c, R) = \min_{r \in R} \mathrm{dtw}(c, r)$$

We find that the min pool technique produces higher-quality results than a single reference maneuver at the expense of additional computation, as we will elaborate in Section VI.

IV. INDIVIDUAL CLASSIFIERS

We have described the concepts used to find turn maneuvers, especially DTW of candidate maneuvers and best-first search. We find these concepts can be used to detect additional maneuvers. However, it is often the case that common maneuvers are not completely defined by X-Y position. In this section we describe three additional maneuver detection techniques. We find that DTW can work well with time-series data besides X-Y position. We show this by detecting sharp deceleration events and lane changes. We also describe a novel technique for finding highway merges and exits.

A. Deceleration

Our maneuver matching technique relying solely on X-Y position does not give any information about speed because DTW stretches maneuvers in time. For detecting turns, this property is useful, because a left turn can happen at any speed. Other maneuvers, however, can be defined in terms of a change in speed, such as decelerating to stop at an intersection. We can utilize speed as an additional component of measurement and search for maneuvers in the X-Y-S space.

For each frame, we start with a coordinate (x_i, y_i) given from DeepV2D for each frame i in the video. We compute the instantaneous speed at each frame by using the position of the next frame as follows:

$$s(x_i, x_{i+1}, y_i, y_{i+1}) = \frac{\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}}{\Delta t}$$

Where $\Delta t = \frac{1}{30}$ seconds (since our video is 30fps) is the time between the two consecutive frames. Because speed tends have a lower absolute value than position, we apply a 5x adjustment to the speed coordinate. We find this provides a balanced normalization of the dimensions for this search technique. We generate our new X-Y-S coordinates for each frame.

$$\{(x_i, y_i)\} \rightarrow \{(x_i, y_i, 5 \cdot s(x_i, x_{i+1}, y_i, y_{i+1})\}$$

We demonstrate the capability of our new search space by finding periods of strong deceleration, such as a sudden braking for a pedestrian, stop sign, or stoplight. We do not search for strong acceleration as this is much more unusual in normal driving behavior. As with turns, given a reference maneuver, we use DTW to find similar maneuvers in the X-Y-S coordinate system.

B. Lane Change

To detect lane change maneuvers, we use an open-source implementation of a lane detection model (LaneNet [20] [14]) to obtain estimated lane line positions relative to the ego-vehicle. Then, given a reference lane change, we use DTW over the lane position to find similar maneuvers.

LaneNet is pretrained on the TuSimple Lane Challenge Benchmark [30]. LaneNet detects four lane lines per image and outputs a classification score for each. A lane "line" may be an actual painted line on the road surface or the edge of the road. We choose the three most prominent lanes based on the classification score. An example of the model's output is shown in Figure 4. On a two-lane surface road, it is typical to have only one lane line (or none), and using only three instead of all four reduces the likelihood of falsely detecting a nonexistent lane.

We wish to identify the position of the car relative to the lanes, so we use the position of the lane line at the base of the frame. To determine this position, we perform a Hough transform [10] on the lane line segmentation output, and then use the position of the center of the line. If the lane line continues off the left or right edge of the image, we project it linearly to determine the coordinate corresponding to the baseline of the image (which is why negative pixel values can arise for the detected lane position, as in Figure 4). We thus obtain a three-dimensional coordinate to represent the camera's position relative to the lanes for each frame.

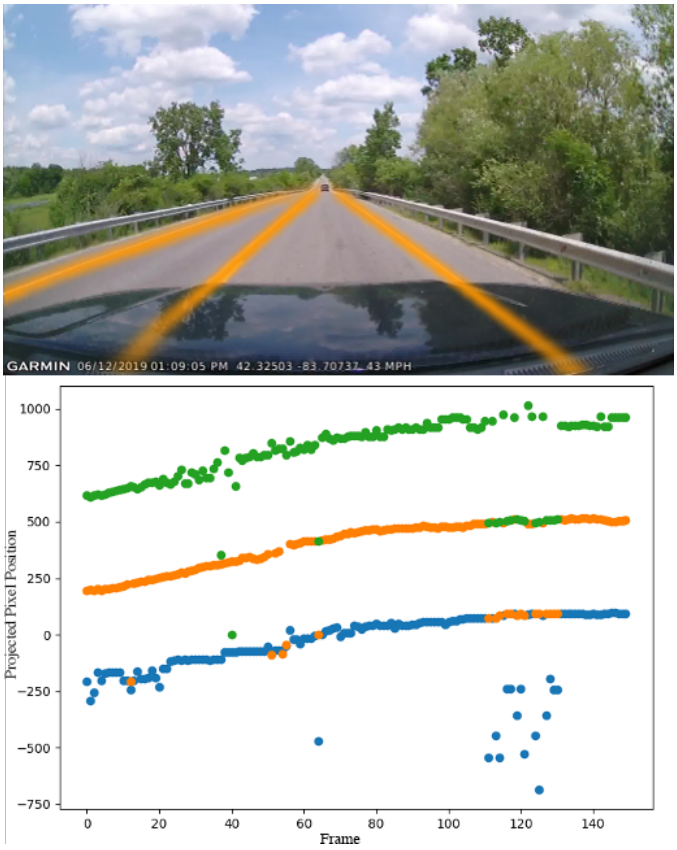

Fig. 4. Graph of the location of each lane (in pixels) in the dashcam against time for a typical lane change. Each color shows the inferred position of a specific lane.

