E2E Parking: Autonomous Parking by the End-to-end Neural Network on the CARLA Simulator

Yunfan Yang, Denglong Chen, Tong Qin*, Xiangru Mu, Chunjing Xu, and Ming Yang

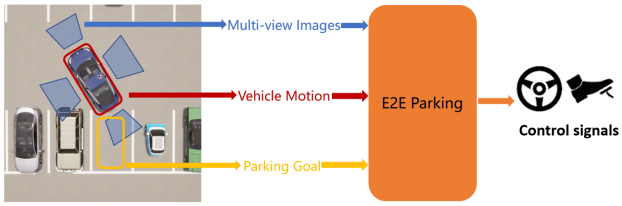
***Abstract*— Autonomous parking is a crucial application for intelligent vehicles, especially in crowded parking lots. The confined space requires highly precise perception, planning, and control. Currently, the traditional Automated Parking Assist (APA) system, which utilizes geometric-based perception and rule-based planning, can assist with parking tasks in simple scenarios. With noisy measurement, the handcrafted rule often lacks flexibility and robustness in various environments, which performs poorly in super crowded and narrow spaces. On the contrary, there are many experienced human drivers, who are good at parking in narrow slots without explicit modeling and planning. Inspired by this, we expect a neural network to learn how to park directly from experts without handcrafted rules. Therefore, in this paper, we present an end-to-end neural network to handle parking tasks. The inputs are the images captured by surrounding cameras and basic vehicle motion state, while the outputs are control signals, including steer angle, acceleration, and gear. The network learns how to control the vehicle by imitating experienced drivers. We conducted closed-loop experiments on the CARLA Simulator to validate the feasibility of controlling the vehicle by the proposed neural network in the parking task. The experiment demonstrated the effectiveness of our end-to-end system in achieving the average position and orientation errors of 0.3 meters and 0.9 degrees with an overall success rate of 91%. The code is available at: `https://github.com/qintonguav/e2e-parking-carla`**

I. Introduction

The advent of autonomous driving has revolutionized the transportation industry, with a strong focus on developing advanced systems capable of safely navigating and parking. Autonomous parking, in particular, poses a set of challenges due to the intricate maneuvering and precise control required in confined spaces. Traditional approaches to autonomous parking [1] often involve a multi-stage pipeline, including perception, mapping, planning, and control modules, each with its complexities and limitations. Due to the sensor's noise and the model's uncertainty, the system accumulates errors step by step. Moreover, the handcrafted rules in planning and control lack flexibility and are easily suffered from accumulated errors. Therefore, the traditional multi-stage rule-based system is fragile and fails to scale up in various complicated environments.

In recent years, the end-to-end learning paradigm has emerged as a promising alternative for tackling complex tasks in autonomous driving. By directly mapping raw sensor inputs to control actions, end-to-end systems offer the potential to simplify the overall pipeline, enhance adaptability, and reduce reliance on handcrafted features and rules. This approach has shown remarkable success in tasks such as lane keeping, object detection, and collision avoidance. Motivated by the advantages of end-to-end learning, we present a novel application of this paradigm, which is E2E Parking.

(a) Autonomous Parking Task



(b) Overall Structure

Fig. 1: Illustration of the overall structure. We propose an end-to-end framework for autonomous parking, which directly maps the visual information into control signals by a neural network. The neural network learns to control the vehicle by imitating experts. The video demonstration can be found at: `https://youtu.be/SfnV13YdQow`.

In our approach, inputs are surround images captured by onboard cameras and basic vehicle motion state, while outputs are vehicle chassis control signals comprising steer angle, acceleration, and gear. By adopting this end-to-end format, we aim to enable vehicles to perform parking maneuvers smoothly and efficiently, without explicitly handcrafted models and rules. In addition, we propose to design a system where no external localization module is required. Vehicles are able to track the parking goal once it has been chosen at the beginning. Through extensive training and fine-tuning, our end-to-end system learns intricate mapping from visual cues to control actions, enabling the vehicle to autonomously navigate and park in parking slots. To evaluate the effectiveness of our proposed approach, we designed quantitative metrics and conducted closed-loop experiments on the CARLA [2] Simulator. The results show that our method can reach an overall success rate of 91% with the

Yunfan Yang, Denglong Chen, Xiangru Mu, and Chunjing Xu are with IAS BU, Huawei Technologies, Shanghai, China. Tong Qin, and Ming Yang are with Glocal Institute of Future Techonology, Shanghai Jiao Tong University, Shanghai, China. {yangyunfan8, chendenglong, muxiangru, xuchunjing}@huawei.com, {qintong, mingyang}@sjtu.edu.cn. * is the corresponding author.

average position and orientation error of 0.3 meters and 0.9 degrees. We will release the CARLA parking datasets and the source code for the convenience of future researchers.

Overall, the main contributions of this work include:

- We proposed a novel and feasible end-to-end visual parking solution that directly maps images and motion states to the **control signals** for parking. By eliminating the need for handcrafted features and rules, our system simplifies the overall pipeline and improves adaptability to diverse parking scenarios.
- We conducted **closed-loop** experiments on the CARLA simulator and proposed comprehensive evaluation metrics to validate the feasibility and reliability of our end-to-end autonomous parking system.
- We established the quantitative benchmark on parking tasks in CARLA and published the parking datasets generated in CARLA for public availability.

