DEEPCRASHTEST: Turning Dashcam Videos into Virtual Crash Tests for Automated Driving Systems

Sai Krishna Bashetty[1], Heni Ben Amor[1], Georgios Fainekos[1]

***Abstract*— The goal of this paper is to generate simulations with real-world collision scenarios for training and testing autonomous vehicles. We use numerous dashcam crash videos uploaded on the internet to extract valuable collision data and recreate the crash scenarios in a simulator. We tackle the problem of extracting 3D vehicle trajectories from videos recorded by an unknown and uncalibrated monocular camera source using a modular approach. A working architecture and demonstration videos along with the open-source implementation are provided with the paper.**

I. INTRODUCTION

Within the last decade the field of autonomous driving has seen tremendous progress, as evidenced by the many industrial and academic entities moving at a rapid pace to win the race for full autonomy. A number of companies are deploying fleets of vehicles on public roads to collect high quality data and, in turn, train their software to ensure safe and secure transportation without human intervention. Test vehicles need to be driven approximately 11 billion miles in the real-world or a simulated environment to verify with 80% confidence that they are 90 percent safer than human drivers [1]. Among the most difficult scenarios to train and verify are unpredictable human actions (as drivers or pedestrians) that may lead to dangerous situations or accidents.

Even though statistics [2] show that in most accidents involving autonomous vehicles (AVs), the human drivers are at fault, humans are still better at handling unpredictable and potentially dangerous driving situations. In order to ensure safety and improve public trust in AV technology, it is critical to train these systems with data that deviates from nominal road behavior, e.g., encounters on the road that may lead to accidents. The current methods for collecting vehicle data primarily consists of normal driving scenarios (i.e, without abnormal driving behaviors). Extending the approach to collecting abnormal driving data, however, would be a dangerous and unsafe endeavor that would put the well-being of the test driver and his/her environment at risk. This contradiction leads to the main dilemma addressed in this paper: collecting information about hazardous road interactions is of vital importance to train and validate perception and control architectures for AVs, but is in compliance with modern ethics and safety regulations.

[1]Sai Krishna Bashetty, Heni Ben Amor and Georgios Fainekos are with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University, 660 S. Mill Ave, Tempe, AZ 85281 {sbashett, hbenamor, fainekos} at asu.edu

This research was partially funded by NSF awards 1350420, 1932068 and 1361926, and the NSF I/UCRC Center for Embedded Systems.

Fig. 1: Top: Two frames from a crash video at two different time instants; Bottom: the same frames in simulation

A common approach to circumvent this problem is by generating manually engineered simulations of driving scenarios [3]. Other approaches attempt to re-generate scenarios from existing police reports [4]. However, police reports show substantial variability and often lack information about critical spatial and temporal details right before the crash.

In this paper, we propose to automatically synthesize a physics based simulation of a crash by extracting relevant information from a video stream. In particular, we focus on replicating hazardous, crash-inducing behaviors found in real-world crashes. Such information can be found in abundance on the internet [5], due to the proliferation of dashboard cameras. We develop a framework which can extract adversarial trajectories of vehicles from these kinds of videos and use them to automatically recreate the scene in a vehicle simulator. The simulations can be used to extract training and test data, or to study the driving behaviors leading to an accident. The process of extracting vehicle trajectories from data recorded with a single low quality unknown monocular sensor source is a challenging problem due to ambiguities in resolving perspective and extracting depths in monocular images.

A modular rather than an end-to-end approach is used in this paper that utilizes multiple existing deep learning components to solve individual sub-problems. The extracted trajectories are processed with custom algorithms to simulate them in the Webots simulation environment [6] using the Sim-ATAV framework [7] (see Fig. 1). Our contributions include: (i) a modular pipeline to extract 3D vehicle trajectories of the vehicles from dashcam videos, (ii) an algorithm for processing and simulating the trajectories in a vehicle simulator and (iii) DEEPCRASHTEST which is an add on to the Sim-ATAV testing framework [8]. We also provide the simulation videos in Webots along with a demonstration of

actual vehicle testing to extract safe/unsafe ego trajectories. The safe trajectories can be further analyzed to design the metrics and actions for collision avoidance systems.

II. RELATED WORK

For detecting traffic flow from aerial video feed, implicit or explicit model-based computer vision techniques and Bayesian filters for tracking can be used [9], [10]. Typically, such approaches require static sensors which must be initially calibrated. This assumption is orthogonal to our approach, since we want to extract detailed vehicle movement from a camera attached to the front of an (unknown) ego car. Inverse Perspective Mapping (IPM) [11] can also be used to extract car positions on the road, since it generates top view images from an ego perspective. Unfortunately, IPM behaves poorly on distant objects making it hard to extract 3D trajectories.