During a lane change, the lane lines move horizontally with respect to the car. This motion causes the positive trend in all three lane lines as seen in Figure 4. When the ego-vehicle is driving straight in its lane, the estimated pixel coordinate of each lane line does not vary. We distinguish between right and left lane changes based on whether estimated lane line positions follow an increasing or decreasing trend. We again apply DTW to find the similarity in trends of the lane line positions over variable length time intervals. We perform DTW on the position of each lane relative to the ego vehicle. Then, using the same procedure as previously described for turns, we identify the candidate maneuvers with the smallest DTW distance to a reference lane change, and find all lane changes.

C. Highway Detection

Identifying highway merge and exit events directly with our maneuver matching technique is challenging, as the length and shape of highway entrance and exit ramps varies considerably. The min pool approach only helps detect known maneuver types and cannot detect unknown ramp configurations. Detecting highway transitions using

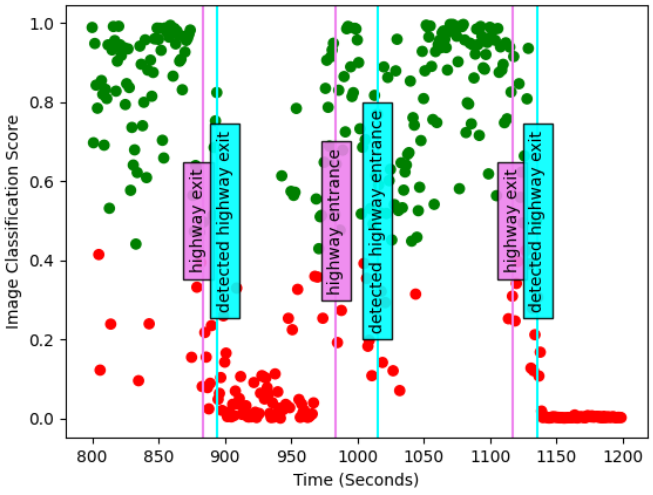

Fig. 5. Image classification score (Y-axis) of image frames from a video including portions of on-highway and off-highway driving. K-means clustering into two groups shows likely on-highway (green) and off-highway (red) frames. The highway entrance and exit events are annotated as purple. These are marked as occurring at the earliest possible interval. Detected highway entrance and exit events are marked in teal.

changes in speed or acceleration is also not viable, as our experiments showed DeepV2D does not perform well on highway environments, where camera pose changes considerably frame-to-frame and there are comparatively few visual features to obtain a good depth map. As such, on highways, DeepV2D yields a noisy velocity signal.

To detect highway merges and exits, we instead use a simple image classifier and k-means sorting. We recorded approximately 68 minutes of training video on local highway and surface roads (a disjoint set of roads from those used in our evaluation dataset). We extract a still frame at a stride of one second. Frames taken on merge and exit ramps were discarded, as are frames where the vehicle is not in motion. Using the resulting still frames, we use transfer learning on an Inception V3 network with two categories: "highway" (n=287) and "surface streets" (n=585) with a 10% validation set and find the model achieves 94% accuracy after 1000 training steps.

We use k-means clustering to best fit the distributions of the on-highway and surface street image clusters in our video dataset. While this is (simple) unsupervised learning, we nonetheless use leave-one-out testing for each video to avoid biasing the distribution. We consider time windows of 30 seconds each from our video dataset: we extract one still frame per second and use the image classifier on each. We then average the scores of these 30 frames and determine to which cluster the average score belongs. This procedure provides a "highway" vs. "surface" classification for each 30-second interval.

To identify highway merges and exits, we consider time windows of 180 seconds in length. We take the average of the first 60 and last 60 seconds, leaving the middle 60 seconds as "dead time", as classification output is unstable during the transition, while the ego vehicle is on an on-ramp or off-ramp (which we assume to last less than 60 seconds). We select these constants such that the windows are sufficiently small to detect brief periods of highway driving while still keeping the classification simple and inexpensive to compute. We report a highway transition when the road type prediction in the first 60 seconds and last 60 seconds differ. We illustrate this clustering technique for a sample video segment in Figure 5. Three highway transition events occur, and each is correctly identified by the classifier.

V. Evaluation Methodology

A. Video Dataset

We collected dashboard view ("dashcam") videos over a total of 15 driving sessions on local surface streets and highways in Washtenaw County, Michigan. Our videos were recorded with a Garmin Dash Cam 55, fixed to the center of the windshield of the vehicle at approximately eye level. We collected our own video, rather than using existing public datasets, because DeepV2D requires camera intrinsics for estimation of camera motion.

We made no preparation for environmental or traffic conditions except for avoiding snow, as DeepV2D performs poorly due to lack of adequate training data in snowy conditions. Road surfaces were mostly (>95% by time) paved, but road surface conditions were highly variable; roads are often patched or in need of repair and lane lines are often faded. Speed limits ranged from 25mph (residential streets) to 70mph (interstate highway). Of 15 video recordings, 13 produced usable camera motion data. One was excluded for low-light conditions (near dusk) and the second for sunlight refracted through a dirty windshield. The 13 videos used for evaluation total 468 minutes of driving footage (about 786,000 individual frames). In addition, approximately 1,000 video frames (from recordings not included in the prior total) were used to train the highway classifier.

