End-to-End Path Planning using Stitched Surround View Images and 2D Object Bounding Boxes

Liguo Zhou, Zhuting Xue, Huaming Liu, Alois Knoll

***Abstract*—Autonomous driving technologies promise to revolutionize the way we navigate, offering safer and more efficient transportation solutions. Particularly, end-to-end autonomous driving systems, which process inputs directly into waypoints or steering commands, represent a significant advancement in this field. This article presents a novel approach to end-to-end autonomous driving that leverages vision and Transformer-based technologies to navigate. Our method incorporates dual Transformer encoders -— one for images and another for past states -— enabling nuanced interpretations of spatial and temporal elements. The integration of a Vision Transformer as the image encoder enhances visual processing capabilities, critical for autonomous navigation. We further advanced our model by integrating object detection model for superior object detection, enriching the ability of the model for improved situational awareness and collision avoidance. Innovations such as gated cross attention in the decoder refine attention mechanisms, enhancing focus on relevant scenarios during driving. Validated on the large-scale nuScenes dataset, our model demonstrates substantial improvements over existing methods in L2 loss and collision rates, affirming the effectiveness of our design in real-world conditions.**

***Index Terms*—End-to-End Autonomous Driving, Transformer, Object Detection, Imitation Learning, Path Planning**

I. Introduction

WHETHER the driver is a human or an autonomous system, the fundamental scenarios encountered while driving on the road can be distilled into two primary situations. First, the vehicle operates in an environment free of obstacles, where it follows the designated path, adhering strictly to traffic rules and road regulations. In this scenario, the vehicle's primary focus is on maintaining its course and speed while ensuring compliance with all applicable laws and signage. Second, the vehicle encounters obstacles on the road. When this occurs, the vehicle must assess the situation and respond accordingly. This may involve braking to reduce speed or come to a complete stop, changing lanes to navigate around the obstacle, or, in some cases, taking no immediate action if the obstacle does not pose a significant threat. The response is determined by the nature of the obstacle, the vehicle's current speed, road conditions, and the proximity of other vehicles.

These two scenarios encapsulate the essence of decision-making in autonomous driving. Autonomous driving, or self-driving or driverless technology, refers to vehicles operating and navigating without human intervention. These vehicles use a combination of sensors, cameras, radar, lidar, GPS, and Artificial Intelligence (AI) algorithms to perceive their environment, interpret data, and make decisions to navigate safely from one point to another.

Common practices in system architecture have evolved considerably over time. Most Automated Driving Systems (ADSs) delineate the intricate task of automated driving into sub-components, integrating an assortment of sensors and algorithms across diverse modules. The modular pipeline undergoes hierarchical decomposition into three components: perception and prediction, path planning and motion controllers. As mentioned above, the decision-making part, or path planning, involves delineating a geometric path from a vehicle's current location to a designated destination while circumventing obstacles.

Notably, end-to-end driving methodologies have recently emerged as an alternative to modular approaches. This approach processes inputs directly from sensors to outputs that control the vehicle's actions or trajectory, all through a single algorithm, typically a deep neural network. The driving model is developed either by using supervised learning, specifically, imitation learning to replicate the behaviour of human drivers [1], or by autonomously generating and refining driving policies from the ground up through reinforcement learning.

Significant advancements have been made by applying deep neural networks. The approach known as policy distillation, initially introduced in LBC [2] and further explored in subsequent studies [3]–[6], has notably enhanced the closed-loop performance by replicating the behavior of a proficient expert. To improve how well these models generalize, owing to differences between the expert's and the learned policies, various researchers [7], [8] have suggested the incorporation of on-policy data [9] during the training phase.

It is important to understand that the end-to-end technological model does not strictly refer to a singular, opaque system producing only final planning or control outputs. It might also be segmented with intermediate stages and results, similar to traditional methods. Indeed, various state-of-the-art systems [10], [11] have adopted this modular architecture while simultaneously optimizing all individual components collectively, thus enhancing overall effectiveness.

While promising, state-of-the-art end-to-end autonomous driving systems face several significant challenges and problems that need to be addressed to ensure their widespread adoption and operational reliability. For example, The technology required for autonomous driving, especially advanced sensors like lidar, is expensive, which currently limits the scalability and mass adoption of fully autonomous vehicles. Furthermore, the current system exhibits significant limitations in its ability to effectively perform obstacle avoidance. This is

Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University of Munich

manifested by the poor performance of vision-based end-to-end autonomous driving system in achieving simultaneously low L2 error and collision rates. Recognizing this shortcoming, we developed a novel approach to address these issues. We propose a vision-based, end-to-end autonomous driving system that integrates advanced object detection capabilities. In summary, the contributions of this article will be as follows:

1) Our method incorporates dual Transformer encoders —- one for images and another for past states. This enables nuanced interpretations of spatial and temporal elements.
2) The integration of a Vision Transformer as the image encoder enhances visual processing capabilities. We further advanced our model by integrating object detection model for superior obstacles avoidance, enriching the visual feature set for improved situational awareness and collision avoidance.
3) We use gated cross attention in the decoder to enhance the focus on relevant scenarios during driving.

II. Related Work

Input Modality: While initial studies [12], [13] were able to accomplish basic autonomous driving tasks such as maintaining a vehicle within its lane using only a single monocular camera, this approach alone proved inadequate for more complicated situations [14]. As a result, various sensors have now been deployed on modern autonomous vehicles to enhance capability and safety. Multi-sensor fusion is typically divided into three groups: early, mid, and late fusion. Early fusion means merging sensory data before feeding it into the feature extractor. A typical method of integrating different inputs, like images with depth information [15], or bird's eye view (BEV) point clouds with High-Definition (HD) maps [16], [17], involves concatenation. This combined data is subsequently processed using a common feature extractor. Late fusion (also known as decision-level fusion) is a technique where data from different sensors are processed separately in individual pipelines, and the features or decisions extracted from each sensor are combined only at the final stage of the process. This topic is often overlooked because it performs poorly [7], [18]. Middle fusion, or feature-level or intermediate-level fusion, combines features or data extracted from different sensor inputs before feeding them into the decision-making algorithms. Simple concatenation is often used to merge features from various modalities [19], [20]. More recent studies have utilized Transformers [21] to explore the interactions among pairs of features. Transfuser [14] integrates image and LiDAR data using a Transformer-based approach, leveraging the self-attention mechanism to process and fuse data from these two sensor modalities. In summary, integrating different sensor modalities often improves the field of view and accuracy of perception. However, effectively merging these to pull essential data for complete autonomous vehicle control still needs more research.