A closely related problem in the computer vision and robotics community is Multi-body Structure From Motion (SFM) [12]. Multi-body SFM methods can extract vehicle trajectories from a monocular camera, but they require proper motion model approximations. Another possibility is to estimate the 6-dimensional pose [13], [14] of vehicles in each frame and then combine it with existing 2D tracking methods to get 3D vehicle tracks. The drawback is that 6D pose networks are designed for indoor robotic applications with a static scene and calibrated sensors.

There are works in the deep neural network literature which are closely related to the problem posed here. Ren et al. [15] extract trajectories from static traffic cameras at an intersection using a pre-calibrated homography matrix mapping the image coordinates directly to simulator's top view. The problem of accurate vehicle trajectory extraction can be reformulated as an end-to-end 3D object detection and tracking. Akshay et al. [16] developed a real-time modular multi-object tracking system for autonomous vehicles which works with any combination of sensors. Markov Decision Process (MDP) are used in [17] to model a tracked object to improve long term tracking performance of agents.

Few methods like [18], a network is designed by leveraging the 3D pose estimation and 2D tracking information for joint detection and tracking of vehicles using monocular video frames. They resort to synthetic data to overcome the deficiency in training data. Scheidegger et al. [19] solves a similar problem by training a neural network to detect and estimate the distance to objects from a single input image.

We use a straight-forward, yet powerful modular approach with existing architectures which only needs a monocular camera source and generalizes well to arbitrary dynamic scenes. In our approach, the problem of absolute trajectory extraction is divided into monocular 2D-object tracking and 3D-bounding box estimation for agent vehicles along with monocular visual odometry, lane tracking, and camera calibration for ego views. We believe that our modular design approach resonates well with the frameworks found in application (e.g, NVIDIA's perception pipeline [20]).

III. PROBLEM DESCRIPTION

The main problem addressed in this paper can be formulated as follows: *Given a dashcam video captured using a monocular camera, extract the vehicle trajectories (including the ego vehicle's) and reproduce the trajectories in a vehicle simulator with a parameterization that enables further test case generation.* To solve the problem, we propose a modular architecture (a pipeline of deep neural network models) called DEEPCRASHTEST (see Fig. 2 for the main modules) under the following (technically necessary) assumptions:

1) Videos are short and generally under one minute in length. This is essential to avoid drift in position estimates over time because of monocular sensor effects.
2) The scene should be a straight or curved road as found in highways. Scenarios like intersections, have little information about the agent vehicles due to their short span of appearance with many occlusions.
3) It is assumed that the camera is attached to the ego car approximately at the center of the dashboard. This assumption can be relaxed with manual position adjustments of the initial simulation frame.
4) When the ego vehicle crashes, we only extract trajectories up to the collision time. After the crash time, the physics engine of the simulator takes over.

IV. TRAJECTORY EXTRACTION

Figure 2 demonstrates our pipeline to extract 3D vehicle trajectories and individual components are explained below.

A. Vehicle Detection and Tracking

For automated vehicle tracking, we use a combination of object detector and tracker along with standard association algorithms since object trackers need annotations of initial bounding boxes for the vehicles. There are many well known fast object detection networks such as [21], [22], [23], [24] but we chose Mask-RCNN [25] which is an instance segmentation method which simultaneously detects objects and estimates segmented masks. Region proposal based networks like Mask-RCNN are flexible with input image dimensions which is an important criterion for our choice. Moreover, instance segmentation is necessary for an alternate approach to extract 3D bounding box centers in sec. IV-C.

For object tracking, traditional trackers [26], [27] use Kalman filters which need proper camera calibrations and fine-tuning of the motion or sensor model is generally difficult. We use the deep-learning based Re3-Tracker [28] which is robust to occlusion and has real-time capabilities.

For data association between object detector and tracker to add or remove excess trackers in a frame, we use the Hungarian algorithm [29] with Intersection Over Union (IOU) of bounding boxes as the matching criteria. As in [30], we define threshold parameters to determine the minimum number of consecutive frames with consistent predictions for removing or adding excess trackers.

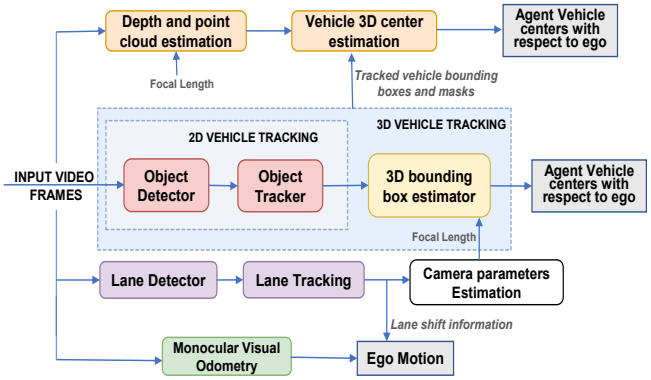


Fig. 2: Vehicle Trajectory Extraction Stage

B. 3D bounding box Detector

We can approximate a vehicle's trajectory (position and heading) by estimating it's sequence of 3D bounding box coordinates at each frame from the video. Using the frame rate of video, we can approximate the relative velocities of the agents with respect to ego at all time instants. We remark that there is a high possibility of variation in the original frame rate during processing and uploading of video to internet which does not guarantee the accuracy of absolute velocity estimates. We can however safely assume that the relative trend in velocity profiles between vehicles remains unaffected. This is found to be sufficient to ensure the timing of vehicle movements and recreate the crash scenario.

