2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Bridging Perception and Planning: Towards End-to-End Planning for Signal Temporal Logic Tasks

Bowen Ye, Junyue Huang, Yang Liu, Xiaozhen Qiao and Xiang Yin

***Abstract*— We investigate the task and motion planning problem for Signal Temporal Logic (STL) specifications in robotics. Existing STL methods rely on pre-defined maps or mobility representations, which are ineffective in unstructured real-world environments. We propose the *Structured-MoE STL Planner* (S-MSP), a differentiable framework that maps synchronized multi-view camera observations and an STL specification directly to a feasible trajectory. S-MSP integrates STL constraints within a unified pipeline, trained with a composite loss that combines trajectory reconstruction and STL robustness. A *structure-aware* Mixture-of-Experts (MoE) model enables horizon-aware specialization by projecting sub-tasks into temporally anchored embeddings. We evaluate S-MSP using a high-fidelity simulation of factory-logistics scenarios with temporally constrained tasks. Experiments show that S-MSP outperforms single-expert baselines in STL satisfaction and trajectory feasibility. A rule-based *safety filter* at inference improves physical executability without compromising logical correctness, showcasing the practicality of the approach.**

I. Introduction

Signal Temporal Logic (STL) has emerged as a widely-used specification language in the design of autonomous systems, primarily due to its quantitative robustness semantics and the wealth of temporal operators it offers. It has been successfully applied across a broad range of engineering applications [1], [31], [33], [34]. Particularly, in the field of robotics motion and task planning, STL has garnered significant attention [7], [8], [26], [32].

Despite its success in robotic task and motion planning, current STL methods remain largely restricted to scenarios where structured system information is available *a priori*. This structured information typically includes: (i) environmental semantics (e.g., atomic predicates), (ii) the abstracted workspace for discrete-level planning (commonly represented as grids or polytopic regions), and (iii) system dynamics when planning over both tasks and motions. Most existing STL planning approaches, including encoding-based [24], [25], gradient-based [9], [17], [21] and learning-based optimization methods [5], [10], [19], [27], [28], fall into this category, as they all fundamentally rely on predefined structural information at some level.

Current methods typically separate environmental perception, structured information modeling, and STL task planning into distinct stages. Consequently, most efforts emphasize planning over pre-structured abstractions, while the influence of perceptual uncertainty and abstraction quality from unstructured observations is comparatively underexplored. In practice, structured information must be inferred from nonlinear system dynamics and high-dimensional sensory inputs (e.g., images), yet conventional pipelines do not support direct reasoning over such raw data, leading to brittle performance under distribution shift or missing map annotations. An end-to-end framework that learns to satisfy STL specifications directly from perceptual inputs can reduce manual discretization and better couple perception with decision-making, consistent with recent successes of end-to-end learning in robotics.

Building on this perspective, recent studies in autonomous driving [11], [13] and robotic manipulation empowered by vision–language–action (VLA) models [2], [35] show that end-to-end learning paradigms are advantageous in unifying representation and decision-making. Inspired by these findings, we believe that such unification could significantly enhance STL-constrained planning. However, solving STL tasks *directly* from raw visual observations remains an unexplored challenge. Existing methods still rely on explicit map abstractions or low-dimensional state inputs. This gap motivates our pursuit of a truly end-to-end framework that integrates perception, temporal logic reasoning, and trajectory generation into a single differentiable architecture.

In this work, we propose an end-to-end autoregressive transformer framework that ingests synchronized multi-view camera streams along with an STL specification, and outputs a physically feasible trajectory guaranteed to satisfy the prescribed temporal-logic constraints. However, two key challenges impede the realization of such an end-to-end architecture. First is the issue of dataset unavailability. To our knowledge, no public corpus offers synchronized multi-view imagery, formal STL task specifications, and expert trajectories simultaneously, complicating supervised training and objective benchmarking. Second is the complexity of spatio-temporal learning. Specifically, the rich temporal operators and spatial predicates of STL create a highly non-convex, long-horizon optimization landscape, making gradient-based learning more demanding than in conventional trajectory-prediction settings. To address these challenges, our work makes the following contributions:

This work was supported by the National Science and Technology Major Project (2025ZD1600701-5) and the National Natural Science Foundation of China (62573291,62533017,62173226).

B. Ye, J. Huang and X. Yin are with the School of Automation & Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {yebowen1025, hjy-564904993, yinxiang}@sjtu.edu.cn. Y. Liu is with the University of Minnesota, Twin Cities. E-mail: {liu03222}@umn.edu. X. Qiao is with the School of Information Science and Technology, University of Science and Technology of China, Hefei, 230026, China. E-mail:{xiaozhennnqiao}@mail.ustc.edu.cn

- **End-to-End STL Planner.** We present the first baseline that performs STL-constrained trajectory synthesis *directly* from synchronized multi-view camera observations. A single autoregressive Transformer generates a physically feasible trajectory satisfying any given STL specification, without requiring intermediate map abstractions.
- **Structure-Aware Mixture of Experts.** We introduce a structure-aware MoE model that decomposes each STL formula into temporally anchored sub-task tokens and routes them to operator-specific experts. This facilitates efficient end-to-end learning of STL-compliant trajectories and enhances logical reasoning by exploiting STL compositionality.
- **Benchmark Dataset and Empirical Validation.** We construct a high-fidelity, Gazebo-based benchmark that combines temporally synchronized multi-view camera streams with formally annotated STL specifications and expert trajectories across a range of factory-logistics scenarios. This is the first publicly available corpus designed for supervised, end-to-end STL learning. We then demonstrate that the proposed model achieves state-of-the-art STL satisfaction and trajectory feasibility, with improved performance and without incurring additional planning latency compared to a single-expert transformer baseline.

II. Related Works

End-to-End Model Paradigm. UniAD [11] pioneered the end-to-end framework in autonomous driving, unifying multiple perception tasks into a single architecture to directly support downstream planning, enhancing planning quality and efficiency. End-to-end designs have since been explored in autonomous driving and robotic manipulation. In autonomous driving, multi-task BEV/Vector transformers and closed-loop differentiable planners unify perception, prediction, and planning [14], [22], [30], while diffusion-based trajectory generators produce multimodal futures and safety-guided sampling [15], [18]. In manipulation, diffusion policies in joint or SE(3) spaces demonstrate strong long-horizon performance [4], and VLA models align semantics with control via instruction grounding and policy distillation [3], [16]. These advancements highlight the advantages of end-to-end approaches, such as better coupling of representation and decision-making, uncertainty-aware generative modeling, and language-conditioned grounding. Building on this, we introduce the first end-to-end solution for solving STL-specified tasks using raw sensory inputs and an STL specification directly.

