LaneLM: Lane Detection as Language Modeling

Yunzhi Zhang and Xiuxiu Bai *

Posted Date: 19 April 2025

doi: 10.20944/preprints202504.1582.v1

Keywords: 2D lane detection; interactive model; visual question answering; autoregressive modeling for vision

Article

LaneLM: Lane Detection as Language Modeling

Yunzhi Zhang [1] and Xiuxiu Bai [1,*]

[1] School of Software Engineering, Xi'an Jiaotong University, No.28 Xianning West Road, Xi'an, 710049, Shaanxi Province, China
[2] Inspur Yunzhou Industrial Internet Co., Ltd., Jinan City, Shandong Province 250101, China
* Correspondence: xiubai@xjtu.edu.cn

Abstract: Lane detection ensures that vehicles remain within drivable areas. However, the auto-annotation of lane detection often underperforms in corner cases. If the output of large visual language models or human feedback could be utilized as prompts in these corner cases, it would greatly enhance the quality of annotations. Nevertheless, existing lane detection models lack the capability for interaction. We present LaneLM, an interactive and promptable framework for lane detection. We are the pioneer who tackles lane detection as a visual question-answering task that progressively estimates different lane sequences in the same image through multi-turn conversations, where the current estimate is prompted by previous context and in turn affects subsequences. This prompt-based approach can generate the correct lanes with some keypoint prompts when the model fails in some corner cases, thus making it faster for image manual annotation and enabling the use of prompts similar to those in language models. Notably, using only 4-point prompts, LaneLM achieves state-of-the-art performance and easily surpasses previous SOTA methods, especially in corner cases. In particular, we get 82.71% F1-score on CULane. Despite its accuracy, LaneLM is lightweight with extremely low trainable parameters.

Keywords: 2D lane detection; interactive model; visual question answering; autoregressive modeling for vision

1. Introduction

Lane detection (**LD**), the process of recognizing lanes as approximated curves, is a foundational task in the realm of autonomous driving, whereby the detector aims to perceive the accurate shape of each lane in an image captured by a front-mounted camera. In the context of autonomous driving or even human-driven vehicles equipped with advanced driver assistance systems (ADAS), accurate lane detection helps keep the vehicle within its proper lane.

Our objective is to develop an interactive lane detection model similar to SAM [1] (Segment Anything Model). Firstly, when encountering complex road conditions where the model fails to recognize, it can accept human feedback in natural language to weight the model's spatial attention or can be prompted by a parallel large visual language model. Secondly, autonomous driving requires a large number of manual annotations for training. Before manual annotation, images are auto-annotated by a pretrained LD model, which often underperforms in corner cases that follow a long-tail distribution. If annotators could directly indicate several keypoints of the lane to the model and the model could autoregress complete high-quality lane based on these semantic cues, it would significantly reduce annotation costs. However, existing lane detection models neither possess such interactive capabilities nor can they be prompted. To address these issues, we train the LD model using the language model training method to explore the improvements gained by the model receiving a few prompts.

Existing lane detection methods, whether based on line-anchor regression, segmentation, or keypoint detection, rely on convolutional neural networks (CNN) and spatial attention mechanisms. Firstly, the locality of CNNs makes it difficult to capture the long-range dependencies of lanes across

 doi:10.20944/preprints202504.1582.v1

the entire image. Secondly, traditional methods lack the interactive capabilities of language models and cannot specify the position of spatial attention weights in the feature map in corner cases. To address these issues, we convert visual and semantic information into a token sequence, utilizing a cross-attention mechanism, in which queries are language tokens and keys and values are visual tokens. Therefore, the model can determine the location where attention is applied on the feature map.

Inspired by [1,2], thanks to Lane2Seq [3], this study proposes a promptable framework, called **LaneLM**, to LD that differs from the mainstream approaches. Similar to pioneer work [3,4], our method is based on two intuitions: First, if the detector knows where lanes are, we just need to teach it to read them out so that we can refine the location of keypoints in this context. If an initial set of keypoints, which are discretized into tokens, is given as prompt for each lane , the detector should generate holistic sequence of the lane respectively in a manner of causal language modeling. Second, since language models are few-shot learners[5], why not use a small number of prompts to enhance the LD performance in corner cases? Besides, we can utilize the instruction-following ability of language models to make LD models follow simple instructions. Furthermore, LaneLM avoids customized heads, relying instead on direct keypoint token classification.

In summary, we make the following contributions:

1. It is the first attempt to treat the LD task as visual question answering (**VQA**). Our work paves the way toward unification for language modeling and lane detection.
2. We develop an interactive lane detection framework that can accept prompts with semantic information to guide the visual modality.
3. With a small number of keypoint prompts (e.g. 4 adjacent points, about 7% of the total length of each lane), LaneLM easily outperforms the previous SOTA method by 1.09%.

Figure 1. Corner cases in lane detection. There are many challenging scenarios in the lane detection task, such as at night, with severe occlusions, resulting in no visual clues for visual task.

2. Related Works

2.1. Traditional Lane Detection Methods

The mainstream approaches can be classified into the following categories:

1) Anchor-based methods. The network structure designing is similar to object detection but [6–9] use line-anchor instead of anchor box as lane prior and regress offsets between sampled points and predefined anchor points. At inference time, Non-Maximum Suppression (NMS) is applied as post-processing. These methods take both speed and accuracy into account. Nevertheless, these methods struggle to detect lanes with complex topology, the predefined line anchor with a fixed shape fails to serve as a correct reference.

2) Segmentation-based methods [10–12] pixel-wisely predict each lane and assign different lanes to different channels, respectively, so that they are flexible enough to detect lanes with complex topology. Segmentation-based methods can be unified with other segmentation tasks. For example, a single model can be used to segment lanes and parking slots simultaneously. However, these methods, which require heuristic post-processing to decode the segmentation map, are not accurate or fast enough compared with anchor-based methods.

 doi:10.20944/preprints202504.1582.v1

3) Keypoint-based methods. [13–15] treat each lane as a set of keypoints and then binarily predict foreground points and the background. Compared with segmentation-based methods, keypoint-based methods reduce the complexity of foreground classification. They can handle complex topological structures while also considering real-time performance. However, the post-processing of keypoint-based methods still remains an issue. Researchers have developed complex clustering and filtering methods to distinguish lanes in the same image.

4) Row-wise methods [16–19], which rasterize lanes in the image and row-wisely predict the grid occupation for each lane, are simple yet effective. These methods develop the row-anchor representation for lane detection instead of a line-anchor. However, side lanes of the Y-shape, often nearly horizontal, make lane classification difficult. These methods can achieve a high inference speed and easily surpass the methods described above in terms of real-time performance while getting comparable results.

5) Curve-based methods [20–23] formulate each lane into a parametric equation, such as polynomials or Bézier curves, predicting the parameters. Benefiting from holistic representation of the lane, inherently eliminate occlusions. These methods predict lanes in an end-to-end manner, require no post-processing but their performance still lag behind other methods by a large margin.

The current SOTA CLRerNet [9], based on the previous SOTA CLRNet [6], has introduced the LaneIoU loss for training, taking into account the topological structure of the lanes. CLRNet [6] integrates high-level features with low-level features that contain more spatial information, and achieves impressive results through the application of a spatial attention mechanism. Previous SOTA GANet [13] use DCN [24] to associate keypoints in the same lane with their belonged start points. In summary, traditional LD methods focus only on the spatial information and topological structure of lanes.

2.2. Autoregressive Modeling for Vision

In light of the aforementioned methods, spatial information of lane perception in 2D space is well-exploited. We aim to borrow semantic-aware capability from Natural Language Processing (**NLP**) so that our model can capture contextual dependency of input sequences and produce high-quality output. . Autoregressive (**AR**) model has found great success in motion prediction [25,26], autonomous driving [27], NLP [5,28,29] and visual tracking [30,31]. Apart from Lane2Seq [3], little work in lane detection focuses on this promising paradigm that utilizes temporal or semantic information, and none of these traditional LD methods are interactive.

Pix2Seq [4] quantizes bounding boxes into language tokens to describe the objects. Therefore, the model can learn to ground the language tokens on visual observation. Following Pix2Seq, ARTrack [30,31] firstly adopts this framework in visual tracking. Flamingo [28] explores the few-shot learning capability of visual language models and it is visually-conditioned AR language model, i.e. a visual encoder is used to perceive the information in images while a language model is utilized to refine the visual information at the semantic level in the few-shot context and employ a connector to bridge the visual encoder and the language decoder. The same architecture can also be found in many VQA models [29,32–34].