II. LITERATURE REVIEW

Over the last decades, there has been a huge breakthrough in neural networks. We first give an overlook of popular components and architectures used in neural networks, e.g. transformer and BEV modules. Then we review how previous work adopts these neural networks into autonomous driving tasks.

A. Transformer Model

Transformer [3] was originally applied in the field of natural language processing (NLP), achieving significant success in sequence modeling tasks, such as GPT [4]–[6], by leveraging its unique self-attention mechanism to capture long-range dependencies. Through continuous exploration, the transformer architecture has also been used in widespread applications of computer vision and autonomous driving tasks. For computer vision, ViT [7] partitioned images into patches and encoded them as sequential inputs, enabling the transformer to capture both global and local visual features, thus enhancing image classification performance. DETR [8] was a transformer-based object detection approach that achieved accurate object detection through an end-to-end process by employing attention mechanisms. For autonomous driving, Transfuser [9] and InterFuser [10] designed a transformer for the multi-scale fusion of features from image and LiDAR. UniAD [11] built a unified autonomous driving framework, including perception and prediction modules based on the transformer decoder. CIL++ [12] proposed an end-to-end autonomous driving framework that used a transformer encoder to aggregate features from multi-view images, velocity, and navigation commands. ParkPredict+ [13] used a transformer to predict waypoints in parking lots. Inspired by the above-mentioned works, we adopt the transformer framework to directly output control signals in the parking scenarios.

B. BEV Representation

The bird's-eye-view (BEV) is a commonly used data representation in the field of autonomous driving, like semantic segmentation M2Bev [14] and CVT [15], and object detection BEVDet [16] and BEVFormer [17]. It provides a top-down perspective of the vehicle's surrounding environment, capturing essential 3D information about obstacles, lanes, and other traffic participants. Without depth information, the 2D image feature can not be converted to the BEV directly. Numerous works studied how to transfer 2D information into BEV. LSS [18] learned the depth distribution of each pixel and used camera parameters to transform the frustum into a BEV representation. BEVDepth [19] added explicit depth supervision based on LSS to improve depth estimation accuracy. Transfuser and its variants [9, 20] employed a transformer to generate BEV representations from the fused Lidar and image features. ThinkTwice [21] relied on LSS to extract intermediate BEV features for subsequent decoding. ST-P3 [22] learned a spatio-temporal BEV feature from multi-view cameras in the past several timestamps. To balance accuracy and efficiency, we adopt modules used in LSS [18] to generate BEV representation.

C. End-to-end Autonomous Driving

In recent years, end-to-end autonomous driving [23] based on neural networks has become a hot topic. Compared with the traditional hierarchical approach [24], end-to-end methods do not rely on heavy rules and have stronger generalization capability for various scenarios.

Urban Scenario: Wayve [25] and Openpilot [26] proposed an end-to-end neural network to predict the waypoints directly from the camera images. Many methods adopted GRU [27] for auto-regressive waypoints generation from BEV features, including Transfuser [9], Transfuser++ [20], and InterFuser [10]. The above methods of directly outputting waypoints required a controller (such as PID) to achieve final actions. However, the controller has tracking errors in some complex scenarios. CIL [28] proposed a neural network that directly mapped front-view image, current measurements, and navigation commands to control signals. To reduce training data bias and causal confusion, CILRS [29] added a ResNet backbone and a speed prediction branch based on CIL. LSD [30] designed a hybrid imitative learning model with multiple strategies to adapt to multiple driving scenarios. TCP [31] fused a waypoints GRU branch and a control GRU branch to boost performance. ThinkTwice [21] designed a scalable decoder module to generate waypoints and control signals by the coarse-to-fine strategy.

Parking Scenario: Currently, several methods for parking tasks focus on traditional hierarchical framework [32, 33], while few end-to-end neural network parking methods appear. Rathour *et al.* [34] designed a CNN model to directly map front and rear images to steering and gear. Li *et al.* [35] only used the rear image as input to a CNN to output steer angle and speed. ParkPredict [36] proposed a parking spot and waypoints prediction network based on CNN and LSTM, but the inputs did not contain any raw camera images either. ParkPredict+ [13] replaced the LSTM in ParkPredict with a transformer, while the inputs consisted of semantic BEV images. None of the above methods can directly obtain

