A Path Planning Approach for Tractor-Trailer System based on Semi-Supervised Learning

Xian Zhang, Johannes Eck, and Felix Lotz

***Abstract*— Autonomous semi-trucks (tractor-trailer combination) call for dedicated path planners for on-road driving due to their unique articulated structures and large dimensions, which are often ignored in the current path/trajectory planner design for normal autonomous cars, and which could lead to off-tracking without proper modification. While numerical optimization approaches such as model predictive control-based path planning algorithms do exist, these complex planners usually require large computation resources to meet real-time requirements. In this paper, we propose a novel path planning approach based on semi-supervised learning. We train an encoder-decoder type of deep neural network to generate / plan paths with the objective to minimize the off-track of the tractor-trailer swept area. The encoder encodes input information such as lane markings, static obstacles, and potentially other features, and pass it to the decoder to generate a planned path. The key to our approach is the construction of a path cost function that scores / penalizes each network-generated path based on its deviation from the lane center, the path smoothness and collision with any static obstacles, and backpropagates the cost of the paths through the encoder-decoder network to train it. As the path cost function acts as a critic of the path quality, our approach requires no collected data from expert driving for training, but only randomly generated samples of many possible combinations of lane shapes and obstacles arrangements. The proposed approach is finally verified in simulation environment.**

I. Introduction

The trucking industry has been seen as one of the most promising industries where the revolution of the autonomous driving (AD) technology will first take place. So far, the majority of the AD research and tests have been conducted using passenger vehicles as the study subjects. While the problems of perception, mapping and localization, and prediction are largely the same for the autonomous semi-trucks as for the autonomous passenger cars, with tractor-trailer combinations (see Figure 1. for an example), new challenges do rise when planning the path or trajectory due to the articulated structures and large dimensions of the systems. Human drivers, when maneuvering these tractor-trailer systems, tend to go to the outer side of the lane when making a turn, since the trailer tends to sweep the inner side of the turn. However, current path planners designed for normal autonomous cars do not consider these attached trailers, and if applied directly to autonomous semi-trucks, could lead to infeasible trailer paths that go off track, especially when making sharp turns. While numerical optimization approaches such as model predictive control (MPC) based path planning algorithms do exist, these complex planners usually require large computation resources to meet real-time requirements.

Figure 1. A semi-truck (tractor-trailer system), the parameters of which is used in this paper

Our goal in this paper is to design and train a neural network that, at inference time, can relatively quickly generate a feasible path with the objective to minimize the off-track of the tractor and the trailer swept areas and avoid collisions with static obstacles. The intuition behind the idea is that, when a human driver plans such a low-level path or trajectory, there is not always a complex, meticulous calculation involved, for most cases, the plan is made rather quickly and intuitively from the driver's past experiences. For the neural network to learn collision avoidance with static obstacles, it would require seeing many data samples with a large variety of obstacles in terms of dimension, position, and the number of the obstacles (in each data sample), which are virtually impractical to collect from real vehicle tests. Therefore, we present in this paper a novel "semi-supervised" approach that guides the training of the network by a direct measure of the quality of the network-generated path with completely artificial data randomly sampled from predefined distributions. This contrasts to the more typical supervised learning approach, where expert driving paths are used as targets in the supervision of the training process.

The main contributions of this paper can be summarized as:

- We present an AI path planner in an encoder-decoder architecture that generates feasible paths for autonomous semi-trucks to minimize the off-track of the tractor and the trailer swept areas and at the same time avoid collisions with static obstacles.

Xian Zhang is with Continental Autonomous Mobility US, LLC., Auburn Hills, USA (e-mail: xian.2.zhang@ continental.com).

Johannes Eck is with Continental Teves AG & Co. oHG, Frankfurt, Germany (e-mail: johannes.eck@ continental.com).

Felix Lotz is with Continental Teves AG & Co. oHG, Frankfurt, Germany (e-mail: felix.lotz@ continental.com).

Manuscript 1148 submitted to 2022 25th IEEE International Conference on Intelligent Transportation Systems. Received March 31, 2022.

- We design a path cost function that evaluates the "true" cost of a path in an absolute, objective sense and use it to guide the training of the AI path planner. This eliminates the need for expert driving data collection / augmentation and enables our training with completely artificial data.
- We infer the corresponding trailer path from a given tractor path using a kinematic model of the tractor-trailer system, which plays a key role in constructing the path cost function.