Mixture-of-Experts Models. Sparse MoE improves model capacity by replacing dense layers with a bank of experts, activated conditionally via a learned router, enabling task specialization and computational efficiency. In large language models, MoE reduces perplexity and improves multitask accuracy without increasing inference costs [6]. In autonomous driving, expert routing disentangles maneuver styles and scenario primitives [29], while in robotics, MoE handles task heterogeneity and long-tail data [12]. Conditional routing directs rare or context-specific tasks to specialized experts, enhancing performance, data efficiency, and generalization. Motivated by these findings, we integrate a structure-aware MoE into our model, adapting routing and expert designs to align with the compositional and temporal nature of STL tasks.

III. Signal Temporal Logic Specifications

Let X denote the state space and let $\mathbf{x}_{0:T} = (x_0, \dots, x_T) \in X^T$ be a finite trajectory. The syntax of Signal Temporal Logic (STL) is given by

$$\phi ::= \top \mid \pi^\mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathbf{U}_{[a,b]} \phi_2 \tag{1}$$

where $\top$ is the true predict, π^μ is a predicate whose truth value is determined by the sign of its underlying predicate function $\mu : \mathbb{R}^n \to \mathbb{R}$ and is true if $\mu(x_k) \geq 0$; otherwise it is false. Notation $\neg$ and $\wedge$ are the standard Boolean operators "negation" and "conjunction", respectively, which can further induce "disjunction" $\vee$ and "implication" $\rightarrow$. Notation $\mathbf{U}_{[a,b]}$ is the temporal operator "until", where $a, b \in \mathbb{R}_{\geq 0}$ are the time instants. STL formulae are evaluated on state sequences. We use $(\mathbf{x}, t) \models \phi$ to denote that the sequence $\mathbf{x}$ satisfies STL formula ϕ at instant t. The semantics of STL are given by: $(\mathbf{x}, t) \models \phi_1 \mathbf{U} \phi_2$ iff $\exists t' \in [t+a, t+b] : (\mathbf{x}, t') \models \phi_2, \forall t'' \in [t+a, t'] : (\mathbf{x}, t'') \models \phi_1$. The reader is referred to [20] for more details on the semantics of STL. Furthermore, we also induce two important temporal operators "eventually" and "always" by $\mathbf{F}_{[a,b]}\phi := \top \mathbf{U}_{[a,b]} \phi$ and $\mathbf{G}_{[a,b]} := \neg \mathbf{F}_{[a,b]} \neg\phi$, respectively. For a given sequence $\mathbf{x}$, we write $\mathbf{x} \models \phi$ whenever $(\mathbf{x}, 0) \models \phi$.

Apart from the syntax and semantics of STL, we also briefly review the quantitative semantics of STL as follows. Given a signal $\mathbf{x}$ and time instant t, the robustness semantics of $\mathbf{x}$ at t is recursively defined as follows: $\rho^\mu(\mathbf{x}, t) = \mu(x_t)$; $\rho^{\neg\mu}(\mathbf{x}, t) = -\mu(x_t)$; $\rho^{\phi_1 \wedge \phi_2} = \min(\rho^{\phi_1}(\mathbf{x}, t), \rho^{\phi_2}(\mathbf{x}, t))$; $\rho^{\mathbf{G}_{[a,b]}\phi}(\mathbf{x}, t) = \min_{t' \in [t+a, t+b]} \left(\rho^\phi(\mathbf{x}, t')\right)$; $\rho^{\phi_1 \mathbf{U} \phi_2}(\mathbf{x}, t) = \max_{t' \in [t+a, t+b]} \left(\min(\rho^{\phi_2}(\mathbf{x}, t')), \min_{t'' \in [t, t']} \rho^{\phi_1}(\mathbf{x}, t'')\right)$. Naturally, for a signal $\mathbf{x}$, we have $\mathbf{x} \models \phi \Leftrightarrow \rho^\phi(\mathbf{x}, 0) > 0$.

Following [23], we adopt a smooth surrogate of the robustness to enable end-to-end training. Because $\min / \max$ and $\inf / \sup$ introduce non-differentiable truncations and flat regions, direct backpropagation through ρ^ϕ is unstable. We therefore replace $\max$ (sup) and $\min$ (inf) operators with temperature-controlled smooth counterparts $\max / \min$:

$$\widetilde{\max}_k(x_1, x_2, \dots) := \frac{1}{k} \log(e^{kx_1} + e^{kx_2} + \dots) \tag{2a}$$

$$\widetilde{\min}_k(x_1, x_2, \dots) := -\widetilde{\max}_k(-x_1, -x_2, \dots) \tag{2b}$$

where k is a scaling factor for this approximation. If $k \to \infty$, the operator $\widetilde{\max} = \max$ and similarly $\widetilde{\min} = \min$. We use $k = 300$ in our training, the ablation study on k is in Table II.

As the general STL is highly expressive, its hypothesis class can be overly rich relative to the available data, which makes it difficult for a learning-based end-to-end model to recover precise semantics and yields unstable gradients during training. To ensure learnability and tractable monitoring,

we restrict attention to a widely used finite-horizon fragment that still captures the majority of robotic tasks. Specifically, we adopt a discrete-time semantics with horizon $T \in \mathbb{N}$ and timestamps $t \in \{0, 1, \ldots, T\}$. All temporal operators are *bounded* in time, with integer bounds $0 \leq a \leq b \leq T$. We further limit temporal nesting depth to 1: each temporal operator applies directly to a temporal-free Boolean formula. Formally, let atomic predicates be $\mu(x_t)$ with robustness $\rho_\mu[t] \in \mathbb{R}$, and let χ denote any Boolean combination of atoms using $\wedge, \vee$ (no temporal operators inside). Our fragment $\mathsf{STL}_{\mathrm{B,depth}\leq 1}$ is generated by

$$\varphi \ ::= \ \chi \ \mid \ \mathbf{F}_{[a,b]}\, \chi \ \mid \ \mathbf{G}_{[a,b]}\, \chi \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \varphi_1 \vee \varphi_2$$

This fragment covers canonical reach-avoid, liveness ($\mathbf{F}$), and safety ($\mathbf{G}$) specifications frequently used in robotics, while keeping monitoring and gradient-based training tractable.