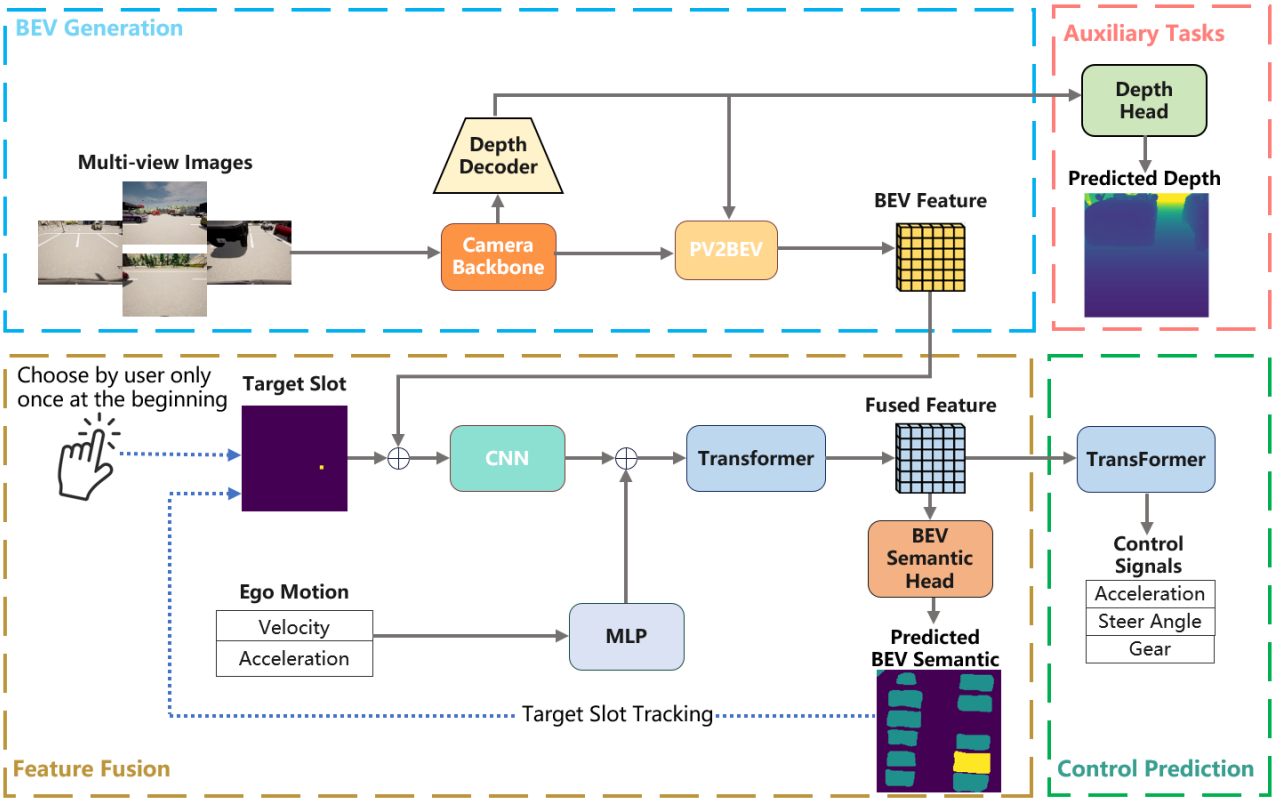


Fig. 2: The framework of the proposed system contains BEV generation, feature fusion, control prediction, and auxiliary task. The BEV Generation module converts surrounding images into a BEV feature map using the LSS [18] method. Then, the BEV feature, vehicle motion, and target slot are fused via the transformer. The relative position of the target slot in the BEV grid (yellow point) is drawn on an extra channel and concatenates to the BEV feature map. A language-modeling based transformer decoder receives the fused feature and predicts the control signals. Depth prediction is supervised as an auxiliary task. The green square in the BEV semantic prediction is the surrounding vehicle, while the yellow square is the target parking slot.

complete control signals (steer angle, acceleration, gear) from the four raw images. To take full advantage of the end-to-end format, we use a transformer to map visual images to the control signals directly.

III. METHODOLOGY

In this work, we propose a novel architecture for visual end-to-end parking. Conventional approaches to address the autonomous parking task typically rely on a multi-stage pipeline, which consists of multiple independent modules. However, our approach integrates those modules into one unified network that maps visual measurements to vehicle control signals directly.

A. Problem Formulation

Given the sensor measurements, the goal is to safely and precisely control the vehicle into the desired parking slot. The model is trained in a supervised manner with datasets, $\mathcal{D} : \boldsymbol{\tau}^n, n \in (0, N)$, where contains N parking trajectories from experts. Each trajectory contains observations and control signals at every instant: $\boldsymbol{\tau}^n : [\mathcal{X}_t^n, \mathcal{C}_t^n], t \in (0, T^n)$. $\mathcal{X}_t^n$ contains raw images from surround cameras, target spot position, and the vehicle's velocity in the nth route at time t. Similarly, $\mathcal{C}_t^n$ contains acceleration, steer angle, and gear state in the nth route at time t. We consider the training as a supervised manner, where the input is $\mathcal{X}$ and the output is supposed to be $\mathcal{C}$. A policy π is learned to map the observation to the control signals. Therefore, the system can be formulated as:

$$\arg\min_{\pi} \mathbb{E}_{(\mathcal{X},\mathcal{C})\sim\mathcal{D}}[\mathcal{L}(\mathcal{C}, \pi(\mathcal{X}))], \tag{1}$$

where $\mathcal{L}$ is the loss function. To enhance the interpretability of the neural network, we further add depth prediction and BEV segmentation tasks, which are discussed in more detail in Sect. III-D.

B. Input and Output

Input Representations: We use camera images, vehicle's states, and the position of the target parking slot as the inputs to the model. We adopt 4 surrounding cameras (front, left, right, and rear) at the position where the lines on the parking slot can be captured. The vehicle's states contain the velocity and acceleration of the ego-vehicle, which come from the wheel odometer and IMU (Inertial Measurement Unit). It's necessary for the model to perceive the current status to generate smooth and reasonable control signals in parking scenarios. As a parking task, a target slot is required.