3. Proposed LaneLM

LaneLM has neither complex trick nor sophisticated structure. First, we tokenize the float keypoints and convert them into word embeddings. Then, we use a visual encoder to transform the visual information into visual embeddings. Next, a language decoder (shown in Figure 4) is designed for prompting. We simply use cross-attention layers to condition the language model [35] on visual inputs.

3.1. Lane Representation

A lane is a sequence of xy coordinates represented as tuples of float numbers. First, we quantize the float numbers in these tuples into integer tokens. Then, we integrate the x and y coordinates

 doi:10.20944/preprints202504.1582.v1

information into a single embedding, transforming it into a contextual representation that can be processed by a transformer decoder.

Lane tokenization. In order to represent lanes, following [6,8], we use equally-spaced 2D-points as lane representation. Each lane L is expressed as a sequence of keypoints, i.e.

$$L = \{(x_1, y_1), (x_2, y_2)..., (x_T, y_T)\} \tag{1}$$

in which y coordinate is equally and vertically sampled at regression time step t, i.e. $y_t = \frac{H}{T} \cdot t$, where H is image height. Inspired by row-wise methods mentioned above, we transfer continuous keypoints of lane into discrete counterparts. Then we can rasterize H pixel rows into integer space T, which is also the vocabulary size for y_t. Specifically, the keypoint at time step t is composed of two tokens, i.e. (x_t, t). For each column, x_t is quantized to $[0, n_{bins})$, in which n_{bins} is the vocabulary size for x_t.

Figure 2 shows the row-wise location for a lane. We process the n_{bins}-classes classification in each row and T-classes classification at time step t in each column.

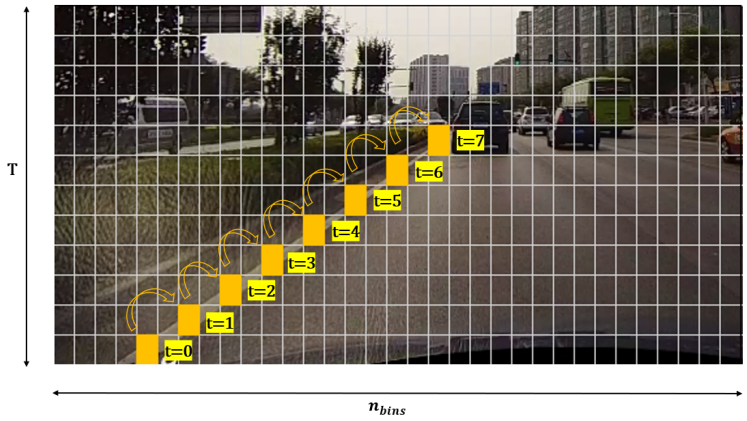


Figure 2. LaneLM selects row-wise location from the rasterized image and rollouts keypoints step by step.

It should be noted that 0 and T are padding tokens for x, y coordinates respectively (i.e. if there is no lane at time step t, let $x_t = 0$ and $y_t = T$). Here, T also refers to the number of rollouts.Then we use the quantized term to index a learnable vocabulary to get the token counterpart, which allows the model to depict the location of keypoints.

Keypoint token encoding. If $a_t^i = (x_t^i, y_t^i)$ in the i-th lane at time step t is given, we use two simple embedding layers E_x and E_y to get the token embeddings $e \in \mathbb{R}^D$, where D is the size of each vector. Therefore, each integer tuple a_t^i is mapped to a high-dimensional vector.

$$e_t = E_y(y_t) + E_x(x_t) + \mathrm{PE}_{keypoint}(t) \tag{2}$$

where $\mathrm{PE}_{keypoint}(\cdot)$ is positional encoding in language modeling [35] to provide transformer model with information about the position of each token in the sequence. Eq. 2 encodes the information of the x-coordinate and y-coordinate of tuple a_t^i into embeddings, which are then fused through addition, followed by the incorporation of positional information. The sequence composed of these embeddings conditioning on visual features is subsequently processed by a language decoder. In the output layer, each embedding will go through the linear head and predicts the next token by classification:

$$a_{t+1} = \mathrm{argmax}(\mathrm{softmax}(Linear(e_t))) \tag{3}$$

Therefore, we embed a integer tuple lane sequence L (see Eq. 1) to its contextual representation

$$\mathbf{h} = \{e_1, e_2, ..., e_T\} \tag{4}$$

 doi:10.20944/preprints202504.1582.v1

3.2. Probabilistic Rollouts

Probability factorization. Following [25], let $a_t^n = (x_t^n, y_t^n)$ represent the target keypoints for the n-th lane at time t. We cast LD as a next-token prediction task, formulated as a conditional probability $P(A_1, ...A_T|\mathbf{X_v})$, where $A_t = \{a_t^1, ..., a_t^N\}$ represents the set of target keypoints for all lanes at time t and $\mathbf{X_v} \in \mathbb{R}^{B\times C\times H\times W}$ is the visual context, RGB image inputs from camera. Thus, we factorize the distribution over future sequences as a product of conditionals:

$$p_\theta(A_1, ...A_T|\mathbf{X_v}) = \prod_{t=1}^{T} p_\theta(A_t|A_{<t}, \mathbf{X_v}) \tag{5}$$

$$p_\theta(A_t|A_{<t}, \mathbf{X_v}) = \prod_{n=1}^{N} p_\theta(a_t^n|A_{<t}, \mathbf{X_v}) \tag{6}$$

Similar to [25], Eq. 6 represents the fact that we treat each lane as conditionally independent for its keypoint at time t, given the previous keypoints and visual context.

n-lanes parallelism at time step t. In this formulation, we duplicate the visual context n times for each lane in an image, where duplication is repeated in batch dimension in our implementation to calculate A_t in parallel because they are assumed to be conditionally independent as described in Eq. 6.

3.3. Network Architecture

In section 3.1, we obtained the embedding sequence for the keypoint modality. Similarly, we first encode the image into an embedding sequence using a visual encoder. Then, the information from both the visual modality and the keypoint modality is fused within the AR language decoder. Ultimately, each output embedding can be utilized by the classification head to predict the next token id. We use an encoder-decoder structure for learning and inference. Such a network architecture is widely used in vision-language model [28,29,36] and language model [37].

Visual encoder. Our visual encoder has to extract visual features from images and then transform them into embedding sequences. The pyramid feature extractor f shown in Figure 3 adopts the standard ResNet[38] and DLA[39] as visual feature extractor. And we add a FPN [40] neck to provide integrated multi-scale features:

$$\{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2\} = f(\mathbf{X_v}) \tag{7}$$

where C_i, H_i and W_i is the channel, the height and the width of the i-th level feature $\mathcal{F}_i \in \mathbb{R}^{B\times C_i\times H_i\times W_i}$ extracted from CNN and FPN [40].This structure leverages ConvNet's pyramidal feature hierarchy and demonstrates its efficiency in [6,13,16,41]. Then we split $\mathcal{F}_i$ into fixed-size patches, linearly embed each of them, add position embeddings:

$$\mathcal{L}_i = E_v(\mathcal{F}_i) + \mathrm{PE}_{vision}(H_i, W_i) + \mathrm{LE}(i) \tag{8}$$

where $\mathrm{LE}(\cdot)$ is level embedding that embeds level information into vectors, $\mathrm{E_v}(\cdot)$ is patch embedding in ViT[42], $\mathrm{PE}_{vision}(\cdot)$ is its standard positional encoding layer that retains the positional information of each patches and the result value $\mathcal{L}_i \in \mathbb{R}^{B\times N_i\times D}$ represents the token sequence extracted from level feature $\mathcal{F}_i$, in which N_i is the number of patches and we linearly embed them into D-dimensional visual embeddings to aligned with keypoint embeddings $e \in \mathbb{R}^D$.

 doi:10.20944/preprints202504.1582.v1

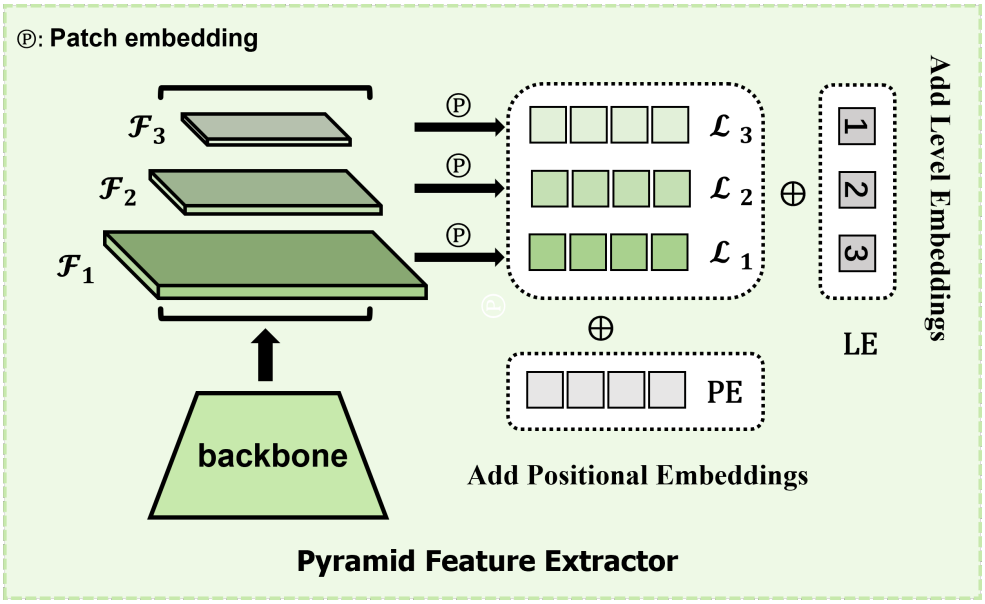


Figure 3. Our visual encoder f transforms pyramid feature into sequences of embeddings $\mathcal{L}$.