Remark 1: We temporarily exclude "implication" ($\rightarrow$), "negation" ($\neg$), and the "until" operator ($\mathbf{U}$). Negation satisfies $\rho_{\neg\varphi}[t] = -\rho_\varphi[t]$, which flips the objective's sign and amplifies gradient-direction uncertainty under compositional nesting; implication reduces to $\neg\varphi \vee \psi$ and inherits the same pathology. For Until, the discrete-time robustness takes a nested min–max form:

$$\rho_{\varphi\, \mathbf{U}_{[a,b]}\, \psi}[t] = \max_{\tau \in \{a,\ldots,b\}} \Big(\min\Big(\rho_\psi[t{+}\tau], \min_{s \in \{0,\ldots,\tau-1\}} \rho_\varphi[t{+}s] \Big) \Big)$$

which introduces deep non-smooth min–max structure and strong nonlocal temporal coupling, yielding ill-conditioned objectives and highly irregular (Clarke) subgradients that impede stable learning.

IV. Our Scenario and Problem Formulation

Our overall objective is to design an end-to-end STL planner whose inputs are:

- raw multi-view camera data of the workspace, without any preprocessing such as semantic recognition or region discretization; and
- a structural STL formula that the agent needs to fulfill.

The output is a feasible trajectory for the ego-agent, ensuring that it both satisfies the STL task and is feasible for the agent's dynamics. The trajectory is represented as a sequence of waypoints $\tau = (x_t, y_t, \psi_t)_{t=1}^{T_f}$, where T_f is the planning horizon, and (x_t, y_t, ψ_t) denotes the position and heading of the agent at time t, expressed in a fixed world (absolute) reference frame.

During the training phase, we assume the availability of a dataset containing task-related trajectories with raw multi-view environment RGB data. Specifically, each data entry includes synchronized multi-view RGB streams with calibrated intrinsics and extrinsics, an STL formula, and an expert reference trajectory. Particularly, we make the following assumptions for training data and execution environment

- **Consistent environmental semantics:** The semantic information of the environment remains unchanged, and the training data includes all possible image semantics that the agent may encounter during execution.
- **Labeled target areas:** To enable the model to learn the correspondence between target regions and STL task specifications from data, we mark different regions with distinct colors. Note that the model input consists only of multi-view concentrated image observations, without any explicit geometric annotations (e.g., region coordinates or masks).
- **Consistent system dynamics:** The dynamics of the system in both the training and execution environments are assumed to be the same.
- **Complete multi-view coverage:** The multi-view camera setup is assumed to fully cover the operating scene, with no blind spots. For example, in autonomous driving, this could be achieved using cameras mounted at various points around the vehicle. In our factory scenario example, multiple cameras suspended from different positions on the ceiling may achieve full coverage.

Furthermore, without loss of generality, we assume that the training and execution environments share the same physical dimensions. However, the positions of task zones and obstacle regions are allowed to vary, thereby generating diverse scenarios.

V. E2E Structural MoE Model

The overall pipeline of **S-MSP** is shown in Fig 1, which is a Transformer-based, end-to-end planner that maps multi bird-eye-view images and a Signal Temporal Logic (STL) task to a future waypoint sequence. The **S-MSP** model fuses four conditioning pathways: vision, STL text, pose/region context, and a *structural* encoding of the STL formula and decodes trajectories with a Hierarchical Key-Value (HKV) Mixture-of-Experts (MoE) Transformer. Formally,

$$f_\theta : \mathcal{I} \times \varphi \times \mathbf{p}_0 \times \mathcal{R} \ \mapsto \ \hat{\mathbf{Y}}_{1:T}; \hat{\mathbf{y}}_t = (x_t, y_t, \psi_t) \qquad (3)$$

where $\mathcal{I}$ is the multi-view concatenated image tensor, φ is the STL specification, $\mathbf{p}_0 = (x_0, y_0, \psi_0)$ is the starting pose, and $\mathcal{R}$ is the region bounds.

Components. The **S-MSP** architecture comprises: (i) an image encoder that yields a compact token memory; (ii) a text encoder over a semantic rendering of φ; (iii) a pose–region encoder; (iv) a structural STL encoder that parses φ into typed tokens aligned with time; and (v) an HKV–MoE trajectory decoder that autoregressively predicts waypoints under multi-source cross-attention.

Baseline. The baseline we proposed is a plain, non-MoE Transformer, depicted in Fig. 1 by replacing the red-boxed module with a standard Transformer block. It shares the same I/O interface as our final model: it ingests raw multi-view RGB images and STL task specifications and outputs dynamically feasible trajectories. But it contains no Mixture-of-Experts modules and no STL-conditioned routing

A. Encoders

1) Image Encoder: We adopt a multi-view image encoder initialized from a pretrained backbone and fine-tuned end-to-end in our pipeline. Given a RGB concatenated image

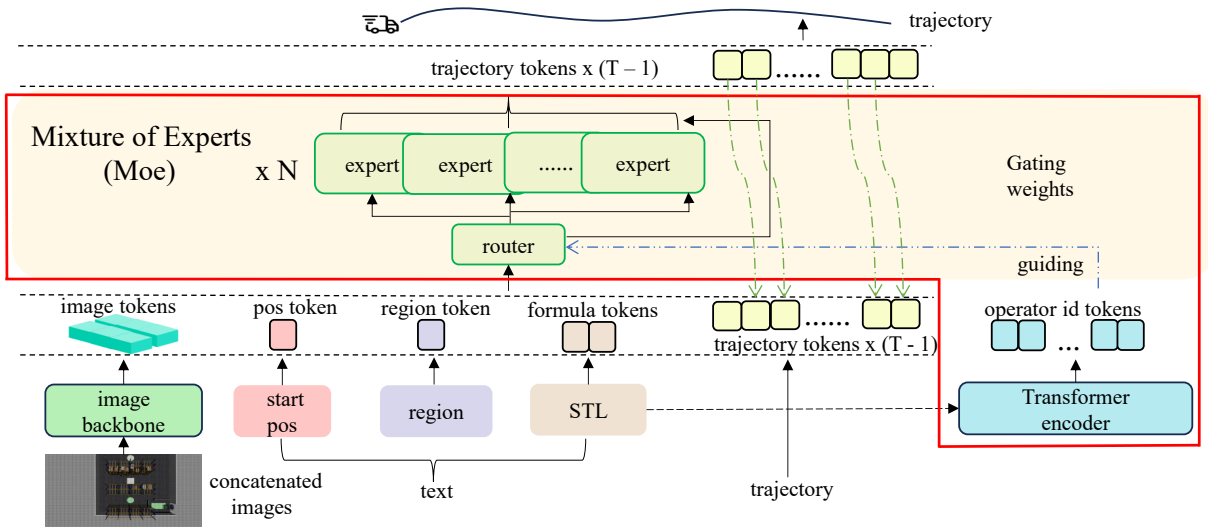


Fig. 1: The Overall Architecture of Model.

tensor $\mathcal{I} \in \mathbb{R}^{3\times H\times W}$, the encoder outputs a compact token sequence:

$$\mathbf{V} = \text{ImageEncoder}(\mathcal{I}) \in \mathbb{R}^{N_v\times d}, \tag{4}$$

where d is the embedding dimension and N_v is the post-aggregation token count.

