IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 March 2024, accepted 13 April 2024, date of publication 22 April 2024, date of current version 29 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3392289

RESEARCH ARTICLE

End-to-End Path Planning Under Linear Temporal Logic Specifications

CHAEEUN YANG, SOJEONG YOON, AND KYUNGHOON CHO
Department of Information and Telecommunication Engineering, Incheon National University, Incheon 22012, South Korea

Corresponding author: Kyunghoon Cho (ckh0923@inu.ac.kr)

This work was supported in part by Incheon National University Research Grant, in 2023; and in part by the National Research Foundation of Korea (NRF) grant funded by Korean Government (MSIT) under Grant 2021R1G1A1014256.

ABSTRACT This paper presents a novel deep learning framework for robotic path planning that seamlessly integrates Linear Temporal Logic (LTL) with trajectory optimization to meet mission specifications efficiently. Our approach innovates on several fronts: First, by training a neural network end-to-end to generate control sequences that are not only cost-effective but also fully compliant with LTL-defined mission objectives. This negates the need for generating traditional automatons, as our network is capable of directly interpreting LTL formulas to guide path planning. Key to our framework is the use of a Conditional Variational Autoencoder (CVAE), which is adept at identifying the optimal distribution of trajectories. This enables the extraction of practical control sequences through a process of sampling latent variables and inferring control outputs, thus addressing the critical challenge of trajectory optimization under uncertainty. Moreover, our model incorporates transformer networks to refine these trajectory distributions into nearly optimal control sequences, further enhanced by a Gaussian Mixture Model (GMM) to manage uncertainty and fine-tune adjustments effectively. Empirical validation through comparative simulations showcases the superior performance of our model. It achieves significant advancements in trajectory optimality and mission success rates over existing deep learning-based path planning strategies. This work underscores the potential of integrating LTL in deep learning models for robotic path planning, marking a significant leap forward in the domain.

INDEX TERMS Deep learning-based control synthesis, formal methods, mission-based path planning.

I. INTRODUCTION

Path planning is a critical component in the field of robotics, advancing from straightforward two-dimensional navigation tasks to tackling complex systems such as robot manipulators [1], [2], [3] and intricate scenarios [4], [5], [6]. This progression reflects the broadening scope of robotic tasks, highlighting the need for advanced path planning algorithms capable of navigating both physical environments and the complex requirements of these tasks.

Translating mission specifications, often described in human language, into computational models presents a significant challenge in path planning. Formal methods like Linear Temporal Logic (LTL), Computation Tree Logic (CTL), and μ-calculus have played a crucial role in this context. Notably, LTL has been favored for its adaptability and expressive power in defining complex missions [7], [8], [9], providing a structured yet flexible means to encode mission objectives.

Beyond defining mission parameters, identifying trajectories that are both cost-effective and meet computational efficiency standards is paramount. For example, in environments characterized by varying communication strengths, finding low-cost paths that minimize exposure to areas with poor communication is essential. Traditional methods like Rapidly-exploring Random Tree Star (RRT*) [10], while effective, can be computationally intensive, especially in environments laden with constraints.

The incorporation of deep learning techniques into the realm of path planning presents an innovative alternative

The associate editor coordinating the review of this manuscript and approving it for publication was Ton Duc Do.

that excels at deriving optimal paths directly from datasets, thus effectively addressing the computational limitations associated with conventional methodologies. Demonstrating wide-ranging applicability, these techniques have significantly advanced the field in various domains. Specifically, in the context of robotic systems, deep learning has facilitated sophisticated control strategies for robot manipulators [1], a domain characterized by complex interactions within mechanical systems. Concurrently, in the domain of autonomous vehicles, it has been instrumental in navigating the multifaceted challenges of autonomous driving [11], [12], further underscoring the versatility and computational efficiency of deep learning-based approaches.

This paper proposes a novel deep learning framework for robotic path planning that effectively integrates LTL for mission specification with trajectory optimization techniques. Our model utilizes a Conditional Variational Autoencoder (CVAE) and a Transformer network to generate control sequences that comply with LTL mission specifications while prioritizing cost efficiency, representing a substantial leap forward in the integration of deep learning and formal methods in path planning.

The introduction of the Transformer network for interpreting LTL formulas and generating control sequences [13], alongside the CVAE's ability to explore complex trajectory manifolds [14], underscores the innovative nature of our approach. The use of a Gaussian Mixture Model (GMM) to refine the output further highlights our method's ability to address uncertainties, enhancing both the precision and reliability of path planning under LTL constraints.

Our contributions set new standards for cost-efficiency and computational performance in robotic path planning. The effectiveness and superiority of our model over existing deep learning-based strategies are validated through comparative simulations, demonstrating its potential to significantly impact the field.

Figure 1 delineates the process flow of the proposed path planning method. Starting with the environment and the LTL formula on the left, the network generates an anchor control sequence in the middle step. Subsequently, the process arrives at the final solution, which adheres to the GMM distribution, as shown on the right. The prescribed LTL formula, $\phi = \Diamond(a \wedge \Diamond(b))$, requires that the solution trajectory visit regions a and b in sequence. Comparative simulations illustrate that our method not only achieves greater efficiency in generating solution trajectories but also results in lower costs compared to existing path planning methods.

II. RELATED WORK

Path planning is a cornerstone of robotics, striking a delicate balance between achieving low-cost trajectories, navigating complex dynamics, and adhering to precise mission specifications. The literature in this domain reflects a diverse array of strategies, each addressing these challenges to varying degrees.

