CS231a Final Project Report
Adaptive cruise control - corner or change of lanes?

Jacob Donley Jung-Suk Lee Vladimir Tourbabin

github.com/jdonley/SingleCameraAdaptiveCruise

Abstract

Adaptive Cruise Control (ACC) is a feature in many vehicles from leading manufacturers that has become popular in recent years. There are cases where ACC fails in determining a speed correction when vehicles in front make corner turns or lane changes. In this work, we describe in detail the shortcomings of existing methods and propose several solutions using Structure From Motion (SFM), 2D keypoint skewness and deep-learning -based yaw prediction. A unique dataset is created for the specific problem and the proposed models are compared using a classifier on the dataset. We show that the proposed models can perform better than chance at predicting the correct control signal for the ACC under the challenging conditions.

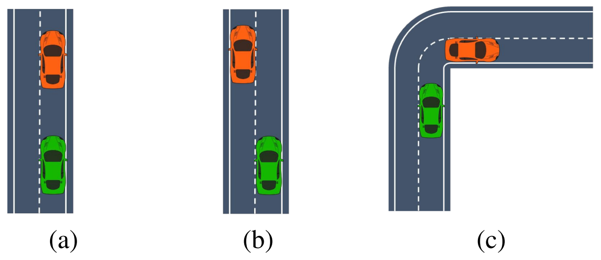


Figure 1: Illustration of potential leading vehicle dynamics: (a) green vehicle is trailing on a highway, blocked by the orange leading vehicle in front, (b) leading vehicle changes lanes allowing the trailing vehicle to safely accelerate, (c) leading vehicle going around a corner while staying on the same lane, trailing vehicle may appear to be unblocked and dangerously accelerate into the corner

.

1. Introduction

ACC is a popular feature of modern vehicles. In addition to the basic cruise control feature, which simply maintains the vehicle's speed constant, adaptive cruise control can also automatically slow down or accelerate based on distance from a leading vehicle.

ACC is popular and is recently being included as a standard feature in vehicles from major manufacturers. However, many existing implementations suffer from erroneous estimation of the leading vehicles dynamics, which can lead to potentially dangerous situations and contribute to increased fatigue or motion sickness.

One particular issue is related to miss-classification of the leading vehicle dynamics when turning around corners. To better understand the issue, consider a typical scenario schematically illustrated in Figure 1.a with vehicle trailing on a highway in ACC mode (green) and a leading vehicle (orange) blocking its way. In this situation, at least two alternative scenarios are possible, which would ideally require two different actions from the ACC system:

1. The leading orange car may change lanes, as illustrated in Figure 1.b. In this case, the green car is no longer blocked and can accelerate freely until it regains its target speed. **Desired action: accelerate.**

2. The leading orange vehicle enters a corner, as illustrated in Figure 1.c. In this case, the adaptive cruise control system of the trailing green vehicle may decide that the way is no longer blocked and dangerously accelerate when entering the corner. This is an existing problem with current implementations of ACC by major vehicle manufacturers. **Desired action: keep constant speed or slow down, do NOT accelerate.**

There are numerous potential approaches to addressing the issue described above. In this project, the goal was to explore feasibility of computer vision-based approaches to tracking the leading vehicle's orientation with relatively simple monocular camera systems. The main hypothesis is that orientation of the leading vehicle over time is sufficiently different between the two above scenarios, changing lanes and turning around a corner. Hence, accurately tracking the leading vehicle's orientation may help to accurately classify its maneuvers.

The approach we had taken in this work is detailed in the next section; it consists of several steps including keypoint detection, leading vehicle identification and tracking, orientation estimation using SFM, 2D skewness of keypoint distributions and 3D bounding box orientation estimates from Deep Learning (DL) models. In order to facilitate the development and statistical evaluation of performance, we had also created and annotated a small dataset specifically tailored to the 'corner' versus 'change of lanes' classification problem.

The remainder of the report is organized as follows. Next section introduces the technical approach, the processing pipeline we had followed, and the created dataset. Then, in Section 3 we present a detailed performance report and a discussion of the results. Conclusion and main learnings complete the report.

2. Technical Approach

In this section, we describe the techinical approach to solving the ACC control signal problem for leading vehcile lane changes and cornering. We provide descripton of a unique, and newly created, dataset. We also provide detailed descriptions of several proposed solutions.