The target slot is appointed by the user for the first time, who points a point under the vehicle's BEV coordinate. In the following, the target slot is continuously tracked by the segmentation task.

Output Representations: Our model predicts vehicle control signals for the next 4 steps with 0.1s time interval. We predict normalized acceleration, $acc_t \in [-1, 1]$, steer angle, $steer_t \in [-1, 1]$ and gear (forward and backward), $gear_t \in \{0, 1\}$. In the control module, the acceleration is linearly mapped to throttle and brake, and the steering value is linearly mapped to the steering wheel angle. Inspired by Pix2seq [37], we take the multi-step control signal prediction as a language modeling task. The model outputs control signals one by one, similar to sequentially generating every word to form the sentence. In order to tokenize the control signals, the acceleration and steer angle are discretized with a resolution of 0.01. Therefore, the acceleration and steer are discrete values in $[0, 200]$. For acceleration, 0 indicates full brake, while 200 indicates full throttle. 0 in steering means maximum left, and 200 means maximum right. By adding an extra begin (BOS, begin of sequence) and end (EOS, end of sequence) token, the sequence can be described as below:

$$\begin{aligned} S =& [BOS, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, EOS] \\ \mathbf{c}_n =& [acc_n, steer_n, gear_n]. \end{aligned} \tag{2}$$

By multi-step prediction, we hope that the model can generate smooth and consistent outputs in a short future.

C. Network Architecture

As illustrated in Fig. 2, our proposed architecture consists of several submodules: BEV Generation, Feature Fusion, and Control Prediction. The following sections will detail each of them.

1) ***BEV Generation****:* Camera-view to BEV conversion follows the LSS method [18]. Multi-view images $I \in \mathbb{R}^{3\times H\times W}$ are first processed by the backbone to acquire the image features $F_{img} \in \mathbb{R}^{C_i\times H_i\times W_i}$, where (H, W), (H_i, W_i) are the height and width of the image and the image feature respectively, and C_i is the feature channels. The image feature F_{img} is then used to estimate the discrete depth distribution D. The feature frustum for each image is created by the dot product of the depth distribution and image feature, $F_{frus} \in \mathbb{R}^{D\times C_i\times H_i\times W_i}$. With the camera extrinsics and intrinsics, feature frustums are projected into the BEV voxel grid. Voxel summing is implemented to merge the features at the same voxel. The final BEV feature map is represented as $F_{bev} \in \mathbb{R}^{C_b\times X_b\times Y_b}$, where (X_b, Y_b) is the BEV grid size and C_b is the number of channels.

2) ***Feature Fusion****:* Besides surrounding visual information from the BEV feature, ego motion and target slot are also required. In the Feature Fusion submodule, the BEV feature map, target slot, and ego motion are fused into one unified feature. We first project the target slot into an extra BEV grid, $F_{target} \in \mathbb{R}^{1\times X_b\times Y_b}$, where the target slot position is marked as one, while other grids are set to zero. By concatenating F_{bev} and F_{target} together, we get the final bev feature, $F_{bev} \in \mathbb{R}^{(C_b+1)\times X_b\times Y_b}$. Through addition 2D convolution and reshape, we get feature $F_{bev'} \in \mathbb{R}^{C'_b\times L}$ with C'_b channels and L length. Then, the current ego motion (velocity, acceleration) is represented as $F_{ego} \in \mathbb{R}^{2\times L}$ through an MLP motion encoder. Finally, we concat $F_{bev'}$ and F_{ego} together to get the fused feature $F_{fuse} \in \mathbb{R}^{(C'_b+2)\times L}$. We utilize the self-attention mechanism by taking the feature F_{fuse} into the Transformer encoder. The visualization of the BEV features' attention is shown in Fig.5.

3) ***Control Prediction****:* We use a language-modeling based transformer decoder to predict the control signal sequence in an auto-regressive manner. By leveraging the global receptive field through attention, the transformer decoder takes spatial information from the fused feature F_{fuse} and the temporal information from the sequence positional embedding to generate control signals. The decoder takes a direct mapping from perceived environmental features to the vehicle controls.

In detail, the fused feature and embedded tokens correlate to each other via the cross-attention mechanism. The keys K and values V of the attention come from the output of the Feature Fusion submodule F_{fuse}, while the empty sequence served as the queries Q. The decoder iteratively generates the control signal tokens. The newly generated token is spliced into the sequence and then the decoder takes the new sequence to generate the next token until the maximum length is reached.

The output tokens are detokenized into normalized acceleration, normalized steer angle, and gear. Then they are linearly mapped into throttle, break, steer angle, and gear status.

D. Loss and Auxiliary Task

We have two main loss functions and one auxiliary task.

1) ***Control Signal Loss****:* We use cross-entropy loss to measure the gap between the predicted control sequence and the ground truth control sequence.

2) ***Semantic Segmentation Loss****:* For semantics, the model classifies BEV grid objects into three categories: vehicle, target slot, and background. The BEV semantic head works on top of the fused feature F_{fuse}. The semantic output is also supervised with cross-entropy loss.

We use the predicted semantic target slot to achieve target slot tracking. In detail, we extract the center (the mean of x, y) of the predicted target slot and treat it as the target slot input for the next time. In this way, the user only needs to appoint the target slot at the first time. The network will track this target slot automatically. In the train time, we give a noisy target slot position into the network to imitate the tracking process.

