Applied Intelligence (2023) 53:19186–19198
https://doi.org/10.1007/s10489-023-04478-8

Learning vision based autonomous lateral vehicle control without supervision

Qadeer Khan[1] · Idil Sülö[1] · Melis Öcal[1] · Daniel Cremers[1]

Accepted: 17 January 2023 / Published online: 25 February 2023

Abstract
Supervised deep learning methods using image data as input have shown promising results in the context of vehicle control. However, these supervised methods have two main disadvantages: 1) They require a copious amount of labeled training data, which is difficult and expensive to collect. 2) Such models do not perform well, when situations that are not in the distribution of the training set are encountered. This includes deviations from the designated driving behavior. We therefore provide a framework to mitigate these problems from merely an unlabeled sequence of images. Visual Odometry is first used to determine the vehicle trajectory. Model Predictive Control (MPC) then uses this trajectory to implicitly infer the steering labels. Meanwhile, synthesized images at deviated trajectories are included in the training distribution for enhanced robustness of the neural network model. Experimental results demonstrate that the performance of our network is at par with methods requiring additional data collection or supervision. Code and supplementary information is available here: https://github.com/idilsulo/nn-driving

Keywords Intelligent driving · Computer vision · Deep learning · Neural networks

1 Introduction

Vehicle autonomy has always been a subject of interest, whether it be for the task of driving [1–3], marine navigation [4–6] or even aerial agents [7, 8]. The focus of this work is on self-driving cars using deep learning. Over the last decade, supervised deep learning methods have achieved exemplary performance in various applications.

This work was supported by the Munich Center for Machine Learning.

✉ Qadeer Khan
qadeer.khan@tum.de

Idil Sülö
idil.sulo@tum.de

Melis Öcal
melis.oecal@tum.de

Daniel Cremers
cremers@tum.de

[1] Chair of Computer Vision and Artificial Intelligence, TUM School of Computation, Information and Technology, Technical University of Munich, Boltzmannstrasse 3, Garching, 85748, Bayern, Germany

One domain of interest to us is vehicle control, where convolutional network architectures have proven to be instrumental in directly mapping images to steering commands [9, 10]. Despite their great success, supervised deep learning methods are prone to two severe limitations:

1. They require tremendous amounts of labeled training data. In the context of a vision-based driving application, this would mean having a camera placed at the front of the car which records a video as the vehicle is being driven. The corresponding steering angle executed by the driver is also recorded and time synchronized for each image frame. The necessity of having an expert driver who collects data from a vehicle with such a controlled setup limits the scalability of the underlying application.
2. The trained neural network model struggles at inference time when data that is not in the distribution of the training set is encountered. In the context of self-driving cars, such out-of-distribution data would constitute those scenarios where the vehicle is driving off-course[1] [11–13].

[1]Note that in this work, the ego-vehicle is considered to be *on-course* if it is driving safely within its own lane. Otherwise, it is defined to be *off-course*. Driving off-road or in the lane of oncoming traffic is included in *off-course*.

It is worth noting that the second limitation is a consequence of the first one, as anomalous off-course data is scarce during data collection with an expert driver. Specifically, the traffic rules and lane regulations are strictly necessary to be followed to ensure safety. During inference, the network predictions may cause the car to diverge far from the boundaries of the driving lane. Since there are no such aberrant scenarios in the training set, it may not be possible for the car to take corrective measures to recover from this deviation.

In this work, we propose a framework which addresses these problems associated with supervised deep learning models. The framework merely requires an unlabeled sequence of RGB images from which the vehicle trajectory is determined using visual odometry. The steering labels can then be inferred by applying Model Predictive Control (MPC) using this trajectory and modeling the kinematics of the ego-vehicle. Meanwhile, off-course images are synthesized at novel views and are also included in the training set to enhance robustness. The primary contributions of our framework are enumerated below:

1. We demonstrate training a lateral vehicle control network from only an unlabeled sequence of images. This eliminates the need for a specialized setup on the car that has to retrieve the steering angles executed by the driver. Moreover, the task of synchronizing the recording of the images with the corresponding steering angles is also eliminated.
2. The data collection process does not necessitate driving off-course for recording anomalous cases. Rather images at divergent positions are synthesized using only on-course image sequences and included in the training data.
3. We demonstrate in the experiments that the proposed fusion of MPC with synthesized novel views leads to improved robustness at inference time.

2 Related work

2.1 Supervised methods for vehicle control

Toromanoff et al. [1, 10] train a supervised network that directly maps image data to the steering commands. One limitation of these methods is scaling, as they require an expert driver whose steering maneuvers need to be recorded during the course of data collection. Moreover, performance of such models tends to be constrained to the domain on which they were trained [14]. Attempts to partially circumvent this problem involve using multiple laterally displaced cameras while recording data [3], shearing the images [15] or using a fish-eye camera to generate laterally displaced images [10]. An appropriate label correction is applied to each of the laterally displaced images. However, such approaches may cause visual distortions or are constrained by the maximum lateral displacement in the images. For robustness, [16] adapts the strategy from [17] by injecting noise into the control command and letting the expert recover from these disturbances during data collection. While this method may be expedient in controlled environments, it is impractical and too dangerous to be deployed in the real world. The injected noise may cause the vehicle to veer off-course and result in a potential collision. In our approach, we synthesize images at arbitrary locations from a single on-course trajectory without having the need to drive off-course. Moreover, recording the steering commands of the driver is not required. Rather, the steering labels are inferred from MPC.

2.2 Methods not requiring supervision

Zhang et al. [18–20] train a neural network in a simulated setting for vehicle control using Reinforcement Learning (RL). RL methods do not require explicit supervision; but involve a random exploration of the environment as part of learning a suitable policy [21]. The trajectory resulting from the exploration strategy may cause the driving agent to depart from the driving lane, thereby violating traffic rules and causing accidents. One way of partially alleviating this issue is to use the training data from a virtual environment which tends to be less intrusive [22]. Therefore, the authors of [23] train an RL model in a virtual environment and evaluate it in the real world. However, an intermediate semantic representation to translate virtual images to the real world is required.

In comparison, our method is capable of training directly on images that the model is expected to see at inference and does not require any additional semantic information. Kendall et al. [11] also demonstrates training an RL policy directly in the real world. However, the approach necessitates a safety driver to seize control whenever the car diverges from the lane. An another issue with RL-based control policies is that they tend to require tremendous amounts of data and computational resources for training [19, 24]. In [25], supervised data is not needed and training can potentially be done on real world data. However, they additionally utilize images from multiple trajectories by aligning them to the reference. In contrast, our framework only requires a single reference trajectory from which additional images at arbitrary positions are synthesized.