2.1. Dataset

We have selected **BDD100k**[1] [7] as the dataset to work with for the project. **BDD100k** is a dataset that consists of 100k videos of driving captured from more than 50K rides. Each video is 40-second long and 30fps. For diversity, the dataset has abundant varieties not only in scene types including city streets, residential areas, and highways, but also in weather conditions. In order to support various complex single and multi tasks such as Lane detection, object detection, semantic segmentation, instance segmentation, panoptic segmentation, multi-object tracking, segmentation tracking, the dataset contains many useful relevant annotated assets.

We examined the dataset to ensure if it includes video clips that capture the situations that we are addressing. Videos were found showing not only scenes of vehicle moving straight ahead but also scenes of vehicles making turns behind another vehicles in diverse road situations, as seen in Fig. 2 and Fig. 3.

A new dataset was created, called *DS1* from hereon, using a subset of videos from the BDD100k dataset. DS1 was created with videos that contained a leading vehicle making either a left turn around a corner, right turn around a corner, a left lane change or a right lane change. 1000 videos were examined manually and annotated. The original, longer length, videos from the BDD100k dataset were clipped to lengths that contained only the maneuver of interest. The DS1 dataset consists of 62 video clips with 42 left and right corner turns and 20 left and right lane changes. A feature detector was used, called *Openpifpaf* [2], on the 62 video DS1 dataset and the output features were analyzed for validity. The output features consisted of vehicle keypoints for the leading vehicle and the feature detection failed for 34 of the 62 videos, leaving a final DS1 dataset size of 28 total clips with 19 corner turns and 9 lane changes. The final DS1 dataset is the one used throughout the analysis and evaluation of the models and classifiers mentioned later in this paper.

[1]https://bdd-data.berkeley.edu

Figure 2: A scene of a vehicle moving straight

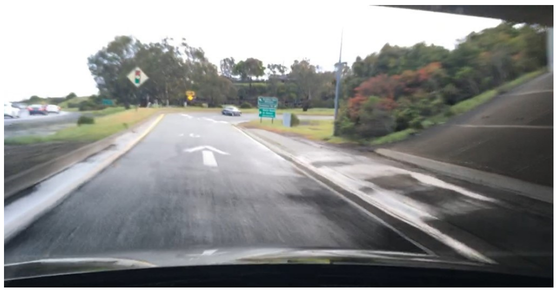


Figure 3: A scene of a vehicle making turn behind another vehicle

2.2. Processing Pipeline

2.2.1 Keypoint Detection and groundtruth bounding box

For vehicle keypoint estimation task, we employed *Openpifpaf* [2] framework. *Openpifpaf* is designed to detect keypoints of an object both spatially and temporally, thus capable of further estimating and tracking poses of the object by associating detected keypoints over space and time in a semantic way. Originally the framework was purposed for human pose estimation and tracking, but the main ideas generalize well to other types of objects, such as vehicles. We have verified that *Openpifpaf* works reasonably well with videos from **BDD100k** dataset when trained on ApolloCar3D Dataset [5]. As can be seen in Figure.(5), some im-

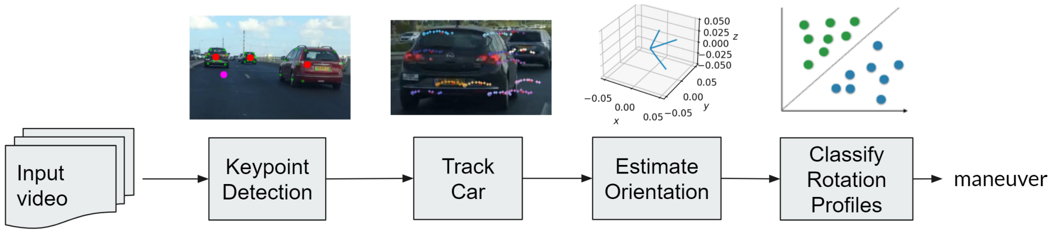


Figure 4: Adaptive cruise control pipeline.

portant keypoints relevant to a vehicle type of object, such as the edges of the license plate, top and bottom corners of the vehicle body and wheels are estimated.