3) ***Auxiliary Task****:* BEVDepth [19] demonstrated that explicit depth supervision can bring a significant performance boost to the LSS method. As an end-to-end method, adding depth prediction can also improve the interpretability. Therefore, we add the depth prediction as an auxiliary task. The ground truth depths are converted into the one-hot encoding of the discrete depth bins to supervise the depth distribution prediction.

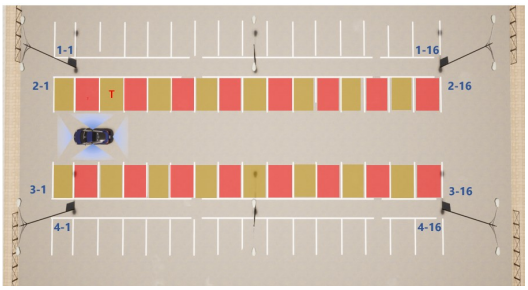
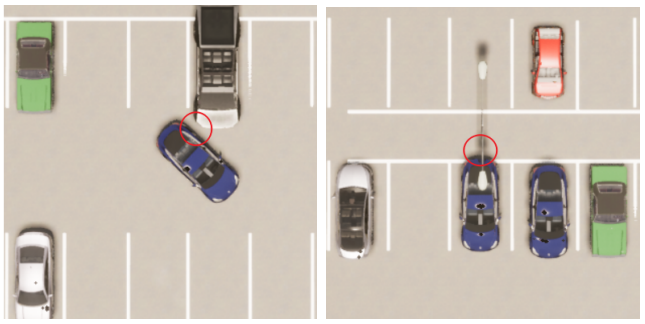


Fig. 3: The parking lot top view. The red parking spaces are used for training and the brown spaces are used for closed-loop evaluation. The text indicates the index of the slot. The red 'T' text on the parking slot indicates the target parking slot which is appointed randomly.



(a) Collision With Vehicle (b) Collision With Street Lamp

Fig. 4: Example of two scenes with high collision probability. The big truck (cyber truck) and street lamp next to the target slot are likely to cause a collision when the ego vehicle is backing into the slot.

IV. Experiments

A. Implementation Details

We used the CARLA 0.9.11 simulator for data collection and closed-loop experiments. The scenario was a parking lot provided in the map Town04-Opt, which includes 64 parking spaces, as shown in Fig. 3.

For the details of the network, the size of the BEV features is 200 $\times$ 200, corresponding to the actual spatial range of $x \in [-10m, 10m]$, $y \in [-10m, 10m]$ with a resolution of 0.1 meters. Pre-trained EfficientNet-B4 is used for the image backbone. The depth range of the depth map is $[0.5m, 12.5m]$ with a resolution of 0.25 meters. The transformer structure in the network, the encoder (feature fusion) and the decoder (control prediction), both have 4 attention layers and 6 attention heads.

During experiments, we observed that excessive vehicle speed negatively affects parking performance. Hence, we limited the speed of the ego vehicle to 12km/h and 10km/h for forwarding and backing respectively. The effectiveness of this trick will be shown in ablation studies, Sect. IV-E.

We utilized the Adam optimizer with a weight decay of $1e^{-4}$, and the beta values of Adam are set to 0.9 and 0.999. We implemented our method using the PyTorch framework. The network was trained using an NVIDIA Tesla V100 GPU with a batch size of 12, and the training took approximately 96 hours for 150 epochs with 22K frames of data.

B. Dataset collection

To collect training set data, we developed a parking data collection pipeline based on CARLA. The top view of the parking lot is illustrated in Fig. 3, which consists of 4 rows of parking spaces with 16 parking spaces per row. The target parking slot is marked with a red 'T' sign on the ground. Under each scene, every parking slot is either left empty or allocated with a random vehicle, while the target slot is always free of occupancy. The preset static vehicles by CARLA on that parking lot are removed. Then the ego vehicle is initialized at a random position of the path outside the target parking slot as long as the distance to the target slot is within 7 meters. The heading of the ego vehicle is parallel with the path. The experienced driver controls the ego vehicle parking into the target slot by the keyboard.

In each collection process, we collected camera images, corresponding depth images, ego-vehicle motion state and BEV semantic maps. The ground truth BEV semantic maps are generated by projecting the 3D bounding boxes of vehicles onto the BEV plane. To acquire ground truth depths, the ego vehicle is equipped with depth cameras at the same location as the RGB cameras during dataset generation. The control signals are also fully saved for network training.

The parking results with undesirable routes or large deviations were discarded. Specifically, the deviation between the final parking position of the ego vehicle and the desired parking position should be less than 0.5 meters for distance and 0.5 degrees for orientation (yaw angle). In total, 128 routes of parking data with random scenes were collected by 4 experienced human drivers. The data sampling frequency is set to 10Hz. We have gathered 22K frames for training. The datasets and the tool for CARLA parking data generation will be released to inspire future research.

C. Metrics

To quantitatively evaluate model performance, we chose 16 park slots and the self-driving vehicle parked 24 times for each target slot. 9 metrics used for evaluation are described as follows.

Target Success Rate (TSR) The probability that the ego vehicle successfully parks into the target parking slot. To be considered a successful park, the distance between the centre of the ego vehicle and the centre of the target slot should be less than 0.6 meters horizontally and 1 meter longitudinally, while the orientation difference should be less than 10 degrees.