We concurrently collected high-resolution GPS positional information using a GoPro Hero 5 Black to obtain 18 Hz GPS metadata time-aligned for a baseline comparison in five videos. We chose GPS as a fair comparison for location tracking, as GPS metadata is readily available in a number of consumer devices.

We manually labeled all maneuvers that occur in recorded videos. Most maneuvers are simple to score unambiguously, such as turns and lane changes. When in doubt about whether a particular maneuver occurs (such as a slight left turn), we err on the side of labeling it. We found the deceleration maneuver is challenging for humans to score objectively, however, so we followed the following protocol:

- Human scorer identifies all *unambiguous* deceleration maneuvers.
- Run evaluation and obtain a list of all false positive maneuvers.
- For each false positive, consider changing to a true positive if upon re-examination the maneuver contained a speed drop of >10mph.

B. Data Preprocessing

Videos were converted to JPEG frames using ffmpeg for processing by DeepV2D, running on an Intel Core i7 workstation with an NVIDIA Titan Xp GPU. This workstation ran Tensorflow 1.12 with CUDA 9.0 on Ubuntu 16.04. The higher-resolution GPS data extracted from the GoPro was time-aligned with the Dash Cam 55 video.

Prior to classification, the trajectory data must be reoriented. While DeepV2D motion estimation allows knowledge of the ego-vehicle's orientation at any arbitrary point, GPS does not. Orientation is important, as a maneuver cannot be detected if the positional track is rotated. Therefore, we estimate orientation from GPS data by taking the average direction vector of the half-second period prior to the start time of interest. To allow a fair comparison, we use the same half-second reorientation algorithm for DeepV2D trajectories.

C. Selection of Reference Maneuvers for Evaluation

Scored maneuvers account for a total of 1.7 hours (about 22%) of video. We then select 10% of scored maneuvers at random to use as reference maneuvers to identify candidate maneuvers (analogous to training data, though in this case the system is not actually being trained). These reference maneuvers are excluded from the overall statistical calculations. Following the conceptual goal of using k-fold cross-validation to split our reference and test data, effectively each reference maneuver (or set of maneuvers) serves as the "training" set for a given run of a classifier and every other maneuver is part of the "test" set. We manually select ground truth endpoints to best capture the given maneuver.

D. True vs. Detected Time Difference

Our time-invariant classification approach effectively filters periods of no motion, leading to cases where the human-scored event time does not match the classifier's detected time. For example, a car creeping forward into a turn while waiting for oncoming traffic to clear: our classifier will find the best match to the reference maneuver, which can differ by a few seconds from the onset time scored by a human. In practice, we observe that it is typical for human-scored vs. detected values to differ by one to two seconds.

We introduce a parameter to our pipeline called "Maneuver Time Error" (MTE), which controls the maximum allowed difference between predicted time and actual time for a maneuver to be considered correctly classified. Importantly, MTE is used *only* when matching within a single maneuver class. We found the number of instances of a single-class maneuver recurring in less than four seconds is zero. Therefore, we chose four seconds to be the default MTE for all results, expecting the absolute number of errors where the wrong maneuver is detected due to the human-scored event time differing from the classifier-assigned time to be very small.

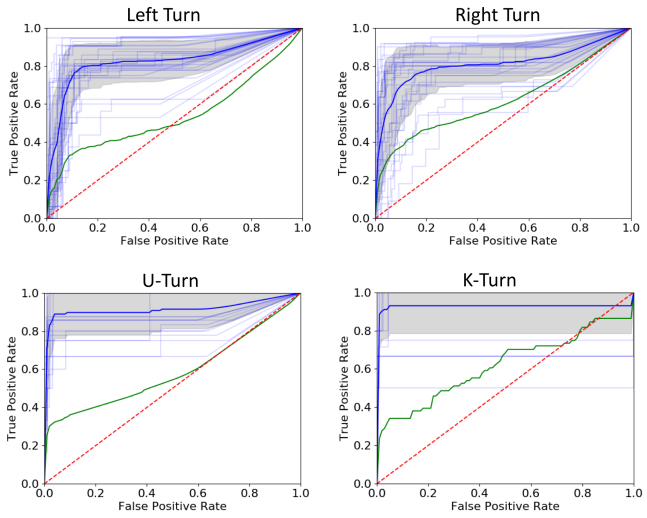

Fig. 6. Average ROC of cross-validation DTW classification for DeepV2D motion estimates (blue) and GPS (green).

VI. Results

We assess the accuracy of each maneuver's classification as follows: A reference maneuver of a particular category is chosen at random. The classifier then finds the best matches to this reference maneuver throughout the duration of the video, allowing no maneuvers to overlap. These candidate maneuvers are compared to the human-annotated ground truth. As described in Section V, if a maneuver is detected within four seconds of the ground truth annotation, it is considered correctly classified; otherwise it is considered a false positive. We then compute the ROC curve using the list of distance measures of detected maneuvers. False negatives (that is, ground-truth maneuvers not detected by the classifier) are included in the ROC calculation with the minimum-detected distance measure (to penalize the classifier for failing to detect these maneuvers).