Over methods like [31], [32], [33], [34], we chose 3D-Deepbox network [35] because of its simplicity and good generalization of the approach to random images. They use a VGG-Net backend with additional fully connected branches to extract the bound box dimensions along with the global orientation of the vehicles. The 3D center of the bounding box with respect to camera is further optimized by placing geometric constraints on the images.

Camera calibration is required at this stage to extract real world distances. We observed that choosing a camera intrinsic close to the dataset which the network is initially trained on (KITTI Dataset [36] in this case) gives a satisfactory output on quite a number of random images. We review an existing method to approximate camera parameters from lane lines in sec. IV-F. The extracted trajectories are sent into the second stage of framework for simulation.

C. Point Cloud method for 3D vehicle centers

In the previous method for 3D center estimation, we have directly utilized the monocular 3D object detection networks. There is an alternative approach using depth maps of a scene for extracting only the 3D positions or specifically the distance and lateral position of the vehicle in a calibrated sensor environment with good quality images. Given a depth map, the 3D point cloud can be generated by estimating the real world (x, y, z) coordinates of individual pixels (u, v) considering the depth D of scene to be in the z direction using notation and equations as in [37] (see Fig. 3):

$$z = D(u, v) \quad x = \frac{(u - c_u) \times z}{f_u} \quad y = \frac{(v - c_v) \times z}{f_v} \tag{1}$$

where (c_u, c_v) is the principal point and (f_u, f_v) is the horizontal and vertical focal length of the camera in pixel coordinates. We assume the principal point to be the center of the image with equal horizontal and vertical focal length.

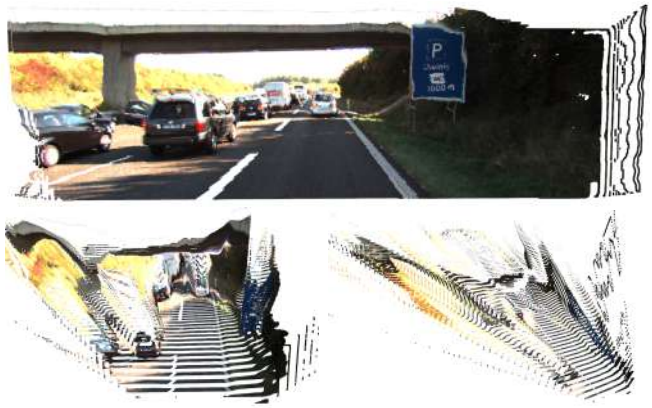


Fig. 3: Top: Front view pseudo point cloud of an image from the KITTI dataset; Bottom: Arbitrary views of the same point cloud; The point cloud is generated from the depth images estimated using a monocular depth extraction technique [38].

From the existing CNN architectures for monocular depth extraction [38], [39], [40], we use Monodepth [38] because of its longer range for accurate depth prediction. Initial instance segmentations from the mask-RCNN network are used to crop the 3D point cloud corresponding to a specific vehicle. We simply find the mean of these 3D points to approximate the vehicle's position. An important clarification to be made is that, we do not consider the 3D coordinates estimated using this approach to be the actual center of the vehicle, but just an approximate position in the scene. This is due to the fact that extracted point clouds belong only to the part of the vehicle which is exposed to the camera. However, the error in Euclidean distance between actual center and the estimated position does not exceed the dimensions of vehicle. Hence, the path formed by this sequence of 3D points at all frames provides information about the vehicle trajectory.

Other works on similar lines are [37], [41], [42]. Their methods are much more refined and accurate since they use additional network architectures for processing. We are using a simple method which is sufficient for extracting approximate positions. There are even end-to-end networks [43], [44] for joint monocular depth extraction and ego-motion estimation but we prefer the modular approach due to it's good generalization capabilities.

D. Ego Motion Estimation

For the simulations, we need the ego vehicle's motion or camera motion to find the absolute trajectories of the agent vehicles with respect to the world frame. In case of simple highway scenarios with the ego moving in a straight

line, we can assign constant velocity and a linear path for the ego vehicle. To generalize well to arbitrary scenarios, we resorted to monocular Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO), and Optical Flow based approaches to estimate the ego motion.

There are numerous works on Visual SLAM and VO and comparisons are provided by [45], [46], [47]. We evaluated several methods like DSO [48], LSD-SLAM [49], ORB-SLAM [50] and even Deep learning based approaches like Deep-VO [51]. We chose the former model based approaches because of their lightweight real time capabilities.