Concretely, we start from a Swin-B backbone (IN22K-ft-1K), take image features from stages 2–4 and align them to a unified resolution (`out_indices`=(1,2,3), `unify_level`=2). Two cross-attention aggregation layers with learnable latent tokens (512 queries) compress the fused pyramid and cap the key–value memory to ∼4K tokens for efficient decoder cross-attention. We add 2D spatial and temporal positional encodings (with optional coordinate channels) before passing $\mathbf{V}$ to the decoder.

2) STL Text Encoder: We encode the formatted STL string into a vector $\mathbf{l} \in \mathbb{R}^d$ and map it to

$$\mathbf{L} = \text{Proj}(\mathbf{l}) \in \mathbb{R}^{m_{\text{text}}\times d}, \tag{5}$$

where m_{text} is the number of text-conditioning tokens (we use $m_{\text{text}} = 1$ by default). $\mathbf{L}$ is appended to the decoder's cross-attention memory.

3) Pose/Region Encoder: Given the start pose $\mathbf{p}_0 = (x_0, y_0, \psi_0)$ and the rectangular feasible task region $\mathcal{R} = [x_{\min}, y_{\min}, x_{\max}, y_{\max}]$, the encoder produces

$$\mathbf{c_1}, \mathbf{c_2} = E^{\text{ctx}}_{\text{pose/reg}}(\mathbf{p}_0, \mathcal{R}) \in \mathbb{R}^d; \mathbf{p}_0^{\text{abs}} = \mathbf{p}_0 \in \mathbb{R}^3. \tag{6}$$

Here $\mathbf{c_i}$ serves as a context vector (concatenated to the decoder's cross-attention memory), while $\mathbf{p}_0^{\text{abs}}$ is passed to the decoder to seed the initial state and anchor the predicted trajectory in the world frame.

4) Structural STL Encoder: We parse the STL specification φ into an operator–predicate abstract syntax tree(AST) and linearize it in preorder with a fixed sibling order to obtain a typed token sequence. For each node m we form an embedding by summing token, time-interval, and depth cues:

$$\mathbf{S} = [\mathbf{s}_1; \ldots; \mathbf{s}_L] \in \mathbb{R}^{L\times d}, \tag{7a}$$

$$\mathbf{s}_m = E_{\text{tok}}(\text{id}_m) + E_{\text{interval}}(\tilde{\Delta}_m) + E_{\text{depth}}(d_m), \tag{7b}$$

where L denotes the number of subformulaes, id_m indexes a unified vocabulary covering operators (`F/G/and/or`) and targeted region tokens; $\tilde{\Delta}_m = [\tilde{a}_m, \tilde{b}_m] \in [0,1]^2$ is the normalized time interval (obtained by dividing by the planning horizon), and d_m is the AST depth. We prepend a learnable $\langle\text{CLS}\rangle$ token and feed $[\langle\text{CLS}\rangle; \mathbf{S}]$ to a small Transformer encoder; the $\langle\text{CLS}\rangle$ output serves as the global formula embedding, while per-token outputs are exposed for downstream structural routing.

B. Autoregressive Trajectory Decoder with STL-Structured HKV–MoE

We decode trajectories autoregressively using a causal Transformer with cross-attention over a shared memory

$$\mathbf{M}_0 = [\mathbf{V}; \mathbf{L}; \mathbf{c_1}, \mathbf{c_2}] \in \mathbb{R}^{(N_v+m_{\text{text}}+2)\times d},$$

where $\mathbf{V}$ are image tokens (Eq. (4)), $\mathbf{L} \in \mathbb{R}^{m_{\text{text}}\times d}$ are STL text tokens (Eq. (5)) and $\mathbf{c_i} \in \mathbb{R}^d$ is the pose/region context (Eq. (6)). With $\mathbf{y}_0 = \mathbf{p}_0^{\text{abs}}$, the step-$t$ query

$$\mathbf{q}_t = E_{\text{frame}}(\mathbf{y}_{t-1}) + E_{\text{time}}(t) + E_{\text{pos}}(t)$$

attends to the autoregressive prefix and $\mathbf{M}_{t-1}$, yielding

$$\mathbf{h}_t = \text{DecBlock}(\mathbf{q}_t; \mathbf{M}_{t-1}); \quad \hat{\mathbf{y}}_t = W_{\text{head}}\,\mathbf{h}_t \in \mathbb{R}^3.$$

During training, schedule sampling with rate ρ replaces $\hat{\mathbf{y}}_t$ by ground-truth position token $\mathbf{y}_t$ with probability ρ:

$$\tilde{\mathbf{y}}_t = \rho \times \mathbf{y}_t + (1-\rho) \times \hat{\mathbf{y}}_t$$

The chosen next-frame embedding is appended to update the memory:

$$\mathbf{M}_t = [\mathbf{M}_{t-1}; E_{\text{frame}}(\tilde{\mathbf{y}}_t) + E_{\text{time}}(t) + E_{\text{pos}}(t)].$$

STL-Structured MoE. Beyond the standard MoE architecture, we develop a *STL-conditioned* expert-routing scheme

that harnesses the compositional structure inherent in the temporal logic specification. $\mathbf{S}$ are structural STL tokens (Eq. (7)) denote the per-token structural memory obtained from the structural STL encoder. For each decoding step t, we determine the *innermost* temporal operator whose bounded interval covers t, denoted by $\mathrm{op}(t)$, and assign a coarse time band within that interval, $\mathrm{band}(t) \in \{0, \ldots, K-1\}$, where K is the pre-defined interval gap. A deterministic map $b(\cdot)$ combines operator family and time band into one of B routing buckets:

$$o_t = b\big(\mathrm{op}(t),\, \mathrm{band}(t)\big) \in \{0, \ldots, B-1\}; t = 1, \ldots, T \quad (8)$$

This leverages STL compositionality: formulas decompose into operator-scoped subgoals with finite temporal support, yielding semantically meaningful buckets.