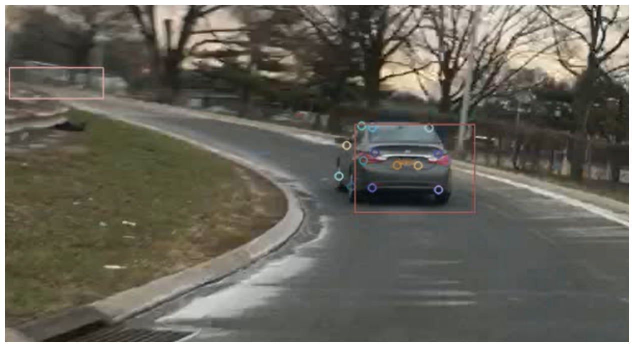


Figure 5: Keypoint detection of the vehicle in front.

In addition, we implemented importing groundtruth bounding box information provided in the dataset into our framework. The update rate of the bounding box in the data set is 5 times sporadic than the video rate, nearest neighbor scheme is employed to fill the gaps in between the video frames. This is to utilize the groundtruth information to evaluate our system later and to practice the exercise of fetching and leveraging groundtruth data. As seen in Figure.(5), the groundtruth bounding boxes are accurately located on the vehicles in the scene.

2.2.2 Keypoint Tracking

After keypoints are detected on each vehicle, we make assumptions about the target leading vehicle and then reduce the set of detected vehicles to a single leading vehicle. To do this, we first assume that the leading vehicle is in front (in the future, we may perform lane detection and choose the vehicle in the lane in front) and that the frontal direction is determined by an offset from the center of the image. Denote this hypothesized front vehicle location by $t \in R^2$. Next, the centroids of all detected vehicle keypoints are computed, c_n, $n = 1...N$, where n is index for vehicle and N is the total number of the vehicles. The smallest Euclidean distance to t is used to determine the vehicle in front, i.e. the leading vehicle, in the first frame. From then on, the tracking is performed using the clusters of detected keypoints, which are compared statistically to the history of the target vehicle's detected keypoints as further detailed below.

A running ring buffer of F frames (current implementation uses $F = 10$) is used to store history of the leading vehicle's keypoints locations. Denote the buffer by $\{f_i,\, i = k, k-1, ...k-F+1\}$, with $f_i = \{x_{i,1}, ...x_{i,M}\}$ holding the set of M keypoints in frame i. From the running buffer, the expected value is determined for each individual keypoint on the target vehicle:

$$\bar{x}_j = \frac{1}{F} \sum_i x_{i,j}. \tag{1}$$

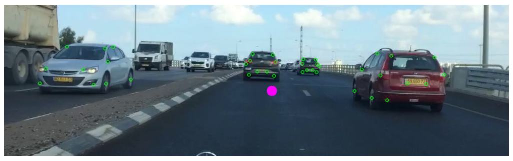
Then, the vehicle in front is tracked by selecting the one that has the keypoints closest to the set $\{\bar{x}_j\}$. In order to compute this, we denote the set of vehicles detected in the current frame by $\{v_n,\, n = 1...N\}$. Each detected vehicle contains M key points: $v_n = \{y_{n,1}, ..., y_{n,M}\}$. Distance of each detected vehicle from $\{\bar{x}_i\}$ is defined as

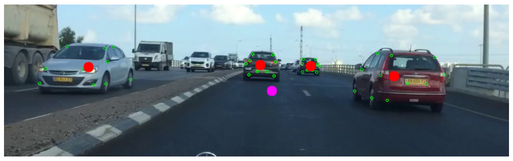
$$d_n = \frac{1}{M} \sum_j \|\bar{x}_j - y_{n,j}\|^2 \tag{2}$$

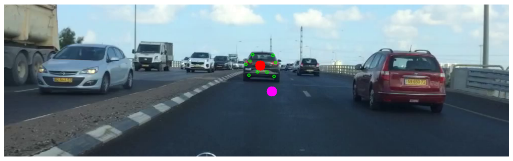
and the vehicle of interest is selected as $n^* = argmin_n\, d_n$. Finally, the circular history buffer gets updated with v_n and the process repeats with the next frame. All missing or undetected keypoints are excluded from the expected value estimates. Outlier keypoints and clusters are removed for each frame when exceeding a pre-determined threshold distance in pixels (e.g. 20 pixels). The history is not updated when there is no resulting vehicle keypoints that meet the criteria.