2.3 Path planning and longitudinal control

The task of autonomous driving can be decomposed into two integral components: 1) High level planning to determine the optimal path for the driving agent to reach its destination. 2) The low level steering commands executed by the ego-vehicle using data received from its immediate surroundings. Li et al. [26, 27] demonstrate path planning methods for generating safe vehicle trajectories. In contrast, our work is concerned with predicting low level vehicle control commands. These low level commands can be further segregated into lateral and longitudinal control. Matute et al. [28, 29] generate velocity profiles such that possible passenger inconvenience resulting from sudden acceleration/deceleration can be mitigated. Similar to our case, they also test their approach in simulation. However, our network is focused on predicting the steering command for lateral vehicle control and does not require state estimation at inference time. Uebel et al. [30] also determines the longitudinal dynamics of the vehicle. However, they additionally take the current and future state of the multiple traffic lights ahead to furnish energy efficient driving. Our framework only requires a single RGB camera for immediate lateral control. It is not focused on long term velocity profiling and trajectory planning.

2.4 Methods combining deep learning and optimal control

Bansal et al. [31] combines optimal control with deep learning for vision-based navigation of a robot in an indoor setting. It is assumed that the environment is static and the robot state is perfectly known. Mohseni et al. [32] combines MPC and uses an ensemble of neural networks for collision avoidance. The sensory input to the network is obtained from Lidar which tends to be more expensive [33]. In [34], a network is trained to predict the output obtained from MPC for controlling the moisture content produced from a paper-making machine. They generate training data in simulation. In comparison, our framework is not limited to just the synthetic domain and can equally be applied on real world data. We show image synthesis for the real world KITTI dataset [35].

3 Framework

Figure 1 provides a high-level overview of our proposed framework. Note that the ultimate goal of the framework is to train a neural network that takes an image as input and predicts the appropriate steering angle for lateral vehicle control. The network is trained only from an unlabeled sequence of images; which are obtained from a camera setup placed at the front of a car. No ground truth steering labels are available. Rather, the steering labels are inferred from MPC. The framework comprises of four main components:

- *Visual Odometry*, which provides the trajectory traversed by the vehicle using a sequence of RGB images.
- *View Synthesis,* that generates additional images at arbitrary positions lateral to the original (on-course) trajectory.

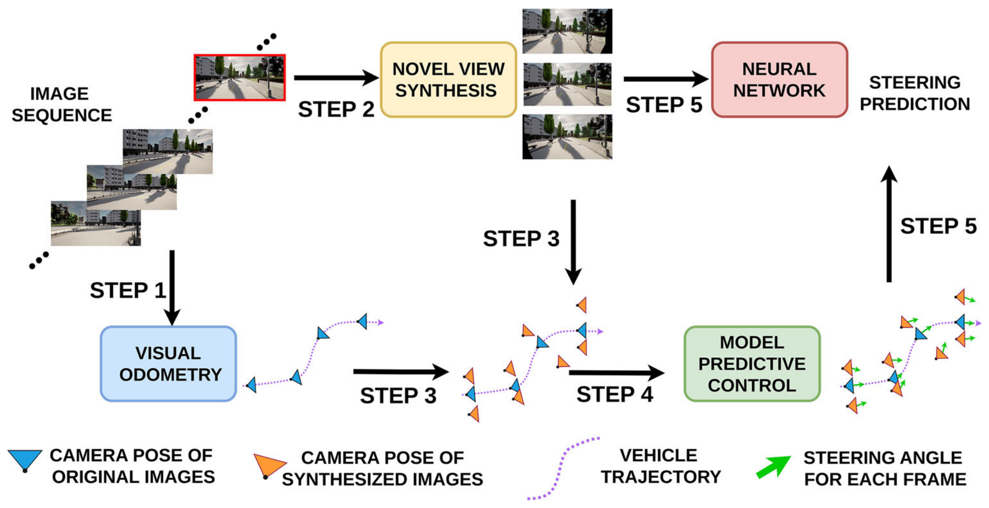

Fig. 1 Overview of the proposed framework describing the steps for data and label generation (Steps 1-4), training (Step 5) and inference. **Step 1:** A sequence of images obtained from a camera rigidly attached to the car is fed through a visual odometry algorithm. It generates the pose of each image along the ego-vehicle's trajectory. We refer to it as the *reference trajectory*. **Step 2:** Images at arbitrary positions are synthesized. **Step 3:** The positions of the synthesized images are aligned in relation to the reference trajectory. **Step 4:** MPC generates the appropriate steering angles at locations of the reference and synthesized images. **Step 5:** A network is trained to predict the appropriate steering angle with an image frame as input. The target labels for training are obtained from MPC for each frame. **Inference:** Once the training is done, only the neural network component of the entire framework is used at inference time to make steering predictions from raw images directly taken from the camera. Note that no view synthesis, visual odometry or MPC calculation is required at the inference step

- *Model Predictive Control*, which supplies the steering labels for both the synthesized images and those in the original trajectory.
- *Neural Network*, which maps the input images to the lateral steering commands.

We first describe the motivation for using model predictive vehicle control and how it works in Section 3.1. Its limitations with regards to real time implementation are also discussed (in Section 3.1.3). Next, view synthesis (Section 3.2) and visual odometry (Section 3.2) are explained as solutions to overcoming these limitations. Finally, Section 3.4 gives information about the neural network.

3.1 Model predictive control (MPC)

We use MPC to implicitly acquire the steering labels. Note that MPC is an optimization based approach, whereas our neural network is trained with a learning based approach. As depicted in Fig. 1, we therefore use MPC to provide the target labels which are then used to train the neural network.

In driving-related tasks, MPC has shown to outperform other controllers [36–38], as it allows to achieve desired behaviours by appropriately adjusting the cost function [39]. We therefore treat MPC for the task of self-driving as a receding horizon problem. As depicted in Fig. 2, the controller optimizes to predict the future set of actions that need to be executed for the ego-vehicle (point A) to reach a goal state (point C) along the original reference trajectory (in purple). The first control action is executed and the vehicle attains a new state at time T_1. The process is repeated at the new vehicle state. Successive optimizations with the updated goal state (point C) for timestamps T_i, where i ∈ [1, n], reduce the deviation between the ego-vehicle (point A) and the closest state (point B) on the reference. Note that we only execute the first action rather than all actions predicted by MPC because the motion model of the ego-vehicle is only an approximation of the real world. Therefore, attempting to execute all actions in order to follow the entire path predicted by MPC may cause the ego-vehicle to deviate far away from the reference.

We now define the motion model of the ego-vehicle and cost function used for the optimization.