Imitation Learning: Imitation Learning (IL) is a machine learning method where models learn to perform tasks by observing demonstrations of those tasks from human experts or sophisticated software agents. In practice, this involves collecting data from expert performances, which typically includes sequences of states, actions, and rewards, and then using this data to train a model. There are two types of imitation learning: Behavioral Cloning (BC) [22] and Inverse Reinforcement Learning (IRL) [23]. Initial uses of BC for driving tasks, as referenced in studies [12], [13], involved using an end-to-end neural network to create control signals directly from visual inputs captured by cameras. Subsequent improvements have included the integration of data from multiple sensors [24], [25], the incorporation of auxiliary tasks [14], [26], and enhancements in expert system design [3], [4]. These advancements have been developed to improve the capacity of BC-based end-to-end driving systems to effectively navigate complex urban environments. Generative Adversarial Imitation Learning (GAIL) [27], [28] elegantly circumvents the need for a manually designed reward function using the adversarial framework, where the discriminator continuously refines the reward function based on the perceived similarity between the agent's and the expert's actions. Various studies have suggested optimizing a cost volume or function alongside auxiliary perceptual tasks.

Reinforcement Learning: Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by performing actions in an environment and receiving feedback as rewards. Reinforcement learning, when integrated with supervised learning, has proven effective in the realm of autonomous driving. Implicit affordances [29] and GRI [30] share a methodology that combines model-free reinforcement learning with direct visual input processing in an end-to-end manner. This integrated approach allows autonomous systems to adaptively learn from both the structured imitation of human drivers and the unstructured environments encountered in real-world urban settings. Reinforcement learning has also effectively been applied to refine entire architectures on CARLA that were initially pre-trained through imitation learning [31], [32]. In the field of modular autonomous driving, RL has also proven successful in planning or controlling scenarios where the network utilizes confidential simulator data [33]–[35].

Evaluation Methods: In autonomous driving systems, evaluation methods are crucial for assessing the safety and effectiveness of driving algorithms under various scenarios [36], [37]. These methods are generally categorized into online and offline evaluations. Online evaluation involves testing the autonomous driving algorithms directly in the environment where they will be deployed, whether in real-world scenarios or highly realistic simulators. The key is that the testing occurs in a dynamic environment where the model interacts in real-time, and decisions are made on the fly. However, testing autonomous driving technologies in actual environments can be expensive and hazardous. As a result, using simulations represents a practical alternative [38]–[40]. On the other hand, offline evaluation uses previously collected data to assess the performance of driving algorithms. This approach doesn't require real-time interaction with the environment; instead, it relies on a static dataset that includes various driving scenarios. This methodology necessitates a thorough collection of trajectory data. Prominent datasets employed for this purpose

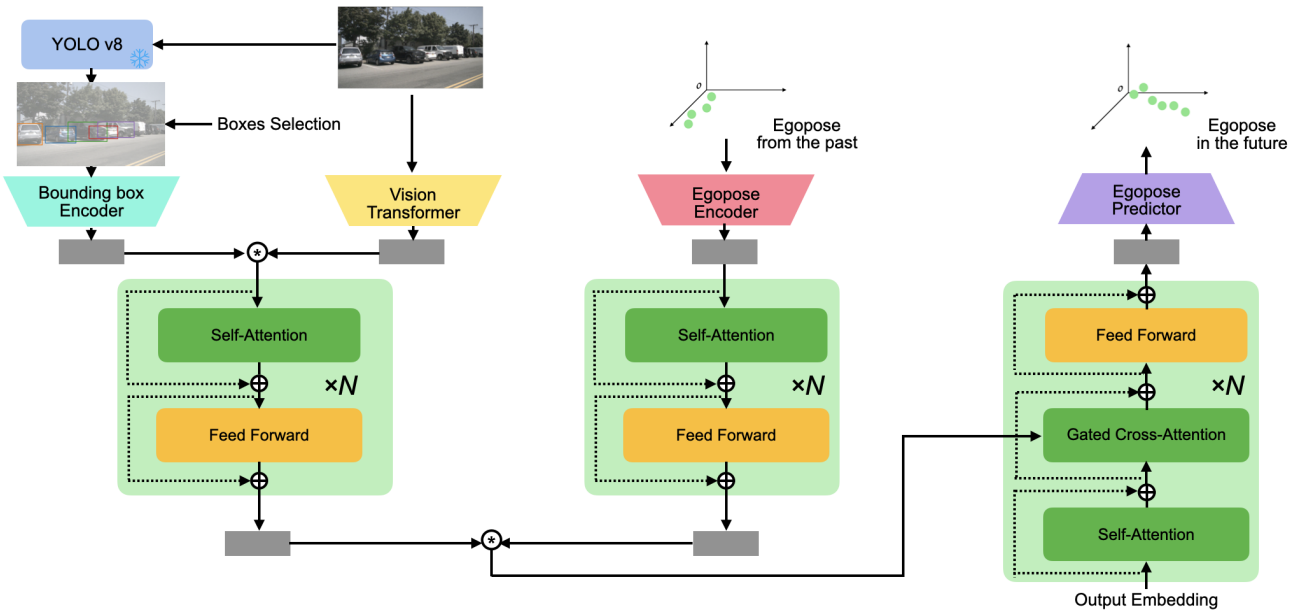

Fig. 1: Framework of proposed method

include nuScenes [41], Argoverse [42], [43], Waymo [44], and nuPlan [45]. These datasets detailed, annotated trajectory and sensor data captured from real-world driving scenarios. However, results from open-loop testing do not offer definitive proof of enhanced driving behavior in closed-loop settings.