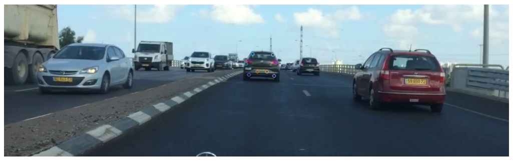
2.2.3 Vehicle orientation estimation with SFM

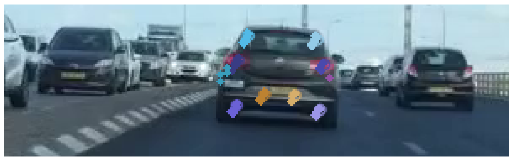
The previous step generates a list of length M that contains all detected keypoints belonging to the leading vehicle. Each entry in this list is a set of keypoints that were detected in a particular video frame number $i = 0, ... M-1$. Note that each particular frame may contain different number of detected keypoints. Hence, as a first preprocessing step, a


(a) Step 1: Predict keypoints for vehicles.


(b) Step 2: Determine centroids of keypoints for each vehicle.


(c) Step 3: Compute distance of centroids to initialization point.


(d) Step 4: Begin tracking of closest vehicle keypoints.


(e) Step 5: Update keypoint history for each frame.

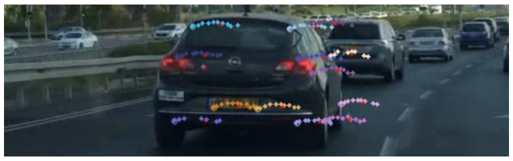

(f) Step 6: Track vehicle with smallest keypoint error to expected value.

Figure 6: The figures above show the process of initializing and tracking a leading vehicle to obtain keypoints that drive statistics in downstream tasks, such as structure from motion. The intialization point, t is magenta dot, the initially predicited vehicle keypoints are green circles, the vehicle keypoint centroids $\bar{x}_i$, are red dots and the leading vehicle keypoints are qualitatively colored circles based on their location (various colors). The smaller points show the ring buffer history sample points and the small red points are the ring buffer vehicle keypoint centroids.

set of keypoints was computed by selecting only the keypoints that are common to all frames in the video. Limitations of this approach as applicable to our problem are discussed in the next section. This step resulted in the final set of keypoints denoted by $\mathcal{K} = \{x_{i,j}, i = 0, ..., M-1; j = 0, ..., P-1\}$.

The set of keypoints is then fed into the Tomasi-Kanade factorization-based SFM algorithm [6]. The algorithm results in two matrices $R = [R_0^T \, R_1^T \, ..., R_{M-1}^T]^T$ and S, with R_i being the rotation matrix corresponding to frame i and S holding the structure, i.e. 3D position of the tracked keypoints relative to their centroid coordinate system.

In order to resolve the affine ambiguity inherent to the decomposition approach, metric constraints were imposed, as suggested in the original paper. These constraints insure that the resulting axes of all camera coordinate systems are orthogonal and have unity norms:

$$I_i^T QQ^T I_i = 1, i = 0, ..., M-1, \quad (3)$$

$$J_i^T QQ^T J_i = 1, i = 0, ..., M-1, \quad (4)$$

$$I_i^T QQ^T J_i = 0, i = 0, ..., M-1, \quad (5)$$

where Q is the affine transformation to be applied to all camera matrices in order to impose the metric constraints and I_i, J_i, K_i are the estimated rotation camera matrices, such that $R_i^T = [I_i \, J_i \, K_i]$. The constraints in Eqs 3-5 result in $3M$ equations leading to an over-determined quadratic system. We solve the system for Q by using the `minimize()` function from `scipy.optimize` toolkit. As an initial point for the optimization, we feed in a least-squares solution of the problem with norm constraint only (only using Eqs 3-4) obtained by solving for QQ^T and by applying Cholesky decomposition to obtain Q. Note that solving the problem for QQ^T using least-squares with all three constraints did not lead to a desired result. This is because the least squares approach does not generally lead to a positive-definite QQ^T, hence not having a straightforward solution for Cholesky decomposition.

Finally, yaw angles of all rotation matrices R_i are computed w.r.t to the first frame. These angles capture the camera rotation with respect to the tracked vehicle and vice versa. Hence, the angles are used as the primary feature for tracking vehicle orientation. The results are reported and discussed in Section 3.