However, we found them to introduce scale ambiguity and, also, they cannot provide real-world depth estimates which further deteriorates velocity estimates. Florian Raudies et al. [52] present a survey of optical flow based ego-motion and depth estimation methods. We use the work by Andrew Jaegle et al.[53] where they introduce Expected Residual Likelihood (ERL) to remove outliers in optical flow estimates and a novel formulation of camera motion equation using lifted kernel optimization framework.

We observe that any of the above methods have stable values in the longitudinal direction of the motion of the vehicle, but they suffer from lateral drifts in the ego position. Currently, we are using lane detection and tracking methods (see next section) to localize the vehicles within lanes and prevent too much lateral drift on the road. Lane detection and tracking helps in camera calibration as well.

E. Lane Detection and Tracking

The Ego-motion estimator returns a qualitatively correct trajectory which may not be accurate enough to laterally localize the ego vehicle in the correct lane. The lane change information can even be used to completely replace the lateral ego-motion estimates in case of high drifting predictions.

Standard algorithms like the CHEVP [54] or the latest ones like [55] could be used for lane detection, but they fail when there are too many occlusions on the lane lines in high traffic scenarios and do not work well with partial lane markings. We use the deep learning based LaneNet [56] network which is a segmentation based approach comparatively more robust to partial or occluded lane markings. We use a density based spatial clustering algorithm (DBSCAN) along with B-spline curve fitting with the segmented outputs of LaneNet to cluster and assign individual lane IDs.

For lane tracking, there are existing methods (e.g, [57], [58]) involving kalman filters, but we use a simple approach where we fit straight lines through the clustered lane pixels and the lane fitting is completely in image pixel domain. We only need information about the lower part of the lane lines which is close to the ego vehicle to localize within the lanes. The Hungarian algorithm with directed Hausdorff distance [59] as a cost metric is used for association between detected and tracked lanes at each frame.

Considering the bottom left corner of the image as the origin, we find the x-intercepts of the lane lines in pixel coordinates. Since the perspective effects or distortions in an image increase only with depth of the scene, we can safely consider that the distance between any of the x-intercepts in image space is proportional to the actual distance between the corresponding lanes. Since, the average lane width on a specific road segment remains fairly constant, we can estimate the lateral position of the vehicle within the lanes.

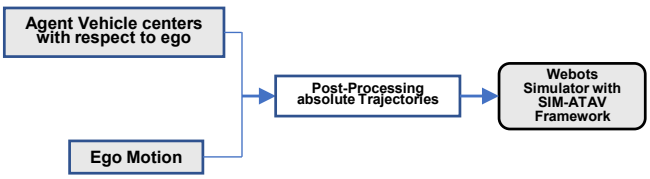


Fig. 4: Vehicle Trajectory Simulation Module

F. Camera Calibration

Camera calibration is required by the 3D bounding box detection, depth extraction network and optical flow based ego-motion estimation. Due to the monocular camera setting, it is very challenging to form a closed form solution using the image features. One approach is to use learning methods like [60][61] trained on existing datasets [62] to estimate the focal length given an RGB image.Fung et al. [63] demonstrate a more simple and light-weight approach which is the reformulation of perspective camera equations and using the lane line annotations to estimate the camera parameters. We use this method only when the predicted lane lines are pretty accurate and in the case of errors, we use the parameters from the datasets used for training the networks. We decide this based on the visual correctness of the final 3D bounding boxes when plotted on the video frames.

V. TRAJECTORY SIMULATION AND TESTING

For all the simulations, the global (or world) origin is defined at the initial position of the ego. The absolute positions of all the vehicles throughout the simulation are calculated with respect to this origin.

A. Trajectory smoothing

We use a Savitzky-Golay filter followed by a spline smoothing for the raw trajectories. Two levels of spline smoothing is used, first is local windowed with high smoothness factor and, then, a global fitting with low smoothing factor. This ensures an overall smooth curve while still preserving the sudden abnormal vehicle movements that lead to collisions. Moreover, splines provide the needed trajectory parameterization to use test generation frameworks such as sim-ATAV [7]. The extracted lateral localization from lane predictions is added to the corresponding ego position at each frame in this stage.

B. Extrapolating Vehicle Trajectories

The agent vehicle trajectories can be categorized based on their heading direction and the time of appearance or disappearance from the scene. Agent vehicles can either move in same or opposite direction of ego i.e, ongoing or oncoming, respectively.

The ongoing vehicles can be classified into three types:
D0T1: Agent vehicle is in the scene since the first frame.
D0T2: Agent vehicle enters later into the scene from behind i.e, agent overtaking the ego.
D0T3: Agent vehicle enters later from front in the scene i.e, a far off vehicle slowing down or pulling over or stopped by the roadside. This can even be considered as a case where the ego overtakes an agent vehicle which is initially far ahead from the ego.