In L_{moe} decoder layers, the position-wise feed-forward network (FFN) is replaced by a sparse MoE gated *within* the selected bucket o_t. Let $\{\mathcal{E}_b\}_{b=0}^{B-1}$ be a partition of experts into buckets and $\mathcal{B}(o_t) := \mathcal{E}_{o_t}$ the active expert set at step t. Given hidden state $\mathbf{h}_t$, the MoE output is

$$\mathrm{FFN}^{\text{MoE}}(\mathbf{h}_t; o_t) \;=\; \sum_{e \in \mathcal{E}_{\text{top-}k}(o_t)} g_{t,e}\, \mathrm{FFN}_e(\mathbf{h}_t), \quad (9)$$

where $\mathcal{E}_{\text{top-}k}(o_t) \subseteq \mathcal{B}(o_t)$ denotes the top-k experts selected *from the bucket* and the mixture weights $g_{t,e}$ are normalized over $\mathcal{B}(o_t)$ by a lightweight hierarchical key–value router:

$$g_{t,\cdot} \;=\; \mathrm{softmax}_{e \in \mathcal{B}(o_t)}\!\left(\frac{\langle W_q \mathbf{h}_t,\, \mathbf{K}_e \rangle}{\sqrt{d_k}}\right), \sum_{e \in \mathcal{B}(o_t)} g_{t,e} = 1. \quad (10)$$

A shared dense pathway runs in parallel and the outputs are blended by a learned gate $\alpha_t \in [0, 1]$ (scalar or channel-wise):

$$\mathrm{FFN}_{\text{final}}(\mathbf{h}_t) \;=\; \mathrm{FFN}_{\text{shared}}(\mathbf{h}_t) \;+\; \alpha_t\, \mathrm{FFN}^{\text{MoE}}(\mathbf{h}_t). \quad (11)$$

This STL-structured routing (i) promotes expert specialization by operator family and coarse time band; (ii) restricts competition to the active bucket, reducing cross-family interference and gradient contention; (iii) enforces sparse activation that allocates capacity where needed, improving sample efficiency; and (iv) yields interpretable routing, as expert activations align with subformulae and their temporal support. To prevent expert collapse and maintain utilization, we employ standard auxiliary regularizers (e.g., load-balancing and entropy terms) and recover the dense decoder in the limit $\alpha_t \to 0$.

C. Training Objective

We minimize a weighted sum of reconstruction, feasibility, obstacle, STL, and MoE regularization terms. For $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_T)$ with $\hat{\mathbf{y}}_t = (\hat{x}_t, \hat{y}_t, \hat{\psi}_t)$ and ground truth $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$,

$$\begin{aligned}\mathcal{L} \;=\; w_{\text{rec}} \sum_{t=1}^{T} \big\|\hat{\mathbf{y}}_t - \mathbf{y}_t\big\|_2^2 \;+\; w_{\text{end}} \big\|\hat{\mathbf{y}}_T^{xy} - \mathbf{y}_T^{xy}\big\|_2^2 \quad (12)\\ + w_{\text{obs}}\, \mathcal{L}_{\text{obs}}(\hat{\mathbf{Y}}) \;+\; w_{\text{feas}}\, \mathcal{L}_{\text{feas}}(\hat{\mathbf{Y}})\\ + w_{\text{stl}}\, [\,\gamma - \rho_\varphi(\hat{\mathbf{Y}})\,]_+ \;+\; w_{\text{moe}}\, \mathcal{R}_{\text{MoE}},\end{aligned}$$

where ρ_φ is a differentiable STL robustness score (positive when the formula is satisfied), $[\cdot]_+ = \max(\cdot, 0)$ is a hinge, and $\gamma \geq 0$ is a is the truncation factor, which is used to encourage the policy to improve "hard" trajectories (with robustness scores $< \gamma$) rather than further increasing "easy" trajectories that already achieve high robustness scores ($\geq \gamma$).

Feasibility uses a hinge on step length and heading change:

$$\begin{aligned}\mathcal{L}_{\text{feas}}(\hat{\mathbf{Y}}) \;=\; \sum_{t=2}^{T} \Big[\, \big\|\hat{\mathbf{y}}_t^{xy} - \hat{\mathbf{y}}_{t-1}^{xy}\big\|_2 - d_{\max} \Big]_+ \; + \quad (13)\\ \lambda_\psi \sum_{t=2}^{T} \Big[\, \big|\hat{\psi}_t - \hat{\psi}_{t-1}\big| - \delta_{\max} \Big]_+ .\end{aligned}$$

Obstacle avoidance applies a smooth barrier/soft-distance surrogate to forbidden regions $\{\mathcal{O}_k\}$:

$$\mathcal{L}_{\text{obs}}(\hat{\mathbf{Y}}) \;=\; \frac{1}{T} \sum_{t=1}^{T} \Phi\big(\hat{\mathbf{y}}_t^{xy};\, \{\mathcal{O}_k\}\big), \quad (14)$$

with Φ any differentiable proxy (e.g., softplus of negative signed distance).

MoE regularization discourages expert collapse and promotes balanced routing:

$$\mathcal{R}_{\text{MoE}} \;=\; \lambda_{\text{bal}}\, \mathcal{L}_{\text{bal}} \;+\; \lambda_{\text{ent}}\, \mathcal{L}_{\text{ent}}, \quad (15)$$

where $\mathcal{L}_{\text{bal}}$ encourages uniform expert importance and $\mathcal{L}_{\text{ent}}$ penalizes overly peaky gate distributions; either can be instantiated with standard choices.

Schedules are kept simple and effective: the teacher-forcing probability decays linearly $\rho(e) = \max\{0,\, 1 - e/E\}$ over epochs $e = 0, \ldots, E$, and secondary weights are warmed up $w_j(e) = r(e)\, w_j^\star$ with $r(e)$ increasing from $r_{\min} \in (0, 1]$ to 1 over the first E_{warm} epochs for $j \in \{\text{obs}, \text{feas}, \text{stl}, \text{moe}\}$. The MoE blend is initialized toward the dense path to recover the non-MoE decoder at early training and is learned end-to-end thereafter.

D. RRT-Guided Local Replanning with STL Awareness

Given a nominal trajectory $\hat{\mathbf{Y}} = \{(x_t, y_t, \theta_t)\}_{t=1}^{T}$, we run a rollout-and-check loop. If a forward lookahead at time τ indicates (i) collision/kinematic violation, denoted by $CK(Y, \tau)$, or (ii) *STL robustness* of the local window is negative, we trigger short-horizon RRT-style replanning on $[\tau, \tau+H]$. Expansion is goal-biased: with probability β targets are sampled near nominal anchors $\{\hat{p}_t\}_{t=\tau}^{\tau+H}$, otherwise uniformly. Candidate edges must satisfy local limits $\|p_t - p_{t-1}\| \leq d_{\max}$ and $|\theta_t - \theta_{t-1}| \leq \Delta\theta_{\max}$. Among feasible branches that reach $\tau+H$, we select the path minimizing

$$\mathcal{J} = \sum_{t=\tau}^{\tau+H} \Big(\lambda_1 \|p_t - \hat{p}_t\|_2^2 + \lambda_2\, \kappa_t^2\Big) \;-\; \lambda_3\, \tilde{\rho}_\varphi\big(\widetilde{\mathbf{Y}}_{\tau:\tau+H}\big), \quad (16)$$

where $p_t = (x_t, y_t)$, κ_t is a discrete curvature proxy, and $\tilde{\rho}_\varphi$ is the STL robustness of the segment (higher is better). The repaired segment is optionally smoothed (e.g., cubic spline) and spliced back into $\hat{\mathbf{Y}}$. The full procedures, termed **TSP**, is shown in Algorithm 1.