Target Failure Rate (TFR) The probability that the ego vehicle parks into the target parking slot but the error is unacceptable.

Non-Target Rate (NTR) The probability that the ego vehicle parks into a non-target parking slot.

Collision Rate (CR) The probability that collision occurs during the parking process.

TaskIdx	TSR (%) ↑	TFR (%) ↓	NTR (%) ↓	CR (%) ↓	TR (%) ↓	APE (m) ↓	AOE (deg) ↓	APT (s) ↓
2-1	95.83	0.00	0.00	0.00	4.17	0.23	2.23	15.49
2-3	100.00	0.00	0.00	0.00	0.00	0.29	0.37	15.38
2-5	100.00	0.00	0.00	0.00	0.00	0.35	0.48	16.13
2-7	83.33	12.50	0.00	4.17	0.00	0.35	1.07	15.90
2-9	70.83	0.00	8.33	20.83	0.00	0.34	0.44	14.50
2-11	100.00	0.00	0.00	0.00	0.00	0.31	0.67	14.68
2-13	91.67	8.33	0.00	0.00	0.00	0.32	0.53	14.22
2-15	100.00	0.00	0.00	0.00	0.00	0.18	0.55	14.53
3-1	79.17	0.00	4.17	4.17	12.50	0.28	1.11	17.53
3-3	100.00	0.00	0.00	0.00	0.00	0.22	1.25	15.61
3-5	100.00	0.00	0.00	0.00	0.00	0.42	0.60	15.68
3-7	100.00	0.00	0.00	0.00	0.00	0.24	0.37	16.39
3-9	45.83	8.33	41.67	4.17	0.00	0.36	0.83	15.00
3-11	95.83	4.17	0.00	0.00	0.00	0.40	1.38	16.75
3-13	100.00	0.00	0.00	0.00	0.00	0.33	1.12	17.98
3-15	100.00	0.00	0.00	0.00	0.00	0.25	0.91	15.70
Avg	**91.41**	**2.08**	**3.39**	**2.08**	**1.04**	**0.30**	**0.87**	**15.72**

TABLE I: **Closed loop results.** The closed-loop experiments consist of 384 test cases in 16 different scenes. Each scene is repeated 24 times with different vehicle starting positions. All the scenes are randomly generated and are different from the training data.

Timeout Rate (TR) The probability that the ego vehicle fails to park successfully within a specified time.

Average Position Error (APE) The average error between the final position of the ego vehicle and the center of the target slot among successful parking cases.

Average Orientation Error (AOE) The average error between the final orientation (yaw angle) of the ego vehicle and the desired target slot orientation among successful parking cases.

Average Parking Time (APT) The average time spent on successful parking.

Average Inference Time (AIT) The average time for network inference one step.

D. Results

Closed-loop results. We evaluated the model's generalization ability via closed-loop experiments in the CARLA simulator. Each evaluation scene was generated with random neighbors and random target slots. Different random seeds were utilized to ensure that the scenes in the closed-loop evaluation were different from those in the training dataset. The ego vehicle initial position had the same setting with the training dataset. The vehicle was required to complete the parking task within a specified time (30s).

Table I shows the closed-loop experiments results. For the total 384 randomly generated test cases, the network reached an overall TSR (Target Success Rate) of 91.41%, which demonstrated the effectiveness of our end-to-end method. Among the 16 scenes, the network completed 9 scenes with 100% TSR. Since evaluation scenes differed from the ones used for network training, it could be shown that the network gains the generalization ability.

In most cases, the vehicle could track and park to the desired parking slot, with only 2.08% NTR (Non-Target Rate). The network successfully learned to predict the position and shape of the target slot from the one-time input by the user at the beginning.

TaskIdx	TSR (%) ↑	TFR (%) ↓	CR (%) ↓	APE (m) ↓	AOE (deg) ↓	APT (s) ↓
baseline	91.41	2.08	2.08	0.30	0.87	15.72
Expert	**100.00**	**0.00**	**0.00**	**0.23**	**0.48**	**14.96**
Rookie	75.00	18.75	6.25	0.35	4.00	20.13

TABLE II: **Comparison with Expert and Rookie Drivers.**

However, there were a small portion of failures cases contributing to a 2.08% CR (Collision Rate). We observed that the too-big trucks, and unseen objects (street lamps) near the target slot were more likely to cause collisions with the ego vehicle. Similar scenes were shown in Fig. 4. If large vehicles were removed from the scene, CR could drop to around 0.5%. Increasing the training dataset with more scenes with trucks and street lamps can potentially mitigate the problem.