Since ViTAdapter [43] proposes a pretraining-free adapter and achieves SOTA result in dense prediction and more and more visual-language model [28,29] use a frozen pretrained visual encoder and with how well the spatial feature in LD has been exploited, there is good reason to believe that we can directly adopt a frozen backbone from pretrained to reach comparable performance.

Language decoder. Our language decoder causally models keypoint sequence and conditions each embeddings e on visual sequences $\mathcal{L}$. As shown in Figure 4, we use a transformer decoder for target sequence generation. This decoder consists of 3 layers of LaneLM blocks. For every LaneLM block, we insert new cross-attention layers between OPT[35] causal attention blocks, trained from scratch. The visual feature sequence $\mathcal{L}_i$, a sequence of flatten 2D-patch embeddings will serve as keys and values in cross-attention layer of block i while queries are derived from keypoint prompts:

$$\mathbf{h_i} = \text{CrossAtt}(\text{Q} = \text{CausalSelfAtt}(\mathbf{h_{i-1}}), \text{K} = \mathcal{L}_i, \text{V} = \mathcal{L}_i) \tag{9}$$

in which we denote $\mathbf{h_i}$ as hidden state output from block i and specially, $\mathbf{h_0} = [e_1, e_2, ..., e_N]$ is original keypoint embedding sequence input. $\mathbf{h_0}$ should go through word embedding layer and add position embedding as OPT[35] does before feeding into block i.

Our LaneLM layer can avoid information leak. cross-attention ensures that the outputs for a certain position in a sequence is based only on the known query tokens at previous positions and not on future positions. Besides, each query embedding is conditioned on the whole visual information. Theorem A1 demonstrates the feasibility of the network architecture.

Decoupled head. In training phase, every embedding (except for the last embedding of each sequence) output from language decoder will go through x head and y head to predict the next token id of x and y as illustrated in Eq. 3. Considering that predicting ordered y sequence is a simple task, we use two decoupled classification heads, x head, y head, to predict the next token of x and y, respectively, instead of predicting xy tokens like [25]. Such decoupled prediction strategy greatly reduces complexity of classification from $O(|X||Y|)$ to $O(\max(|X|, |Y|))$. To further reduce computational overhead, these heads share the same hidden state input because the embedding e_t contains the information of both x-coordinate and y-coordinate (see Eq. 2).

 doi:10.20944/preprints202504.1582.v1

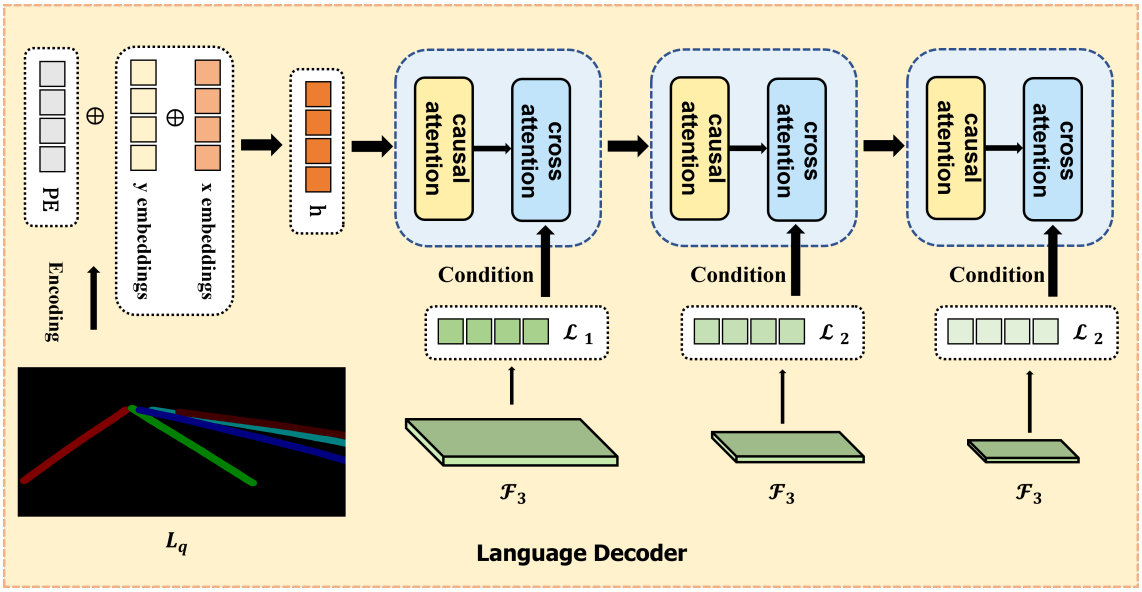


Figure 4. To condition the LM on visual inputs, we insert cross-attention layers between causal attention layers. The keys and values are obtained from the visual features $\mathcal{L}$ while queries are derived from keypoint prompts L_q

3.4. Training and Inference

Visual question answering. If the detector knows where lanes are, we just need to teach it to read them out so that we can refine the location of keypoints in this context. For instance, during training, each image is pre-annotated by other lane detection models. The output labels are converted into pseudo labels using the method described in Section 3.1. These pseudo labels are then used as lane priors and fed into the language decoder as queries to extract spatial information from the keys and values in its cross-attention modules. In this way, the model gains an approximate understanding of the lane locations. Our goal is to refine these locations or just read them out if pseudo labels are accurate enough. Moreover, These pseudo-labels can be regarded as questions in VQA, while their corresponding ground truth serves as the answers to these questions. In this way, the LD task is transformed into a VQA task.

Self-supervised labels. Following [28,29], we design a muti-turn conversation between LaneLM and a pretrained teacher. For an input image $\mathbf{X_v}$, we consider the pretrained detecor [6], which provides pseudo keypoint labels L_q as init queries for laneLM. Then we generate multi-turn conversation data $(L_q^1, L_{gt}^1; ...; L_q^N, L_{gt}^N)$, where N, the max number of turns is also the max number of lanes in each image and L_{gt}^1 is the first lane of ground truth in the image. We organize them as a sequence by treating ground truth as the answer of pseudo label. Therefore, the multi-modal self-supervised label S for image $\mathbf{X_v}$ can be expressed as follows:

$$S = (L_q^1 \circ L_{gt}^1, ..., L_q^N \circ L_{gt}^N, \mathbf{X_v}) \tag{10}$$

where $\circ$ means concatenating two sequences. We adopt the bipartite matching to find the matching that minimizes the distance of the start points between the query sequence L_q^i and the answer L_{gt}^j, and then take the matched pair (L_q^i, L_{gt}^j) as the self-supervised label for each lane.

Loss. We only adopt standard loss in the decoder-only language models. We train our model by minimizing negative log-likelihoods of keypoint tokens conditioned on the visual inputs $\mathbf{X_v}$ with cross entropy loss at each time stamp t:

$$\text{maximize} \sum_{t=1}^{T} (\log P(x_t | x_{<t}, \mathbf{X_v}) + \log P(y_t | y_{<t}, \mathbf{X_v})) \tag{11}$$

where T is the length of the sequence.

 doi:10.20944/preprints202504.1582.v1

Inference. We sample tokens from the model likelihood $P(x_t|x_{<t}, \mathbf{X_v})$ and $P(y_t|y_{<t}, \mathbf{X_v})$ using the argmax sampling as illustrated in Eq. 3.The same as language models, we apply the standard greedy search with fixed length and EOS (the End of Sequence token) stop criteria to generate xy tokens at the same time (i.e. we stop prediction when EOS token is predicted or the current sequence reaches the max length). After obtaining the discrete tokens, we de-quantize them to get continuous coordinates.