3.1.1 Motion model

We use the bicycle model [40] to describe the kinematics of a 4-wheeled ego-vehicle with planar motion. The state of the ego-vehicle is described by its orientation (θ) and location coordinates X, Y. It can be controlled by regulating the acceleration (a) and steering angle (δ). We assume there is no slip in the vehicle, which is a valid assumption for vehicles that execute turns at low or moderate speed [41]. If L is the wheelbase and V is the ego-vehicle's velocity, then the equations of motion can be formulated as:

$$\dot{X} = V cos\theta;\ \dot{Y} = V sin\theta;\ \dot{V} = a;\ \dot{\theta} = V\frac{\tan\delta}{L} \tag{1}$$

3.1.2 Cost function

The cost function aims to produce the optimal sequence of control actions such that the difference between the goal state and an ego-vehicle state at any of the N timesteps into the horizon ahead is minimized. The state at each of the N timesteps ahead can be estimated by iteratively applying the equations of the motion model. Longer horizons ought to produce better estimates of the control actions to be taken. However, this comes at the expense of longer optimization cycles. If X_g, Y_g and θ_g describe the state variables of the

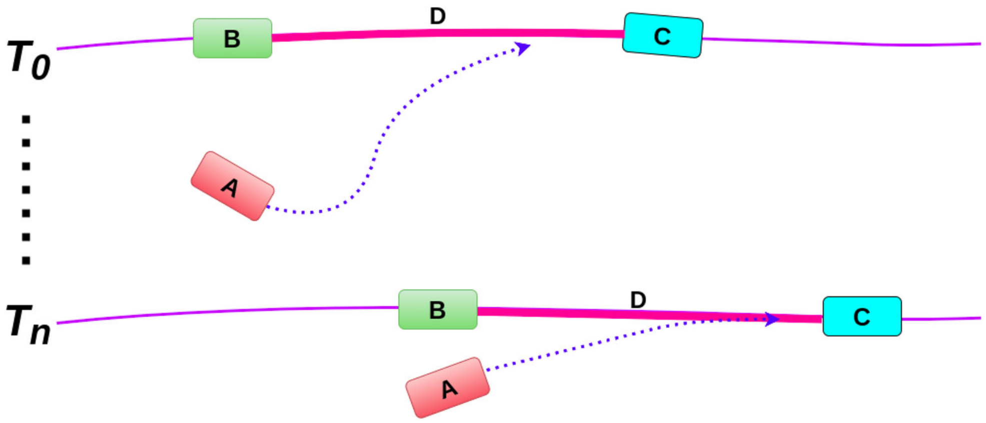

Fig. 2 Depicts MPC treated as a receding horizon problem. At time T_0, the ego-vehicle is at position A. B is the point on the purple reference trajectory closest to A. The dotted blue line shows the estimated trajectory of the car, if the sequence of control actions optimized by MPC are executed. The first set of action(s) predicted by the controller is executed and the ego-vehicle attains a new state. The process is repeated all the while the distance between point B and the goal state is maintained to be D. The state at some later time T_n is also shown. Figure should be viewed in colour

goal state, then the objective function to be minimized can be expressed as:

$$\min_{a,\delta} \sum_{i=1}^{N} [\alpha_1 (X_\mathrm{i} - X_\mathrm{g})^2 + \alpha_2 (Y_\mathrm{i} - Y_\mathrm{g})^2 + \alpha_3 (\theta_\mathrm{i} - \theta_\mathrm{g})^2], \quad (2)$$

where $\alpha_{1-3} \in \mathbb{R}$ are values to appropriately weight the different terms in the cost function. Also, note that the cost function is dynamically modified to bring the goal state closer to the current state when making turns. This is to ensure that the optimization does not cause the vehicle to cut corners, while simultaneously reducing the velocity and abiding by the no-slip condition.

The optimization yields a sequence of N control actions. However, only the first is executed and the vehicle attains a new state. This process is repeated at this new vehicle state.

3.1.3 MPC limitations and solutions

Optimization Time: In principle, the entire set of N control actions can be executed after a single MPC optimization. This would relieve the computational burden from optimizing at each time step. However, the ego-vehicle model is only an approximation of the real world. Therefore, attempting to execute all control actions that are farther into the future may yield an ego-vehicle trajectory which significantly deviates from the estimated trajectory. Hence, in order to obtain an optimal solution, the calculations need to be performed at each time step which may not be feasible for real time execution. To alleviate this issue, we perform the optimizations offline and train a neural network to predict the first control action in the sequence. Offline computation allows inference time to remain the same, irrespective of how long the optimization cycles were to train the network. The input to the network is an image corresponding to the ego vehicle position.

Reference Trajectory: The reference trajectory required to implement MPC in real time is no longer needed when a neural network is used at inference. Nevertheless, it would still be required to determine the target labels for training the network. We utilize a visual odometry system in order to track the moving camera and obtain the reference trajectory for the training stage. Please see Section 3.3 for further details on visual odometry.

Localization: Even if the reference trajectory is known, another issue with real time MPC implementation is that the position of the ego-vehicle needs to be localized against the reference. Methods used for localization would incur additional cost and further constrain the hardware resources. In contrast, the neural network directly maps the input image to the appropriate steering command, thereby obviating the intermediate localization step. Nevertheless, to make the network robust to deviations from the reference, we would still like to train it with images at arbitrary locations that are not on the reference trajectory. For this, we formulate a self-supervised pipeline which uses images from the reference trajectory to synthesize images at arbitrary locations away from the reference. Hence, we have inverted the problem by synthesizing images at desired locations and training the network offline. This is in contrast to localizing the car in relation to the reference in real time. Please see the next Section 3.2 on view synthesis.

3.2 View synthesis

Figure 3 provides the schematics on how images at novel view points are generated using a single image from the reference trajectory traversed by the car. This image is first fed to a depth estimation network to output the corresponding depth. The depth estimation problem is framed similar to a view synthesis one by constraining the network to perform image synthesis using an intermediary variable such as a depth map. If I_t is the target image and

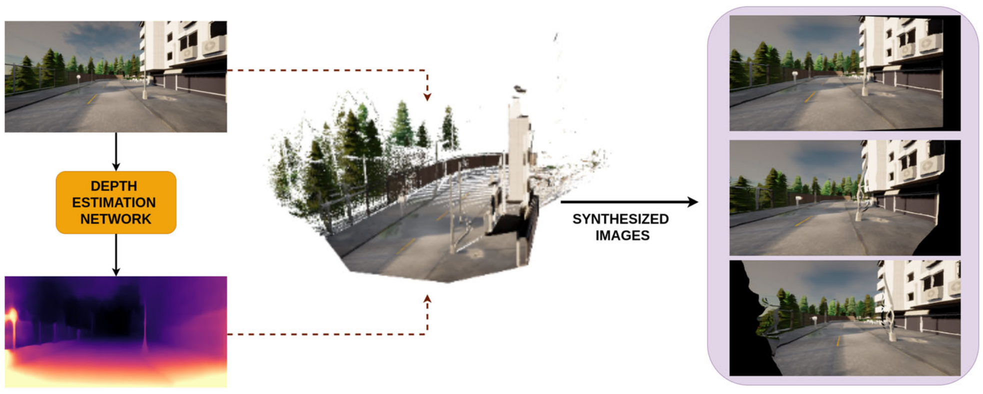

Fig. 3 Describes the schematics of synthesizing images at novel view points using a source image from the reference trajectory

$I_{t' \to t}$ is the reconstructed image from the source $I_{t'}$, then the objective function to be minimized can be formulated as a combination of photometric reconstruction error L_p and edge-aware depth smoothness L_s, as follows [42]:

$$L_D(I_t, I_{t' \to t}) = \mu\, L_p(I_t, I_{t' \to t}) + \lambda\, L_s(I_t) \tag{3}$$