II. Related Work

Path planning for tractor-trailer systems. Literature in this area focused mainly on off-road, or unstructured driving of tractor-trailer systems [1-6]. Classical motion planning approaches, which can be broadly grouped into three categories – graph search-based (e.g., hybrid A*) [2][3], incremental sampling and searching (e.g., rapidly exploring random trees, or RRT) [4], and optimization-based (e.g., model predictive control, or MPC) [5][6] – are employed and extended to tackle this new problem with consideration of the size and the dynamics of the attached trailer. We found roughly equal numbers of work representing each category of the three approaches.

For on-road, or structured driving of tractor-trailer systems, i.e., the scenarios of interest of this paper, only a few papers exist in the literature investigating the path planning of the systems, all of which are optimization-based. In [7], the path planning problem is formulated as a nonlinear Optimal Control Problem (OCP) with the control objective being minimizing the off-track of the vehicle bodies swept area and avoiding collision with obstacles. The OCP problem is then solved using a Sequential Quadratic Programming (SQP) approach.

Motion planning with deep learning. While the majority work applying deep learning techniques to autonomous driving problems focus on perception, recent years have seen more and more applications of deep learning to prediction and planning as well [8-12]. Two of the most representative deep learning paradigms for motion planning are imitation learning and deep reinforcement learning [11].

With imitation learning, the idea is to replicate the driving behaviors of some driving "experts", usually human drivers, via supervised learning. However, pure imitation is insufficient for handling complex driving scenarios even with huge amount of expert driving data, as found in [9], where the authors proposed ChauffeurNet, a neural network that takes map and perceived environment information in the form of pixel images as input and outputs a planned trajectory at each time step. Rather than purely imitating all data, the authors synthesized the collected data in the form of perturbations to the expert's driving and augment the imitation loss with additional losses that penalize undesirable events, leading to robustness of the learned model and achieved improved performance.

In contrast, a deep reinforcement learning (DRL) based agent learns to drive a vehicle without any supervision, but by exploring a simulated environment with defined reward functions. A trained DRL agent does not explicitly plan a path or trajectory, rather, it learns a policy that maps a state to an action, executing which would yield a trajectory implicitly. The "action" can be a low-level action such as a steering wheel angle command, or a high-level action such as abstracted driving strategies. In [10], the authors decomposed their motion planner into a learned driving strategies planner and a trajectory planner with hard constraints that is not learned. The DRL was then applied to learn a policy that maps a state in the (agnostic) state space into a set of Desires, which basically represent high-level driving strategies. The goal of Desires is to enable comfort of driving, while hard constraints guarantee the safety of driving.

III. Our Approach

This section introduces our so-called "semi-supervised" learning approach for path planning of a tractor-trailer system. As mentioned before, our goal is to design and train a neural network that maps lane features and nearby static obstacles to a feasible path with the objective to minimize the off-track of the tractor and the trailer swept areas and avoid collisions with static obstacles. In this section, we first introduce our vector form representations of the input and the output of the neural network and the proposed encoder-decoder network architecture. We then discuss the key concept of our approach, i.e., how do we guide the training of the neural network using a path cost function. Lastly, we present the design of the path cost function and the kinematic model of the tractor-trailer system as it plays an important role in constructing the path cost function.

A. Input Output Representation

For the input to our network, i.e., lane features and static obstacles, we use intermediate and vector representation. As pointed out in [8], using vector representation "avoid lossy rendering and computationally intensive ConvNet encoding steps"; while using intermediate representation enhances the transferability of the model compared to using raw sensor data, meaning we can more easily apply a model trained with intermediate representation in simulation to real scenarios expecting similar model performance. Also, using vector and intermediate representation facilitates the artificial data generation used for our model training, which we will discuss in the next section.

As depicted in Figure 2. , we use a series of waypoints (each point in x, y coordinate) to represent the target lane center, or the reference path the tractor-trailer is trying to follow (in colored rendition: black crosses are behind the current tractor rear axle center position, and cyan ahead); we use 6 points (red hexagons) to represent a static obstacle; and the output planned path is represented using a series of waypoints (green dots). To avoid confusion, in this paper, we also call the input reference path "anchor path", as each waypoint in the planned path corresponds to and is supposed to stay not too far from (hence "anchor") a waypoint in the input reference path (except for the first point of the planned path). The output planned path is for the tractor (as opposed to the trailer) and can be used as a reference path to a lower-level trajectory planner, which adds time or velocity information to the path.

Manuscript 1148 submitted to 2022 25th IEEE International Conference on Intelligent Transportation Systems. Received March 31, 2022.