To speed up the inference, in addition to adopting the parallel strategy outlined in Section 3.2, each level of visual sequence $\mathcal{L}_i$ is cached into the decoder's corresponding cross-attention layer. Further more, we adopt the KV-cache strategy in each causal self-attention layer only at inference time.

Prompting strategy. We have the following three prompting strategies. (1) A regression network is employed to provide the two initial keypoints, for each lane. LaneLM is responsible for completing the remaining keypoints. The regression network (we use CLRNet [6]) only gives start points for each lane rather than the holistic lane, which is easier than the LD task. Keypoint-based methods [13–15] use the similar start point regression. However, they are struggling to design keypoint decoding strategy while we leverage contextual representation of lane and just rollout the keypoints. (2) Pseudo labels in section 3.4 are given as questions, our language decoder leverage the few-shot capability to generate the answers and refine the read input or just read them out. (3) We simulate the annotation process performed by annotators by introducing a certain level of noise to the ground truth (randomly shifting the x-coordinates by -5 to 5 pixels) to demonstrate LaneLM's capability of interaction. (4) We give some simple instruction tokens to LaneLM to accomplish some post-processing tasks.

3.5. Hallucination Removal

Language models are prone to hallucination, generating plausible but non-factual content [44]. As shown in Figure 6(a), like most language models, LaneLM may exhibit hallucination on side lanes. In order to mitigate such hallucination, we have developed a threshold filtering strategy **HR** (Hallucination Removal) based on clustering at the 85th percentile, in which points with offsets of adjacent x-coordinates exceeding twice the 85th percentile, along with their subsequent points, will be filtered out. Figure 6(a) shows the mitigation of hallucination after filtering by HR.

4. Experiments

4.1. Implementation Details

Model variants. We have three model variants and their trainable parameters are shown in Figure 5. With the different hidden size of the decoder, the visual encoders of LaneLM-128, LaneLM-256, and LaneLM-512 are ResNet18, ResNet34 [38] and DLA34 [39].

Hyperparameters. We generate 40 points ($T = 40$) for CULane [45], 56 for TuSimple [46] and LLAMAS [47]. We only apply HR for lanes whose valid number of points is greater than $\mathcal{N} = 10$ in Alg. 1. On CULane [45] and LLAMAS [47], the max number of lanes $N = 4$, while on TuSimple [46] $N = 5$. The padding value for x-offsets is 0, while T for y. EOS is a special token that represents the end of the sentence. (x, y) is an EOS token when $x = 0$ or $y = T$. In our additional study, we use EOS token as cmd to control the output and LaneLM stops its inference until the second EOS token is predicted.

 doi:10.20944/preprints202504.1582.v1

Algorithm 1: HR($\cdot$)

Data: **x**:x-coordinates of each lane.
Result: **x**: x-coordinates after hallucination removal.

if $len(\mathbf{x}) > \mathcal{N}$ **then**
  $\text{diff} \leftarrow \text{abs}(\mathbf{x}[1:] - \mathbf{x}[:-1])$;
  $\theta \leftarrow 2 \cdot \text{percentile}(85^{th}, \text{diff})$;
  $p \leftarrow \text{argmin}(\text{diff} > \theta)$;
  if $p > 0$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}[:p+1]$;
  end
end
Return **x** ;

Training environment. All experiments are carried out with a single NVDIA GeForece RTX 4090, 128 batch size, 800 n_{bins} and 100 training epochs and we report the best results in Table 3, Table 2 and Table 4. All visual encoders are frozen in the training phase and pretrained from [6].

4.2. Real-Time Performance and Computational Overhead

Figure 5 shows that the trainable parameter of LaneLM is extremely low. We aim to make LaneLM like an adapter, bridging the visual and language modalities. Similar to multimodal models, there is no need for pre-training the visual encoder. LaneLM is easy to plugin into any LD models.

Table 1 reports FPS (Frames Per Second) of LaneLM. We did not measure the inference time for the visual encoder f in Eq. 7 because it is frozen and shared with the teacher model CLRNet [6], which can also be replaced with other pretrained models. We set batch size to 1 and measure the max memory allocated during inference using the corresponding function in pytorch. T=80 indicates that we rollout 80 keypoints for each lane. The impact of sequence length on memory overhead is minimal when the batch size is relatively small.

The real-time performance of LaneLM can be attributed to our cache strategies. We cache the visual sequence of each level feature in each cross-attention layer in language decoder and adopt KV-cache at inference time. All lanes in the same image can be batch processed after padding. Lane2Seq [3] is an AR LD model, which can not reach real-time performance.

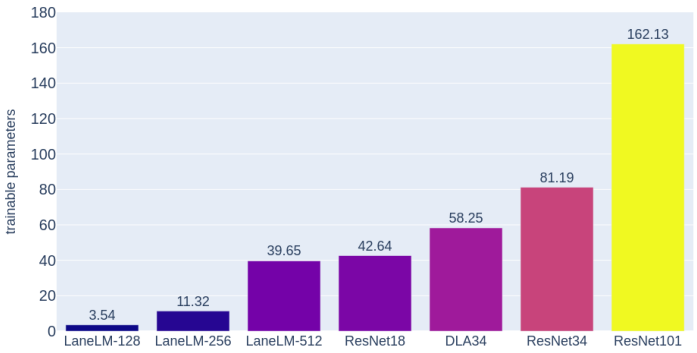


Figure 5. The comparison of trainable parameters in LaneLM with some backbones.

 doi:10.20944/preprints202504.1582.v1

Table 1. Performance on different model variances

Model	FPS	Max Memory Allocated	Trainable Parameters
LaneLM-128	370	190MB (T=80)	3.54MB
LaneLM-256	290	239MB (T=40)	11.32MB
LaneLM-512	227	281MB (T=40)	39.65MB
Lane2Seq [3]	Extremely slow	Not reported	Not reported

4.3. Datasets

Our experiments are conducted on three lane detection benchmark datasets: **CULane** [45] and **TuSimple** [46] and **LLAMAS** [47].

CULane [45] is a large-scale dataset that poses significant challenges for lane detection. It encompasses nine challenging categories, including crowded scenes, nighttime conditions, crossroads, etc. It has 100,000 images. Each image in CULane has a resolution of 1640 × 590 pixels.

Tusimple [46] lane detection benchmark is among the most widely used datasets in lane detection. It consists solely of highway scenes and contains 3268 images for training, 358 for validation, and 2782 for testing. These images have a resolution of 1280 × 720 pixels.

LLAMAS [47] is a large lane detection dataset that has more than 100,000 images. The annotations in LLAMAS are automatically generated using high-definition maps. The images in LLAMAS are all from highway scenarios. For evaluation purposes, the CULane's F1 metric is utilized. The annotations of the testing set are not publicly available.

4.4. Evaluation Metric

For **CULane** [45] and **LLAMAS** [47], we use the F1@50 measure as the evaluation metric. We compute intersection-over-union (IoU) of two lanes, which is defined as the IoU of the two lane masks within 30 pixels and used to determine whether a sample is true positive (TP), false positive (FP), or false negative (FN), between prediction lane and ground truth. Predicted lanes with an IoU greater than a threshold (50%) are regarded as true positives (TP). Further, F1 score is calculated as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{Precision} = \frac{TP}{TP + FP} \quad \text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

In particular, since there are no lanes in the cross-road scenes, the false negative will be adopted as the evaluation metric.

For **Tusimple** [46], we report three official metrics: false positive rate, false negative rate and accuracy:

$$\text{accuracy} = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}} \tag{13}$$

where S_{clip} is the total number of ground truth lane points and C_{clip} is the number of correct lane points in prediction of a clip. A predicted lane is considered correct if more than 85% of its predicted lane points are within 20 pixels of the ground truth. In addition, we also report the F1 score on TuSimple.