To evaluate the quality of the DTW classifiers, we use area under the receiver operating characteristic curve (AUROC) as our primary metric. The output from each classifier is essentially a sorted list of distance measures and segments of video. As the distance increases, there is worse similarity to the reference maneuver. Interpretation as a binary classifier requires choosing a cutoff threshold for each classifier's distance measure. Choosing a threshold is an arbitrary exercise as it is not clear what constitutes a "good" value. Therefore, we instead use an ROC curve, which does not require choosing a threshold. The ROC curve is a plot of false positive rate vs. true positive rate, and integrating this curve gives a measure of quality of the classifier without requiring a specific numeric threshold.

A. Estimated Vehicle Motion Compared to GPS

We find estimating vehicle motion with visual odometry significantly outperforms off-the-shelf GPS. To perform this comparison, we replace DeepV2D's motion estimates in

our maneuver detection pipeline with 18 Hz GPS data. We collected this data simultaneously with a subset of our overall video library (5 video recordings). We show the average ROC curves [22] for each of the four turn maneuvers detected in Figure 6. Each individual ROC curve for a particular maneuver (shown in thin blue lines) is averaged with equal weighting at each step (shown in the bold blue line). An interval of one standard deviation above and below the mean is shown in grey.

The green line represents the reconstructed trajectories when using GPS instead of the DeepV2D portion of our computational pipeline. We find the DeepV2D motion estimates consistently outperform our GPS measurements, due to the spatial and temporal resolution advantage of DeepV2D's 30 Hz estimates relative to the 18 Hz 3-meter resolution of the GPS. The typical GPS accuracy of approximately 3 meters [31] is simply not sufficient for tight maneuvers, as a standard U.S. highway lane is 3.7 meters wide and city roads are narrower. Therefore, a maneuver like a U-turn is difficult to reconstruct accurately if the maneuver is performed on a two-lane road that is at most 8 meters wide.

B. Turn Classifier

The results are shown in Figure 7. We use an average ROC (shown in orange) across every reference maneuver, meaning each individual true or false classification is evaluated with respect to ground truth and summed. We show the results of our min pool technique described in Section III as the ROC curve in green. We find the min pool technique exceeds the individual average for all maneuver types, as this allows the classifier greater flexibility in matching maneuvers to a library of known maneuvers. This improvement occurs because even with randomly chosen maneuvers, different road intersections and common maneuvers will have slightly different geometry, such as a U-turn across a narrow road vs. a wide road.

C. Calculation of Discarded Frames

While we prefer AUROC as the primary metric, we are also interested in using our system as a binary classifier that finds all instances of a particular maneuver. We wish to avoid discarding positive events, meaning that we prioritize a high recall at the expense of precision. This use case requires the selection of a threshold to meet the desired recall rate, and then calculating the resulting precision and F1 score. Our DTW classifier partitions the video into many different segments, each representing a potential maneuver with a DTW score representing how close it is to the reference maneuver(s). By introducing a specific score threshold, we convert the DTW score to a binary classification. This system is intended as support for a human looking for maneuvers, so we are particularly interested in the *fraction of discarded frames* for a particular recall. We can approximate this measure as follows: Let N be the number of potential maneuvers in the partition, and $TP+FP$ be the number of events that were predicted positive by our classifier. $\frac{TP+FP}{N}$ is the fraction that is kept, so we define $F_{\text{eliminated}}$ to be the percentage of eliminated sections of video.

$$F_{\text{eliminated}} = 1 - \frac{TP+FP}{N}$$

We can derive $F_{\text{eliminated}}$ as a function of a fixed precision and recall with the following algebra:

$$\begin{aligned} F_{\text{eliminated}} &= 1 - \frac{TP+FP}{N} \\ &= 1 - \frac{TP+FP}{TP} \cdot \frac{TP}{TP+FN} \cdot \frac{TP+FN}{N} \\ &= 1 - \frac{\text{Recall} \cdot (TP+FN)}{\text{Precision} \cdot N} \end{aligned}$$

Where the first step is multiplying by 1, and the second step is substitution from the definition of precision and recall. Because maneuvers differ slightly in length, we refer to this as the *approximate percent of discarded frames*. We show the calculation of $F_{\text{eliminated}}$ for individual classifiers in Table III.

TABLE III
INDIVIDUAL CLASSIFIER PERFORMANCE BY RECALL THRESHOLD

Classifier	Recall	Precision	F1 Score	Frames Eliminated
Deceleration	0.98	0.06	0.12	**34%**
	0.95	0.07	0.13	**50%**
	0.90	0.13	0.23	**74%**
	0.80	0.23	0.36	**92%**
	0.61	0.46	0.53 (max)	**95%**
Left Turn	0.98	0.05	0.10	**46%**
	0.95	0.19	0.32	**86%**
	0.90	0.34	0.49	**93%**
	0.80	0.43	0.56	**95%**
	0.74	0.52	0.61 (max)	**96%**
Right Turn	0.98	0.12	0.22	**34%**
	0.95	0.53	0.67	**86%**
	0.92	0.85	0.89 (max)	**91%**
	0.90	0.85	0.88	**91%**
	0.80	0.89	0.84	**92%**
Left Lane Change	0.98	0.01	0.02	**31%**
	0.95	0.02	0.03	**64%**
	0.90	0.02	0.04	**77%**
	0.80	0.06	0.10	**92%**
	0.33	0.61	0.43 (max)	**99%**
Right Lane Change	0.98	0.01	0.03	**56%**
	0.95	0.02	0.03	**57%**
	0.90	0.02	0.03	**62%**
	0.80	0.03	0.06	**84%**
	0.18	0.79	0.30 (max)	**99%**