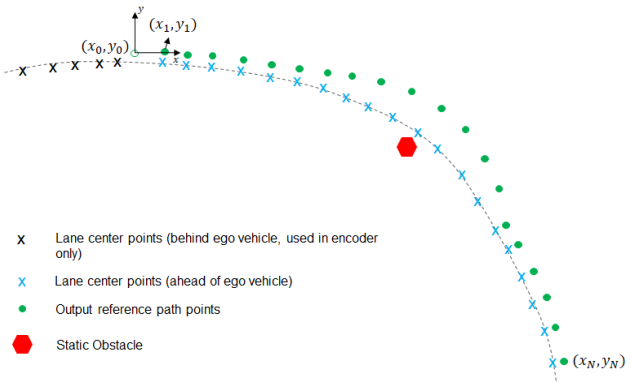
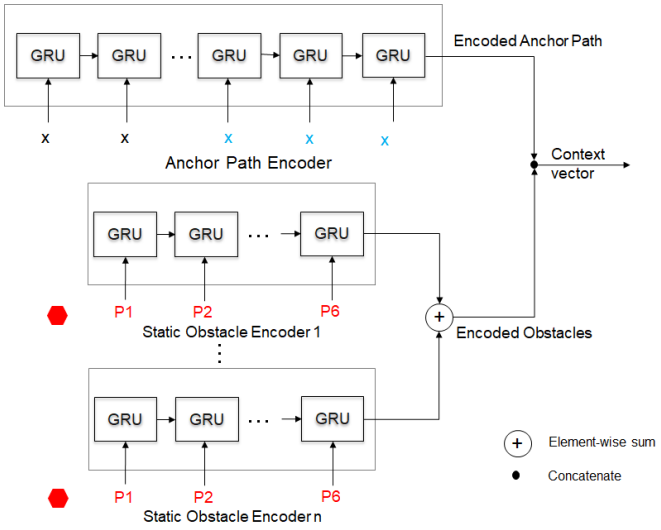


Figure 2. Illustration of input features and the output path of the network

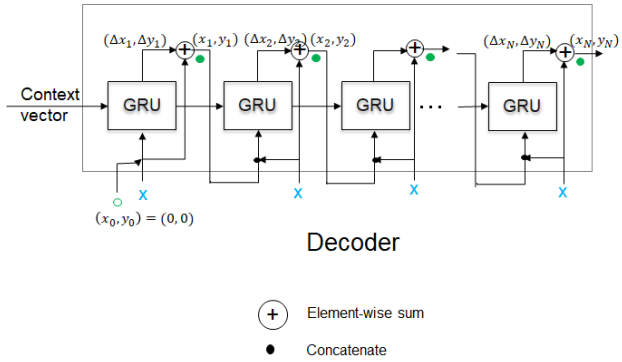
Note that everything is in the tractor coordinate at current step, with the origin at the rear axle center of the tractor projected to the ground plane and the x-axis aligning with the tractor body heading. Therefore, the planned path would always start from the origin of the coordinate, i.e., $(x_0, y_0) = (0, 0)$.

B. *Model Architecture*

In this paper we use an encoder-decoder type of neural network as our path planner, with all the encoders and the decoder implemented using GRUs (gated recurrent units). For clarity, the GRU cells are unrolled in Figure 3. with corresponding inputs and outputs denoted. In the encoder case, each cell takes a point (either a waypoint or a vertex of an obstacle hexagon, in the form of x, y coordinates) as its input and produces a hidden state; in the case of the decoder, the input to a cell at each step is the concatenation of the output from the last step and the corresponding anchor waypoint at this step, while the output at this step is the sum of the GRU cell output and the anchor point at this step.



(a)



(b)

Figure 3. The encoder-decoder architecture for the AI path planner (a) the anchor path encoder and the static obstacle encoders (b) the decoder

For simplicity, we combine all the static obstacle encodings using the summation operation, the encoded obstacles are then concatenated with the encoded anchor path to form the context vector the decoder uses to generate the planned path. Other ways to combine the encodings from different encoders can also be used, e.g., graph neural network (GNN), self-attention mechanism.

C. *The Semi-Supervised Approach*

Once the model is ready, the next problem is to train the model to generate a desired path that, if the tractor follows, yields minimal off-track of the tractor and the trailer swept areas and avoids collisions with static obstacles, while being smooth and feasible at the same time.

With expert driving data, it is straightforward to train the model in a supervised learning fashion, with the expert driven path as the target, and a function such as mean squared error (MSE) as the loss function, which basically evaluates the point-to-point proximity between the network-generated path and the target path. Any discrepancy between the two path is calculated as the training loss and is backpropagated through the encoder-decoder network for the weight update. The training pipeline for the supervised approach is illustrated in Figure 4. (a).