III. Methodology

We propose an end-to-end learnable motion planner designed to generate precise space-time trajectories over a planning horizon of several seconds. Our model processes inputs from a front-view camera and state variables to produce the final states for the specified planning horizon. Crucially, we integrate object detection model, which enhances the model's ability to comprehend and interpret the visual environment. In this section, the architecture of proposed method in Fig. 1 for end-to-end autonomous driving is introduced, which consists of two primary components:

1) The integration of an image encoder and a states encoder for extracting information from front-camera and IMU & GPS.
2) A states prediction decoder with gated cross attention.

A. Problem Setting

We consider the task of end-to-end autonomous driving where the ultimate goal is developing vehicles that can navigate and operate safely and efficiently without human intervention. In order to solve this task, we use Imitation Learning (IL). In our setup, the policy π is a mapping from vision inputs to states (including position and orientation). The expert policy $\hat{\pi}$ utilized is sourced from the nuScenes dataset. This dataset is defined as $D = \{(X_i, S_i)\}_{i=1}^{Z}$, where X_i represents a sequence of five frames comprising the current frame and the four preceding frames captured by the front camera. Correspondingly, S_i denotes the states associated with these frames, specifically including the position and orientation of the current frame, the previous four frames, and the subsequent six frames. These states are formatted as $S = \{(x_t, y_t, yaw_t)\}_{t=1}^{T}$ and are converted into the ego-vehicle's coordinate system.

The policy π, which governs decision-making in this context, is trained using a supervised learning approach with the dataset D. The objective of the training is to minimize the loss function L, which quantifies the discrepancy between the policy's output and the ground-truth data. The formulation of the training process can be mathematically represented by the following optimization problem:

$$\pi = \arg\min_{\hat{\pi}} L(\pi(X_i, S_i^{Past}), S_i^{Future}) \tag{1}$$

where $\pi(X_i, S_i^{Past})$ predicts the optimal states based on the input frames, and L measures the L1 loss. S_i^{Future} are the states of future six frames.

B. Input and Output Representation

Our input consists of two key components: the image data and the states (including position and orientation). We only use images from front camera. For each current frame, along with the past four frames, we perform resizing, cropping and normalization on the image data. For the state data, which includes both the position and yaw of the vehicle, we take different approaches depending on the time frame. For the current frame, we convert the states from the global coordinate system to the sensor coordinate system, resulting in the $Ego_{current}$. For the states in the past four frames, as well as for the next six frames, we first convert these values from the sensor coordinate system back to the global coordinate system. Then, we apply a transformation using the $Ego_{current}$

to convert all these values into the current coordinate system. It is important to highlight that the state of the current frame is represented as $[0, 0, yaw_{current}]$. This ensures that all position and orientation data are aligned consistently in the current coordinate frame, facilitating accurate processing and analysis.

In terms of output representation, we predict the future states of the ego-vehicle, with the states being centered in the current coordinate frame of the ego-vehicle. These predicted states are represented as a sequence of 3D coordinates, denoted as $\{s_t = (x_t, y_t, yaw_t)\}_{t=1}^{T}$, where each state s_t corresponds to the vehicle's position and yaw at a future time step t. In this context, T represents a prediction horizon of 3 seconds, allowing the model to forecast the ego-vehicle's 6 states over a short-term future.

C. Bounding-Boxes for Better Image Representation

To enhance the model's ability to avoid obstacles, we employed object detection model on the images. The pre-training process of object detection model utilize the nuImages dataset [46], which includes a wide range of images and corresponding bounding boxes (as ground truth). The choice of the nuImages dataset for pre-training was strategic, as it closely resembles the nuScenes dataset, making it highly relevant for our application. The nuImages dataset offers a diverse set of object categories for bounding boxes, such as "$animal$", "$human.pedestrian.adult$" and "$vehicle.car$". However, for the purpose of our path planning task, such detailed categorization is not particularly beneficial. To address this, we consolidated all the categories into a single class: "has_object". This simplification allowed the model to focus on the presence of objects in the scene, which is more relevant to our objective.

For the evaluation of object detection performance, we used the Mean Average Precision at 50% Intersection over Union (mAP50) as our metric. After pre-training, we observed a significant improvement in mAP50, increasing from 0.687 to 0.82. This improvement indicates that the model became more adept at detecting objects on nuImages dataset, thereby enhancing its overall understanding of the environment on the nuScenes dataset.

In our proposed method, we utilize this pre-trained object detection model, which is kept frozen during the process to ensure consistent feature extraction. It's important to highlight that multiple objects may be detected within a single image. Through our experiments, we have observed that retaining a large number of detected objects does not necessarily improve the model's performance. Therefore, we strategically choose to retain only the closest five detected objects per image for further processing. The box selection process involves calculating the distance from each detected object's bounding box to the ego vehicle, as shown in Algorithm 1. The object detection model we used defines the origin of the coordinate system at the upper-left corner of the image. In this context, the midpoint of the lower edge of the image, represented by the coordinates $Ego = (ego_x, ego_y) = (W/2, H)$, is considered the closest point to the ego vehicle. x_1, y_1, x_2, y_2 are $x_{min}, y_{min}, x_{max}, y_{max}$ of a bounding box. We calculate the distance from each detected bounding box to this reference point by considering the following three scenarios:

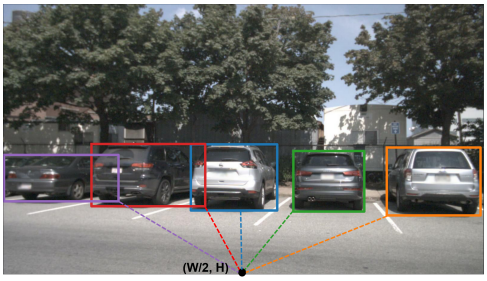

Fig. 2: Selection of 5 closest boxes

1) If the bounding box is positioned to the left of the reference point Ego, as shown by the two boxes on the left in Fig. 2. we calculate the distance between the lower-right vertex x_2, y_2 and the reference point
2) If the bounding box is located above the reference point Ego, as shown by the blue box in Fig. 2. The distance is computed as the absolute value $|ego_y - y_2|$
3) If the bounding box is positioned to the right of the reference point Ego, as shown by the two boxes on the right in Fig. 2. we calculate the distance between the lower-left vertex x_1, y_2 and the reference point

