Supplementary Material: Self-Supervised Steering Angle Prediction for Vehicle Control Using Visual Odometry

Qadeer Khan[1,2] **Patrick Wenzel**[1,2] **Daniel Cremers**[1,2]
[1] Technical University of Munich
[2] Artisense
{qadeer.khan, patrick.wenzel, cremers}@tum.de

1 Global to Local Frame Transformations

Figure 1, shows two trajectories generated using the same steering commands from two different starting positions. Note that in the global frame of reference, the two paths despite having the same steering commands produce different relative translation vectors. This is due to the fact that the 2 trajectories are oriented differently in the global frame of reference.

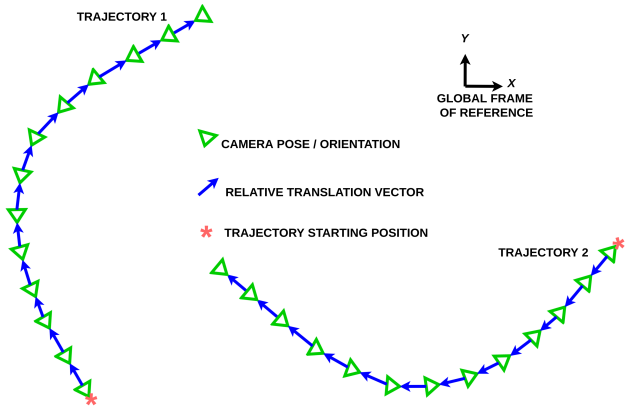

Figure 1: This figure shows two trajectories traversed by the ego-vehicle in the global frame of reference. Despite executing the same sequence of steering commands, the corresponding relative translation vectors point in different directions.

To deal with this, we redefine the vectors to a local frame of the reference such that the forward direction is always in the x'-direction and the lateral movement is defined in the y'-direction.

Note that the relative translation vector for a previous image gives the direction of motion of the car for the current time step. V^t and V^{t+1} are the relative translation vectors for the previous and current image in the global frame of the reference, respectively. We have already defined the local frame of reference such that the direction of the current motion of the car is in the x'-direction. Hence, in this new local frame, V^t will be aligned with the x'-axis. The rotation matrix which aligns V^t in the x'-direction can be formulated as:

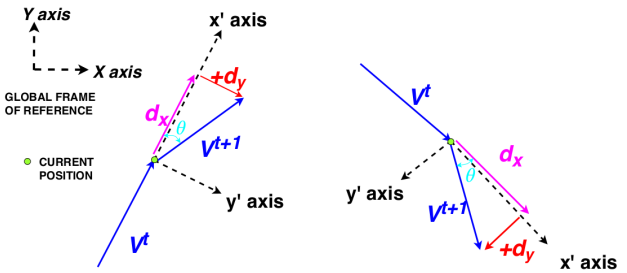

Figure 2: This figure demonstrates the effects of retrieving the relative position vector in the local frame of reference. V^t and V^{t+1} represents the relative position vectors at time t and $t+1$, respectively in the global frame of the reference. Note that despite the same lateral movement of the car in the 2 examples, these vectors have different coordinates. However, after projecting them in their respective local frames of reference, the new vectors dx and dy have the same values in the 2 cases.

$$\frac{1}{||V^t||} \cdot \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} V_x^t \\ V_y^t \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1}$$

After having solved for the rotation matrix, we can multiply the vector V^{t+1}, to get dx and dy indicating the forward and lateral movements in the local frame of reference.