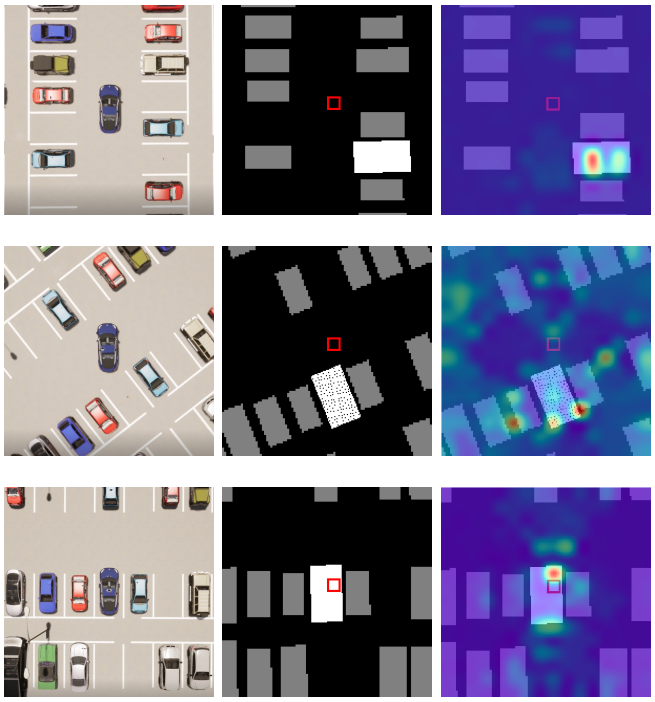
The average parking position error was 0.3m and the orientation error was 0.87°. We visualized the BEV attention in the transformer mechanism, as shown in Fig. 5. The attention focused on the target slot the most of time. Specifically, the attention focused on the stop line (rear) of the target parking slot when the ego vehicle was about to stop. This indicated that the network implicitly relied on stop line detection to decide when to fully stop the vehicle.

Parking Performance Compared with Expert and Rookie. We conducted a contrast experiment that compared control performance from our model with an expert and new driver. The data captured by the expert and rookie is the same as our training data. As is shown in Table II, the expert had the highest TSR and the lowest APE, AOE and APT. The result of our model was slightly lower than the expert in these metrics. However, compared to the rookie driver, our model achieved 16 points higher in TSR and 3 degrees smaller in AOE. In addition, the rookie driver took 5 seconds more than our model to park in the slot. The above comparison indicates that our parking model gets similar performance as the expert and is able to help the rookie driver.

Runtime. We conducted parking experiments using a

TaskIdx	TSR (%) ↑	TFR (%) ↓	NTR (%) ↓	CR (%) ↓	TR (%) ↓	APE (m) ↓	AOE (deg) ↓	APT (s) ↓	AIT (ms) ↓
baseline	**91.41**	2.08	3.39	2.08	**1.04**	0.30	0.87	15.72	74.92
w/o depth	77.08	5.21	5.47	6.25	5.99	0.29	0.80	16.37	73.66
w/o speed limit	81.51	4.43	5.21	4.43	4.43	0.39	1.25	**13.17**	79.34
MLP decoder	83.33	**1.30**	**2.86**	**1.04**	11.46	**0.25**	**0.54**	16.59	**65.72**

TABLE III: **Ablation studies.** The w/o depth removes depth supervision from the baseline. The w/o speed limit disables the speed limit trick in close-loop evaluation. The MLP changes the decoder from transformer to MLP.



(a) BEV Real (b) BEV Semantic (c) BEV Attention

Fig. 5: The visualization of self-attention in the BEV. (a) is the real BEV scenarios in the CARLA simulator. (b) illustrates the ground truth of the BEV semantic where gray rectangles are the static vehicles and the white rectangle is the target parking slot. The red square indicates the query position. The attention map to the red square is shown in (c). The attention focuses on the target slot during the parking process, which enhances the interpretability of the network.

Quadro RTX 5000 graphics card, completing a total of 384 parking tasks. The network AIT (Average Inference Time) is 74.92 milliseconds for one step, which enables the network to work at around 13 Hz. The design goal of the network is to generate control signals at 10 Hz. Therefore, the runtime performance of our network satisfies the requirement, which is crucial for an autonomous parking system to ensure safe and efficient driving in practical applications.

E. Ablation Studies

Auxiliary Task. In Table III, we compared the effects of the depth auxiliary task. Depth supervision can improve BEV performance. Therefore, the TSR decreased by about 14 points after removing depth supervision. The experimental results prove the effectiveness of depth supervision.

Speed Limit. In Table III, we discussed the necessity of the speed limit trick. Without the speed limit, the TSR decreased by about 10 points. The fast driving speed reduced the APT but produced more failures (TFR, NTR, CR and TR). With the speed limit trick, the system was more stable.

Decoder Architecture. We also compared the performance of different kinds of decoders. As shown in Table III, the MLP decoder gained about 83 points in TSR. The baseline, transformer decoder, raised about 8 points in TSR. The performance difference mainly benefited from the cross-attention mechanism of the transformer, enabling our approach to effectively use the BEV features.

Control Steps. The decoder control steps in the training may affect the closed-loop performance. The baseline used 4 control steps. We tested with different numbers of control steps: 1 and 6. As shown in Table IV, 4-steps and 6-steps had similar performance in TSR. However, when the step was set to 1, the TSR dropped about 8 points. The result showed that the baseline achieved a better balance between performance and efficiency.

V. Conclusion

In this paper, we presented our work on adopting an end-to-end format for autonomous parking, utilizing surround images and motion measurements as inputs and directly generating control signals. Our method is able to track the parking goal during the inference. We leveraged camera-view to BEV projection and the attention mechanism in Transform to make full use of the visual information. Our method has proved its feasibility and effectiveness in the closed-loop experiments conducted on CARLA.