The oncoming vehicles can be classified into four types:
D1T1: Agent vehicle already in scene since the first frame and the ego vehicle eventually passes it before the completion of the video sequence.
D1T2: Agent vehicle is in scene since the first frame and the ego vehicle *doesnot* pass it through the entire video sequence. This may happen when the agent vehicle collides with ego or the video clip ends before the agent reaches the ego.
D1T3: Agent vehicle enters later into the scene and the ego passes it.
D1T4: Agent vehicle enters later into the scene and the ego *doesnot* pass it.

This kind of classification is necessary for appropriate trajectory processing and is made purely based on our observations of the dashcam crash videos.

Processing Ongoing vehicle trajectories: D0T1 paths can be directly used without any processing, but the *D0T2* paths need extrapolation of path and velocity profiles. Delay needs to be added to *D0T3* vehicles such that the frame and position at which they appear in the scene with respect to ego is properly synchronized with the original video. For *D0T2* vehicles, until the frame where they appear in the scene, their initial positions and velocity profile are extrapolated such that the agent mimics the ego motion. Sufficient distance is maintained such that it does not collide with ego or does not appear in the scene. *D0T3* vehicles are initially far ahead of ego and appear much later in the video. To recreate this, we initialize agents at their initial position and then introduce some delay in the controller calculated based on frame of appearance and frame rate of video. In all the ongoing vehicle trajectories, if the ego overtakes other vehicles and the agent leaves scene before end of video, we extrapolate these trajectories until the end of the simulation following the road or lane at a velocity less than the ego.

Processing oncoming vehicle trajectories: The *D1T1* and *D1T3* vehicles are passed by the ego in the video and we do not have information once they move out of the scene. We extrapolate these trajectories such that they follow the road with a constant velocity after they pass the ego. *D1T3* and *D1T4* enter later into the scene and similar to *D0T3* vehicles, we generate initial start delays.

C. Generating Road Waypoints

Generating proper road structure is important for recreating plausible simulation. In most of the cases, since the crash happens on an highway, we can simply generate a straight road structure. However, it is also important to reproduce curved roads as observed in one of our video demonstrations which involves an oncoming collision on a sharp turning. We use the ego vehicle's qualitative trajectory as reference to generate the road structure and perform spline smoothing with a high smoothness factor. For oncoming collisions, the ego trajectory does not have any information about the road structure beyond the point of collision. However, it is necessary to extend the road beyond the collision point. One such example can be observed in the head on collision simulation in our video submission. Similar to the road structure generated using ego vehicle paths, we use the oncoming agent paths to extend the road by using spline interpolation to generate a continuous road structure. The entire process is automated in the framework. This approach does not guarantee accurate road estimation but is found to generate qualitatively correct simulations.

D. Generating Trajectories in Webots

Typically, in the available video clips, the vehicles move with non zero velocity from the initial frame. Since starting vehicles with non-zero velocities in Webots simulations does not result in convincing natural movements, we need the vehicles from zero initial velocity until all the vehicles reach the intended velocities and relative positions at the right time.

The time taken or distance traveled in a straight path to reach that target velocity under constant acceleration can be directly calculated from kinematic equations. We will refer to this as step-back time and step-back distance, respectively. Since the target velocities of all the vehicles where they first appear in the scene are different, each vehicle needs varying step-back times and distances. Additionally, we need to synchronize the relative positions of all the vehicles with those estimated in the initial frame of the video by the time they reach their respective target velocities.

To ensure this, we first determine the maximum of all step-back times $t^s_{max} = \max(t^s_1, t^s_2, ..., t^s_n)$, compute the total step-back distance D^s_i for each vehicle i with step-back time t^s_i as $D^s_i = d^s_i + (t^s_{max} - t^s_i) * v^t_i$, where d^s_i is the step-back distance for vehicle i to reach its target velocity v^t_i, estimated for its initial frame of appearance and n is the number of vehicles of interest to be simulated. We use the calculated distances as the length for extending the initial segment of the paths. The vehicles follow a straight path until they merge with their actual velocity and path profiles in the video. The part of the simulation which represents or recreates the actual video starts after time t^s_{max}.

Additionally, we provide visualization and editing tools for minor modifications to the lateral or longitudinal displacements of individual trajectory waypoints either to easily change the initial conditions or to ensure that vehicles do not overlap during initialization. We reiterate that since our target application is testing in a virtual environment which is going to be further modified, we do not need a quantitatively accurate reproduction of the real driving scenario. In other words, a visual inspection for qualitative correctness is sufficient for our application.