Algorithm 1 DistanceCalculation(H, W, box)

$ego_x, ego_y \leftarrow W/2, H$
$x_1, y_1, x_2, y_2 \leftarrow$ box
if $x_2 \leq ego_x$ **then**
  $distance \leftarrow \sqrt{(x_2 - ego_x)^2 + (y_2 - ego_y)^2}$
else if $x_1 \geq ego_x$ **then**
  $distance \leftarrow \sqrt{(x_1 - ego_x)^2 + (y_2 - ego_y)^2}$
else
  $distance \leftarrow |ego_y - y_2|$
end if
return $distance$

After calculating the distances of all detected boxes from the specified reference point, we proceed with a filtering process as outlined in Algorithm 2. For each image, the computed distances are stored in a list. This list is then sorted in ascending order, starting with the shortest distance. From this sorted list, we select only the top 5 distances, which correspond to the 5 boxes closest to the reference point. If the total number of detected boxes is fewer than 5, we fill the remaining positions with zeros to ensure a consistent output size. This approach allows us to focus on the most relevant obstacles while maintaining a uniform structure for subsequent processing steps.

Algorithm 2 BoxesSelection(H, W, boxes, n_object)

$distances \leftarrow []$
for box **in** $boxes$:
    $distances.append(DistanceCalculation(H, W, box))$
$indices \leftarrow argsort(distances)[: n\_object]$
$boxes \leftarrow boxes[indices]$
return $boxes$

To ensure the reliability of the pre-trained model, we employed visualization to examine the performance of our model on nuScenes dataset. The example result is shown in Fig. 3. More examples can be found in Appendix. The consistent performance of our pre-trained object detection model across the diverse scenarios presented in the nuScenes dataset reinforces its reliability. This is crucial for application in our project, where accurate and dependable object detection is essential. Through these visual assessments, we confirm that our pre-trained model maintains high detection accuracy, making it a trustworthy component in the whole task.

Fig. 3: Example result of our pre-trained model

D. Transformer Encoders with Object Detection

Our proposed method consists of two encoders, one is for images, the other is for states. As illustrated in Fig. 1, the input images undergo a dual processing pipeline, where each path plays a critical role in preparing the data for subsequent stages of the model. The first step involves feeding the images into Vision Transformer (ViT) for feature extraction. The resulting features from ViT, denoted as $I_{in} \in \mathbb{R}^{N \times D_f}$, where N corresponds to the sequence length, and D_f represents the dimension of the feature vector for each image. To incorporate positional information, I_{in} is then subjected to positional encoding.

Simultaneously, the normalized images are processed through the pre-trained object detection model for obstacle avoidance, as demonstrated in section III-C. This step identifies and localizes objects within the images, generating bounding box coordinates for the detected objects. The information extracted from this step, denoted as $B_{in}^{'} \in \mathbb{R}^{N \times N_{object} \times 4}$, includes the sequence length N, the number of objects detected per image N_{object} and the four coordinates that define each bounding box. Following object detection, a box selection process is employed. The selected box information is then encoded using a box encoder, which transforms the bounding box coordinates into a feature representation B_{in} of the same dimension as I_{in}, specifically $\mathbb{R}^{N \times D_f}$.

Once both I_{in} and B_{in} are prepared, they are combined through an element-wise multiplication to produce the input $X_{in} \in \mathbb{R}^{N \times D_f}$ for the image encoder. This operation ensures that the features derived from the overall image and those specific to detected objects are integrated, capturing both global and localized information. The resulting input X_{in} is then fed into the image encoder, which utilizes linear projections to compute keys, values, and queries, as shown in Eq. 2

$$Q = X_{in} * M_q, \quad K = X_{in} * M_k, \quad V = X_{in} * M_v \tag{2}$$

where $M_q \in \mathbb{R}^{D_f \times D_q}$, $M_k \in \mathbb{R}^{D_f \times D_k}$ and $M_v \in \mathbb{R}^{D_f \times D_v}$ are weight metrics. It uses matrix product between Q and K to compute the attention weights and then multiply the values for each queries.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) * V \tag{3}$$

Finally, the image encoder uses a non-linear transformation to calculate the output $X_{out} \in \mathbb{R}^{N \times D_f}$.The Transformer model leverages the attention mechanism extensively throughout its architecture, applying it multiple times across a series of attention layers. Each of these layers is composed of multiple parallel attention "heads", resulting in distinct sets of Q, K and V matrices. And d_k is the dimension for each head.

The states encoder employed in our architecture follows the traditional Transformer encoder structure. The input to this encoder, denoted as $S_{in} \in \mathbb{R}^{N \times 3}$, where N represents the length of the sequence, and 3 corresponds to the three state components: (x, y, yaw). Initially, S_{in} is transformed to a dimension of (N, D_f) through a process known as coordinate encoding. Then the data undergoes positional encoding, self-attention mechanism and feed-forward layer. The result of these transformations is the output $S_{out} \in \mathbb{R}^{N \times D_f}$.

After obtaining X_{out} and S_{out}, we perform an element-wise multiplication of these two tensors to produce the combined output, referred to as $En_{out} \in \mathbb{R}^{N \times D_f}$. We did some ablation studies on how to combine X_{out} and S_{out}, which are described in detail in subsection IV-E. This multiplication effectively merges the information, allowing the model to integrate features from front camera and states. The resulting En_{out} tensor serves as the input for the subsequent decoder, where it will be utilized in the gated cross-attention mechanism to compute keys and values.

E. Transformer Decoder with Gated Cross Attention

The decoder of our proposed method adheres to the traditional Transformer decoder structure, but two key points are particularly noteworthy:

1) As outlined in Algorithm 3, the training process incorporates a prediction loop. Initially, a zero vector of dimension (B, 1, 3) is initialized, where B represents the batch size, 1 is the length of sequence and 3 represents the 3 coordinates (x, y, yaw). This vector serves as the initial input to the Transformer decoder. After processing through the decoder, which includes a gated cross

attention mechanism, an output $De_{out} \in \mathbb{R}^{B\times 1\times D_f}$ is generated. This output is then passed through a fully connected layer, where the dimension D_f is reduced to 3, corresponding to the predicted state of the next frame. The predicted states are then concatenated with the start symbol to form the input $pred_{in}$ for the next step. This process continues, with the last state in the output $pred_{out}$ being appended to $pred_{in}$, until the states for all 6 frames are predicted.