The supervised approach is a pure imitation of the "expert" driving and while works for the simplest scenarios (i.e., without static obstacles), suffers from several drawbacks: 1) it requires collection of a large amount of data of "good" driving with enough "interesting" scenarios (turns of various curvatures); 2) it might learn the "random wandering" about the lane center, as human drivers never stay perfectly aligned with the lane center; 3) since the network learns from the expert, it can never hope to better the expert driving; and 4) it is extremely hard, if not impossible, to teach the network to avoid static obstacles using the supervised approach, as the training data needed for that purpose would require so many combinations of the lane shapes and different sizes and placements of static obstacles relative to the lanes that it is impractical to collect from real vehicle tests.

To overcome these drawbacks and difficulties, we propose the semi-supervised approach, where we replace the MSE kind of loss function (that requires a target path from expert driving and evaluates the discrepancy between the network output path

the target path) with an absolute, "true" path cost function that evaluates the quality of a path without having to compare it to some expert path, but by applying a set of predefined criteria. The training pipeline for the semi-supervised approach is illustrated in Figure 4. (b).

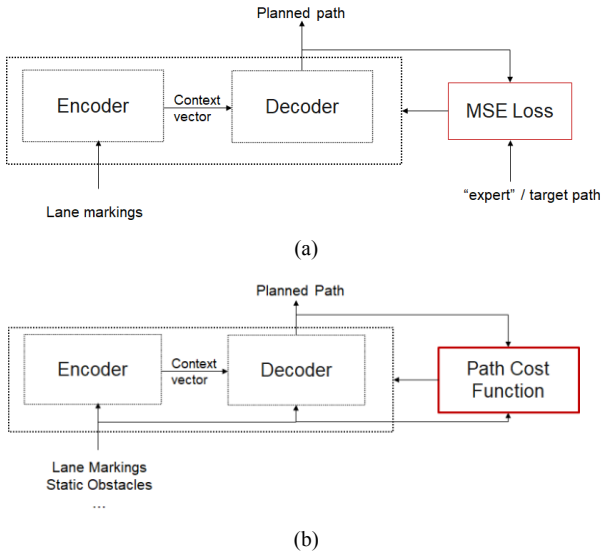


Figure 4. Training pipelines / concepts for the supervised approach (a) and the semi-supervised approach (b).

The path cost function calculates a cost for each path based on its relationship with the anchor path (e.g., the deviation from the anchor path) and its own attributes (e.g., the path smoothness). The higher the cost, the "worse" the path: a path with zero cost is a "perfect" path (in terms of the definition of the cost function). The path cost is then backpropagated through the neural network to guide the end-to-end training, updating the parameters of the encoders and the decoder in the direction that would minimize the cost. This, of course, requires design of a differentiable path cost function, which is presented next.

D. Path Cost Function

We designed our path cost function with the following criteria of a "good" path in mind: a good path should be safe (collision free), rule-following (e.g., keeping in the lane, or following the anchor path in our case), feasible and comfortable (smooth). Based on this set of criteria, the total cost of a path can be represented as a linear combination of several individual costs, as in the following equation, each corresponding to one criterion.

$$C_{total} = (w_{11}C_{tractor-dev} + w_{12}C_{trailer-dev}) + w_2C_{yaw-acc} + w_3C_{collision} + w_4C_{other} \quad (1)$$

Below, we explain each individual cost in detail.

1) Path Deviation Cost

$(w_{11}C_{tractor-dev} + w_{12}C_{trailer-dev})$ consists of the path deviation cost (which is inspired by the optimization objective 3 in [7]). It penalizes the deviation of both the tractor path and the trailer path from the anchor path and corresponds to the rule-following criterion.

We quantify the deviation of a path from the anchor path using the mean squared lateral distances of all the points of that path to the anchor path. In the case of the tractor path, the deviation cost can be written as

$$C_{tractor-dev} = \frac{1}{N}\sum_{j=1}^{N}\left[lat\_dist(\boldsymbol{x_j}, path_{anchor})\right]^2 \quad (2)$$

Where $\boldsymbol{x_j} = (x_j, y_j)$ is the j-th point of the tractor path; the anchor path $path_{anchor}$ is piece-wise linear as we represent it with a series of waypoints; and $lat\_dist(\cdot,\cdot)$ is a function that calculates the lateral distance from a point to a path.

Note the AI path planner plans a path for the tractor to follow, i.e., the network outputs only the tractor path. To evaluate the trailer path deviation from the anchor path, we need to infer the trailer path from the output tractor path, which would require a kinematic model of the tractor-trailer system, which we discuss in the next subsection (III.E).