For the photometric reconstruction error L_p, we use a combination of SSIM and L1 loss terms, and set $\alpha = 0.85$. μ denotes the per-pixel mask to eliminate pixels which remain stationary between adjacent frames in the sequence.

$$L_p(I_t, I_{t' \to t}) = \min_{t'} pe(I_t, I_{t' \to t}) \tag{4}$$

$$pe(I_a, I_b) = \alpha \frac{1 - \mathrm{SSIM}(I_a, I_b)}{2} + (1 - \alpha) \left\| I_a - I_b \right\|_1 \tag{5}$$

Edge-aware smoothness is used to prevent shrinking of the predicted depth, where $d_t^* = d_t / \overline{d_t}$ denotes the mean-normalized inverse depth. The combined loss L_D is averaged over each pixel, scale, and batch.

$$L_s(I_t) = \left|\partial_x d_t^*\right| e^{-\left|\partial_x I_t\right|} + \left|\partial_y d_t^*\right| e^{-\left|\partial_y I_t\right|} \tag{6}$$

This depth map is then projected to yield a 3D coloured point cloud. Multiple images can then be synthesized from this point cloud at desired imaginary positions [43]. Note that certain regions in the synthesized image may be beyond the visible field of view of the reference image. In this case, the synthesized image may have voids. We make sure that the imaginary camera position is chosen such that the void regions do not occlude the drivable regions in the image. The depth estimation network can be trained in an entirely un/self-supervised manner [42, 44]. However, such methods tend to suffer from bleeding artifacts at the object boundaries [45]. This is further exacerbated for thin objects [46]. Hence, attempting to synthesize images at novel views may produce curved boundaries for certain edges. Nevertheless, for the task of lateral vehicle control, the trained neural network tends to focus on the high level features of the image [47]. Hence, for all intents and purpose, the network trained with synthesized images yields similar performance as the network trained with original images. This is also shown in Table 1 in Section 4 and further discussed in Section 5.7.

3.3 Visual odometry

Recall that by using a neural network, we are absolved from the requirement of having a reference trajectory at inference time. The network also has the possibility to generalize itself to control the vehicle in new unseen environments where no reference trajectory is available. This is done by shifting the requirement of having the reference trajectory at inference time to having it during offline training of the network. This can be obtained by running a state-of-the-art visual odometry system [48–50]. These approaches generate the 6 Degree-of-Freedom (DoF) pose information for each frame in the trajectory. Since the camera setup is rigidly attached to the car, the pose of the camera can also be used to determine the pose of the car at the corresponding frame. Using stereo image pairs can additionally provide the scale information [51, 52]. The camera poses are represented by a transformation matrix $\mathbf{T}$ belonging to the special Euclidean group $\mathbf{SE}(3)$ representing rigid body motions. It comprises of $\mathbf{R} \in \mathbf{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$.

3.4 Neural network

The neural network is trained to take an image as input and produce the appropriate steering angle (δ) for lateral vehicle control. The training is done by minimizing the L1 loss between the value predicted by the model and the first δ of the N steering commands optimized by model predictive

Table 1 Ratio of time the car remains within its driving lane

	Train	Test
Our method	0.9441	0.9413
Single trajectory	0.5944	0.3117
Online MPC	0.9799	–
Supervised	0.4931	0.3945
Supervised with noise injection [16]	0.9704	0.9517
3-Camera model [3]	0.8374	0.7125
Ours + Ground Truth	0.9385	0.9355

The evaluation is done for different model configurations on both the seen (train) and unseen (test) trajectories. Please refer to Section 5 for a detailed discussion of the different configurations control. This image can either be one from the reference trajectory or can also be synthesized.

4 Experiments

There are many publicly available real world datasets [35, 53–56] that have been instrumental for benchmarking and evaluating various autonomous driving tasks such as object detection/tracking, SLAM, semantic segmentation etc. However, none of these real world benchmarks provide steering labels for evaluation of vehicle control algorithms. More importantly, they also do not facilitate interaction with the environment. This interaction is necessary to quantitatively evaluate sensorimotor control methods for true driving quality. Codevilla et al. [57] had conducted extensive studies to demonstrate that the true driving quality obtained from online interaction does not necessarily correlate with offline evaluation. In fact, they showed that two models with similar offline performance metrics can have significantly different driving behaviours. Hence, offline evaluation on static images of the available real world benchmarks would be inadequate for our task of evaluating online sensorimotor control. In contrast, simulation engines [58–61] aid online interaction with the driving environment. They are therefore increasingly being used to quantitatively evaluate contemporary vehicle control algorithms [62, 63]. We use the CARLA simulator [60] for evaluation and benchmark against the strategies from [16] and [3]. Further details on our experimental setup on the CARLA simulator are given in the following subsections.

Note that in addition to the quantitative results on CARLA, we also provide qualitative evaluation on the real world KITTI dataset. Please refer to Section 4.4 for more details.

4.1 CARLA simulator

The CARLA simulator has been designed to validate various autonomous driving algorithms encompassing both perception and control. It also supports a diverse suite of sensors. The primary objective of this paper is to demonstrate how a vehicle control model can be trained merely from an unlabeled sequence of images. Therefore, the RGB camera is the only category of sensor used in this paper. Note that the CARLA simulator also provides the ability to violate traffic rules which would otherwise be infeasible and costly in the real world. Hence, we are able to compare the performance of our method with the supervised approach adapted from [16], wherein noise is injected into the control signal during data collection. This causes the vehicle to swerve off-course, thereby breaking traffic rules. The action taken by the expert driver to bring the vehicle back on-course is recorded. We show that the performance of our model is comparable to this supervised method. This is despite the fact that our method does not require dangerous traffic violations during data collection. Recall from the Introduction Section 1, that we have defined the ego-vehicle to be *on-course* if it is driving safely within its own lane. Otherwise, it is considered to be *off-course*. Driving off-road or in the lane of oncoming traffic would be categorized as *off-course*.

4.2 Data collection

We place the camera setup at the front of the ego-vehicle. Images of size 1200×600 with a field of view of $110°$ are recorded as the vehicle traverses the road in the autopilot mode. Note that recording the steering angle executed by the ego-vehicle during data collection is not needed for our method. Nevertheless, for the purpose of comparison with the supervised method, these steering commands are also collected. We use [51] as the visual odometry algorithm for determining the reference trajectory. This is needed for determining the control values with MPC (Section 3.1), which serve as the training labels for our neural network (Section 3.4). Meanwhile, [42] is used to train the depth estimation model in an entirely self-supervised manner. The estimated depth is then utilized to warp the original colour image as if the scene would be seen from a new perspective.

4.3 Quantitative evaluation results

Table 1 reports the online evaluation results for different starting positions both on trajectories that the model had seen during training and also the testing trajectories which were not seen during training. Each episode is run up to 250 timesteps for natural turns. The online metric used for evaluation is the mean ratio of time the ego-vehicle remains within its driving lane [47]. The car is considered to be within its own lane if no portion of it is in the lane of the oncoming traffic or off the road and it does not collide with other traffic participants/obstacles. For the purpose of comparison, we additionally report the evaluation scores for other model configurations. They are described in further detail in Section 5.