Algorithm 3 Predict(encoder_out, predict_n, coord_enc, decoder, pred_fc, B)

$encoder\_out$: output of transformer encoder
$predict\_n$: number of frames for prediction
$coord\_enc$: coordinates encoder
$decoder$: transformer decoder
$pred\_fc$: fully connected layer for final output
B : batch size
$preds \leftarrow zeros(B, 1, 3)$
for $i = 1$ to $predict\_n$ **do**
  $preds\_i \leftarrow preds$
  $preds\_i \leftarrow coord\_enc(preds\_i)$
  $dec\_out \leftarrow decoder(preds\_i, encoder\_out)$
  $new\_preds \leftarrow pred\_fc(dec\_out)$
  $preds \leftarrow concatenate((preds, new\_preds[:, -1 :]), dim = 1)$
end for
return $preds[:, 1:]$

2) Inspired by [47], we use gated cross attention instead of normal cross attention in the decoder. The use of gated cross attention is a critical innovation in our model. Gated cross attention allows the model to selectively focus on relevant parts of the input while filtering out irrelevant information. This mechanism enhances the effectiveness of the cross-attention process by introducing a gating mechanism that controls the flow of information. Mathematically, the gating mechanism introduces an additional gating vector g:

$$Gated\_Att(Q, K, V, g) = g * (softmax(\frac{QK^T}{\sqrt{d_k}}) * V) \tag{4}$$

Here, g is a learnable gating vector, and $*$ denotes element-wise multiplication. The gated cross attention mechanism plays a crucial role in refining the predicted coordinates at each step, ensuring more accurate and context-aware predictions.

F. Training Objective

In our approach, the network is trained using an L1 loss function, which measures the absolute difference between the predicted states and the ground truth states provided by an expert. These states are registered to the current coordinate frame to ensure accurate alignment. Specifically, let $\hat{s}_t$ represent the ground truth states at time-step t, and s_t represent the predicted states at the same time-step. The loss function can then be defined as:

$$L = \sum_{t=1}^{T} |s_t - \hat{s}_t| \tag{5}$$

The use of L1 loss is advantageous in this context because it provides a robust measure of error, penalizing deviations from the expert's trajectory in a straightforward and interpretable manner.

IV. Experiments

A. nuScenes Dataset

The nuScenes dataset [41] is a public dataset designed to advance research in autonomous driving and computer vision. It features 1000 manually selected 20-seconds driving scenes from Boston and Singapore, chosen for their complex and diverse traffic situations. This dataset includes approximately 15 hours of driving data, annotated at 2Hz with accurate 3D bounding boxes for 23 object classes, along with attributes like visibility, activity, and pose. Besides, camera images from six directions, LiDAR and RADAR sweeps data are also available. The data collection spans various challenging scenarios, weather conditions, and environments, enhancing the dataset's utility for studying algorithm generalization across different contexts.

We consider 2.0 seconds of historical context to predict the subsequent 3.0 seconds in the nuScenes dataset, corresponding to four past frames and six future frames. For each input sequence, we transform all past and future states to the current coordinate system (ego pose), utilizing relative coordinates.

B. Evaluation Metrics

Following ST-P3 [48], we focus primarily on two crucial performance metrics to evaluate our model: L2 error and collision rate, reporting each metric over durations of 1s, 2s, and 3s. The L2 error metric quantifies the accuracy of our planned trajectories by calculating the Euclidean distance between each point on the planned trajectory and the corresponding point on the human-driven trajectory over the specified durations. Conversely, the collision rate is measured to assess the safety aspects of our model. This metric determines how frequently the ego vehicle, under autonomous control, would potentially collide with other road agents, such as other vehicles and pedestrians. Specifically, we calculate the overlap rate, which involves assessing the instances where the predicted trajectories of the ego vehicle intersects with the trajectories of nearby moving agents.

In our experiments, achieving lower values for both L2 error and collision rate is indicative of superior model performance, showcasing not only the accuracy of trajectory planning but also the model's robustness in maintaining safety across varied and dynamic traffic scenarios.

C. Implementation Details

We use only the front camera image as visual input, which is encoded by a Vision Transformer (ViT). The patch size and depth of the ViT are set to 8 and 2, respectively. To

enhance the visual representation, we further employ object detection model to predict the bounding boxes of obstacles. These bounding boxes are encoded and subsequently fused with image features from the ViT through element-wise multiplication. We limit the number of bounding boxes to 5 per image, selecting the closest ones, as detailed in section III-C. The enhanced visual features are then fed into a Transformer encoder to model temporal dependencies. Additionally, another Transformer encoder is used to encode the past states. The outputs from both Transformer encoders are fused by multiplication, and the fused representation is passed to a Transformer decoder. Both the Transformer encoders and decoder have a depth of 3 and a hidden dimension of 256. During training, we use a batch size of 8 and train each model for 20 epochs. We select SGD [49] as the optimizer, with a fixed learning rate of 5e-4.

D. Experimental Results

During this end-to-end self-driving project, we explored several sequence models, including LSTM [50], Transformer [21], and the newly proposed state-space model, Mamba [51]. Among these, we focused primarily on Transformers, upon which we developed a novel technique that enhances visual features using object detection results. Our methods are compared with state-of-the-art (SOTA) approaches in Tab. I.

Among the SOTA approaches, none simultaneously achieve a low L2 loss and a low collision rate. The NMP [16] presents a holistic model that integrates raw LiDAR data and High-Definition (HD) maps to create interpretable intermediate representations. These representations include 3D detection of objects and their predicted future trajectories. They also create cost volume, which evaluates the ”goodness” of potential positions for the self-driving vehicle within a given planning horizon. Freespace [52] introduces a novel concept of forecasting future free space as an alternative to the traditional object-centric representations used in autonomous driving. This approach leverages self-supervised learning, allowing the model to forecast without the need for costly manual annotations. The model inputs include raw sensor data from LiDAR scans, and its output are predictions of future free space. Freespace's effectiveness is demonstrated using two datasets: the nuScenes dataset for real-world evaluation and the CARLA simulator for controlled experimental conditions. The result on nuScenes is shown in Tab. I. ST-P3 [48] integrates spatial and temporal information from camera inputs to enhance feature representations for perception, prediction, and planning tasks. This technique preserves geometric information in 3D space before transforming it into a bird's eye view, aiding perception accuracy. ST-P3 is vision-based and as in Freespace, is evaluated on the open-loop nuScenes dataset and closed-loop CARLA simulator.