Algorithm 1: STL-Aware RRT-Guided Local Replanning (Triggered Segment Repair, **TSP**)

Input : Nominal trajectory $\hat{\mathbf{Y}} = \{\hat{y}_1, \ldots, \hat{y}_T\}$; STL formula φ; horizon H; goal-bias β; sampling times M; limits $(d_{\max}, \Delta\theta_{\max})$

Output: Repaired trajectory $\widetilde{\mathbf{Y}}$

$\widetilde{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}}$; $t \leftarrow 1$

while $t \leq T$ **do**

  $H_t = \min(H, T - t - 1)$

  if $CK(\widetilde{\mathbf{Y}}, t)$ ***or*** $\rho(\widetilde{\mathbf{Y}}_{t:t+H_t}, \varphi) < 0$ **then**

    Initialize tree $\mathcal{T}$ with root $z_0 = \widetilde{y}_{t-1}$; set $z_{H_t+1} = \widetilde{y}_{t+H_t+1}$

    for $m = 1$ **to** M **do**

      Sample target q: with prob β sample near anchors $\{\hat{p}_\tau, \ldots, \hat{p}_{\tau+H_t}\}$; else sample uniformly

      $z_{\text{near}} \leftarrow \texttt{nearest}(\mathcal{T}, q)$

      $z_{\text{new}} \leftarrow \texttt{run}(z_{\text{near}}, q; d_{\max}, \Delta\theta_{\max})$

      insert z_{new} and edge $(z_{\text{near}}, z_{\text{new}})$ into $\mathcal{T}$

    $\mathbf{Y}_{new} = \texttt{path}(\mathcal{T}, z)$, with min $\mathcal{J}$ via (16), $\forall z$ satisfies `height(z)` $= H_t$ **and** $feasible(z, z_{H_t+1})$

  $t \leftarrow t + 1$

return $\widetilde{\mathbf{Y}}$

VI. Experiments

A. Dataset and Metrics

Dataset. We construct a high-fidelity, Gazebo-based benchmark for factory–logistics navigation under Signal Temporal Logic (STL) guidance. Each episode includes synchronized multi-view RGB streams with calibrated intrinsics/extrinsics, ego states, polygonal maps of static obstacles and task regions, a formally annotated STL specification from a bounded-horizon fragment, and an expert reference trajectory. The benchmark covers representative warehouse layouts (aisles, racks, pallets) with domain randomization in geometry, placement, lighting, and textures. To assess generalization, we adopt two regimes: *ID-layout with novel STL tasks* (ID-L/Novel-Φ), where layouts are seen during training but task specifications are new, and *OOD-layout with novel STL tasks* (OOD-L/Novel-Φ), where both layouts and task compositions are unseen. Splits are disjoint by layout and STL templates. We standardize a fixed prediction horizon and provide per-episode metadata for reproducibility.

Metrics. We evaluate predicted trajectories along only one ax: *Success Rate (SR)* for episodes that satisfy the stl task and remain dynamically feasible. Unless otherwise specified, we report per-regime episode-level point estimates for ID-L/Novel-Φ and OOD-L/Novel-Φ, together with scenario-wise breakdowns.

B. Implementation details

We train on a single NVIDIA A800 (80 GB) with batch size 12 for 24 epochs using AdamW (betas $0.9/0.95$) and three parameter groups: (i) weight-decayed (10^{-2}), (ii) norm/bias/embedding without decay, and (iii) router centroids (`expert_key`) with a $2\times$ larger learning rate and a small decay 10^{-3}; the base learning rate is 1×10^{-5}. Unless noted, we set d=512, horizon T=80, and a 4-layer decoder with 4 heads and hidden width 128; MoE is enabled in the last 2 layers with B=6 buckets, E'=2 experts per bucket, temperature τ=1.0, and in-bucket top-k=2 (dropout 0.1). What's more, the sampling probability β in **TSP** is 0.5. The source codes are available at `https://github.com/yc7421cmd/E2E-Signal-Temporal-Logic-Planner.git`.

C. Ablation Study

Table II reports All(ID) ablations on three factors while keeping splits, schedule, and optimiser settings fixed: (a) the *loss margin* γ in the STL hinge term of Eq. (12), (b) the *sharpness* k of the smooth $\max / \min$ used in robustness aggregation, and (c) the image resolution/backbone.

Loss margin. γ enters only through the hinge $[\gamma - \rho_\varphi(\hat{\mathbf{Y}})]_+$. Small margins under-penalize near-violations and admit borderline solutions; overly large margins keep the hinge active for most samples, slowing optimization and trading off with feasibility/collision terms. We observe a wide stable regime for moderate margins (e.g., $\gamma \in [0.1, 0.5]$); pushing γ higher yields diminishing or negative returns. We therefore adopt γ=0.2 in all main results.

Smooth-aggregation sharpness. k controls the tightness of the differentiable $\widetilde{\max}/\widetilde{\min}$ used to approximate formulae robust. Very small k over-smooths and biases robustness; very large k approaches hard max/min and produces sparse, ill-conditioned gradients. A mid-range setting achieves the best bias–variance trade-off; we use k=300 by default.

Image resolution and backbone. Under an iso-compute setting, increasing input resolution improves geometric fidelity and adherence to fine spatial constraints, but exhibits clear diminishing returns alongside higher memory/FLOPs. Within Swin-B, using larger attention windows (w12) at medium–high resolutions expands the effective receptive field without increasing the number of encoder tokens. In our ablations, scaling from 320×480 to 480×720 delivers the most pronounced gains, while a further jump to 640×960 yields only marginal improvement at disproportionate cost. We therefore adopt w12 at a medium–high resolution (e.g., 512×768) as the default.