D. Comparison of Turn Classifiers

We wish to quantify how well DTW distance measures can function as as a metric across different classes of maneuvers. This experiment shows the potential to search for multiple maneuver types instead of only one at a time. To answer this question, we use a greedy approach to classify these four turn maneuvers, recursively selecting the candidate maneuver with the lowest distance measure among all four turn types. (We purposely ignore the fact that certain maneuvers may have different distributions of

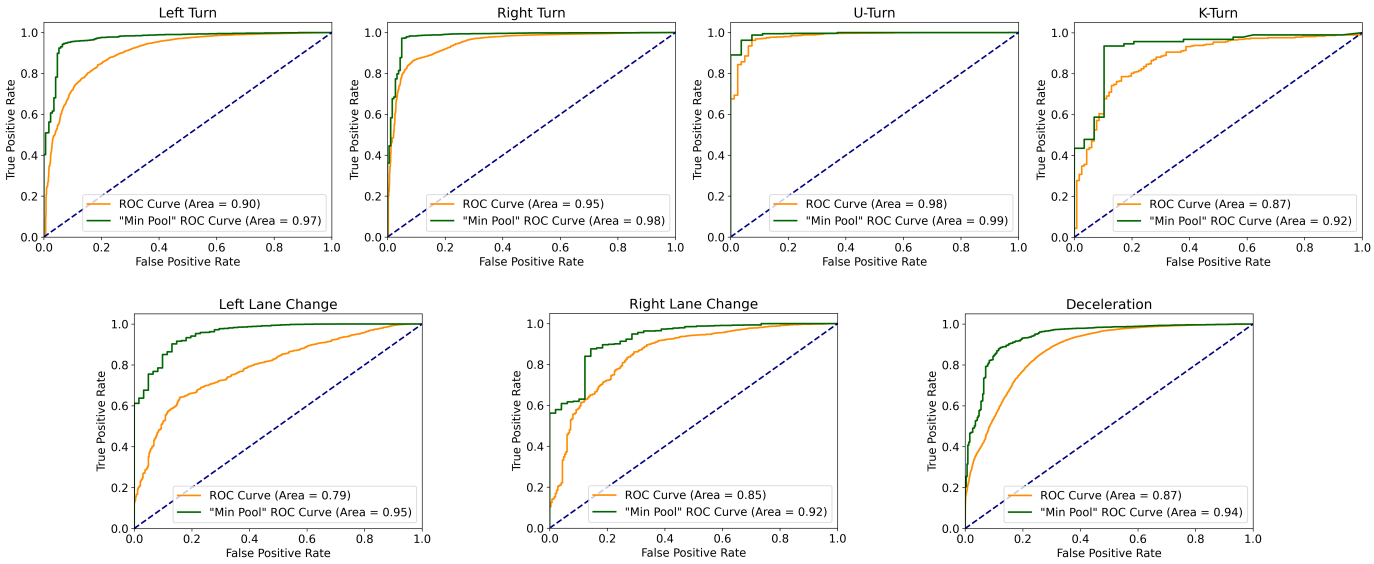

Fig. 7. ROC curves for seven individual classifiers. We find that the min pool approach of looking for the best match among several maneuvers (green) is superior to the average of individual randomly chosen maneuvers (orange).

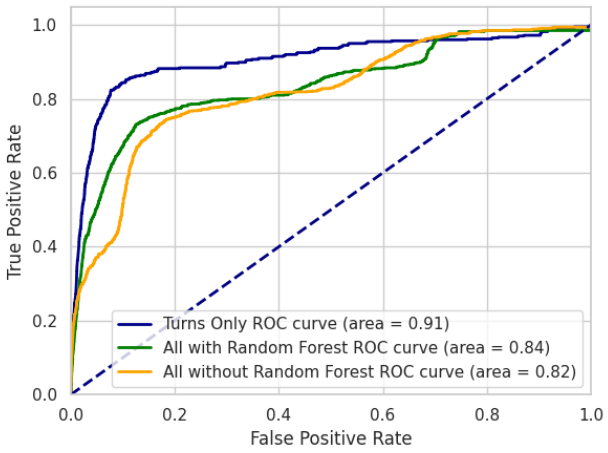

Fig. 8. Receiver Operating Characteristic of turns only, overall classifier with random forest, and overall classifier without random forest.

distance measure values.) We then check the time duration of this candidate maneuver to ensure no prior selected maneuver is in conflict. We find achieve an overall AUROC of 0.91; the ROC curve is shown in Figure 8. Note that we count each maneuver instance in our dataset equally, so the more prevalent maneuvers are weighted more in this AUROC curve.

E. Additional Classifier Results

Below we describe the additional maneuver classification techniques we have developed for common driving maneuvers. For each maneuver type, we show average X-distance, Y-distance, and time duration for each correctly-matched maneuver in Table IV (standard deviation is shown in parentheses). The units are in estimated meters from the visual odometry model.

TABLE IV
DESCRIPTIVE STATISTICS OF DETECTED MANEUVERS.