However, the three methods discussed above exhibit significant shortcomings in terms of L2 loss and collision rates, particularly over a 3-second predictive horizon. Specifically, the L2 loss exceeds 2.9 meters, which is close to the typical width of a lane on an urban road. This substantial error suggests a significant deviation between the predicted trajectory and the actual path taken by a human driver, indicating that the model's predictions may not align well with real-world driving behaviors. Furthermore, the collision rate surpassing 1% within 3 seconds. Such a rate implies a non-trivial likelihood of accidents, which is unacceptable for any autonomous driving system aiming to ensure safety and reliability. Our method addresses this gap. The initial model we investigated combines ResNet-50 [53] and LSTM. In this model, the ResNet-50 extracts image features, which are subsequently processed by the LSTM to model temporal dependencies. This approach yields a significant improvement in L2 loss over existing SOTA techniques. However, it underperforms in terms of collision rate when compared to the ST-P3 model.

Although LSTMs address the issue of modeling long-term dependencies more effectively than traditional recurrent neural networks, they still fall short in this regard. Consequently, we transitioned to using Transformer, a sequence model renowned for its exceptional capability to model long-term relationships through its self-attention mechanism. Additionally, we replaced the ResNet-50 image encoder with a ViT to enhance image representation, as detailed in section III. We also conducted ablation studies to evaluate various image encoders, discussed in section IV-E. As demonstrated in Tab. I, our proposed Transformer-based method significantly outperforms existing state-of-the-art methods in both L2 loss and collision rate, highlighting the efficacy of our model design.

As our objective is to solely utilize visual information, another intuitive approach is to enhance the visual representations by explicitly incorporating object information. To achieve this, we employ object detection model to integrate obstacles' features into the image features. The results presented in Tab. I indicate that the inclusion of object detection model significantly improves the collision rate while maintaining an acceptable L2 loss. This improvement suggests that the model effectively accounts for entities such as pedestrians and vehicles in its route planning, thereby substantially reducing the collision rate.

Additionally, we investigated the recently proposed state-space model, Mamba, known for its efficiency in processing long sequence inputs and its low computational demands. Due to time constraints, our exploration of Mamba was not as extensive as Transformers. Therefore, while Mamba did not demonstrate results as competitive as those of the Transformers, we believe there is potential for further enhancement. We plan to address this in future research, for instance, also applying object detection model to Mamba.

E. Ablation Study

In this section, we present ablation studies on how to fuse image and state feature, type of image encoder, number of selected objects, bounding box selection metric, effectiveness of object detection model and gated cross attention.

How should we fuse image and state features? In our experiments, we explored three different methods for fusing X_{out} and S_{out}: element-wise summation, element-wise multiplication, and concatenation along the sequence length dimension. The results are shown in Tab. II. When

TABLE I: Experimental Results

Method	L2 Loss (m) ↓				collision rate (%) ↓			
	1s	2s	3s	Average	1s	2s	3s	Average
NMP [16]	0.61	1.44	3.18	1.74	0.66	0.90	2.34	1.30
Freespace [52]	0.56	1.27	3.08	1.64	0.65	0.86	1.64	1.05
ST-P3 [48]	1.33	2.11	2.90	2.11	0.23	0.62	1.27	0.71
ResNet-50+LSTM	0.48	0.94	1.51	0.98	0.34	0.66	1.44	0.81
Transformer	0.31	0.62	1.05	**0.66**	0.30	0.57	1.06	0.64
Transformer + YOLO	0.33	0.65	1.10	0.69	0.13	0.30	0.81	**0.42**
Mamba	0.39	0.76	1.23	0.79	0.43	0.65	1.43	0.84

TABLE II: Ablation Study on How to Fuse Image and Coordinate Feature

How to fuse	L2 Loss (m) ↓				collision rate (%) ↓			
	1s	2s	3s	Average	1s	2s	3s	Average
Addition	0.59	0.91	1.41	0.97	1.46	1.32	1.94	1.57
Multiplication	0.36	0.70	1.15	**0.74**	0.32	0.56	1.07	**0.65**
Concatenation	0.39	0.78	1.32	0.83	0.28	0.47	1.25	0.67

TABLE III: Ablation Study on Image Encoder

Image encoder	L2 Loss (m) ↓				collision rate (%) ↓			
	1s	2s	3s	Average	1s	2s	3s	Average
ResNet-50	0.36	0.70	1.15	0.74	0.32	0.56	1.07	0.65
VIT	0.31	0.62	1.05	**0.66**	0.30	0.57	1.06	**0.64**

element-wise multiplication method is employed, we observe a significant reduction in average L2 error. Furthermore, this method achieved the best average collision rate, demonstrating superior performance in terms of safety and reliability in autonomous driving scenarios. Based on these observations, we conclude that element-wise multiplication is the most effective method for feature fusion in our architecture.

Which image encoder is most effective for image feature extraction? We maintain the image fusion technique using element-wise multiplication while evaluating the effectiveness of two different image encoders: ResNet-50 and ViT. The results are shown in Tab. III. Despite the average collision rate remaining consistent between the two encoders, we observe a notable improvement in the average L2 error when using ViT, with a reduction of 11%. This significant decrease in average L2 error indicates that ViT is more effective than ResNet-50 in extracting relevant features from the vision input, ultimately leading to more accurate trajectory predictions.

How many objects should we select? In object detection tasks, a single image may contain multiple objects, and determining the optimal number of objects to retain for further processing is an important hyperparameter that can significantly impact performance. To explore this, we experimented with retaining 5, 10, or 15 objects per image. The results are shown in Tab. IV. Our results reveal that retaining 5 objects per image yields the best outcomes, achieving the lowest average L2 error and the lowest average collision rate simultaneously. Based on these findings, we chose to retain 5 objects per image in subsequent experiments.