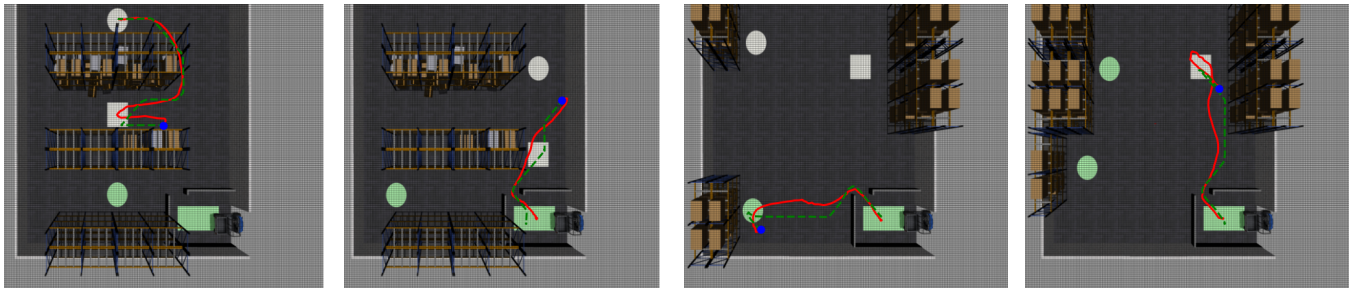
D. Main Results

We evaluate on five in-distribution subsets, `single-F`, `single-G`, `only-AND`, `only-OR`, `only(AND-OR)` and their union set `All (ID)`, and three aggregated out-of-distribution (OOD) suites (`OOD-1/2/3 (All)`). The primary metric is *success rate*, defined as jointly satisfying the STL specification, remaining collision-free, and meeting kinematic feasibility. All entries in Table I are percentages.

Overall, the **Baseline (no-MoE)** shows limited generalization, particularly on composite formulas (notably `only(AND-OR)`) and under OOD shift. A post-hoc **TSP** improves feasibility and safety but yields only modest

Method	single-F	single-G	only-AND	only-OR	only(AND–OR)	All (ID)	OOD-1 (All)	OOD-2 (All)	OOD-3 (All)
Baseline (no-MoE)	72.75	60.75	61.25	64.88	60.50	61.50	7.50	8.62	12.62
Ours(S-MSP)	77.75	77.00	63.50	79.00	62.50	71.00	7.88	6.67	14.00
Baseline + TSP	**90.88**	79.00	77.75	82.25	74.33	77.88	12.50	**12.88**	19.25
Ours(S-MSP) + TSP	89.00	**90.00**	**85.83**	**93.00**	**84.50**	**88.00**	**13.67**	10.82	**23.00**

TABLE I: Success rate on ID subsets and aggregated OOD scenes.



(a) ID (b) OOD-1 (c) OOD-2 (d) OOD-3

Fig. 2: Qualitative rollouts on representative tasks from ID and OOD suites.

a) Satisfaction threshold γ

γ	Val Acc.	Test Acc.
0.0	66.00	66.67
0.1	71.50	69.50
0.2	73.50	71.00
0.5	71.00	68.50
0.8	66.75	65.00
1.0	61.67	62.00

b) Sharpness k for $\widetilde{\max}/\widetilde{\min}$

k	Val Acc.	Test Acc.
1	55.33	55.00
5	53.00	53.00
10	63.33	63.00
100	64.88	65.00
1000	70.75	69.50
10000	69.00	68.00

c) Image resolution and backbone

Model	Resolution.	Window.	Val Acc.	Test Acc.
Swin-B w7	320×480	7	63.00	61.50
Swin-B w7	480×720	7	66.33	66.00
Swin-B w7	640×960	7	67.50	65.00
Swin-B w12	384×576	12	66.00	68.75
Swin-B w12	512×768	12	73.50	71.00

TABLE II: Different setups in training S-MSP.

gains in STL satisfaction, since downstream filtering cannot correct upstream semantic planning errors. Relative to a simple transformer baseline, **S-MSP** uses structured MoE routing aligned with STL operator families and temporal segments, reducing upstream semantic mismatches and improving reliability under shift. Quantitatively (Table I), on ID it lifts "All (ID)" from 61.50% to 71.00% ($+9.50$ pp), with pronounced gains on `single-G` ($+16.25$ pp) and `only-OR` ($+14.12$ pp). With the **TSP** post-processor, the advantage amplifies: "All (ID)" reaches 88.00% versus 77.88% ($+10.12$ pp), and `only(AND-OR)` improves by $+10.17$ pp. Under OOD shift, S-MSP+TSP attains the best mean across suites (15.83% vs. 14.88%), led by `OOD-3` ($+3.75$ pp), and achieves best-in-column on $5/6$ ID subsets (plus the ID aggregate) and $2/3$ OOD suites. Here, pp denotes percentage points.

To better illustrate the behavior of our planner, we further include four full-width qualitative rollouts—one each from the ID, OOD-1, OOD-2, and OOD-3 suites, showing successful trajectories that satisfy the STL specification while remaining collision-free and kinematically feasible in Fig. 2. We use a factory-transport environment with four zones, white circle (A), white square (B), green circle (C), green rectangle (D), and static shelves/walls delimiting the drivable free space $\mathcal{S}$. The planner is trained on the ID layout and evaluated (without finetuning) on one ID and three OOD layouts. Blue point marks the start, green is the expert trajectory, red is the planned trajectory and v means vehicle. Four STL tasks are showing below:

$$\phi_a = (\mathbf{F}[18,28](v \in C) \vee \mathbf{G}[19,22](v \in B)) \wedge \mathbf{F}[14,28](v \in B) \wedge \mathbf{F}[32,77](v \in A); \tag{17a}$$

$$\phi_b = (\mathbf{F}[2,19](v \in C) \vee \mathbf{G}[15,18](v \in B)) \wedge (\mathbf{G}[53,54](v \in D) \vee \mathbf{G}[50,53](v \in D)); \tag{17b}$$

$$\phi_c = (\mathbf{F}[18,26](v \in D) \vee \mathbf{G}[25,26](v \in C)) \wedge (\mathbf{F}[34,76](v \in B) \vee \mathbf{G}[67,71](v \in D)) \wedge \mathbf{F}[14,27](v \in C); \tag{17c}$$

$$\phi_d = \mathbf{F}[13,19](v \in B) \wedge \mathbf{G}[71,72](v \in D). \tag{17d}$$

The rollouts provide evidence that the model has learned *image- and task-conditioned* planning, producing collision-free and kinematically admissible trajectories across both ID and OOD layouts. Given only multi-iew image observations and finite-horizon STL specifications, the planner consistently satisfies the prescribed reach/sequence predicates while remaining within the drivable set and respecting per-step motion limits, without per-layout tuning. This indicates transfer of both scene understanding and symbolic goal structure rather than memorization of a fixed particular map.

Such behavior confers several system-level benefits: (i) *compositionality*, as STL-defined objectives can be recombined without retraining; (ii) *robustness to moderate distribution shift*, as performance persists under shelf/aisle perturbations; (iii) *verifiability*, since satisfaction of $\mathbf{F}/\mathbf{G}$ predicates and feasibility constraints yields a principled certificate of correctness; (iv) *operational simplicity*, reducing hand-engineered interfaces between perception and planning by training end-to-end; and (v) *diagnosability*, as robustness margins and violation sets localize potential failure modes for targeted remediation. Collectively, these properties suggest a practical path toward deployable, specification-driven

planning in structured industrial environments.