4.5. Benchmark Results

Performance on TuSimple. Table 2 presents the performance comparison with SOTA methods on TuSimple. * indicates that LaneLM only uses prompting strategy (1) from section 3.4. The performance difference on this dataset is very small because TuSimple is a dataset of small scale and it contains single scene only. Our prompting strategy has a limited effect on performance. The result in this dataset seems to be saturated. Therefore, The prediction can easily overlap with the ground truth. It is hard to get a significant improvement on TuSimple.

 doi:10.20944/preprints202504.1582.v1

Table 2. Comparison with other methods on TuSimple

Methods	Encoder	F1↑	Acc↑	FP↓	FN↓
UFLD [17]	ResNet18	87.87	95.82	19.05	3.92
UFLD [17]	ResNet34	88.02	95.86	18.91	3.75
CondLaneNet [16]	ResNet18	97.01	95.48	2.18	3.80
CondLaneNet [16]	ResNet34	96.98	95.37	2.20	3.82
CondLaneNet [16]	ResNet101	97.24	96.54	2.01	3.50
CLRNe [6]	ResNet18	97.89	96.84	2.28	1.92
CLRNet [6]	ResNet34	97.82	96.87	2.27	2.08
CLRNet [6]	ResNet101	97.62	96.83	2.37	2.38
LaneATT [8]	ResNet18	96.71	95.57	3.56	3.01
LaneATT [8]	ResNet34	96.77	95.63	3.53	2.92
LaneATT [8]	ResNet122	96.06	96.10	5.64	2.17
LaneLM-128*	ResNet18	96.41	96.33	4.08	2.54
LaneLM-128 (0-kp)	ResNet18	96.36	96.12	3.03	2.62
LaneLM-128 (4-kp)	ResNet18	97.16	96.45	4.66	2.50
LaneLM-256*	ResNet34	96.50	96.35	3.62	2.52
LaneLM-256 (0-kp)	ResNet34	96.49	96.31	4.03	2.54
LaneLM-256 (4-kp)	ResNet34	97.01	96.43	3.27	2.32
LaneLM-512*	ResNet101	97.61	96.69	2.13	2.65
LaneLM-512 (0-kp)	ResNet101	97.64	96.76	2.41	2.29
LaneLM-512 (4-kp)	ResNet101	97.85	96.80	2.21	2.65

Performance on CULane. Table 3 reports the main results on CULane. Our model receives two adjacent keypoints output from CLRNet [6] as init prompts for each lane and rollouts the remaining keypoints in the * version. To explore the interactive capability of LaneLM, (2-kp) denotes that the holistic lane predicted from CLRNet is given and two adjacent ground truth keypoints with ramdom shift at the commencement of each lane are also supplied to enhance model performance.

We observed that with the increase in the number of keypoint prompts, F1-score has significantly improved in most corner cases, which can be attributed to the powerful language modeling capability. The results on CULane demonstrate that our model has surpassed the SOTA [9] after prompting and is particularly suitable for application in the annotation of corner cases. More results can be found in Figure A1

Analysis on Limitations. **(1)** In the * version, LaneLM underperforms CLRNet because, in Eq. 10, LaneLM actually predict pseudo-labels from CLRNet i.e. the knowledge of this part in LaneLM is distilled from the CLRNet. **(2)** LaneLM with fewer keypoint prompts is worse than the * version because, in the training sequence, a sudden jump occurs at the junction between the pseudo-label and the ground truth (see Eq. 10), which disrupts the contextual semantic information and confuses the model. It has been observed that the model often hallucinates on the side lanes, indicating that the model struggles to cope with abrupt changes in semantic information.

Performance on LLAMAS. The result on the LLAMAS is shown in Table 4. LaneLM-512 outperforms PolyLaneNet [20] by 7.05 and LaneATT [8] by 2.08. The training strategy is slightly different with CULane and TuSimple. We directly use L_{gt} as self-supervised label S and Lq is not used during training, which is different with Eq. 10. Thus, we can avoid knowledge distilling from the teacher model.

 doi:10.20944/preprints202504.1582.v1

Table 3. Comparison with SOTA methods on CULane

Methods	Encoder	Normal↑	Crowd↑	Dazzle↑	Shadow↑	No line↑	Arrow↑	Curve↑	Night↑	Cross↓	Total↑
Segmentation based											
SCNN [45]	ResNet50	90.60	69.70	58.50	66.90	43.40	84.10	64.40	66.10	1900	71.60
LaneAF [11]	DLA34	91.80	75.61	71.78	79.12	51.38	86.88	72.70	73.03	1360	77.41
AtrousFormer [48]	ResNet34	92.83	75.96	69.48	77.86	50.15	88.66	71.14	73.74	1054	78.08
Lane2Seq-s [3]	ViT-Base	93.39	77.27	73.45	79.69	53.91	90.53	73.37	74.96	1129	79.64
Anchor based											
LaneATT [8]	ResNet122	91.74	76.16	69.47	76.31	50.46	86.29	64.05	70.81	1264	77.02
Laneformer [49]	ResNet50	91.77	75.74	70.17	75.75	48.73	87.65	66.33	71.04	**19**	77.06
ADNet [50]	ResNet34	92.90	77.45	71.71	79.11	52.89	89.90	70.64	74.78	1499	78.94
Lane2Seq-a [3]	ViT-Base	93.11	77.43	73.25	79.46	53.74	90.02	72.44	75.12	1173	79.27
CLRNet [6]	ResNet34	93.49	78.06	74.57	79.92	54.01	90.59	72.77	75.02	1216	79.73
CLRerNet [9]	DLA34	94.36	80.62	75.23	84.35	57.31	91.17	79.11	76.92	1540	81.43
Row wise methods											
UFLD [17]	ResNet34	90.70	70.20	59.50	69.30	44.40	85.70	69.50	66.70	2037	72.30
UFLDv2 [18]	ResNet34	92.5	74.8	65.5	75.5	49.2	88.8	70.1	70.8	1910	76.0
CondLaneNet [16]	ResNet34	93.38	77.14	71.17	79.93	51.85	89.89	73.88	73.92	1387	78.74
Keypoint based											
FOLOLane [14]	ERFNet	92.70	77.80	75.20	79.30	52.10	89.00	69.40	74.50	1569	78.80
GANet [13]	ResNet-101	93.67	78.66	71.82	78.32	53.38	89.86	77.37	73.85	1352	79.63
RCLane [15]	SegFormer-B1	93.59	78.77	72.44	84.37	52.77	90.31	78.39	73.96	907	80.03

 doi:10.20944/preprints202504.1582.v1

Table 3. *Cont.*

Methods	Encoder	Normal↑	Crowd↑	Dazzle↑	Shadow↑	No line↑	Arrow↑	Curve↑	Night↑	Cross↓	Total↑
Parameter based											
LSTR [22]	ResNet18	64.00	-	-	-	-	-	-	-	-	64.00
BezierLaneNet [21]	ResNet18	90.22	71.55	62.49	70.91	45.30	84.09	58.98	68.70	996	73.67
Lane2Seq-p [3]	ViT-Base	93.03	76.42	72.17	78.32	52.89	89.67	72.67	73.98	1319	78.39
BSNet [51]	ResNet34	93.75	78.01	76.65	79.55	54.69	90.72	73.99	75.28	1445	79.89
LaneLM-128*	ResNet18	92.29	73.53	66.14	75.11	44.34	88.15	62.37	68.85	1255	76.98
LaneLM-128(0-kp)	ResNet18	92.24	72.58	65.80	74.94	42.87	88.38	59.78	67.79	1216	73.66
LaneLM-128(1-kp)	ResNet18	92.36	73.04	66.07	75.06	43.69	88.42	61.36	68.30	1216	74.05
LaneLM-128(2-kp)	ResNet18	92.58	74.50	67.14	75.67	47.19	89.08	62.32	70.41	1216	75.46
LaneLM-128(4-kp)	ResNet18	92.87	75.88	69.40	78.55	49.03	89.68	71.69	72.02	1216	77.08
LaneLM-256*	ResNet34	92.33	73.77	69.57	74.12	45.01	88.69	65.53	69.22	1216	77.63
LaneLM-256(0-kp)	ResNet34	92.36	73.04	66.07	75.06	43.69	88.42	61.36	68.30	1224	74.05
LaneLM-256(1-kp)	ResNet34	92.25	74.19	69.18	74.85	44.77	87.91	66.03	69.23	1224	74.87
LaneLM-256(2-kp)	ResNet34	92.48	74.70	69.88	74.63	49.85	87.77	55.51	70.43	1112	75.78
LaneLM-256(4-kp)	ResNet34	93.05	78.56	74.16	77.96	**58.52**	89.48	60.20	75.26	1112	79.37
LaneLM-512*	DLA34	92.37	76.79	71.24	77.28	53.26	88.70	67.44	72.78	1135	78.74
LaneLM-512(0-kp)	DLA34	92.75	75.07	68.84	77.17	46.87	88.81	61.72	69.63	1102	75.54
LaneLM-512(1-kp)	DLA34	93.19	76.93	71.17	77.89	50.64	88.68	70.88	71.99	1102	77.80
LaneLM-512(2-kp)	DLA34	93.32	78.36	72.89	79.96	55.65	89.44	63.17	75.19	1155	79.04
LaneLM-512(4-kp)	DLA34	93.42	**81.73**	**79.26**	81.64	**68.34**	90.23	63.08	**79.51**	1155	**82.71**

 doi:10.20944/preprints202504.1582.v1

Table 4. Comparison of the performance of different models on LLAMAS.