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} V_x^{t+1} \\ V_y^{t+1} \end{bmatrix} = \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{2}$$

Figure 2 shows the effect of this transformation. Here, 2 examples with different vectors V^t and V^{t+1} (corresponding to the same steering command) in the global frame of reference have the same dx and dy in their respective local frame of reference. Alternatively, the angle θ can be determined from the cosine similarity between the vectors

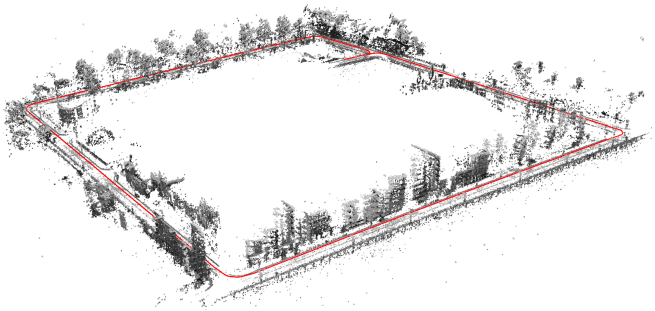

Figure 3: This figure shows a point cloud generated with visual odometry. We use the estimated poses (in red) to train our models.

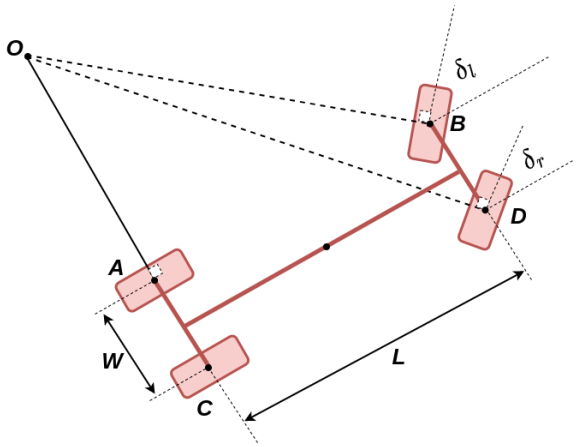

Figure 4: Depicts the steering angles of the 2 front wheels which are different as the car executes a turning maneuver. The 2 front wheels are oriented in a manner such that they have the same instantaneous center of rotation.

V^t and V^{t+1}. The sign of dy can be found by determining which side the vector V^{t+1} is with respect to the line formed by the vector V^t. We train a neural network to predict the lateral movement dy, by minimizing the $L1$ loss between the prediction and this calculated label.

2 Data Collection and Testing

The CARLA simulator [Dosovitskiy et al., 2017] data used for training was collected by running the car in autopilot mode at a frame rate of 30fps. The car is controlled by adjusting the throttle and steering command. The throttle influences the speed whereas the steering command controls the steering angle of the car. The throttle ranges between a value of 0 and 1. The car is at rest when the throttle is at zero and moves faster as it is increased to a maximum value of 1. In the autopilot mode, the mean throttle value is around 0.5. Moreover, the average speed at which the car executes the turn is around 20 km/h. This is within the range of values at which the car does not slip and therefore also matches with our assumption described in Section 2.2 of the main paper. The steering command varies between -1 and -1 with 1 corresponding to 70°.

Visual Odometry:

During the data collection phase images of size 512×512 pixels are recorded along with the corresponding ground truth steering angles. Note that this ground truth steering angle data is only used to train the Oracle (supervised model). Whereas our model is trained with the visual odometry camera poses. Figure 3 shows the trajectory of the estimated poses (in red) and the resulting point cloud generated when running Stereo DSO [Wang et al., 2017].

Front Wheel Steering:

Note that in the bicycle model described in Section 2.2 the left and right front wheels were both modeled by a single front wheel with a steering angle δ. As depicted in Figure 4, while executing a turning maneuver the left and right front wheels, will have slightly different steering angles denoted by δ_l and δ_r for the same instantaneous center of rotation O. When the car is making a left turn, than $\delta_l > \delta_r$ and vice versa, i.e., the steering angle of the inner front tyre would be greater than that of the outer front tyre. The difference can be approximated to be [Rajamani, 2012]:

$$\Delta\delta = \delta^2 \frac{W}{L} \tag{3}$$

Where, L is the length of the wheelbase, and W is the track width of the car and

$$\delta = \frac{\delta_l + \delta_r}{2}. \tag{4}$$

The Ackermann steering mechanism [Zhao et al., 2013] can be used to ensure that the $\Delta\delta$ between the two front tyres is maintained while the car is making a turn. The CARLA simulator already caters for this difference and no additional correction needs to be performed. Figure 5 depicts the difference in steering angles of the front tyres when the inner tyre is at a maximum of 70°