2) Tractor Yaw Acceleration Cost

The term $w_2C_{yaw-acc}$ is the tractor yaw acceleration cost and enforces smoothness of the planned path. $C_{yaw-acc}$ can be computed as the mean squared yaw acceleration at all (tractor) path points:

$$C_{yaw-acc} = \frac{1}{N-1}\sum_{j=0}^{N-2}\left(\frac{\omega_{j+1}-\omega_j}{\Delta s_j}\right)^2 \quad (3)$$

Where Δs_j is the delta distance between path point j and j+1; ω_j is the tractor yaw rate at path point j and is defined as $(\theta_{j+1}-\theta_j)/\Delta s_j$; the tractor yaw angle θ_j is in turn defined as $\theta_j = \arctan(\frac{y_{j+1}-y_j}{x_{j+1}-x_j})$.

3) Collision Cost

The term $w_3C_{collision}$ is the collision cost that promotes collision avoidance of the tractor and the trailer bodies with static obstacles and corresponds to the safe criterion. With n static obstacles, $C_{collision}$ can be written as

$$C_{collision} = \sum_{i=1}^{n}(C^{[i]}_{tractor,col} + C^{[i]}_{trailer,col}) \quad (4)$$

Where $C^{[i]}_{tractor,col}$ is the collision cost incurred by the i-th static obstacle with the tractor body ($C^{[i]}_{trailer,col}$ the trailer), and is implemented as a penalty to small distance between an obstacle and the tractor path:

$$C^{[i]}_{tractor,col} = \max\{0, -k \cdot [\min_j(lat\_dist(P_j^{[i]}, path_{tractor})) - thresh]\} \quad (5)$$

Where $P_j^{[i]}$ is the j-th vertex point of the i-th static obstacle; k and *thresh* are tunable constants. Obstacle-to-path distance smaller than the threshold distance is penalized linearly with respect to the distance difference (obstacle-to-path distance minus *thresh*), while obstacle-to-path distance larger than the threshold distance is not penalized.

4) Other Costs

In addition to the three main costs introduced before, we also include some other costs (grouped in the C_{other} term) for different purposes, namely initial yaw cost and anchor points approximation cost, which we only briefly mention here without going to details.

The initial yaw cost penalizes large yaw angle at the first point of the planned path and can be written as $\|\theta_0\|^2$. This cost enforces a (soft) constraint that the tractor path starts with zero heading since we chose the tractor coordinate at the current time as our coordinate system (Figure 2).

The anchor points approximation cost is equivalently a regularization term and is simply the mean squared error of the tractor path and the anchor path, or $mse(path_{tractor}, path_{anchor})$.

When implemented properly, all the individual costs are differentiable, thus the total cost can be backpropagated through the neural network for training. In addition to being used as the loss function in training time, the path cost function can also be used in execution time to evaluate the quality of a planned path to see if it is feasible.

E. Tractor-Trailer Model and Trailer Path Inference

An integral piece in constructing the path cost function is a kinematic model for the tractor-trailer system, as it enables the inference of the corresponding trailer path from a given tractor path.

Similar to the idea of a bicycle model for two-axle vehicles, we can use a tricycle model to represent the articulated structure of a truck and trailer system, as illustrated in Figure 5. below.

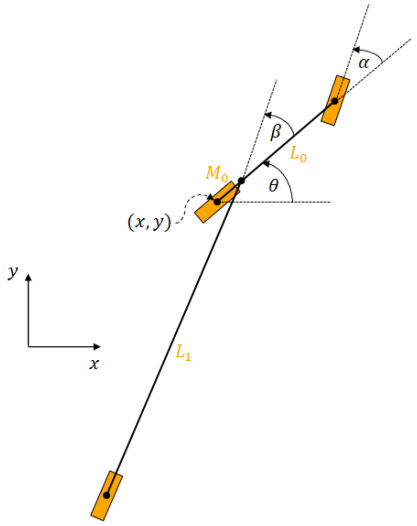


Figure 5. Truck-trailer tricycle model

In Figure 5. , α is the steering angle of the front wheel of the tractor; (x, y) is the position of the rear axle center of the tractor; θ is the yaw angle of the tractor; β is the trailer angle; L_0 is the tractor wheelbase; M_0 is the distance from the trailer hinge point to the tractor rear axle; and L_1 is the distance from the trailer hinge point to the trailer rear axle.