4.4 Qualitative results

Evaluating the online performance of the neural network for steering angle prediction is not possible on existing real world datasets, since it requires interaction with the captured environment. Nevertheless, we can depict the qualitative performance of the other three components of our framework, namely view synthesis, visual odometry and MPC on the real world KITTI dataset [35]. The results of visual odometry for calculating the vehicle pose have been included in the supplementary material. Moreover, view synthesis and control labels from MPC are depicted in the supplementary video on the KITTI dataset. For visual odometry and view synthesis, images at their native resolution of 1241 x 376 are used without further pre-processing. However, for view synthesis, locations at farther distances from the source image lead to visible voids at the boundaries. This is because the field of view (FOV) of the source image does not capture the entire FOV of the synthesized images. Therefore, for illustration purposes, we center crop the image to 1000 x 376 in the video. We conduct the same qualitative evaluation on CARLA as well. These results can also be visualized in the video.

5 Discussion

In this section, we make some observations on the experimental results of our approach when compared with other models.

5.1 Single trajectory model

The power of our framework comes from its ability to synthesize additional off-course data from a single on-course reference trajectory. To examine the significance of utilizing the synthesized images, we train another model only with images from the single reference trajectory. Note that the reference trajectory is always on-course. As can be seen in Table 1, this model's performance drops significantly on both the train and test sequences. There are two plausible explanations for this:

1. *Over-fitting:* Firstly, the single trajectory model was trained with fewer images, which has the tendency to lead to over-fitting. But then, why does online evaluation on the training trajectories yields dismal results? This is despite that images in these trajectories were seen by the model during training. As alluded to in [57], the training loss is an offline evaluation metric and it does not necessarily correlate with the true driving quality at inference. So, if the car were to deviate even slightly from the reference training trajectory at inference, it would come across a scene whose corresponding image was not available in the training set. This brings us to our second point.

2. *Lack of anomalous scenarios in the training data:* The training data for the single trajectory model did not contain such anomalous off-course scenarios. Therefore, it has difficulty figuring out what ought to be the correct action to take to return the ego-vehicle to its original course. This would cause the model in taking the wrong decision, leading the ego-vehicle to go further astray. This accumulation in errors will eventually cause a violation of traffic rules or even a collision. This is despite conducting an evaluation on the same trajectories the model had seen in the training set.

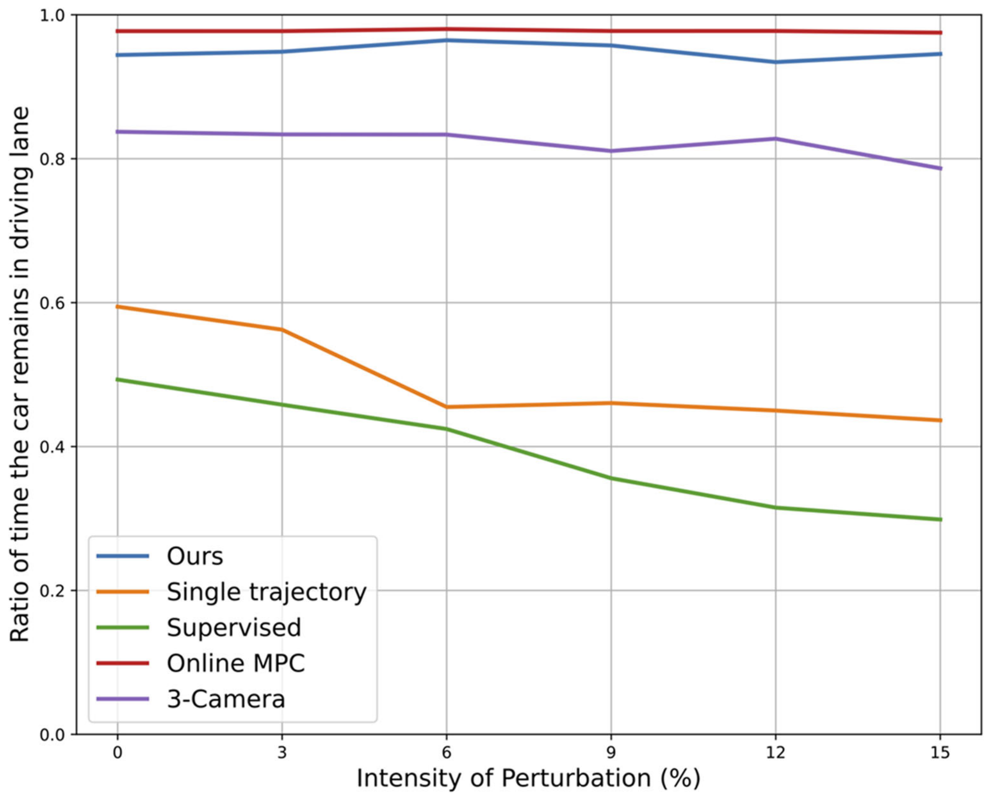

Fig. 4 Shows the effect on the performance of the various model configurations when different levels of perturbation are added to the final steering command. The performance is reported as the mean of the ratio of time the car remains within its driving lane. Higher ratio values correspond to better performance of a method. Note that perturbations are introduced regularly every 15 timesteps for a continuous duration of 5 timesteps. In the remaining 10 timesteps of the cycle, no perturbation are added to allow the models to recover from deviations. Figure should be viewed in colour

5.2 Effect of perturbations

To further investigate this issue with the single trajectory model, we introduce perturbation into the predicted steering command. This is to see if and how well it can recover from deviation on the training trajectories. As can be seen in Fig. 4, the greater the intensity of perturbations, the more difficult it is for the model to execute a recovery and the worse is the performance. This is in contrast to our model, trained additionally with synthesized images representing potential anomalous driving scenarios. Our model therefore has the ability to cater to such circumstances and bring the vehicle back to its original course. It maintains a fairly consistent performance even in light of perturbations.

5.3 Online MPC

This is the only approach in Table 1 that is not data driven. Rather, it adapts the approach described in Section 3.1 by performing real time optimizations at every ego-vehicle state. It assumes that the ground truth reference trajectory and the ego-vehicle state are always perfectly known, as they can be easily obtained from the CARLA simulator. It is not surprising to see that this online MPC approach outperforms our method on the training set. However, ground truth trajectory and precise ego-vehicle state are unknown in the real world. One can determine the ego-vehicle state in relation to the reference trajectory using visual re-localization methods such as [64] or by using visual image descriptors [65, 66] on the PnP algorithm in a RANSAC scheme. Although, these visual localization methods are highly accurate in determining the ego-vehicle state to centimeter accuracy, they are however slower when compared to our network (please also see Section 5.8 on *'Computational Cost'*). Moreover, our framework neither requires a ground truth trajectory nor needs to determine the ego-vehicle state at inference time. Instead of using the ground truth trajectory, our method derives the reference trajectory from visual odometry. In addition to this, instead of localizing the ego-vehicle against the reference in real time, we synthesize images at arbitrary locations from the reference offline. Note that since the online MPC approach requires a reference trajectory, its performance for unseen test trajectories cannot be reported. However, Fig. 4 shows that it is robust to perturbations on the *"training"* trajectories.