2.2.4 Vehicle orientation estimation with 2D skewness

As the orientation of the leading vehicle changes, the distribution of the estimated keypoints changes in a correlated way. For example, if the leading vehicle is in left turn, the distribution of the keypoints associated with the rear side of the vehicle would likely be skewed to the right and the keypoints associated with the left side of the car may be

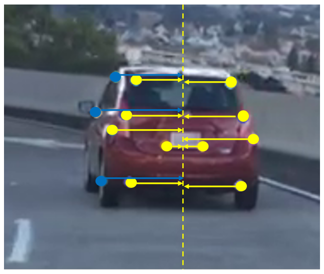


Figure 7: Yellow dots, blue dots and the dashed vertical line represent $K_{paired,rear}$, K_{left} and the horizontal centroid of $K_{paired,rear}$, respectively.

seen. At each frame of a video, the rear side keypoints that are left-right paired are grouped as $K_{paired,rear} = [x_{pr,1}, \cdots, x_{pr,N_{pr}}]$ and the left side and right side keypoints are grouped as $K_{left} = [x_{l,1}, \cdots, x_{l,N_l}]$, $K_{right} = [x_{r,1}, \cdots, x_{r,N_r}]$, respectively.(Fig.7) The horizontal centroid of $K_{paired,rear}$ is computed and the sum of the horizontal distances from all the points left and right to this centroid are computed as $D_{left} = \sum^{N_l+N_{pr}/2} |x_{l,1} - x_{centroid}|$, $D_{right} = \sum^{N_r+N_{pr}/2} |x_{r,1} - x_{centroid}|$, respectively. The decibel of the ratio of, $20log_{10}(D_{left}/D_{left})$ is used as a statistic to estimate the vehicle orientation. If the statistic is higher or lower than the pre-defined threshold by certain margin, the frame is labeled as 'left' or 'right'. If the statistic is near the threshold, then being labeled as 'straight'. 'left','right' and 'straight' indicates the orientation from the camera perspecitve.

2.2.5 Vehicle orientation estimation with DL 3D bounding boxes

While the SFM and 2D skewness approaches estimate orientation using a mix of signal processing and deep learning based methods, e.g. from DL based keypoint detection, we also investigate a completely DL approach by using keypoint, 2D bounding box and 3D bounding box DL estimators. The method uses the keypoint detection and tracking based on the *Openpifpaf* [2] model, as described in 2.2.1 and 2.2.2, to determine the target leading vehicle of interest. A 2D bounding box estimator from *Openpifpaf* that is trained on the *NuScenes* dataset [1] is used to crop the leading vehicle image. We then feed the cropped image of the leading vehicle into a trained 3D bounding box estimation model, known as *Deep3DBoX* [3].

The *Deep3DBoX* model returns a local orientation angle, θ_l, of the vehicle as shown in Fig. 8. The model is a Convolutional Neural Network (CNN) that is trained by regressing the 3D bounding box orientation and dimensions. The model uses features from the VGG 19-layer pre-trained model with batch normalization [4]. The network consists

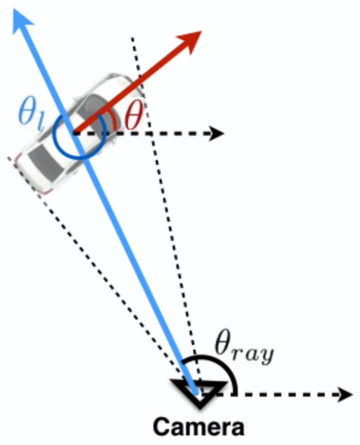


Figure 8: The angles required to compute the leading vehicle's yaw angle when using the *Deep3DBoX* model [3]. θ is the yaw angle, θ_{ray} is determined from the projection matrix and 2d bounding box center, and θ_l is the local orientation predicted from the model.

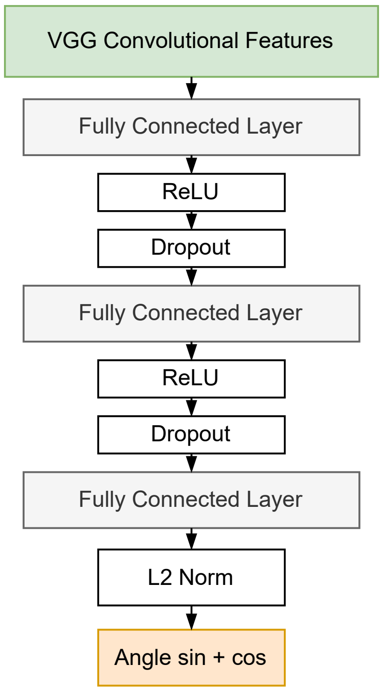


Figure 9: The Deep3DBoX network architecture.

of three fully connected layers with ReLU's and dropout layers with the final output computed from from sin and cos angles as shown in Fig.9

In order to compute the yaw angle, θ, of the leading vehicle we also need to calculate the relative angle to the vehicle, θ_{ray}. The relative angle to the vehicle can be calculated by using the projection matrix and the center location of the estimated 2D bounding box. We can then calculate the yaw angle using the predicted local orientation and the relative angle to the vehicle simply as,

$$\theta = \theta_l + \theta_{ray}. \tag{6}$$

The yaw angle is then used to determine the leading vehicles rotation profile over a given duration. The rotation profile is then continuously fed to a classifier to determine the correct maneuver for the ACC to make.