We can now present the system dynamics of the tractor-trailer kinematic model with respect to s, the distance along the path, as we are interested in path instead of trajectory.

$$\begin{cases} {dx}/{ds} = \cos\theta \\ {dy}/{ds} = \sin\theta \\ {d\theta}/{ds} = \dfrac{\tan\alpha}{L_0} = u \\ {d\beta}/{ds} = -\dfrac{1}{L_1}\sin\beta + (\dfrac{M_0}{L_1}\cos\beta - 1)\dfrac{\tan\alpha}{L_0} \end{cases} \quad (6)$$

Notice that, given a tractor path in form of a series of waypoints (x_j, y_j), we can calculate the (modified) control input $u = {\tan\alpha}/{L_0}$ along the path, which can then be substituted into the trailer angle dynamics equation to calculate the trailer angle. Once we have the trailer angles, we can compute the full trailer path.

IV. Results

We describe in this section how we generate data for training and validation, some details of the training process, and present the training results with a validation dataset and in closed loop simulation in IPG CarMaker.

A. Data Generation

To train the AI path planner, we randomly generate a very large number (in the order of millions) of combinations of different lane shapes and placements of static obstacles by independently sampling from a (infinite) set of anchor paths and a (infinite) set of static obstacles with respective predefined distributions.

The anchor paths set we sample from contains two types of lane shapes: a fifth order polynomial, and an arc preceded by a line segment. The polynomial coefficients are sampled from some uniform distributions with zero means; the arc angle is uniformly sampled from $[{\pi}/{3}, \pi]$; and the path length varies in the range of [80, 250] meters.

We then randomly generate 0 to n (in this paper we chose n to be 2) static obstacles along the generated anchor path, each obstacle positioned in the range of [0, 10] meters from some randomly selected anchor point with a random angle, the size of each obstacle (the side length of the obstacle hexagon) is in the range of [0.2, 2] meters.

Figure 6. shows two examples of the generated data.

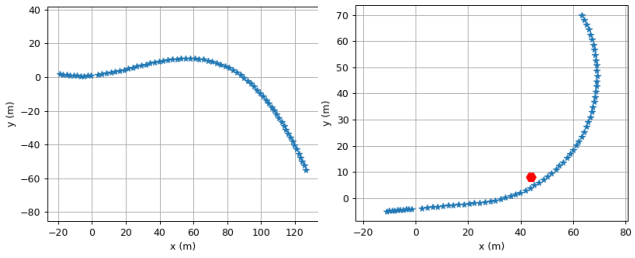


Figure 6. Random generated data samples. Left: a polynomial anchor path without static obstacle; Right: a line segment + arc type of anchor path with one static obstacle near the path

B. *Training*

We implemented our model in PyTorch, with each encoder hidden size selected to be 1000, and the decoder hidden size 2000; the number of recurrent layers of the GRU is selected to be 3. Since we use GRU for our encoders and decoder, we chose Adam as the default optimizer. The learning rate is set to 2e-5 initially and is decreased towards the end of the training.

For each epoch, we generate 50000 new data samples (we have unlimited data). The learning process takes about 40 to 100 epochs, meaning our model would have seen 2-5 million data examples to learn to plan a decent path.

C. *Validation Result*

In Figure 7. , we demonstrate some examples of the results applying our model to the validation dataset after training.