VII. Conclusion

In this work, we presented, to the best of our knowledge, the first end-to-end model **S-MSP** for STL-specified robotic control, mapping synchronized sensory inputs and symbolic specifications directly to feasible trajectories. A structure-aware sparse Mixture-of-Experts exploits STL compositionality and temporal scope, yielding faster convergence and high satisfaction and feasibility at low latency. For deployability, we augment the policy with a runtime repair layer that invokes an RRT-style local replanner to modify only violating subsegments while respecting dynamics and obstacles. The experimental results validate the effectiveness and robustness of the proposed approach.

In this work, we restrict attention to STL fragments without nested temporal operators. Future work will extend the framework to nested operators and incorporate reinforcement learning for preference alignment and adaptation under distributional shifts to improve deployment robustness.

References

[1] C. Belta and S. Sadraddini. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):115–140, 2019.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A Vision–Language–Action Flow Model for General Robot Control. *arXiv:2410.24164*, 2024.

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv:2212.06817*, 2022.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[5] K. Cho and S. Oh. Learning-based model predictive control under signal temporal logic specifications. In *IEEE International Conference on Robotics and Automation)*, pages 7322–7329, 2018.

[6] D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv:2401.06066*, 2024.

[7] E. Dietrich, E. C. Gezer, B. Zhong, M. Arcak, M. Zamani, R. Skjetne, and A. J. Sørensen. Symbolic control for autonomous docking of marine surface vessels. *arXiv:2501.13199*, 2025.

[8] M. Fossdal, A. H. Brodtkorb, M. Arcak, and A. J. Sørensen. Past-time signal temporal logic hybrid switching control for underwater vehicles. In *IEEE/OES Autonomous Underwater Vehicles Symposium*, pages 1–6, 2024.

[9] Y. Gilpin, V. Kurtz, and H. Lin. A smooth robustness measure of signal temporal logic for symbolic control. *IEEE Control Systems Letters*, 5(1):241–246, 2020.

[10] Z. Guo, W. Zhou, and W. Li. Temporal logic specification-conditioned decision transformer for offline safe reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024.

[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 *IEEE/CVF conference on computer vision and pattern recognition*, pages 17853–17862, 2023.

[12] S. Huang, Z. Zhang, T. Liang, Y. Xu, Z. Kou, C. Lu, G. Xu, Z. Xue, and H. Xu. Mentor: Mixture-of-experts network with task-oriented perturbation for visual reinforcement learning. *arXiv:2410.14972*, 2024.

[13] X. Jia, J. You, Z. Zhang, and J. Yan. Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. *arXiv:2503.07656*, 2025.

[14] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *IEEE/CVF International Conference on Computer Vision*, pages 8340–8350, 2023.

[15] X. Jiang, Y. Ma, P. Li, L. Xu, X. Wen, K. Zhan, Z. Xia, P. Jia, X. Lang, and S. Sun. Transdiffuser: End-to-end trajectory generation with decorrelated multi-modal representation for autonomous driving. *arXiv:2505.09315*, 2025.

[16] I. Leal, K. Choromanski, D. Jain, A. Dubey, J. Varley, M. Ryoo, Y. Lu, F. Liu, V. Sindhwani, Q. Vuong, et al. Sara-rt: Scaling up robotics transformers with self-adaptive robust attention. In *IEEE International Conference on Robotics and Automation*, pages 6920–6927, 2024.

[17] K. Leung, N. Aréchiga, and M. Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, 42(6):356–370, 2023.

[18] B. Liao, S. Chen, H. Yin, B. Jiang, C. Wang, S. Yan, X. Zhang, X. Li, Y. Zhang, Q. Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *Computer Vision and Pattern Recognition Conference*, pages 12037–12047, 2025.

[19] R. Liu, A. Hou, X. Yu, and X. Yin. Zero-shot trajectory planning for signal temporal logic tasks. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*, 2025.

[20] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 152–166, 2004.

[21] Y. Meng and C. Fan. Signal temporal logic neural predictive control. *IEEE Robotics and Automation Letters*, 8(11):7719–7726, 2023.

[22] C. Pan, B. Yaman, T. Nesti, A. Mallik, A. G. Allievi, S. Velipasalar, and L. Ren. Vlp: Vision language planning for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14760–14769, 2024.

[23] Y. V. Pant, H. Abbas, and R. Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *IEEE Conference on Control Technology and Applications*, pages 1235–1240, 2017.

[24] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87, 2014.

[25] R. Takano, H. Oyama, and M. Yamakita. Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation. In *IEEE International Conference on Robotics and Automation*, pages 8409–8415, 2021.

[26] C. Wang, X. Yu, J. Zhao, L. Lindemann, and X. Yin. Sleep when everything looks fine: Self-triggered monitoring for signal temporal logic tasks. *IEEE Robotics and Automation Letters*, 9(10):8983–8990, 2024.

[27] S. Wang, S. Li, L. Yin, and X. Yin. Synthesis of temporally-robust policies for signal temporal logic tasks using reinforcement learning. In *IEEE International Conference on Robotics and Automation*, pages 10503–10509, 2024.

[28] S. Wang, X. Yin, S. Li, and X. Yin. Tractable reinforcement learning for signal temporal logic tasks with counterfactual experience replay. *IEEE Control Systems Letters*, 8:616–621, 2024.

[29] Z. Yang, Y. Chai, X. Jia, Q. Li, Y. Shao, X. Zhu, H. Su, and J. Yan. Drivemoe: Mixture-of-experts for vision-language-action model in end-to-end autonomous driving. *arXiv:2505.16278*, 2025.

[30] B. Ye, B. Zhang, and H. Zhao. DAP: A discrete-token autoregressive planner for autonomous driving, 2025.

[31] X. Yin, B. Gao, and X. Yu. Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges. *Annual Reviews in Control*, 57:100940, 2024.

[32] X. Yu, W. Dong, S. Li, and X. Yin. Model predictive monitoring of dynamical systems for signal temporal logic specifications. *Automatica*, 160:111445, 2024.

[33] X. Yu, Y. Zhao, X. Yin, and L. Lindemann. Signal temporal logic control synthesis among uncontrollable dynamic agents with conformal prediction. *Automatica*, 183:112616, 2026.

[34] J. Zhao, K. Zhu, M. Feng, S. Li, and X. Yin. No-regret path planning for temporal logic tasks in partially-known environments. *The International Journal of Robotics Research*, 44(9):1526–1552, 2025.

[35] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183, 2023.