TABLE I: The table provides the modified (refer Sec. VI-B) KITTI tracking benchmark results on three KITTI [36] tracking sequences 3, 8, 10

Sequences	MOTA	MOTP	MODA	MODP	recall
SEQ 3	58.08 %	80.35 %	59.28 %	83.33 %	78.17 %
SEQ 8	67.36 %	79.55 %	67.36 %	84.54 %	72.08 %
SEQ 10	77.75 %	85.04 %	78.44 %	86.96 %	83.01 %

Sequences	precision	F1	TP	FP	FN	FAR
SEQ 3	81.03 %	79.75 %	265	62	74	42.75 %
SEQ 8	94.35 %	81.73 %	736	44	285	11.25 %
SEQ 10	94.90 %	88.56 %	484	26	99	8.81 %

Sequences	objects	trajectories	MT	PT	ML
SEQ 3	355	10	37.5 %	62.5 %	0.00 %
SEQ 8	796	21	47.61 %	38.09 %	14.28 %
SEQ 10	510	14	23.07 %	76.92 %	0.00 %

VI. EXPERIMENTS

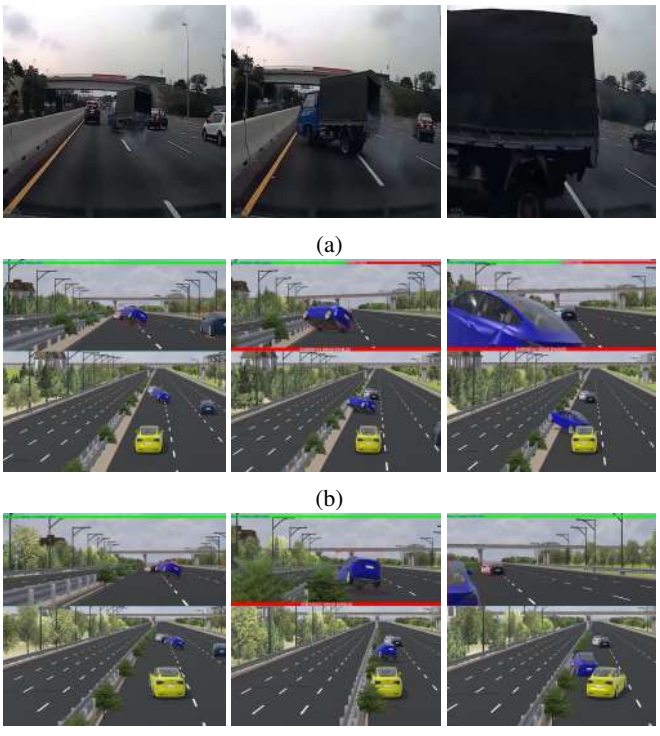
A. Qualitative Evaluation

Here, we describe our procedure for qualitative assessment of our framework through a typical usage of the extracted adversarial trajectories in a test environment for a collision avoidance system. We use a naive Automatic Emergency Braking (AEB) controller from the Sim-ATAV [7] framework. The AEB in sim-ATAV uses a sensor fusion module to perceive the environment by utilizing the sensors attached to the ego vehicle. In our tests, we activate the GPS/IMU, monocular dashcam and a single Radar sensor attached to the front of the ego vehicle to localize the agents in the scene.

Figure 5a shows three time ordered images of an original crash video. Fig. 5b includes the corresponding frames in a simulation generated using DEEPCRASHTEST on the original video. The second and third images in Fig. 5b show that though the AEB activates before the collision, it collides with the agent. This is because, the velocity of ego vehicle is very high (around 90 Kmph) and the distance to obstacle is very low compared to the required braking distance. At such velocities only way to completely avoid a collision is to steer away from the obstacle. To extract safe maneuvers of the ego vehicle which avoids a collision, we generated 128 simulations. For these simulations, we first manually sampled 32 initial positions within a box of 4x8(m) around the original position of ego. For each of these initial positions, we added 2D guassian noise to the spline control points of the original ego vehicle trajectory generating 4 random trajectories. Eight of the simulations did not have a collision. We picked four meaningful ego motions and Fig. 5c includes frames sampled from one of the safe simulation. As can be seen from the third image of 5c, before collision, the ego steers away from the agent thus avoiding collision.

B. Quantitative Evaluation

Even though, our application focuses on qualitative performance, we provide quantitative evaluations for the trajectory extraction module (see fig. 2). This is not straightforward since, DEEPCRASHTEST deals with an unknown dashcam video without ground truth, however, we provide evaluations by testing the framework with the KITTI tracking dataset [36]. The KITTI benchmark provides tracking evaluation



(a)

(b)

(c)

Fig. 5: (a): Three frames from a crash video at different time instants; (b): The corresponding frames in simulation with original trajectories; (c): The same frames in simulation with safe ego behavior without collision which is generated by randomly searching various initial conditions.

where the object positions are all relative to the ego vehicle but we want to evaluate the absolute trajectory positions by including the ego motion. For this, we added the ground truth odometry information to the ground truth tracking data to evaluate our framework. Table I provides the tracking evaluation results on vehicles using such a modified version for three tracking sequences from the dataset (sequences 3, 8, 10). The actual performance can be slightly lower because of errors in camera calibrations.