5.4 Supervised network

Table 1 also compares the performance of our method with a supervised model; which is trained with ground truth steering labels recorded during data collection. Note that our approach does not require supervised labels. Yet, it far outperforms the network trained with supervision. The supervised model suffers from the same issue as that of the single trajectory model described earlier in Section 5.1. It was trained with only images on the reference trajectory. Hence, if the vehicle deviates off-course, it is not capable of returning back to course. The model is also not robust against perturbations as depicted in Fig. 4.

5.5 3-Camera model

This approach is adapted from [3]. It is also a supervised approach trained with ground truth steering labels but it uses three cameras during data collection. Here, two cameras are placed to the left and right of the central camera at a distance of 0.3 meters each. This provides additional deviated trajectories during training. The steering labels for the images from these two cameras are obtained by adding a slight bias to the steering label of the central camera. The bias is a hyper-parameter and in our case is obtained by respectively taking the mean of the steering values obtained by MPC for all the left and right camera images. The performance of the model is better than both the supervised and single trajectory model as can be seen in Table 1 and Fig. 4. This is because the images from the deviated trajectories expand the training distribution. However, this comes at a cost of using three time-synchronized cameras. Moreover, note that the performance of this model is still worse than our approach. One explanation is that our method of view synthesis can create an arbitrary number of additional trajectories used for training rather than only three.

5.6 Supervised network with noise injection

We would like to investigate whether the lack of images depicting anomalous scenarios is the pivotal reason for constrained performance of the supervised model. For this we adapt the strategy of [16]. It is similar to the supervised model, except that noise is injected into the steering command during data collection. This would cause the car to swerve. The corrective action executed by the expert driver to bring the vehicle back to its original course is recorded. Table 1 shows that the performance of such a model trained with noise injection improves significantly. However, note that attempting to collect supervised data with noise injection in real traffic may be extremely dangerous. This is because the injected noise may not only cause the ego-vehicle to violate traffic rules but also surprise other traffic participants resulting in them potentially taking false decisions. Moreover, it would necessitate having an expert driver with specialized skills to take immediate corrective actions. This is not a pragmatic and scalable solution.

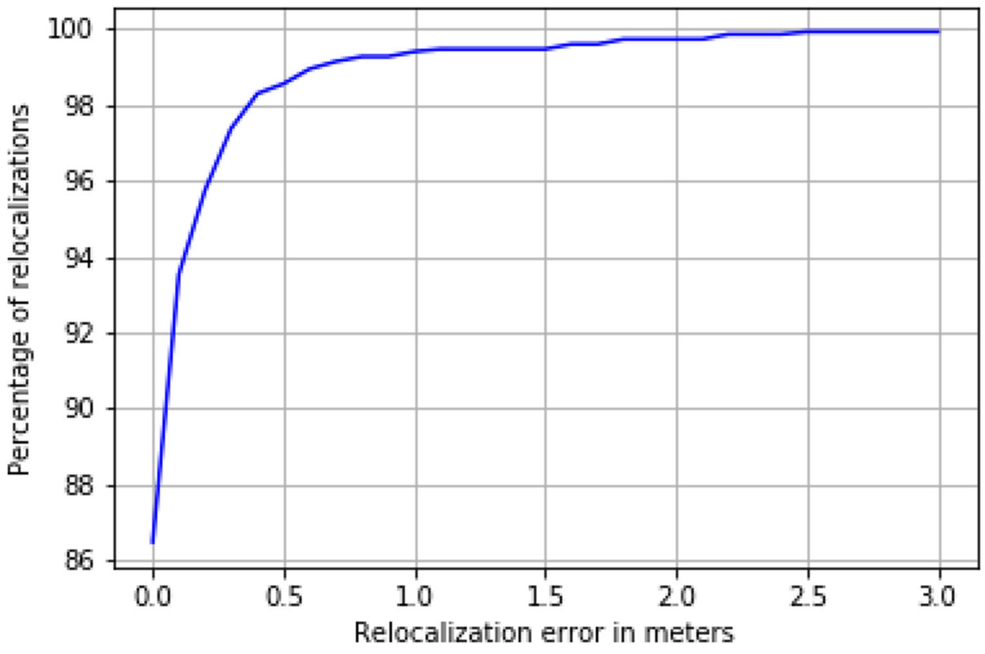

Fig. 5 Shows the cumulative error of re-localization within 3 meters. The error is the translation norm between the predicted and ground truth in meters

5.7 Our method + ground truth data

This method is similar to ours, except that ground truth reference trajectory and images are used for training. Recall that our approach used visual odometry to determine the reference trajectory and synthesized images at different locations. Despite this, our model's performance is at par with the model trained with ground truth data.

5.8 Computational cost

Note that the online MPC method assumed that the state of the vehicle in relation to the ground truth trajectory is precisely known. Although, this information can easily be extracted from the simulation, it will not be available in the real world. As an alternate solution, state of the art visual localization methods can be used to determine the ego vehicle state in relation to a reference trajectory. Among such methods one could use visual descriptors such as [66] on the PnP algorithm in a RANSAC setting. This achieves up-to centimeter (cm) accuracy. Figure 5 shows that the method achieves a cumulative re-localization of 98% within 50 cm.

Although very accurate, it is much slower than our network. This re-localization approach runs at a frequency of 11 Hz on the GeForce GTX TITAN X 12 GB GPU and 1 Hz on an Intel(R) Xeon(R) CPU E5-2637 CPU. In contrast, our network runs faster at 25 Hz on a GPU and 16 Hz on a CPU with the same specifications. This is because our network at inference time directly maps the RGB image to the appropriate steering commands. It completely eliminates the state estimation process at inference.

6 Conclusion

In this paper, we presented a framework for training a lateral vehicle control network from an unlabeled sequence of RGB images. The approach demonstrated improved robustness from using additional images. These views were synthesized from only the available on-course data but appeared as if emerging from a deviated traversal of the vehicle. Hence, data collection did not have to violate traffic rules to record such deviated aberrant situations. Moreover, steering labels were inferred from MPC rather than being recorded by an expert driver. Experimental results demonstrated that our approach yields on par performance with methods that rely on additional data collection and supervision.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Munich Center for Machine Learning.

Data Availability This work has used data from two sources:

- The KITTI dataset [35]: http://www.cvlibs.net/datasets/kitti/index.php
- CARLA simulator [60]: https://carla.readthedocs.io/en/latest/

Code and Supplementary information We have provided the supplementary material, video and scripts for running the experiments. The interested reader may find information about them here: https://github.com/idilsulo/nn-driving.

Declarations

Competing interests The authors see no bias or conflict of competing interest.