Model Architecture:

Note that this method uses stereo visual odometry, thereby also giving the notion of scale. However, the neural network model only requires a single image to predict the lateral component of the translation vector for the next frame at a fixed distance of dx apart. The architecture of the network is described in Figure 6. The image is downscaled to a lower resolution of 128×128, thereby simplifying the architecture. Note that the model comprises of a Feature Extraction Module (FEM) and a Steering Angle Prediction (SAP) Module. The FEM is a series of Convolution, Maxpooling and ReLU activation layers. Meanwhile, the SAP has 2 fully connected layers with one ReLU

Figure 5: Depicts the orientation of the 2 front wheels when the wheel towards the inner circle of the radius of the turn is at the maximum value of approximately 70°. Note that the outer wheel has a lower turning angle since it has to cover a larger curvature distance. Figure is taken from [Subirón, 2018].

activation in between. The training was done with an initial learning rate of 0.0001 using the Adam optimizer [Kingma and Ba, 2015].

Testing at higher speeds:

The equations derived in Section 2.2 of the main paper were based around the critical assumption that the car is moving at moderately low speeds. A car turning at 5 m/s will be subjected to very low lateral forces on the tyres and hence experience negligible slipping [Rajamani, 2012]. While this assumption is reasonable in many urban scenarios, it may not hold in other circumstances. Therefore, we would like to assess how our models will behave if this assumption does not hold when turning. We successively enhance the throttle of the car, which leads to an increase in speed. The online performance of the models trained with 2, 4, 6, 8, and 10 trajectories is reported. This is depicted in Figure 7.

It can be observed that as the throttle is increased, so is the mean speed. Models trained with more trajectories are more robust to the speed than the ones trained with less number of trajectories. At a throttle value of 0.7, when the mean speed of the car is 7.5 m/s the models trained with 6 or more trajectories still maintain the same performance. Moreover, at a mean speed of around 9 m/s the performance of these high trajectory models only drops by about 10%. This is despite an increase in speed by approx. 80% from the assumption of 5 m/s. On the other hand, models trained with fewer trajectories show a dramatic drop in performance as the speed is successively increased. This demonstrates that training with more trajectories tends to be more robust in performance as the speed deviates farther away from our assumption. Nevertheless, an increase in the mean speed beyond 9 m/s leads to a significant drop in performance even for the models trained with a greater number of trajectories. To cater for this limitation, the lateral dynamics of the car would also need to be incorporated into the model to enable the car to perform stable high-speed turning maneuvers. While this is beyond the scope

FEATURE EXTRACTION MODULE			
Layer Number	**Layer Type**	**Layer Input**	**Layer Output**
1	*Convolution*	3 x 128 x 128	30 x 62 x 62
2	*Maxpool*	30 x 62 x 62	30 x 31 x 31
3	*ReLU activation*	30 x 31 x 31	30 x 31 x 31
4	*Convolution*	30 x 31 x 31	30 x 14 x 14
5	*Maxpool*	30 x 14 x 14	30 x 7 x 7
6	*ReLU activation*	30 x 7 x 7	30 x 7 x 7
7	*Convolution*	30 x 7 x 7	30 x 2 x 2
8	*Maxpool*	30 x 2 x 2	30 x 1 x 1
9	Squeeze Dim.	30 x 1 x 1	30
10	Concatenate Conditional	30 + 3	33
STEERING ANGLE PREDICTION MODULE			
Layer Number	**Layer Type**	**Layer Input**	**Layer Output**
11	*Fully Connected*	33	30
12	*ReLU Activation*	30	30
13	*Fully Connected*	30	1

Figure 6: The convolution layers numbered 1, 4, and 7 have a kernel size of 5, with stride 2, and no additional padding. The number of kernels in each of these convolution layers is 30. The max pooling layers numbered 2, 5, and 8 have a kernel size of 2, and stride of 2 with no padding. The concatenation of the 3-dimensional vector is a one-hot encoding, indicating the car to turn left, right or keep move straight.