Maneuver	X Mean in Estimated Meters	Y Mean in Estimated Meters	Time Mean in Seconds
Right Turn	7.89 (2.36)	9.04 (3.12)	4.20 (1.61)
Left Turn	-6.29 (3.26)	8.61 (4.52)	3.52 (1.03)
U-Turn	-0.73 (2.17)	29.38 (20)	6.61 (1.40)
K-Turn	-5.63 (3.13)	1.13 (2.14)	9.69 (3.33)
Left Lane Change	-0.73 (2.17)	29.38 (20)	1.87 (0.87)
Right Lane Change	0.61 (2.59)	30.54 (19.7)	2.37 (0.50)
Deceleration	-0.89 (1.73)	13.6 (7.00)	3.99 (1.08)

1) Deceleration: Overall, we find an AUROC of 0.87 for the deceleration classifier using this technique while selecting uniformly at random 10% of our maneuvers as reference maneuvers. This is seen in Figure 7. When augmented with the min pool technique, we find that AUROC improves to 0.94. While the F1 score is not as high, we find the effective number of frames eliminated can be quite high for a given recall. For example, with a recall threshold of 90% we can successfully discard 74% of the video frames as uninteresting. We note that this discard rate is a bit lower than for turn maneuvers; human scoring of deceleration events is itself a subjective exercise and difficult to quantify compared to the other maneuvers.

2) Lane Change: We find an AUROC of 0.79 for left lane change and 0.85 for right lane change. If we use 10% of of the maneuvers as reference maneuvers and take the minimum distance, the AUROC improves to 0.95 for left lane change and 0.92 for right lane change, showing

TABLE V
HIGHWAY CLASSIFIER AND MERGE/EXIT DETECTOR PERFORMANCE

	Highway On/Off	Highway Merge/Exit
Precision	0.83	0.54
Recall	1.0	0.96
F1 Score	0.90	0.69
Count (True)	85	23
Count (False)	348	116
Effective Frames Eliminated	n/a	70.5%

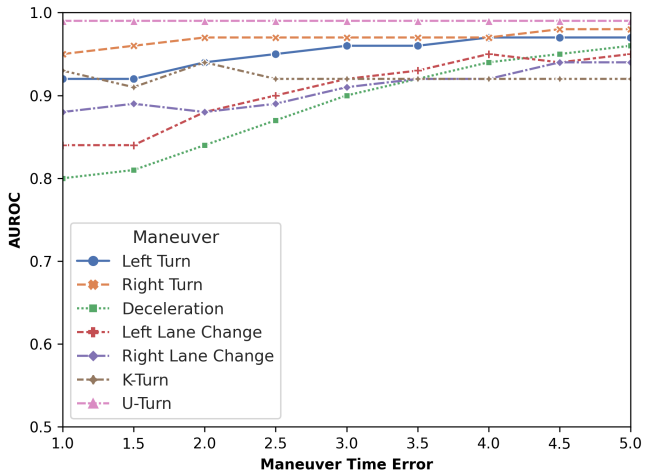

Fig. 9. Effect of Maneuver Time Error (MTE) on AUROC for each maneuver class.

that DTW is a robust technique for finding lane change maneuvers.

3) Highway vs. Surface Streets: We show the results in the "Highway On/Off" column of Table V. We find that for a total of 85 on-highway epochs, we achieve a recall of 1.0 and precision of 0.83 for a total F1 score of 0.90. While our approach correctly identifies every on-highway epoch, we find the false positives from the image classifier occur primarily on major surface streets that have certain features in common with a highway (e.g. medians, few businesses or buildings, trees set back far from the road). As our goal is to maximize recall even at the cost of a lower F1 score, we find this to be an acceptable trade-off in practice.

4) Highway Merges and Exits: The overall performance of the merge/exit classifier is shown in the "Highway Merge/Exit" column of Table V. We find that this technique gives high recall (0.96) and precision of 0.54, allowing us to eliminate approximately 70.5% of the total frames in our video dataset.

F. "Maneuver Time Error" Sensitivity

To justify four seconds as the set value for "Maneuver Time Error" (MTE) as mentioned in Section V, we show a sensitivity study in Figure 9 of MTE values ranging from 1 to 5 seconds in 0.5 second intervals and the resulting effect on the AUROC value. We find that as MTE is decreased, AUROC falls as some detected events are unable to be matched. At four seconds and above, we see relatively small increases or decreases of the AUROC value. Based on these data (and no consecutive maneuvers recurring within four seconds) we believe four seconds is a reasonable choice for MTE.

TABLE VI
OVERALL CLASSIFIER PERFORMANCE BY RECALL THRESHOLD

Recall	Precision	F1 Score	Frames Eliminated
0.98	0.08	0.15	**34%**
0.95	0.10	0.17	**49%**
0.90	0.14	0.24	**65%**
0.80	0.21	0.33	**79%**
0.52	0.47	0.56 (max)	**94%**

TABLE VII
OVERALL CLASSIFIER ACCURACY FOR BEST MATCH (TOP 1) AND TOP 2 DETECTION OF MANEUVERS BY TIME. WE ALSO SHOW THE TOP 2 MATCHES ASSUMING THE RANDOM FOREST MODEL ALWAYS CHOOSES CORRECTLY.

	Best Match Overall	Top 2 Matches	Top 2 Matches (without RF error)
Matched	628	654	703
Missed	169	143	94
Accuracy	78.8%	82.1%	88.2%

VII. OVERALL MANEUVER CLASSIFICATION

Whereas each maneuver classifier can be used individually, we also wish to compose classifiers to determine the best labeling for every point in a video.