References

1. Pomerleau DA (1989) Alvinn: an autonomous land vehicle in a neural network. In: Advances in neural information processing systems, vol 1
2. Thrun S (2006) Stanley: the robot that won the darpa grand challenge: research articles. J Robot Syst 23(9):661–692. https://doi.org/10.1002/rob.v23:9
3. Bojarski M, Testa DD, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, Zhang X, Zhao J, Zieba K (2016) End to end learning for self-driving cars. CoRR arXiv:1604.07316
4. Wang N, He H (2021) Extreme learning-based monocular visual servo of an unmanned surface vessel. IEEE Trans Industr Inf 17(8):5152–5163. https://doi.org/10.1109/TII.2020.3033794
5. Wang N, Ahn CK (2021) Coordinated trajectory-tracking control of a marine aerial-surface heterogeneous system. IEEE/ASME Trans Mechatronics 26(6):3198–3210. https://doi.org/10.1109/TMECH.2021.3055450
6. Wang N, Zhang Y, Ahn CK, Xu Q (2022) Autonomous pilot of unmanned surface vehicles: bridging path planning and tracking. IEEE Trans Vehicular Technol 71(3):2358–2374. https://doi.org/10.1109/TVT.2021.3136670
7. Gille WH, Kutzler RJ (1944) Application of electronics to aircraft flight control. Electr Eng 63(11):849–853. https://doi.org/10.1109/EE.1944.6440576
8. Penicka R, Song Y, Kaufmann E, Scaramuzza D (2022) Learning minimum-time flight in cluttered environments. IEEE Robot Autom Lett 7(3):7209–7216. https://doi.org/10.1109/LRA.2022.3181755
9. Chen Z, Huang X (2017) End-to-end learning for lane keeping of self-driving cars. In: 2017 IEEE intelligent vehicles symposium (IV), pp 1856–1860. https://doi.org/10.1109/IVS.2017.7995975
10. Toromanoff M, Wirbel E, Wilhelm F, Vejarano C, Perrotton X, Moutarde F (2018) End to end vehicle lateral control using a single fisheye camera. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 3613–3619. https://doi.org/10.1109/IROS.2018.8594090
11. Kendall A, Hawke J, Janz D, Mazur P, Reda D, Allen J-M, Lam V-D, Bewley A, Shah A (2019) Learning to drive in a day. In: 2019 International conference on robotics and automation (ICRA), pp 8248–8254. https://doi.org/10.1109/ICRA.2019.8793742
12. Ross S, Bagnell D (2010) Efficient reductions for imitation learning. In: Teh YW, Titterington M (eds) Proceedings of the thirteenth international conference on artificial intelligence and statistics. Proceedings of machine learning research. PMLR, Chia Laguna Resort, Sardinia, Italy, vol 9, pp 661–668. http://proceedings.mlr.press/v9/ross10a.html
13. Bansal M, Krizhevsky A, Ogale A (2018) Chauffeurnet: learning to drive by imitating the best and synthesizing the worst. CoRR arXiv:1812.03079
14. Wenzel P, Khan Q, Cremers D, Leal-Taixé L (2018) Modular vehicle control for transferring semantic information between weather conditions using GANs. In: Conference on robot learning (coRL)
15. Hubschneider C, Bauer A, Weber M, Zöllner J (2017) Adding navigation to the equation: turning decisions for end-to-end vehicle control. In: 2017 IEEE 20th international conference on intelligent transportation systems (ITSC), pp 1–8. https://doi.org/10.1109/ITSC.2017.8317923
16. Codevilla F, Müller M, López A, Koltun V, Dosovitskiy A (2018) End-to-end driving via conditional imitation learning. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1–9
17. Laskey M, Lee J, Fox R, Dragan A, Goldberg K (2017) Dart: noise injection for robust imitation learning. In: Proceedings of the 1st annual conference on robot learning. Proceedings of machine learning research, vol 78, pp 143–156. http://proceedings.mlr.press/v78/laskey17a.html
18. Zhang Q, Luo R, Zhao D, Luo C, Qian D (2019) Model-free reinforcement learning based lateral control for lane keeping. In: 2019 International joint conference on neural networks (IJCNN), pp 1–7. https://doi.org/10.1109/IJCNN.2019.8851766
19. Li D, Zhao D, Zhang Q, Chen Y (2019) Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. IEEE Comput Intell Mag 14(2):83–98. https://doi.org/10.1109/MCI.2019.2901089
20. Pérez-Gil Ó, Barea R, López-Guillén E, Bergasa LM, Gómez-Huélamo C, Gutiérrez R, Díaz-Díaz A (2022) Deep reinforcement learning based control for autonomous vehicles in carla. Multimed Tools Appl 81(3):3553–3576
21. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press
22. Roitberg A, Schneider D, Djamal A, Seibold C, Reiß S, Stiefelhagen R (2021) Let's play for action: recognizing activities of daily living by learning from life simulation video games. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE
23. Xinlei Pan ZW, You Y, Lu C (2017) Virtual to real reinforcement learning for autonomous driving. In: Proceedings of the British machine vision conference (BMVC). https://doi.org/10.5244/C.31.11
24. Vithayathil Varghese N, Mahmoud QH (2020) A survey of multi-task deep reinforcement learning. Electronics, vol 9(9). https://doi.org/10.3390/electronics9091363
25. Khan Q, Wenzel P, Cremers D (2021) Self-supervised steering angle prediction for vehicle control using visual odometry. In: Proceedings of the 24th international conference on artificial intelligence and statistics. Proceedings of machine learning research, vol 130, pp 3781–3789. http://proceedings.mlr.press/v130/khan21a.html
26. Li C, Wang J, Wang X, Zhang Y (2015) A model based path planning algorithm for self-driving cars in dynamic environment. In: 2015 Chinese automation congress (CAC), pp 1123–1128. https://doi.org/10.1109/CAC.2015.7382666
27. Oliveira R, Lima PF, Collares Pereira G, Mårtensson J, Wahlberg B (2019) Path planning for autonomous bus driving in highly constrained environments. In: 2019 IEEE intelligent transportation systems conference (ITSC), pp 2743–2749. https://doi.org/10.1109/ITSC.2019.8916773
28. Matute JA, Marcano M, Zubizarreta A, Perez J (2018) Longitudinal model predictive control with comfortable speed planner. In: 2018 IEEE international conference on autonomous

robot systems and competitions (ICARSC), pp 60–64. https://doi.org/10.1109/ICARSC.2018.8374161