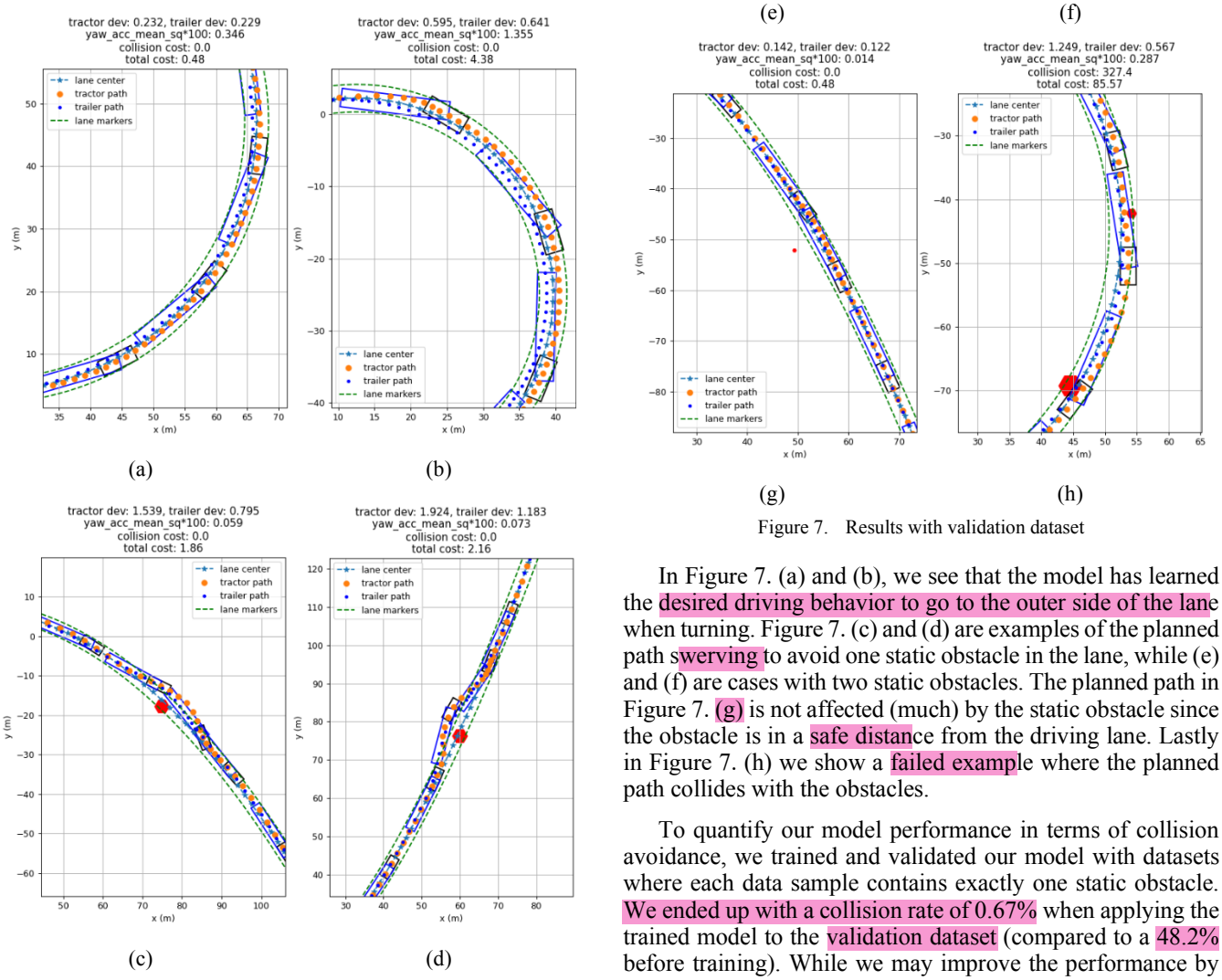


Figure 7. Results with validation dataset

In Figure 7. (a) and (b), we see that the model has learned the desired driving behavior to go to the outer side of the lane when turning. Figure 7. (c) and (d) are examples of the planned path swerving to avoid one static obstacle in the lane, while (e) and (f) are cases with two static obstacles. The planned path in Figure 7. (g) is not affected (much) by the static obstacle since the obstacle is in a safe distance from the driving lane. Lastly in Figure 7. (h) we show a failed example where the planned path collides with the obstacles.

To quantify our model performance in terms of collision avoidance, we trained and validated our model with datasets where each data sample contains exactly one static obstacle. We ended up with a collision rate of 0.67% when applying the trained model to the validation dataset (compared to a 48.2% before training). While we may improve the performance by tuning some hyperparameters and / or modifying the model architecture, we realize that due to the (probabilistic) nature of the deep learning approach, it may be impossible to ever achieve a 0% collision rate, which is a requirement for lower-level motion planners. Nevertheless, by combining our AI path planner with some backup motion planner (such as MPC based), we can achieve inference speed and ensure safety at the same time. One way to complete our AI path planner and

guarantee 0% collision rate with static obstacles is, in the rare cases where the AI planned path fails the feasibility check, instead of using the planned path directly, we only use it as a starting point for further optimization by the backup planner.

D. Closed-loop Simulation

The trained AI path planner has been integrated with a lower-level trajectory planner, which takes the planned path as a reference path and adds velocity information to it. We conducted a closed-loop simulation in CarMaker (for the no obstacle case) and successfully drove the predefined route, proving the model transferability from being trained with randomly generated artificial data segments to being applied to realistic scenarios. As can be seen in Figure 8. , the planned path tends to go to the outer side of the lane when making the turn. First in-vehicle tests on the same track showed comparable results.