3. Results and Discussion

In this section, we discuss the results from the SFM, 2D skewness and 3D deep-learning yaw models that are proposed in this work. We show evaluation and classification results on the DS1 dataset.

3.1. SFM

The SFM method described in Section 2.2.3 was applied to the videos in DS1 dataset. below are a few interesting examples.

Fig. 10 an successful example of SFM-based yaw tracking of the leading vehicle orientation. It can be seen that only keypoints on the rear face of the car were identified and tracked. It can be seen that yaw increases when the leading car enters the curve and then keeps more or less constant when the trailing car follows into the curve. It is interesting to consider the first three singular values of the measurement matrix: $[84076723]$. It can be seen that the third dimension contains very little energy, which probably reflects the fact that all the identified keypoints are almost co-planar as a result of the rear end car geometry (hatchback).

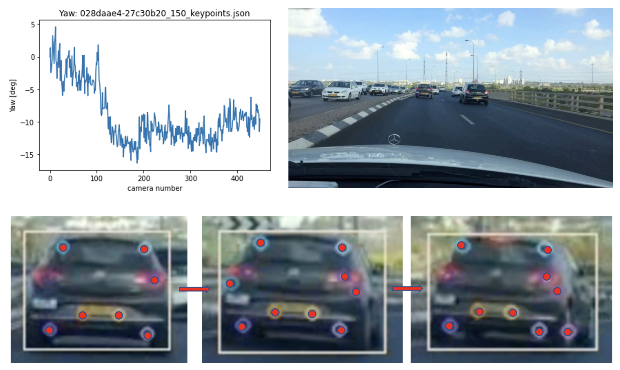


Figure 10: Example of successful SFM-based estimation of yaw angle of a leading car.

An unsuccessful SFM example is shown on 11. In this video, the leading car performs left lane change. However, the estimated yaw angle is almost 0 and does not reflect any relationship to the actual car maneuver. An insight into this behaviour can be gained by again considering the first three singular values of the measurement matrix: $[24295001e - 12]$. The third singular value is practically 0 implying that there is no 3D information captured. it is worth than the previous example because less common keypoints were identified and also probably hindered by a very close to planar rear-end geometry.

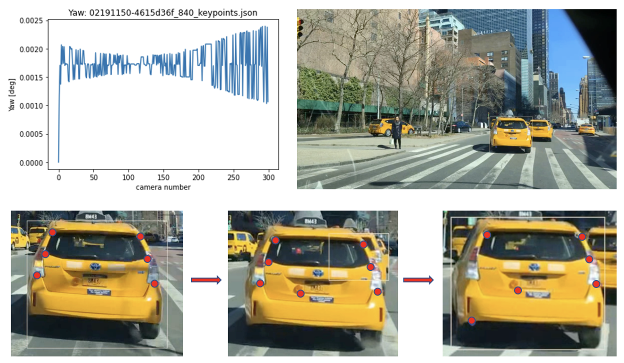


Figure 11: Example of unsuccessful SFM-based estimation of yaw angle of a leading car.

An even more interesting example is shown in 12. In this example, the pipeline erroneously tracked a vehicle on the side, which has two of its orthogonal faces visible to the camera. It can also be clearly seen that a handful of keypoints was successfully detected on both faces. It is therefore not surprising that the ratio of the third and the second singular values in this case is about 0.5; significantly better than in the previous two example.

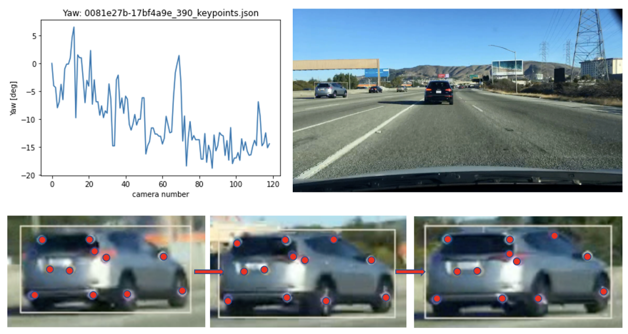


Figure 12: Example of a successful SFM-based estimation of a car on the side with two visible faces.