Methods	Encoder	F1↑
PolyLaneNet [20]	EfficientNet-b0	90.20
BezierLaneNet [21]	ResNet34	96.11
LaneATT [8]	ResNet34	94.96
LaneATT [8]	ResNet122	95.17
LaneAF [11]	DLA34	96.90
CLRNet [6]	ResNet18	96.96
CLRNet [6]	DLA34	97.16
LaneLM-128*	ResNet18	96.24
LaneLM-256*	ResNet34	96.38
LaneLM-512*	DLA34	**97.25**

4.6. Ablation Study

We have maintained a minimalist design for the model. LaneLM has neither complex trick nor sophisticated structure. To investigate the impact of pure language modeling architectures on LD tasks, we did not design complex spatial attention mechanisms.

Table 5 illustrates ablation study on CULane and shows effects of each module and prompting strategy in our method, where cmd refers to the command token. We use LaneLM-512 (0-kp) with DLA34 encoder but without **HR** (Hallucination Removal), as our baseline to conduct our experiments. Experiments have proven that, compared with other methods which add the attention module to LD model, following the prompting protocol for inference can improve the accuracy of the model by a large margin in complex scenarios, demonstrating LaneLM's powerful interactive capability. Modules specifically designed to handle spatial information, such as FPN [40], can also significantly enhance the model's performance.

Table 5. Ablation study on CULane

FPN	LE	HR	1-kp	2-kp	4-kp	cmd	F1-score
-	-	-	-	-	-	-	68.36 (-2.35)
✓	-	-	-	-	-	-	69.24 (-1.47)
✓	✓	-	-	-	-	-	**70.71**(baseline)
✓	✓	✓	-	-	-	-	75.54 (**+4.83**)
✓	✓	✓	✓	-	-	-	77.80 (+2.26)
✓	✓	✓	-	✓	-	-	79.04 (+1.24)
✓	✓	✓	-	-	✓	-	82.71 (**+3.67**)
✓	✓	✓	-	-	✓	✓	83.75 (+1.04)

Additionally, to explore the instruction-following ability of LaneLM, we conduct extra ablation study on command token that sets the predictions to null, in which a cmd token is given at the beginning of the sequence when it comes to no lanes in the image. This instruction can reduce the post-processing of autonomous driving, enabling the visual model to receive semantic feedback from human being and other models (e.g. large language models).

Analysis on hallucination. Current large language models are still struggling with hallucination. Figure 6(a) shows hallucination in LaneLM. Eq. 10 endows the model with the capability of VQA but it makes it easier for the model to predict cyclic sequences. Figure 6(a) illustrates that the model has learned the abrupt change points that connecting L_q and L_{gt} on the side. LaneLM has learned the contextual representation of abrupt change points and consequently results in hallucination.

Study on hallucination removal. As shown in Figure 6(c), we collect the statistic results of lanes with hallucination from TuSimple, in which the red line demonstrates that a significant number of x-offsets with abnormal values have been filtered out by HR. As shown in Figure 6(d), to find the best

 doi:10.20944/preprints202504.1582.v1

percentile, we conducted a search for the performance of θ on the F1 and Acc metrics on TuSimple using LaneLM-128 (0-kp). We report the detailed metric values in Table 6.

Study on n_{bins} per pixel. To investigate the impact of the resolution of bins (i.e., n_{bins} per pixel) on performance, we vary the n_{bins} and conduct our training on TuSimple using LaneLM-512 (0-kp). We report our result in Table 7. Most LD methods resize images into 800 pixel width in training and inference. n_{bins}=800 aligns with our intuition. Using large n_{bins}, which increases the vocabulary size, has no significant improvement and it will slow down training. Choosing a larger n_{bins} does not yield benefits because the model's hidden size is too small relative to n_{bins}, making classification difficult.

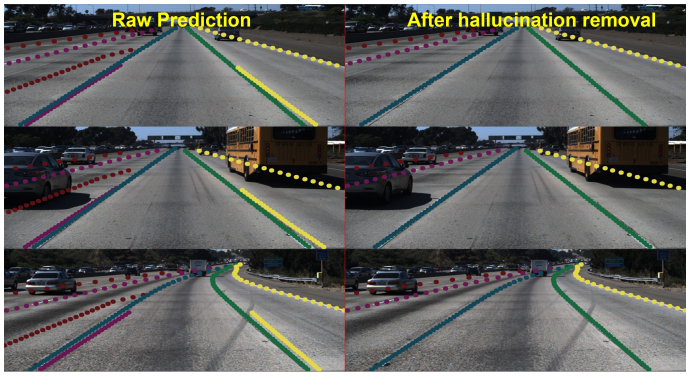
Table 6. Experimental studies on θ-percentile.

θ	0	5	10	20	30	50	70	80	**85**	90
F1	55.96	82.93	88.07	94.26	91.17	92.02	95.51	95.89	**96.41**	96.11
Acc	79.49	89.98	92.88	90.33	94.7	95.11	96.10	96.27	**96.33**	96.32

Table 7. Experimental studies on bins per pixel.

n_{bins} / image_width	F1↑
512 / 1280	96.12
800 / 1280	**97.64**
1200 / 1280	97.55
1600 / 1280	97.01

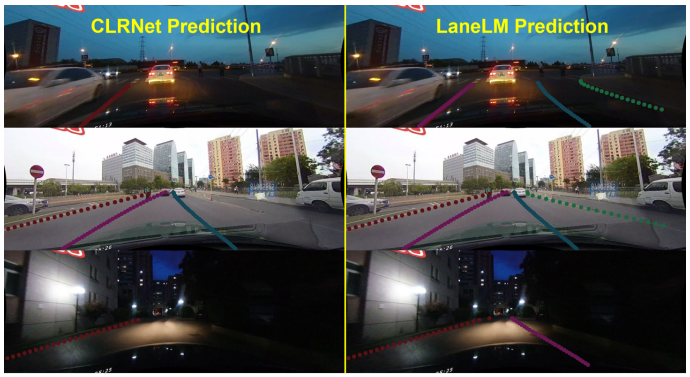
Qualitative results. We display the qualitative results in Figure 6(a-b). After keypoint prompting (2-kp), LaneLM can recall more lanes at night. The results show that LaneLM can effectively detect lanes in the multi-lane scenario (see Figure 6(a) right). The results demonstrate that LaneLM can be prompted from the semantic feedback. Semantic information directly determines the spatial attention of the model.



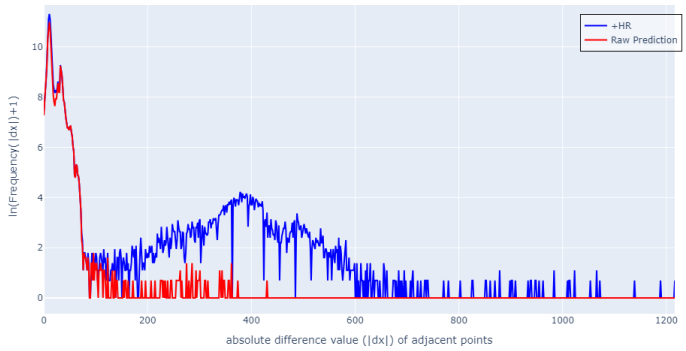
(**a**)

Figure 6. *Cont.*

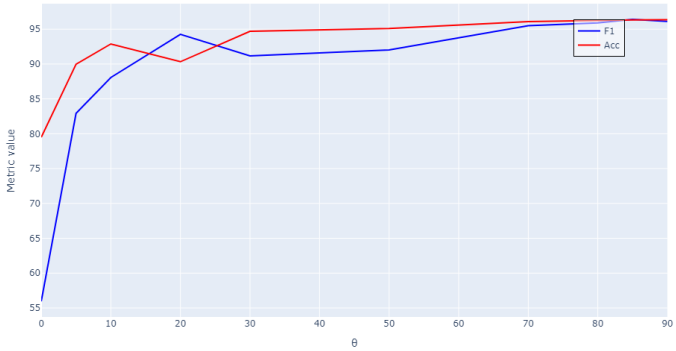
 doi:10.20944/preprints202504.1582.v1



(**b**)



(**c**)



(**d**)

Figure 6. (**a**) The left image shows the hallucination on the side lane, while the right image is post-processed with the HR algorithm. (**b**) The left image shows the CLRNet prediction and the right is LaneLM prediction. (**c**) shows the frequency distribution of horizontal distances between adjacent keypoints, with the x-axis as the ln(frequency) and the horizontal axis as the distance values. (**d**) shows the impact of different θ-percentile thresholds on F1-score and accuracy.