Which objects selection metric should we utilize? In the object selection experiments presented in Tab. IV, we utilize metric that filter objects based on bounding box confidence scores. The confidence score represents the model's certainty that a detected object belongs to a particular class. By default, the pre-trained model outputs bounding boxes ordered according to their confidence scores, ranging from highest to lowest. In our approach, we selected the top n objects, specifically those with the highest confidence scores. Additionally, we explored an alternative selection metric based on the distance between the detected objects and the ego vehicle. A detailed explanation of this distance-based filtering method is provided in section III-C. The results are shown in Tab. V.

Our findings indicate that using distance as a selection metric significantly reduces the collision rates at prediction horizons of 1, 2, and 3 seconds, compared to the confidence-based method. Notably, this improvement in safety metrics is achieved without compromising the accuracy of the trajectory prediction, as evidenced by the average L2 error, which remains consistent between the two methods.

Is object detection model really helpful? Which camera should we use? To evaluate the effectiveness of the object detection model in enhancing accurate path planning and to determine the most suitable type of vision input, we conduct a series of experiments as detailed in Tab. VI. 1 image means that the network only uses images from front-view camera, while 6 images represents the input is images from surround-view cameras.

How do we use six images as input for our model? Our purpose of using six images is to get a global view similar to BEV. Therefore, it was a natural idea to take the back left, back and back right camera images and rotate them 180 degrees around the center of each image. So for each time step,

TABLE IV: Ablation Study on Number of Detected Objects

Objects number	**L2 Loss (m) ↓**				**collision rate (%) ↓**			
	1s	2s	3s	Average	1s	2s	3s	Average
5	0.33	0.66	1.11	**0.70**	0.43	0.57	1.14	**0.71**
10	0.44	0.85	1.38	0.89	0.39	0.63	1.18	0.73
15	0.42	0.79	1.27	0.83	0.43	0.65	1.27	0.78

TABLE V: Ablation Study on Objects Selection Metric

Box selection	**L2 Loss (m) ↓**				**collision rate (%) ↓**			
	1s	2s	3s	Average	1s	2s	3s	Average
Confidence	0.33	0.66	1.11	0.70	0.43	0.57	1.14	0.71
Distance	0.33	0.65	1.09	**0.69**	0.12	0.39	0.82	**0.45**

corresponding to each frame captured by the surround-view cameras, six images are processed. The three images obtained from the rear-facing cameras are first rotated 180 degrees around their center. Following this, we stitch these six images together, forming a unified global image with dimensions of 448×1440 pixels. This global image is then input into the ViT for feature extraction, producing an integrated feature map.

Concurrently, to enhance obstacles avoidance ability, each of the six individual images per frame is fed into the pre-trained and frozen object detection model. We deliberately avoid performing object detection directly on the stitched global image due to potential duplication: objects appearing near image junction might be detected twice, leading to redundancy and potential errors in object detection. At this point, the example result of the object detection model reference is shown in Fig. 4. This figure shows the result in a global manner, but in reality, each image is detected separately. Further, the bounding boxes detected in the images from the rear-facing cameras are also rotated by 180 degrees to align with the global perspective. The coordinates of these bounding boxes are transformed into the unified global coordinate system. These adjusted bounding boxes are then processed through a box encoder to obtain $B_{in} \in \mathbb{R}^{N \times N_{objects} \times 4}$. Here, $N_{objects}$ equals to image numbers times box numbers which is allowed to retain in one image. Subsequent processing is the same as in the case of 1 image.

Fig. 4: Objects detection of stitched 6 images before rotation

For illustrative purposes, Fig. 5 showcases the same example of objects detection after rotation.

The results in Tab. VI indicate that utilizing a single image input in conjunction with object detection model yields the best experimental outcomes. This superior performance can likely be attributed to object detection model's ability to enhance the representation of objects within the image. By providing more detailed and precise information about the

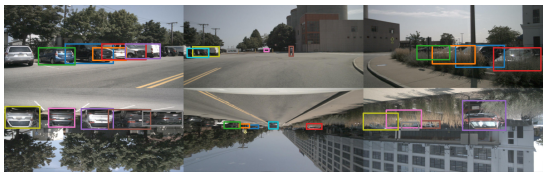

Fig. 5: Objects detection of stitched 6 images after rotation

objects in the environment, object detection model enables the whole system to gain a better understanding of the scene. This improves scene comprehension, in turn, contributes to a significant reduction in collision rate, thereby supporting more effective and safer path planning. However, using six images introduces a significant amount of redundant information, which contributes little to improving the model's performance.

Is gated cross attention useful? Building on insights gained from previous experiments, we introduce a slight modification to the decoder component of the model by replacing the standard cross-attention mechanism with a gated cross-attention mechanism. The results are shown in Tab. VII. Our results indicate that this modification led to a slight reduction in the average collision rate, without compromising the model's accuracy, as evidenced by the unchanged L2 error. This outcome highlights the potential benefits of integrating gated mechanisms in the attention layers of the model for more robust and reliable autonomous driving performance.

V. Conclusion

In this comprehensive project on end-to-end autonomous driving, our primary objective is to develop a robust model that relies solely on visual inputs for path planning. To achieve this, we investigated various sequence models that could effectively handle the complexities of dynamic driving environments. Among the models explored —- LSTM, Transformer, and Mamba —- we placed a particular emphasis on the Transformer model due to its proven efficacy in capturing temporal dependencies within sequential data.