Following the typical examples shown above, a simple heuristic was used to exploit the SFM-based orientation estimation for maneuver classification. The maneuver was considered to be a ”turn” if the yaw angle is sufficiently large - 5 [deg] or more. Otherwise, the maneuver was classified as a lane change.

3.2. 2D Keypoint Skewness

The 2d keypoint skewness was used to detect the vehicle's orientation on a frame-by-frame basis. In Fig.(13), it can be seen that the value of the statistic starts increasing around the 150th frame which corresponds to the beginning of the left-turn maneuver of the leading vehicle. Before this point, the statistic was consistent, mostly being around 0 ,which corresponds to the leading vehicle moving straight. Around the 250th frame, the statistic value starts decreasing as the leading vehicle exiting the corner and staring moving straight again. By properly thresholding the statistic value, one could obtain the estimate of the vehicle orientation. The estimation becomes more accurate when the keypoints from either the left or the right side of the vehicle appear, breaking left-right distance ratio symmetry. The major drawback of this method is its heavy dependency on the keypoint estimation quality. If the keypoints estimate is erroneous, this method fails accordingly.

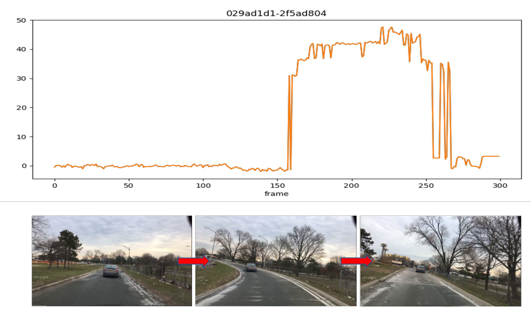


Figure 13: Time-evolution of the statistic used to estimate vehicle orientation over frames.

3.3. DL 3D Bounding Boxes

The deep 3D bounding box outputs can be seen annotated on the video images in Fig. 14. On close inspection it can be seen that that the 3D box closely matches the cars orientation in the examples. By tracking the yaw of the 3D box we can determine the rotation characterities of the car as it enters and holds the corner.

The approach to the corner is often seen through the dataset as a constant yaw angle and when entering the corner the yaw angle rotates either left or right, depending on the angle of the corner. Often, the leading car will enter the corner with a large relative yaw that is soon after reduced as the trailing car follows into the corner behind. This can be seen in the example in Fig. 15. After both cars enter the corner there is a period where the yaw and cornering is sustained. In the particular example shown here, the leading vehicle drifts slightly within the lane during the corner and then makes a correction towards the end of the video.

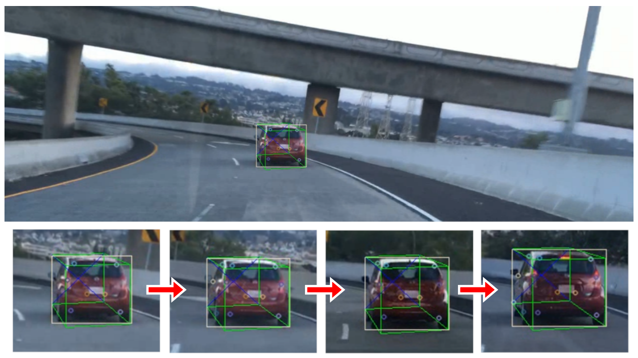


Figure 14: Detected Keypoints, 2D bounding boxes and 3D bounding boxes from inference using the deep learning models of *Openpifpaf*, *Openpifpaf+NuScenes* and *VGG+Deep3DBox*, respectively

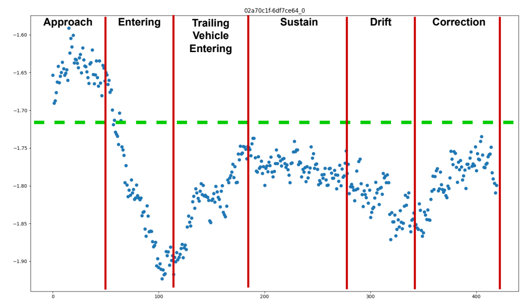


Figure 15: Complete deep learning yaw estimate for vehicle heading straight, entering corner, sustaining turn. The yaw estimates are from the video of Fig. 14. The horizontal green dashed line shows an appropriate threshold for the corner detection.