5. Discussion

Current autonomous driving technologies mainly focus on the fusion of visual sensors like LiDAR [52–54], radar, and cameras. Visual modality often fail in corner cases. Many recent works [55–57] has concentrated on Large Language Models (LLMs) to address the long-tail distribution in corner cases. However, they cannot interact with the visual models. Future autonomous vehicles will incorporate multiple redundant sensors for visual perception. In harsh conditions where sensors fail, a parallel LLM will leverage semantic information to prompt and enhance visual tasks, making the system more robust.

 doi:10.20944/preprints202504.1582.v1

6. Conclusions

In this paper, we present autoregressive LaneLM and treat the LD task as visual question answering. We use the language model to make LD task promptable and interactive. The lanes and images are transformed into multi-modal embedding sequences and interact with each other in LaneLM. Then we alleviate the hallucination by HR. Experiments show our proposed method outperforms previous unpromptable SOTA methods. The design of LaneLM is quite simple and doesn't have overly complicated structures. We hope that LaneLM can serve as a baseline for language-modeling-based approaches. We believe that our work will pave the way towards unification for LLM and visual models.

Author Contributions: Conceptualization Y.Z.; methodology Y.Z.; software, Y.Z.; formal analysis, Y.Z.; resources, Y.Z.; writing—original draft, Y.Z.; writing—review & editing, X.B. and Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (No. 62472349).

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Data associated with this research is available in publicly accessible repository. The associated dataset can be found in the following urls: 1. TuSimple: https://github.com/TuSimple/tusimple-benchmark/issues/3 2. CULane: https://xingangpan.github.io/projects/CULane.html 3. LLAMAS: https://unsupervised-llamas.com/llamas/. Source code will be available on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LD	Lane Detection
VQA	Visual Question Answering
LLM	Large Language Model
HR	Hallucination Removal
SAM	Segment Anything Model
ADAS	Advanced Driver Assistance Systems
CNN	Convolutional Neural Network
NMS	Non-Maximum Suppression
AR	Autoregressive
NLP	Natural Language Processing
SOTA	State-Of-The-Art
FPN	Feature Pyramid Networks
DCN	Deformable Convolutional Network

Appendix A

Appendix A.1

Theorem A1. *cross-attention mechanism does not leak the information of the future query tokens.*

Proof. Let's revisit the cross-attention function:

$$\text{CrossAtt}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{A1}$$

Obviously, the softmax function preserves the order information of the query embeddings and does not leak information about future embeddings. Let n,m,D and $\tilde{\mathbf{q}}^{1\times D}$ be the sequence length of the query tokens, the value tokens, the hidden size and the embedding of the query token, respectively.

 doi:10.20944/preprints202504.1582.v1

According to the principle of matrix block calculation, the information interaction in cross-attention is equivalent to:

$$Q^{n\times D}(K^{m\times D})^T V^{m\times D} = \begin{pmatrix} \tilde{\mathbf{q}}_1^{1\times D} \\ \tilde{\mathbf{q}}_2^{1\times D} \\ \dots \\ \tilde{\mathbf{q}}_n^{1\times D} \end{pmatrix} (K^T V)^{D\times D} = \begin{pmatrix} \tilde{\mathbf{q}}_1 (K^T V) \\ \tilde{\mathbf{q}}_2 (K^T V) \\ \dots \\ \tilde{\mathbf{q}}_n (K^T V) \end{pmatrix} \tag{A2}$$

Therefore, cross-attention preserves the order information of the query tokens, preventing any information leakage and every query embedding is conditioned on the whole visual information $K^T V$. □

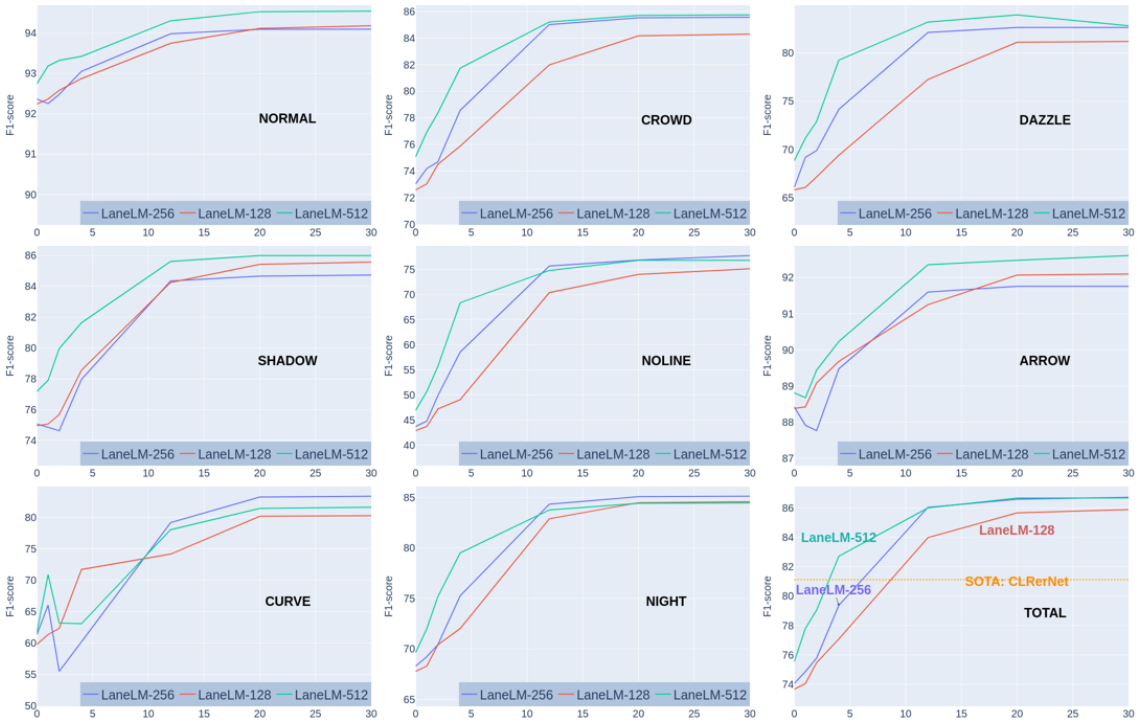


Figure A1. F1@50 in 9 categories on CULane for keypoint prompting. LaneLM reaches the state of the art on CULane [45] with keypoint prompting, significantly outperforming unpromptable SOTA method [9] with as few as 4 keypoint prompts at the beginning of each lane. We found that in corner cases, the marginal benefit of prompting is quiet large.

References

1. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A.C.; Lo, W.Y.; et al. Segment anything. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4015–4026.
2. Xu, N.; Price, B.; Cohen, S.; Yang, J.; Huang, T.S. Deep interactive object selection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 373–381.
3. Zhou, K. Lane2Seq: Towards Unified Lane Detection via Sequence Generation. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 16944–16953.
4. Chen, T.; Saxena, S.; Li, L.; Fleet, D.J.; Hinton, G. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852* **2021**.
5. Brown, T.B. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* **2020**.
6. Zheng, T.; Huang, Y.; Liu, Y.; Tang, W.; Yang, Z.; Cai, D.; He, X. Clrnet: Cross layer refinement network for lane detection. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 898–907.
7. Li, X.; Li, J.; Hu, X.; Yang, J. Line-cnn: End-to-end traffic line detection with line proposal unit. *IEEE Transactions on Intelligent Transportation Systems* **2019**, *21*, 248–258.