VII. CONCLUSIONS

We developed a framework called DEEPCRASHTEST for extracting 3D vehicle trajectories from dashcam videos uploaded to the internet. Currently, DEEPCRASHTEST is an add-on to the test generation framework Sim-ATAV [7] within which we recreate the extracted scenarios. DEEPCRASHTEST can be used for testing collision avoidance systems, or more generally autonomous vehicles. We demonstrated a specific use-case for extracting safe/unsafe vehicle trajectories within Sim-ATAV, but DEEPCRASHTEST could also be used as an add-on to a range of simulation-based automated test generation tools for AV, e.g., [3], [64], [65].

ACKNOWLEDGMENT

We thank Dr. Pavan Turaga (ASU) and Toyota Research Institute of North America for valuable technical discussions.

REFERENCES

[1] K. Nidhi and S. M. Paddock, "Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?" Santa Monica, CA: RAND Corporation, Tech. Rep., 2016.

[2] "https://www.axios.com/california-people-cause-most-autonomous-vehicle-accidents-dc962265-c9bb-4b00-ae97-50427f6bc936.html."

[3] D. J. Fremont, X. Yue, T. Dreossi, S. Ghosh, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: Language-based scene generation," arXiv:1809.09310, Tech. Rep., 2018.

[4] T. Huynh, A. Gambi, and G. Fraser, "AC3R: Automatically Reconstructing Car Crashes from Police Reports," in *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 31–34. [Online]. Available: https://doi.org/10.1109/ICSE-Companion.2019.00031

[5] "Youtube crash videos." [Online]. Available: https://youtu.be/WFKB9BxtZUs , https://youtu.be/vxqBS2-4puw

[6] O. Michel, "Webotstm: Professional mobile robot simulation," 03 2004.

[7] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1555–1562, 2018, to appear in Transacion of Intelligent Transportation Systems.

[8] [Online]. Available: https://cpslab.assembla.com/spaces/sim-atav/

[9] A. Jiří, B. A, H. D, and A. Tomas, "Automatic vehicle trajectory extraction for traffic analysis from aerial video data," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-3/W2, pp. 9–15, 03 2015.

[10] Zu Whan Kim and J Malik, "High-quality vehicle trajectory generation from video data based on vehicle detection and description," in *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, vol. 1, Oct 2003, pp. 176–182 vol.1.

[11] J. Yaghoobi Ershadi N, Menéndez JM, "D. Robust vehicle detection in different weather conditions: Using MIPM," *PLoS One. 2018;13(3):e0191355*, 2018.

[12] R. Sabzevari and D. Scaramuzza, "Multi-body Motion Estimation from Monocular Vehicle-Mounted Cameras," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 638–651, June 2016.

[13] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *CoRR*, vol. abs/1711.00199, 2017. [Online]. Available: http://arxiv.org/abs/1711.00199

[14] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep Iterative Matching for 6D Pose Estimation," *CoRR*, vol. abs/1804.00175, 2018. [Online]. Available: http://arxiv.org/abs/1804.00175

[15] X. Ren, D. Wang, M. Laskey, and K. Goldberg, "Learning Traffic Behaviors by Extracting Vehicle Trajectories from Online Video Streams," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Aug 2018, pp. 1276–1283.

[16] A. Rangesh and M. M. Trivedi, "No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & LiDARs," *CoRR*, vol. abs/1802.08755, 2018. [Online]. Available: http://arxiv.org/abs/1802.08755

[17] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track: Online Multi-object Tracking by Decision Making," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 4705–4713.

[18] H. Hu, Q. Cai, D. Wang, J. Lin, M. Sun, P. Krähenbühl, T. Darrell, and F. Yu, "Joint Monocular 3D Vehicle Detection and Tracking," *CoRR*, vol. abs/1811.10742, 2018. [Online]. Available: http://arxiv.org/abs/1811.10742

[19] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granström, "Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 433–440.

[20] [Online]. Available: https://developer.nvidia.com/drive/drive-software

[21] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[22] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[23] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[24] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving," *CoRR*, vol. abs/1612.01051, 2016. [Online]. Available: http://arxiv.org/abs/1612.01051

[25] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[26] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A Survey," *ACM Computing Surveys*, vol. 38, pp. 1–45, 01 2006.

[27] Y. Xu, X. Zhou, S. Chen, and F. Li, "Deep learning for multiple object tracking: a survey," *IET Computer Vision*, vol. 13, no. 4, pp. 355–368, 2019.