Our innovative approach involves the use of separate Transformer encoders, one designated for processing images and another for past states. This dual-encoder system allows for a nuanced interpretation of both spatial and temporal aspects

TABLE VI: Ablation Study on The Effectiveness of Object Detection Model and Number of Camera

Method		L2 Loss (m) ↓				collision rate (%) ↓			
use YoLo	Num Camera	1s	2s	3s	Average	1s	2s	3s	Average
×	1	0.31	0.62	1.05	**0.66**	0.30	0.57	1.06	0.64
✓	1	0.33	0.65	1.09	0.69	0.12	0.39	0.82	**0.45**
×	6	0.31	0.62	1.06	0.67	0.30	0.39	0.81	0.50
✓	6	0.35	0.68	1.14	0.72	0.20	0.40	0.99	0.53

TABLE VII: Ablation Study on The Effectiveness of Gated Cross Attention

Gated cross attention	L2 Loss (m) ↓				collision rate (%) ↓			
	1s	2s	3s	Average	1s	2s	3s	Average
×	0.33	0.65	1.09	**0.69**	0.12	0.39	0.82	0.45
✓	0.33	0.65	1.10	**0.69**	0.13	0.30	0.81	**0.42**

of driving scenes. To enhance the model's capability in visual representation, we proposed the incorporation of ViT as the image encoder. The ViT is particularly adept at processing complex visual inputs, making it an ideal choice for the intricate demands of autonomous navigation.

Further refining our model, we introduced a novel pipeline that integrates advanced object detection techniques. This integration aims to enrich the visual feature set by accurately identifying and classifying objects within the vehicle's vicinity. By incorporating bounding box information from detected objects, the model gains a heightened awareness of its surroundings, which is critical for safe navigation. This capability enables the autonomous system to avoid collisions with pedestrians and other vehicles more effectively.

Another significant enhancement in our model is the replacement of traditional cross attention with gated cross attention within the decoder. This modification provides a more controlled and contextually aware mechanism for attention, which is crucial for maintaining focus on relevant objects and scenarios during autonomous driving.

We validated our model using the large-scale nuScenes dataset. The results were compelling, demonstrating superior performance in terms of L2 loss and collision rates when compared to current state-of-the-art methods. Additionally, our ablation studies provided clear evidence of the effectiveness of each individual component within our model, confirming that both the innovations in the encoder strategy and the enhancements in the object detection integration substantially contribute to the overall performance and reliability of this end-to-end autonomous driving system.

Limitation: There is a thought-provoking limitation, which is associated with the current evaluation metrics used for assessing autonomous driving systems, specifically focusing on L2 loss and collision rates. These metrics, while informative, do not fully encapsulate the complexities of real-world driving scenarios. First, there are multiple potential routes in driving scenarios, yet only one ground truth route is considered. This may result in penalization of other safe and reasonable routes. Furthermore, the collision rate metric solely accounts for collisions with other agents and does not consider scenarios where the ego vehicle deviates from lane lines. Therefore, some routes that veer off the road might still achieve very low collision rates, highlighting a gap in the current evaluation approach.

Fig. 6: First Example result of our pre-trained model

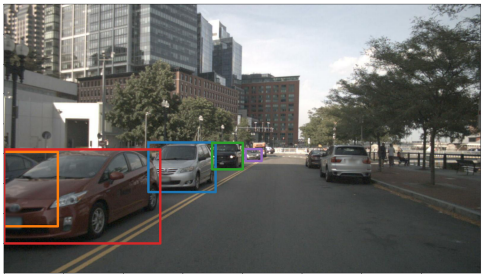

Fig. 7: Second Example result of our pre-trained model

APPENDIX

These are 2 example results of our pre-trained object detection model on nuScenes dataset. Through these visual assessments, we confirm that our model maintains high detection accuracy, making it a trustworthy component in the whole task.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.

[3] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 590–15 599.

[4] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "End-to-end urban driving by imitating a reinforcement learning coach," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 15 222–15 232.

[5] 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.

[6] J. Zhang, Z. Huang, and E. Ohn-Bar, "Coaching a teachable student," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7805–7815.

[7] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger, "Exploring data aggregation in policy learning for vision-based urban autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 763–11 773.

[8] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[10] S. Casas, A. Sadat, and R. Urtasun, "Mp3: A unified model to map, perceive, predict and plan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 403–14 412.

[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] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[13] 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.

[14] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[15] B. Zhou, P. Krähenbühl, and V. Koltun, "Does computer vision matter for action?" *Science Robotics*, vol. 4, no. 30, p. eaaw6661, 2019.

[16] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[17] P. Cai, S. Wang, H. Wang, and M. Liu, "Carl-lead: Lidar-based end-to-end autonomous driving with contrastive deep reinforcement learning," *arXiv preprint arXiv:2109.08473*, 2021.

[18] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, "Multimodal end-to-end autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 537–547, 2020.

[19] I. Sobh, L. Amin, S. Abdelkarim, K. Elmadawy, M. Saeed, O. Abdeltawab, M. Gamal, and A. El Sallab, "End-to-end multi-modal sensors fusion system for urban automated driving," 2018.

[20] P. Cai, S. Wang, Y. Sun, and M. Liu, "Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4218–4224, 2020.

[21] 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.

[22] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995, pp. 103–129.

[23] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[24] 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.

[25] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 222–17 231.

[26] 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.

[27] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[28] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *Advances in neural information processing systems*, vol. 30, 2017.

[29] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7153–7162.

[30] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde, "Gri: General reinforced imitation and its application to vision-based autonomous driving," *Robotics*, vol. 12, no. 5, p. 127, 2023.

[31] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 584–599.

[32] 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.

[33] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[34] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis) design for autonomous driving," *Artificial Intelligence*, vol. 316, p. 103829, 2023.

[35] C. Zhang, R. Guo, W. Zeng, Y. Xiong, B. Dai, R. Hu, M. Ren, and R. Urtasun, "Rethinking closed-loop training for autonomous driving," in *European Conference on Computer Vision*. Springer, 2022, pp. 264–282.

[36] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8271–8277.

[37] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "Ibrnet: Learning multi-view image-based rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4690–4699.

[38] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, p. 2, 2000.

[39] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2575–2582.

[40] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 3, pp. 3461–3475, 2022.

[41] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[42] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8748–8757.

[43] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," *arXiv preprint arXiv:2301.00493*, 2023.

[44] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[45] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[46] Motional, "Pixel-wise segmented result within images," 2024. [Online]. Available: https://www.nuscenes.org/nuimages
[47] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.
[48] 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.
[49] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.
[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[51] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.
[52] P. Hu, A. Huang, J. Dolan, D. Held, and D. Ramanan, "Safe local motion planning with self-supervised freespace forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 732–12 741.
[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.