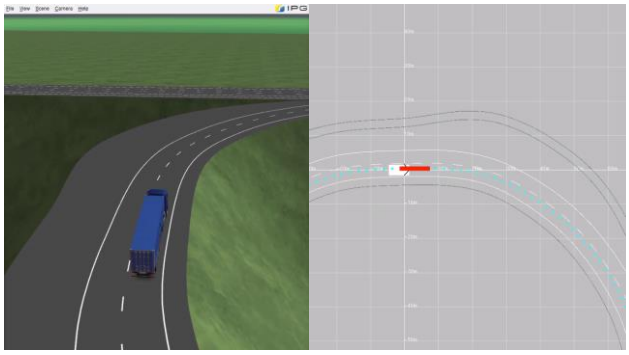


Figure 8. Left: Screenshot of a simulation scene in CarMaker; right: the corresponding visualization of our trajectory planner with the planned trajectory (red) and the AI planned path (cyan)

V. Conclusion

In this paper we presented a novel AI path planner that can be trained entirely on randomly generated artificial data to plan a feasible path for tractor-trailer combinations with the objective to minimize the off-track of the tractor and the trailer swept areas and avoid collisions with static obstacles. To guide the training of the path planner, which is implemented in an encoder-decoder architecture, a differentiable path cost function is constructed to evaluate the cost of a path, which is backpropagated through the network during the training. We call our approach "semi-supervised" approach, as no expert data is needed for the "supervision" of the training. Compared to the supervised approach, our approach 1) requires no collection of expert driven data; 2) can better the expert driving as it is not imitating the expert, rather, it is trained to minimize the "true" cost of a path in an absolute, objective sense, given that the path cost function is properly designed; and 3) makes it possible to teach the network to avoid static obstacles, as the data needed to train the network for that purpose can be easily generated without limitation.

The trained path planning model was validated with a validation dataset and in closed-loop simulation in CarMaker as well. The results showed the model successfully learned the desired driving behavior to go to the outer side of the lane when turning, and it also learned to avoid static obstacles in most cases (> 99%). Due to the probabilistic nature of deep learning approaches, though, a 100% path feasibility can be hard to achieve for the paths generated by the AI path planner. We propose additional safety mechanisms – e.g., feasibility check and backup motion planner – be integrated with the AI path planner to ensure the safety while leveraging the fast inference speed of the AI path planner.

Future work may include experimenting with new model architectures (e.g., other ways to combine the encodings from different encoders) to further boost the model performance, and considering other features, such as free space, as the model input.

References

[1] O. Ljungqvist, N. Evestedt, D. Axehill, M. Cirillo and H. Pettersson. "A path planning and path-following control framework for a general 2-trailer with a car-like tractor." *Journal of Field Robotics* 36 (2019): 1345 - 1377.

[2] K. Bergman, O. Ljungqvist and D. Axehill, "Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control," in IEEE Transactions on Intelligent Vehicles, vol. 6, no. 1, pp. 57-66, March 2021, doi: 10.1109/TIV.2020.2991951.

[3] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill and O. Holmer. "Lattice-based motion planning for a general 2-trailer system," 2017 IEEE Intelligent Vehicles Symposium (IV) (2017): 819-824.

[4] L. Palmieri, S. Koenig and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 2775-2781, doi: 10.1109/ICRA.2016.7487439.

[5] P. Zips, M. Böck and A. Kugi, "An optimisation-based path planner for truck-trailer systems with driving direction changes," 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 630-636, doi: 10.1109/ICRA.2015.7139245.

[6] B. Li, Y. Zhang, T. Acarman, Q. Kong and Y. Zhang. "Trajectory Planning for a Tractor with Multiple Trailers in Extremely Narrow Environments: A Unified Approach." *2019 International Conference on Robotics and Automation (ICRA)* (2019): 8557-8562.

[7] R. Oliveira, O. Ljungqvist, P. F. Lima and B. Wahlberg. "Optimization-Based On-Road Path Planning for Articulated Vehicles." *ArXiv* abs/2001.06827 (2020): n. pag.

[8] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li and C. Schmid. "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation." *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020): 11522-11530.

[9] M. Bansal, A. Krizhevsky and A. Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst." *ArXiv* abs/1812.03079 (2019): n. pag.

[10] S. Shalev-Shwartz, S. Shammah, A. Shashua, "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving." *ArXiv* abs/1610.03295 (2016): n. pag.

[11] S. Grigorescu, B. Trasnea, T. Cocias and G. Macesanu, "A survey of deep learning techniques for autonomous driving." *Journal of Field Robotics* 37 (2020): 362 - 386.

[12] S. Aradi, "Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 2, pp. 740-759, Feb. 2022, doi: 10.1109/TITS.2020.3024655.