At the current stage, this work is limited in simulation. The overall ability of our method still has a clear gap to that of a real human driver in generalization. Besides, the capability to handle dynamic objects has not been taken into consideration yet. In the future, we will conduct real-world experiments in more diverse scenarios such as horizontal and diagonal parking with the exploitation of extra on-board sensors like fish-eye camera and ultrasonic radar. We also plan to explore the possibilities of implementing reinforcement learning on parking tasks to better avoid collisions with dynamic objects and solve corner cases. We believe that this paper will serve as a valuable resource for researchers and engineers working on autonomous driving systems, paving the way for further advancements in the field of self-parking technology.

References

[1] T. Qin, T. Chen, Y. Chen, and Q. Su, "Avp-slam: Semantic visual mapping and localization for autonomous vehicles in the parking lot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5939–5945.

TaskIdx	TSR (%) ↑	TFR (%) ↓	NTR (%) ↓	CR (%) ↓	TR (%) ↓	APE (m) ↓	AOE (deg) ↓	APT (s) ↓	AIT (ms) ↓
baseline (4-steps)	**91.41**	2.08	**3.39**	2.08	**1.04**	0.30	**0.87**	15.72	74.92
1-step control	83.07	1.04	6.77	2.86	6.25	**0.27**	0.94	17.68	74.99
6-steps control	90.63	**0.00**	6.51	**1.04**	1.82	**0.27**	1.06	**15.02**	**74.41**

TABLE IV: **Ablation studies on decoder control steps used in the training.** In the inference stage, we only infer one step at 10 hz for closed-loop control. The baseline chooses 4-steps strategy, which achieves a balance between performance and efficiency.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale. arxiv 2020," *arXiv preprint arXiv:2010.11929*, 2010.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[9] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7077–7087.

[10] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," in *Conference on Robot Learning*. PMLR, 2023, pp. 726–737.

[11] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[12] Y. Xiao, F. Codevilla, D. P. Bustamante, and A. M. Lopez, "Scaling self-supervised end-to-end driving with multi-view attention learning," *arXiv preprint arXiv:2302.03198*, 2023.

[13] X. Shen, M. Lacayo, N. Guggilla, and F. Borrelli, "Parkpredict+: Multimodal intent and motion prediction for vehicles in parking lots with cnn and transformer," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 3999–4004.

[14] E. Xie, Z. Yu, D. Zhou, J. Philion, A. Anandkumar, S. Fidler, P. Luo, and J. M. Alvarez, "M2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation," *arXiv preprint arXiv:2204.05088*, 2022.

[15] B. Zhou and P. Krähenbühl, "Cross-view transformers for real-time map-view semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 13 760–13 769.

[16] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view," *arXiv preprint arXiv:2112.11790*, 2021.

[17] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *European conference on computer vision*. Springer, 2022, pp. 1–18.

[18] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer, 2020, pp. 194–210.

[19] Y. Li, Z. Ge, G. Yu, J. Yang, Z. Wang, Y. Shi, J. Sun, and Z. Li, "Bevdepth: Acquisition of reliable depth for multi-view 3d object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 1477–1485.

[20] B. Jaeger, K. Chitta, and A. Geiger, "Hidden biases of end-to-end driving models," *arXiv preprint arXiv:2306.07957*, 2023.

[21] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li, "Think twice before driving: Towards scalable decoders for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 983–21 994.

[22] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao, "St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning," in *European Conference on Computer Vision*. Springer, 2022, pp. 533–549.

[23] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[24] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[25] J. Hawke, V. Badrinarayanan, A. Kendall *et al.*, "Reimagining an autonomous vehicle," *arXiv preprint arXiv:2108.05805*, 2021.

[26] L. Chen, T. Tang, Z. Cai, Y. Li, P. Wu, H. Li, J. Shi, J. Yan, and Y. Qiao, "Level 2 autonomous driving on a single device: Diving into the devils of openpilot," *arXiv preprint arXiv:2206.08176*, 2022.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[28] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4693–4700.

[29] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9329–9338.

[30] E. Ohn-Bar, A. Prakash, A. Behl, K. Chitta, and A. Geiger, "Learning situational driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 296–11 305.

[31] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6119–6132, 2022.

[32] X. Shen, E. L. Zhu, Y. R. Stürz, and F. Borrelli, "Collision avoidance in tightly-constrained environments without coordination: a hierarchical control approach," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2674–2680.

[33] J. Leu, Y. Wang, M. Tomizuka, and S. Di Cairano, "Autonomous vehicle parking in dynamic environments: An integrated system with prediction and motion planning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 890–10 897.

[34] S. Rathour, V. John, M. Nithilan, and S. Mita, "Vision and dead reckoning-based end-to-end parking for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 2182–2187.

[35] R. Li, W. Wang, Y. Chen, S. Srinivasan, and V. N. Krovi, "An end-to-end fully automatic bay parking approach for autonomous vehicles," in *Dynamic Systems and Control Conference*, vol. 51906. American Society of Mechanical Engineers, 2018, p. V002T15A004.

[36] X. Shen, I. Batkovic, V. Govindarajan, P. Falcone, T. Darrell, and F. Borrelli, "Parkpredict: Motion and intent prediction of vehicles in parking lots," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1170–1175.

[37] T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. Hinton, "Pix2seq: A language modeling framework for object detection," *arXiv preprint arXiv:2109.10852*, 2021.