3.4. Maneuver Classification

A rather rudimentary classifier was put together in order to classify the maneuver of the leading vehicle into two categories: 1) corner-turning or 2) lane-change. The classifier combined two types of inputs: a yaw estimate and the 2D keypoint skewness. Here, the DL 3D bounding boxes-based yaw angle estimate was used, as it demonstrated a higher correlation to true vehicle orientation than the SFM-based method.

The basic heuristic behind the classifier design is the complementary nature of the two inputs. The skewness estimate can provide information about vehicles orientation to disambiguate between the conditions of the vehicle being tilted left, right, or straight. However, it is not sufficient to disambiguate between the two maneuvers, as both can contain varying patterns of rotation. Using the yaw angle estimate can help to reduce this ambiguity. In the case that

	Corner turn	Lane change	Overall
Accuracy	17/19	1/9	18/28

Table 1: Accuracy of the classifier on the DS1 dataset.

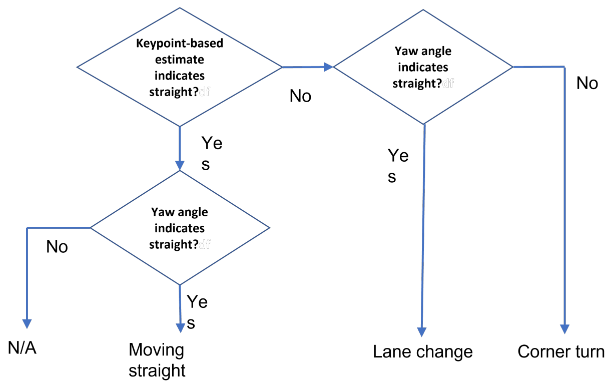

Keypoint-based estimate indicates straight?
No
Yaw angle indicates straight?
No
Ye s
Ye s
Yaw angle indicates straight?
No
Ye s
N/A
Moving straight
Lane change
Corner turn

Figure 16: Logic used to classify the maneuver of each frame in a video.

the yaw angle indicates that the leading vehicle is oriented straight, while the skewness-based orientation indicates the left or the right side, it is likely that the vehicle is moving straight on the side lane indicating a lane-change. On the other hand, in the case where the yaw angle estimate indicates left or right orientation with the skewness-based estimate also indicating a tilted orientation, the vehicle is likely to be turning a corner. This way the aforementioned ambiguity can be resolved, resulting in a classifier that combines both types of input.

The classifier was applied to the DS1 dataset in a batch fashion such that the keypoint-based orientation estimates and the yaw angle estimates over the entire frames for a video are collected first and than fed to the classifier. At each frame, classification into one of the four categories described in the flowchart in Fig. (16) takes place and the category that earns most votes over the entire frames of the video becomes a representative vehicle maneuver for the video. The result of the classification in terms of accuracy is given in the Table.(1)

It can be seen that the proposed classifier gives roughly 64% $(18/28)$ accuracy. This performance is better than chance, but far from being sufficient in relation to driving safety, hence further work on this topic is needed to ensure satisfactory performance.

4. Conclusion

In the current project, we had investigated an issue with ACC feature of modern vehicles related to miss-classification of leading vehicle dynamics, which can lead to erroneous car behaviour and increased risk on the road. The problem boils down to classification of the leading vehicle maneuver into two classes, ”turn around a corner” or ”change of lanes”. We had proposed a computer vision-based pipeline using relatively simple monocular camera system that can help improve classification accuracy. In particular, three different methods for car orientation estimation were examined: (i) SFM-based, (ii) deep neural network-based bounding box orientation estimation, and (iii) 2D keypoint skewness. It was shown that, while each of the three individual approaches were not sufficient to produce a consistent car maneuver classification, fusing the approaches together can improve classification accuracy to around 64%. This result should be seen as only a viability investigation, while much more work can be done to improve generalization and consistency. Future work can include fusion of additional sources of information such as lane marks and leading vehicle trajectory.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[2] S. Kreiss, L. Bertoni, and A. Alahi. OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, March 2021.

[3] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[5] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, and R. Yang. Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving. *CoRR*, abs/1811.12222, 2018.

[6] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 9(2):137–154, 1992.

[7] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.