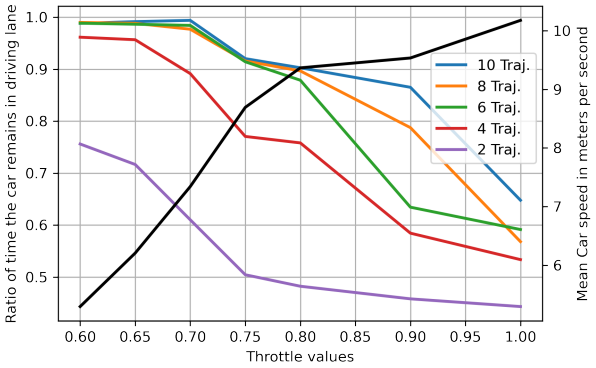

Figure 7: Shows the effect of gradually increasing the throttle on the mean speed of the car and the online performance for the models trained with 2, 4, 6, 8, and 10 trajectories. The speed of the car (right vertical axis) is reported in meters per second. The online performance (left vertical axis) is reported as the ratio of time the car remains within its driving track.

of this paper, we leave it for further work.

Testing on a new Town:

The results from the main paper show that our self-supervised framework for steering angle prediction is comparable to the supervised method. The models were trained and evaluated in the same Town albeit across different weathers. However, generalization to unseen environments is also important. CARLA v0.8.2 provides 2 different towns. Therefore, in this supplementary material, an additional experiment is performed. Our method and the supervised model are trained on Town2 but evaluated on Town1. Figure 8 shows the performance of both methods across all 15 weather conditions. The performance of both methods in an unseen environment drops slightly in comparison to when trained and evaluated in the same towns. However, it is important to observe that our method is still on par with the supervised method. This aligns with the results from the main paper.

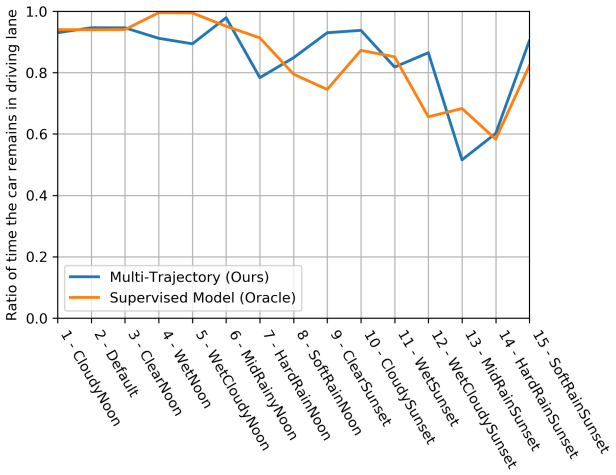

Figure 8: The plot exhibits the ratio of time the car remains within its own driving track for 2 models approach across all the 15 different weather conditions. Both models were trained on Town2 but evaluated on Town1. Higher is better.

3 Video

The video *supplementary_video.mp4* shows the performance between the visual odometry models trained with one and trained with multiple trajectories. As can be observed, the multiple trajectory model is capable of recovering the course despite some deviations from the reference. This is in contrast to the model trained with only one VO trajectory.

References

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[Rajamani, 2012] Rajamani, R. (2012). Vehicle dynamics and control. In *Second Edition, Publisher: Springer*.

[Subirón, 2018] Subirón, N. (2018). Github repository of the carla simulator[accessed on 23.02.2021]. `https://github.com/carla-simulator/carla/issues/106#issuecomment-355985118`.

[Wang et al., 2017] Wang, R., Schwörer, M., and Cremers, D. (2017). Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*.

[Zhao et al., 2013] Zhao, J.-S., Liu, Z.-J., and Dai, J. (2013). Design of an ackermann type steering mechanism. *Journal of Mechanical Engineering Science*, 227.