29. Mizushima Y, Okawa I, Nonaka K (2019) Model predictive control for autonomous vehicles with speed profile shaping. IFAC-PapersOnLine 52(8):31–36. https://doi.org/10.1016/j.ifacol.2019.08.044 . 10th IFAC symposium on intelligent autonomous vehicles IAV 2019
30. Uebel S, Kutter S, Hipp K, Schrödel F (2019) A computationally efficient MPC for green light optimal speed advisory of highly automated vehicles. In: VEHITS, pp 444–451
31. Bansal S, Tolani V, Gupta S, Malik J, Tomlin C (2020) Combining optimal control and learning for visual navigation in novel environments. In: Conference on robot learning. PMLR, pp 420–429
32. Mohseni F, Voronov S, Frisk E (2018) Deep learning model predictive control for autonomous driving in unknown environments. IFAC-PapersOnLine 51(22):447–452. https://doi.org/10.1016/j.ifacol.2018.11.593. 12th IFAC symposium on robot control SYROCO 2018
33. Qian R, Garg D, Wang Y, You Y, Belongie S, Hariharan B, Campbell M, Weinberger KQ, Chao W-L (2020) End-to-end pseudo-lidar for image-based 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5881–5890
34. Pon Kumar SS, Tulsyan A, Gopaluni B, Loewen P (2018) A deep learning architecture for predictive control. IFAC-PapersOnLine 51(18):512–517. https://doi.org/10.1016/j.ifacol.2018.09.373. 10th IFAC symposium on advanced control of chemical processes ADCHEM 2018
35. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on computer vision and pattern recognition (CVPR)
36. Varma B, Swamy N, Mukherjee S (2020) Trajectory tracking of autonomous vehicles using different control techniques(pid vs lqr vs mpc). In: 2020 International conference on smart technologies in computing, electrical and electronics (ICSTCEE), pp 84–89. https://doi.org/10.1109/ICSTCEE49637.2020.9276986
37. Samuel M, Mohamad M, Hussein M, Saad SM (2021) Lane keeping maneuvers using proportional integral derivative (pid) and model predictive control (mpc). J Robot Control (JRC) 2(2):78–82. https://doi.org/10.18196/jrc.2256
38. Liu J, Yang Z, Huang Z, Li W, Dang S, Li H (2021) Simulation performance evaluation of pure pursuit, stanley, lqr, mpc controller for autonomous vehicles. In: 2021 IEEE international conference on real-time computing and robotics (RCAR), pp 1444–1449. https://doi.org/10.1109/RCAR52367.2021.9517448
39. Cheng S, Li L, Guo H-Q, Chen Z-G, Song P (2020) Longitudinal collision avoidance and lateral stability adaptive control system based on mpc of autonomous vehicles. IEEE Trans Intell Transp Syst 21(6):2376–2385. https://doi.org/10.1109/TITS.2019.2918176
40. Wang D, Fing Q (2001) Trajectory planning for a four-wheel-steering vehicle. In: IEEE international conference on robotics and automation (ICRA)
41. Rajamani R (2012) Vehicle dynamics and control. In: Second edn, publisher: Springer
42. Godard C, Mac Aodha O, Firman M, Brostow GJ (2019) Digging into self-supervised monocular depth prediction. The international conference on computer vision (ICCV)
43. Wiles O, Gkioxari G, Szeliski R, Johnson J (2020) Synsin: end-to-end view synthesis from a single image. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7467–7477
44. Godard C, Mac Aodha O, Brostow GJ (2017) Unsupervised monocular depth estimation with left-right consistency. In: CVPR
45. Zhu S, Brazil G, Liu X (2020) The edge of depth: explicit constraints between segmentation and depth. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
46. Imran S, Kyung C-M, Mukaram S, Karim Khan MU (2020) Unsupervised monocular depth estimation with multi-baseline stereo. In: The 31st British machine vision conference. British machine vision virtual conference
47. Khan Q, Wenzel P, Cremers D, Leal-Taixé L (2019) Towards generalizing sensorimotor control across weather conditions. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 4497–4503. https://doi.org/10.1109/IROS40897.2019.8968451
48. Mur-Artal JMM, Montiel R, Tardos JD (2015) ORB- SLAM: a versatile and accurate monocular SLAM system. IEEE Trans Robot 31(5):1147–1163. https://doi.org/10.1109/TRO.2015.2463671
49. Engel J, Koltun V, Cremers D (2018) Direct sparse odometry. IEEE Trans Pattern Anal Mach Intell 40(3):611–625. https://doi.org/10.1109/TPAMI.2017.2658577
50. Engel J, Cremers D (2014) Lsd-slam: large-scale direct monocular slam. In: In ECCV
51. Mur-Artal R, Tardós JD (2017) ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-d cameras. IEEE Trans Robot 33(5):1255–1262. https://doi.org/10.1109/TRO.2017.2705103
52. Wang R, Schwörer M, Cremers D (2017) Stereo dso: large-scale direct sparse visual odometry with stereo cameras. In: International conference on computer vision (ICCV), Venice, Italy
53. Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) Nuscenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
54. Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, Tsui P, Guo J, Zhou Y, Chai Y, Caine B et al (2020) Scalability in perception for autonomous driving: waymo open dataset. In: Conference on computer vision and pattern recognition (CVPR)
55. Wenzel P, Wang R, Yang N, Cheng Q, Khan Q, Von Stumberg L, Zeller N, Cremers D (2020) 4Seasons: a cross-season dataset for multi-weather SLAM in autonomous driving. In: Proceedings of the german conference on pattern recognition (GCPR)
56. Geyer J, Kassahun Y, Mahmudi M, Ricou X, Durgesh R, Chung AS, Hauswald L, Pham VH, Mühlegg M, Dorn S, Fernandez T, Jänicke M, Mirashi S, Savani C, Sturm M, Vorobiov O, Oelker M, Garreis S, Schuberth P (2020) A2D2: audi autonomous driving dataset. CoRR arXiv:2004.06320 [cs.CV]
57. Codevilla F, López AM, Koltun V, Dosovitskiy A (2018) On offline evaluation of vision-based driving models. In: Proceedings of the European conference on computer vision (ECCV), pp 236–251
58. Shah S, Dey D, Lovett C, Kapoor A (2017) Airsim: high-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics. arXiv:1705.05065
59. Koenig N, Howard A (2004) Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS) (IEEE Cat. No.04CH37566), vol 3, pp 2149–21543. https://doi.org/10.1109/IROS.2004.1389727
60. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) CARLA: an open urban driving simulator. In: Conference on robot learning (coRL)
61. Espié E, Guionneau C, Wymann B, Dimitrakakis C, Coulom R, Sumner A (2005) Torcs, the open racing car simulator
62. Wang T, Chang DE (2019) Improved reinforcement learning through imitation learning pretraining towards image-based

autonomous driving. In: 2019 19th international conference on control, automation and systems (ICCAS), pp 1306–1310. https://doi.org/10.23919/ICCAS47443.2019.8971737

63. Wenzel P, Schön T, Leal-Taixé L, Cremers D (2021) Vision-based mobile robotics obstacle avoidance with deep reinforcement learning. In: Proceedings of the IEEE international conference on robotics and automation (ICRA)
64. Von Stumberg L, Wenzel P, Khan Q, Cremers D (2020) GN-Net: the gauss-newton loss for multi-weather relocalization. IEEE Robot Autom Lett (RA-L) 5(2):890–897
65. Dusmanu M, Rocco I, Pajdla T, Pollefeys M, Sivic J, Torii A, Sattler T (2019) D2-Net: a trainable CNN for joint detection and description of local features. In: Proceedings of the 2019 IEEE/CVF conference on computer vision and pattern recognition
66. Revaud J, Weinzaepfel P, De Souza CR, Humenberger M (2019) R2D2: repeatable and reliable detector and descriptor. In: NeurIPS