[28] D. Gordon, A. Farhadi, and D. Fox, "Re3 : Real-Time Recurrent Regression Networks for Object Tracking," *CoRR*, vol. abs/1705.06368, 2017. [Online]. Available: http://arxiv.org/abs/1705.06368

[29] H. W. Kuhn and B. Yaw, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[30] [Online]. Available: https://github.com/kcg2015/Vehicle-Detection-and-Tracking

[31] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation," *CoRR*, vol. abs/1711.10871, 2017. [Online]. Available: http://arxiv.org/abs/1711.10871

[32] X. Du, M. H. A. Jr., S. Karaman, and D. Rus, "A General Pipeline for 3D Detection of Vehicles," *CoRR*, vol. abs/1803.00387, 2018. [Online]. Available: http://arxiv.org/abs/1803.00387

[33] X. Ma, Z. Wang, H. Li, W. Ouyang, and P. Zhang, "Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving," *CoRR*, vol. abs/1903.11444, 2019. [Online]. Available: http://arxiv.org/abs/1903.11444

[34] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D Object Detection for Autonomous Driving," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 2147–2156.

[35] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D Bounding Box Estimation Using Deep Learning and Geometry," *CoRR*, vol. abs/1612.00496, 2016. [Online]. Available: http://arxiv.org/abs/1612.00496

[36] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the KITTI dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[37] Y. Wang, W. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," *CoRR*, vol. abs/1812.07179, 2018. [Online]. Available: http://arxiv.org/abs/1812.07179

[38] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," *CoRR*, vol. abs/1609.03677, 2016. [Online]. Available: http://arxiv.org/abs/1609.03677

[39] Z. Li and N. Snavely, "MegaDepth: Learning Single-View Depth Prediction from Internet Photos," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[40] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," *CoRR*, vol. abs/1806.02446, 2018. [Online]. Available: http://arxiv.org/abs/1806.02446

[41] X. Weng and K. Kitani, "Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud," *CoRR*, vol. abs/1903.09847, 2019. [Online]. Available: http://arxiv.org/abs/1903.09847

[42] Y. You, Y. Wang, W. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving," *CoRR*, vol. abs/1906.06310, 2019. [Online]. Available: http://arxiv.org/abs/1906.06310

[43] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised Learning of Depth and Ego-Motion from Video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6612–6619.

[44] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3D

Geometric Constraints," *CoRR*, vol. abs/1802.05522, 2018. [Online]. Available: http://arxiv.org/abs/1802.05522

[45] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," *Intelligent Industrial Systems*, vol. 1, 11 2015.

[46] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, 12 2017.

[47] J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," in *NA*, 05 2018.

[48] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *CoRR*, vol. abs/1607.02565, 2016. [Online]. Available: http://arxiv.org/abs/1607.02565

[49] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," September 2014.

[50] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *CoRR*, vol. abs/1610.06475, 2016. [Online]. Available: http://arxiv.org/abs/1610.06475

[51] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks," *CoRR*, vol. abs/1709.08429, 2017. [Online]. Available: http://arxiv.org/abs/1709.08429

[52] F. Raudies and H. Neumann, "A review and evaluation of methods estimating ego-motion," *Computer Vision and Image Understanding*, vol. 116, pp. 606–633, 02 2012.

[53] A. Jaegle, S. Phillips, and K. Daniilidis, "Fast, robust, continuous monocular egomotion computation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 773–780.

[54] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake," *Image and Vision Computing*, vol. 22, no. 4, pp. 269 – 280, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885603002105

[55] C. Lee and J. Moon, "Robust Lane Detection and Tracking for Real-Time Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 4043–4048, Dec 2018.

[56] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Towards End-to-End Lane Detection: an Instance Segmentation Approach," *CoRR*, vol. abs/1802.05591, 2018. [Online]. Available: http://arxiv.org/abs/1802.05591

[57] G. Lu, "A Lane Detection, Tracking and Recognition System for Smart Vehicles," in *NA*, 2015.

[58] N. Mechat, N. Saadia, N. K. M'Sirdi, and N. Djelal, "Lane detection and tracking by monocular vision system in road vehicle," in *2012 5th International Congress on Image and Signal Processing*, Oct 2012, pp. 1276–1282.

[59] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, Sep. 1993.

[60] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs, "DEEPFOCAL: A method for direct focal length estimation," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sep. 2015, pp. 1369–1373.

[61] H. Yan, Y. Zhang, S. Zhang, S. Zhao, and L. Zhang, "Focal length estimation guided with object distribution on FocaLens dataset," *Journal of Electronic Imaging*, vol. 26, no. 3, p. 033018, 2017.

[62] h. yan, "FocaLens dataset," May 2016. [Online]. Available: https://figshare.com/articles/FocaLens/3399169/2

[63] G. S. K. Fung, N. H. C. Yung, and G. Pang, "Camera calibration from road lane markings," *Optical Engineering - OPT ENG*, vol. 42, 10 2003.

[64] H. Abbas, M. O'Kelly, A. Rodionova, and R. Mangharam, "Safe at any speed: A simulation-based test harness for autonomous vehicles," in 7^{th} *International Workshop on Cyber-Physical Systems (CyPhy)*, 2017.

[65] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "Paracosm: A language and tool for testing autonomous driving systems," *arXiv preprint arXiv:1902.01084*, 2019.