A. Technique

Building an overall classifier is not as simple as running each classifier individually and comparing the results. We begin by defining a coexistence matrix specifying which maneuver label should be preferred when they overlap (e.g., prefer K-turn over turn, as a turn may be part of a K-turn), and whether two labels may coexist (deceleration may coincide with a turn). For example, we disallow turn maneuver labels when the scene is classified as highway driving. The coexistence matrix can be used to optimize computational performance (skipping classifications that are disallowed by the matrix) or to improve classification accuracy by eliminating false positives.

When classifiers produce a label combination disallowed by the coexistence matrix, we must resolve the conflict. We apply several heuristics:

- If the evaluation scores of two classifiers are comparable, we can choose the more confident score. For example, if we find a time segment that is similar to a left turn with a distance measure of 100, and similar to a right turn with a distance measure of 10,000, we can prefer the lower distance score.
- We filter turns detected while driving on the highway, as the turn detector produces false positives at highway curves.
- Deceleration may overlap with any other maneuver.

TABLE VIII
CONFUSION MATRIX

True \ Detected	Highway Merge/Exit	Left Turn	Right Turn	U-Turn	K-Turn	Left Lane Change	Right Lane Change	Deceleration
Highway Merge / Exit	22 (96%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (4%)	0 (0%)	0 (0%)
Left Turn	0 (0%)	161 (89%)	0 (0%)	0 (0%)	0 (0%)	4 (2%)	15 (8%)	0 (0%)
Right Turn	0 (0%)	0 (0%)	133 (80%)	0 (0%)	0 (0%)	12 (7%)	21 (13%)	0 (0%)
U-Turn	0 (0%)	0 (0%)	1 (3%)	21 (68%)	0 (0%)	2 (6%)	7 (23%)	0 (0%)
K-Turn	0 (0%)	0 (0%)	1 (4%)	0 (0%)	26 (93%)	0 (0%)	1 (4%)	0 (0%)
Left Lane Change	1 (1%)	1 (1%)	14 (18%)	2 (3%)	0 (0%)	5 (7%)	53 (70%)	0 (0%)
Right Lane Change	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (2%)	62 (98%)	0 (0%)
Deceleration	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	251 (99%)

- Lane DTW and turn DTW do not provide comparable distance measures, and there is no obvious heuristic for which label to prefer when both classifiers report a match. To solve this problem, we trained a simple random forest classifier (validated on an independent selection of lane changes and turns) which attempts to determine whether a particular period of time is more likely to be a lane change or a turn, based on the positional change over time.

The overall classification algorithm is as follows: we run each classifier separately and obtain maneuver predictions at each timestep. For the lane-change, turn, and deceleration classifiers, the prediction output is a distance measure from reference maneuvers, but the on-highway classifier is a binary classification. Therefore we perform highway classification first and mask the predicted on-highway time periods. For each subsequent classifier, we check whether coexistence with prior predictions during the same time period is allowed. For maneuvers with comparable distance measures (e.g., right turns and left turns), we choose the maneuver with the lowest distance measure. After all conflicting predictions are pruned, we add the best detected maneuver's time period to the mask to prevent an overlapping maneuver from being detected. After importing and pruning the output from each classifier, the result is a time series with the best match of maneuver(s) for each point in time.

B. Results

Table VI shows the precision and F1 scores at different recall rates. As with the individual classifiers, the primary purpose of our technique is to find as many *possibly interesting* sequences of frames as possible while eliminating a high number of uninteresting frames. We find that while detecting 90% of all maneuvers we can discard nearly two-thirds (65%) of frames. At the maximum F1 score, we detect about half (52%) of all maneuvers correctly while eliminating 94% of frames.

We measure the accuracy of matching one or more detected maneuvers at a given time to the correct maneuver at that time. We first perform this matching for the best single detected maneuver at a given time. We find 654 maneuvers match correctly and 143 are missed, for a total overall accuracy of 82.1% (Table VII). When we expand the classifier to allow selection of the best two maneuvers, we find accuracy improves to 88.2%. Additionally, we compute ROC curves for the overall classifier (excluding highway, as highway is a binary classification) as shown in Figure 8.

The confusion matrix for our overall classifier is shown in Table VIII. We find left turns commonly alias with U-turns and K-turns, as each of these begins with ego-vehicle movement to the left. We also see that left and right lane changes alias frequently, due in part to some variance in the randomly-chosen reference maneuvers for each.

We also find that the overall classifier approach frequently has false positives for left and right turns compared to the labeled ground truth; these arise due to curves in the road. For example, a road curving to the left can be detected by the classifier as a left turn even though a human marked it as forward/unremarkable. To better distinguish curves from turns, our trajectory-based method could be fused with other information sources, such as road segmentation or map data.

VIII. CONCLUSION

We have described, implemented, and evaluated classifiers for common road maneuvers. The input to these classifiers is reconstructed trajectory data from dashboard camera video. We evaluated our approach against human-labeled ground truth for common road maneuvers, and compare against a classification approach which uses measured GPS rather than reconstructed trajectories. We found that utilizing a min pool approach outperforms classification with only a single reference maneuver. We built an overall classifier to operate the individual classifiers and implement our rules for handling conflicts. This classifier had a 78.8% top-1 accuracy and an AUROC of 0.84.