 doi:10.20944/preprints202504.1582.v1

8. Tabelini, L.; Berriel, R.; Paixao, T.M.; Badue, C.; De Souza, A.F.; Oliveira-Santos, T. Keep your eyes on the lane: Real-time attention-guided lane detection. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 294–302.
9. Honda, H.; Uchida, Y. CLRerNet: Improving Confidence of Lane Detection with LaneIoU. In Proceedings of the 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024, pp. 1165–1174. https://doi.org/10.1109/WACV57701.2024.00121.
10. Lee, S.; Kim, J.; Shin Yoon, J.; Shin, S.; Bailo, O.; Kim, N.; Lee, T.H.; Seok Hong, H.; Han, S.H.; So Kweon, I. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2017, pp. 1947–1955.
11. Abualsaud, H.; Liu, S.; Lu, D.; Situ, K.; Rangesh, A.; Trivedi, M.M. LaneAF: Robust Multi-Lane Detection With Affinity Fields. *IEEE Robotics and Automation Letters* **2021**, *6*, 7477–7484.
12. Zhang, X.; Li, Z.; Gao, X.; Jin, D.; Li, J. Channel Attention in LiDAR-camera Fusion for Lane Line Segmentation. *Pattern Recognition* **2021**, *118*, 108020. https://doi.org/https://doi.org/10.1016/j.patcog.2021.108020.
13. Wang, J.; Ma, Y.; Huang, S.; Hui, T.; Wang, F.; Qian, C.; Zhang, T. A keypoint-based global association network for lane detection. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 1392–1401.
14. Qu, Z.; Jin, H.; Zhou, Y.; Yang, Z.; Zhang, W. Focus on local: Detecting lane marker from bottom up via key point. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 14122–14130.
15. Xu, S.; Cai, X.; Zhao, B.; Zhang, L.; Xu, H.; Fu, Y.; Xue, X. Rclane: Relay chain prediction for lane detection. In Proceedings of the European Conference on Computer Vision. Springer, 2022, pp. 461–477.
16. Liu, L.; Chen, X.; Zhu, S.; Tan, P. Condlanenet: a top-to-down lane detection framework based on conditional convolution. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 3773–3782.
17. Qin, Z.; Wang, H.; Li, X. Ultra fast structure-aware deep lane detection. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16. Springer, 2020, pp. 276–291.
18. Qin, Z.; Zhang, P.; Li, X. Ultra fast deep lane detection with hybrid anchor driven ordinal classification. *IEEE transactions on pattern analysis and machine intelligence* **2022**, *46*, 2555–2568.
19. Chen, Z.; Liu, Y.; Gong, M.; Du, B.; Qian, G.; Smith-Miles, K. Generating dynamic kernels via transformers for lane detection. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 6835–6844.
20. Tabelini, L.; Berriel, R.; Paixao, T.M.; Badue, C.; De Souza, A.F.; Oliveira-Santos, T. Polylanenet: Lane estimation via deep polynomial regression. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021, pp. 6150–6156.
21. Feng, Z.; Guo, S.; Tan, X.; Xu, K.; Wang, M.; Ma, L. Rethinking efficient lane detection via curve modeling. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 17062–17070.
22. Liu, R.; Yuan, Z.; Liu, T.; Xiong, Z. End-to-end lane shape prediction with transformers. In Proceedings of the Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2021, pp. 3694–3702.
23. Niu, J.; Lu, J.; Xu, M.; Lv, P.; Zhao, X. Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting. *Pattern Recognition* **2016**, *59*, 225–233. Compositional Models and Structured Learning for Visual Recognition, https://doi.org/https://doi.org/10.1016/j.patcog.2015.12.010.
24. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2017, pp. 764–773.
25. Seff, A.; Cera, B.; Chen, D.; Ng, M.; Zhou, A.; Nayakanti, N.; Refaat, K.S.; Al-Rfou, R.; Sapp, B. Motionlm: Multi-agent motion forecasting as language modeling. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 8579–8590.
26. Jia, X.; Shi, S.; Chen, Z.; Jiang, L.; Liao, W.; He, T.; Yan, J. AMP: Autoregressive Motion Prediction Revisited with Next Token Prediction for Autonomous Driving. *arXiv preprint arXiv:2403.13331* **2024**.
27. Chen, S.; Jiang, B.; Gao, H.; Liao, B.; Xu, Q.; Zhang, Q.; Huang, C.; Liu, W.; Wang, X. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243* **2024**.

 doi:10.20944/preprints202504.1582.v1

28. Alayrac, J.B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems* **2022**, *35*, 23716–23736.
29. Liu, H.; Li, C.; Wu, Q.; Lee, Y.J. Visual instruction tuning. *Advances in neural information processing systems* **2024**, *36*.
30. Wei, X.; Bai, Y.; Zheng, Y.; Shi, D.; Gong, Y. Autoregressive visual tracking. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 9697–9706.
31. Bai, Y.; Zhao, Z.; Gong, Y.; Wei, X. Artrackv2: Prompting autoregressive tracker where to look and how to describe. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 19048–19057.
32. Li, J.; Li, D.; Savarese, S.; Hoi, S. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In Proceedings of the Proceedings of the 40th International Conference on Machine Learning; Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; Scarlett, J., Eds. PMLR, 23–29 Jul 2023, Vol. 202, *Proceedings of Machine Learning Research*, pp. 19730–19742.
33. Li, J.; Li, D.; Xiong, C.; Hoi, S. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In Proceedings of the Proceedings of the 39th International Conference on Machine Learning; Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; Sabato, S., Eds. PMLR, 17–23 Jul 2022, Vol. 162, *Proceedings of Machine Learning Research*, pp. 12888–12900.
34. Wang, C.; Yang, J.; Zhou, Y.; Yue, X. CooKie: commonsense knowledge-guided mixture-of-experts framework for fine-grained visual question answering. *Information Sciences* **2025**, *695*, 121742. https://doi.org/https://doi.org/10.1016/j.ins.2024.121742.
35. Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X.V.; et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* **2022**.
36. Boukhers, Z.; Hartmann, T.; Jürjens, J. COIN: Counterfactual Image Generation for Visual Question Answering Interpretation. *Sensors* **2022**, *22*. https://doi.org/10.3390/s22062245.
37. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* **2020**, *21*, 1–67.
38. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
39. Yu, F.; Wang, D.; Shelhamer, E.; Darrell, T. Deep layer aggregation. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2403–2412.
40. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.
41. Xiao, Z.; Li, Z.; Li, H.; Li, M.; Liu, X.; Kong, Y. Multi-Scale Feature Fusion Enhancement for Underwater Object Detection. *Sensors* **2024**, *24*. https://doi.org/10.3390/s24227201.
42. Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* **2020**.
43. Chen, Z.; Duan, Y.; Wang, W.; He, J.; Lu, T.; Dai, J.; Qiao, Y. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534* **2022**.
44. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ArXiv* **2023**, *abs/2311.05232*.
45. Pan, X.; Shi, J.; Luo, P.; Wang, X.; Tang, X. Spatial As Deep: Spatial CNN for Traffic Scene Understanding. In Proceedings of the AAAI Conference on Artificial Intelligence, 2017.
46. TuSimple: TuSimple benchmark, 2019. https://github.com/TuSimple/tusimple-benchmark.
47. Behrendt, K.; Soussan, R. Unsupervised labeled lane markers using maps. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision workshops, 2019, pp. 0–0.
48. Yang, J.; Zhang, L.; Lu, H. Lane Detection with Versatile AtrousFormer and Local Semantic Guidance. *Pattern Recognition* **2023**, *133*, 109053. https://doi.org/https://doi.org/10.1016/j.patcog.2022.109053.
49. Han, J.; Deng, X.; Cai, X.; Yang, Z.; Xu, H.; Xu, C.; Liang, X. Laneformer: Object-aware row-column transformers for lane detection. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2022, Vol. 36, pp. 799–807.

 doi:10.20944/preprints202504.1582.v1

50. Xiao, L.; Li, X.; Yang, S.; Yang, W. ADNet: Lane Shape Prediction via Anchor Decomposition. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023, pp. 6404–6413.
51. Chen, H.; Wang, M.; Liu, Y. Bsnet: Lane detection via draw b-spline curves nearby. *arXiv preprint arXiv:2301.06910* **2023**.
52. Park, G.; Koh, J.; Kim, J.; Moon, J.; Choi, J.W. LiDAR-Based 3D Temporal Object Detection via Motion-Aware LiDAR Feature Fusion. *Sensors* **2024**, *24*. https://doi.org/10.3390/s24144667.
53. Huang, F.; Liu, S.; Zhang, G.; Hao, B.; Xiang, Y.; Yuan, K. DeployFusion: A Deployable Monocular 3D Object Detection with Multi-Sensor Information Fusion in BEV for Edge Devices. *Sensors* **2024**, *24*. https://doi.org/10.3390/s24217007.
54. Alfeqy, L.; Abdelmunim, H.E.H.; Maged, S.A.; Emad, D. Kalman Filter-Based Fusion of LiDAR and Camera Data in Bird's Eye View for Multi-Object Tracking in Autonomous Vehicles. *Sensors* **2024**, *24*. https://doi.org/10.3390/s24237718.
55. Xu, Z.; Zhang, Y.; Xie, E.; Zhao, Z.; Guo, Y.; Wong, K.Y.K.; Li, Z.; Zhao, H. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters* **2024**.
56. Marcu, A.M.; Chen, L.; Hünermann, J.; Karnsund, A.; Hanotte, B.; Chidananda, P.; Nair, S.; Badrinarayanan, V.; Kendall, A.; Shotton, J.; et al. LingoQA: Visual question answering for autonomous driving. In Proceedings of the European Conference on Computer Vision. Springer, 2024, pp. 252–269.
57. Cui, C.; Ma, Y.; Cao, X.; Ye, W.; Zhou, Y.; Liang, K.; Chen, J.; Lu, J.; Yang, Z.; Liao, K.D.; et al. A survey on multimodal large language models for autonomous driving. In Proceedings of the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 958–979.