Acknowledgements We thank German Ros for his advice in the early stages of the project, and Zach Teed and Jia Deng for their assistance using DeepV2D.

This work was supported by the Toyota Research Institute and the Applications Driving Architectures (ADA) Research Center, a JUMP Center co-sponsored by SRC and DARPA.

IX. Author Biographies

Stephen A. Zekany is a Ph.D. candidate at the University of Michigan, Ann Arbor, Michigan. Stephen's research interests are the efficiency of hardware and software systems at scale, and the intersection of computer hardware and machine learning.

Thomas F. Larsen is an honors undergraduate student studying Computer Science and Mathematics at the University of Michigan, Ann Arbor, Michigan. Thomas' research interests are machine learning and AI safety.

Ronald G. Dreslinski is an Associate Professor of Computer Science and Engineering at the University of Michigan, Ann Arbor, Michigan. He earned the IEEE TCCA Young Computer Architect Award and was recognized by the College of Engineering with the Kenneth M. Reese Outstanding Research Scientist Award in 2015, and recently was named a Morris Wellman Faculty Development Professor.

Thomas F. Wenisch is a Professor of Computer Science and Engineering at the University of Michigan and Director of Engineering at Google. He holds a Ph.D. from Carnegie Mellon University. He has authored numerous papers in computer architecture and systems. He has been recognized with the HPCA and ASPLOS Hall of Fame awards and the University of Michigan's Henry Russel Award.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[2] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[3] Hien Dang and Johannes Fürnkranz. Using past maneuver executions for personalization of a driver model. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 742–748. IEEE, 2018.

[4] Haluk Eren, Semiha Makinist, Erhan Akin, and Alper Yilmaz. Estimating driving behavior by a smartphone. In *2012 IEEE Intelligent Vehicles Symposium*, pages 234–239. IEEE, 2012.

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] Robert Geirhos, David HJ Janssen, Heiko H Schütt, Jonas Rauber, Matthias Bethge, and Felix A Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017.

[7] Jingqiu Guo, Yangzexi Liu, Lanfang Zhang, and Yibing Wang. Driving behaviour style study with a hybrid deep learning framework based on GPS data. *Sustainability*, 10(7):2351, 2018.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle trajectory prediction based on motion model and maneuver recognition. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4363–4369. IEEE, 2013.

[10] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, 1988.

[11] Derick A Johnson and Mohan M Trivedi. Driving style recognition using a smartphone as a sensor platform. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1609–1615. IEEE, 2011.

[12] B-H Juang. On the hidden Markov model and dynamic time warping for speech recognition—a unified view. *AT&T Bell Laboratories Technical Journal*, 63(7):1213–1243, 1984.

[13] Jair Ferreira Júnior, Eduardo Carvalho, Bruno V Ferreira, Cleidson de Souza, Yoshihiko Suhara, Alex Pentland, and Gustavo Pessin. Driver behavior profiling: An investigation with different smartphone sensors and machine learning. *PLoS one*, 12(4):e0174959, 2017.

[14] Github: MaybeShewill-CV/lanenet lane detection. https://github.com/MaybeShewill-CV/lanenet-lane-detection. Accessed: 2019-04-14.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[16] Clara Marina Martinez, Mira Heucke, Fei-Yue Wang, Bo Gao, and Dongpu Cao. Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):666–676, 2018.

[17] John Martinsson, Nasser Mohammadiha, and Alexander Schliep. Clustering vehicle maneuver trajectories using mixtures of Hidden Markov Models. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3698–3705. IEEE, 2018.

[18] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[19] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[20] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach, 2018.

[21] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Xishuai Peng, Yi Lu Murphey, Ruirui Liu, and Yuanxiang Li. Driving maneuver early detection via sequence learning from vehicle signals and video images. *Pattern Recognition*, 103:107276, 2020.

[24] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics (TOG)*, 37(4):138, 2018.

[25] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[26] D Sankoff and J Kruskal. The symmetric time-warping problem: from continuous to discrete. In *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 125–161. Addison Wesley Publishing Company, 1983.

[27] Zachary Teed and Jia Deng. DeepV2D: Video to depth with differentiable structure from motion. In *International Conference on Learning Representations*, 2019.

[28] Gineke A Ten Holt, Marcel JT Reinders, and EA Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, volume 300, page 1, 2007.

[29] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[30] TuSimple. Github: TuSimple-benchmark. https://github.com/TuSimple/tusimple-benchmark. Accessed: 2019-04-14.

[31] Navigation U.S. Air Force National Coordination Office for Space-Based Positioning and Timing. Gps.gov: GPS Accuracy. https://www.gps.gov/systems/gps/performance/accuracy. Accessed: 2019-04-16.

[32] The Verge. How Tesla and Waymo are tackling a major problem for self-driving cars: data. https://www.theverge.com/transportation/2018/4/19/17204044/tesla-waymo-self-driving-car-data-simulation. Accessed: 2019-04-16.

[33] Johan Wahlström, Isaac Skog, and Peter Händel. Smartphone-based vehicle telematics: A ten-year anniversary. *IEEE Transactions on Intelligent Transportation Systems*, 18(10):2802–2825, 2017.

[34] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018.

[35] Stephen Zekany, Ronald Dreslinski, and Thomas Wenisch. Classifying ego-vehicle road maneuvers from dashcam video. *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2019.