A. FINITE DETERMINISTIC SYSTEMS

Studies in finite deterministic systems have explored optimal controls with diverse cost functions, such as minimax for bottleneck path problems [15], and weighted average for cyclic paths [16]. Despite their insights, these approaches face challenges in real-world continuous path planning scenarios, particularly due to limitations in incorporating robot dynamics and the need for high-resolution discretization.

B. SAMPLING-BASED MOTION PLANNING

Sampling-based motion planning methods like the Rapidly-exploring Random Tree (RRT) [17] have gained prominence in addressing path planning challenges that require the integration of temporal logic and handling complex dynamics. Among these methods, the Rapidly-exploring Random Graph (RRG) strategy has been specifically adapted for deterministic μ-calculus specifications [18], demonstrating its utility in optimizing motion planning. Despite their advantages, such strategies encounter scalability and time efficiency issues as the number of nodes expands and the complexity involved in constructing product automata increases.

The Rapidly-exploring Random Tree Star (RRT*) algorithm, known for its asymptotic optimality—that is, its solutions progressively converge to an optimal trajectory that meets the task specifications—has been integrated with process algebra specifications [19] to handle tasks defined by 'alternative' and 'sequential' operators. Despite its strengths, this specificity restricts the range of tasks that can be addressed, signaling a potential area for extending the methodology to accommodate a broader spectrum of task specifications.

C. MULTI-LAYERED FRAMEWORKS

Multi-layered frameworks combine discrete abstractions with automata representing co-safe LTL formulas [20], [21], [22]. These systems guide trajectory formation using sampling-based methodologies. Certain studies [23] concentrate on developing methods for low-cost trajectory planning, taking into account both robot dynamics and temporal logic specifications. However, the geometric decomposition dependency and computational intensity of sampling-based methods at the lower layer remain challenges.

D. OPTIMIZATION METHODS

Optimization approaches, particularly those using mixed-integer programming, have targeted optimal paths under LTL constraints [24], [25]. Challenges in these methods escalate with increasing obstacles and more complex LTL formulas due to the growing number of integer constraints. The cross entropy-based planning algorithm [26] improves efficiency but struggles with extensive LTL formulas.

E. LEARNING FROM DEMONSTRATION (LFD)

LfD in robotics has increasingly intersected with temporal logic. Key advancements include the integration of machine

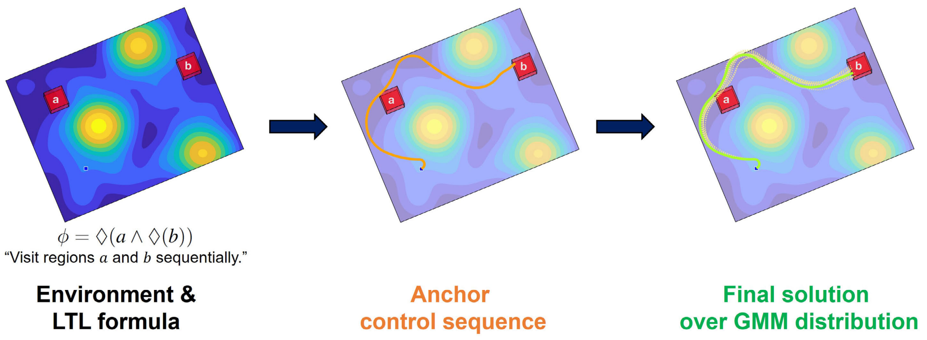

$\phi = \Diamond(a \wedge \Diamond(b))$
"Visit regions a and b sequentially."
Environment & LTL formula
Anchor control sequence
Final solution over GMM distribution
a
b

FIGURE 1. **Visualization of the proposed path planning approach. The leftmost image showcases the initial environment and LTL formula, the center image displays the generated anchor control sequence, and the rightmost image presents the final trajectory solution over the GMM distribution.**

learning and temporal logic for autonomous behavior, employing STL robustness degree to delineate autonomous systems' actions [27]. Another study demonstrates the incorporation of STL into LfD policies using Monte Carlo Tree Search (MCTS), enhancing constraint satisfaction by adjusting the MCTS heuristic with STL robustness values [28].

Further research illustrates that learned continuous policies can simulate discrete plans specified by LTL formulas, highlighting LTL's utility in continuous control contexts [29]. Additionally, a significant method has been proposed for integrating formal task specifications within LfD skills using STL and black-box optimization (BBO) for skill adaptation [30].

F. TRAJECTORY FORECASTING

Advances in trajectory forecasting have leveraged deep learning to predict future movements by analyzing past data, closely aligning with LfD principles. This field utilizes advanced models, such as GMMs for variability and Variational Autoencoders with Transformer architectures for generating action-aware predictions [31], [32]. Recent innovations aim for more precise forecasting by integrating global intention with local movement strategies [33], showcasing a trend towards models that offer enhanced adaptability and accuracy in predicting complex motion patterns.

G. SUMMARY

In the landscape of robotic path planning, foundational strategies like "Finite Deterministic Systems" and "Sampling-based Motion Planning" have made significant contributions, yet often grapple with the demands of navigating real-world, continuous environments, particularly within dynamic settings and against complex mission criteria. Meanwhile, "Multi-layered Frameworks" and "Optimization Methods" have sought to integrate formal logic with optimization techniques to tackle intricate path planning challenges, but still face hurdles in computational efficiency and scalability, limiting their broader application. The emergence of "Learning from Demonstration" and "Trajectory Forecasting", propelled by advancements in machine learning, heralds new possibilities. However, even these cutting-edge approaches encounter challenges in seamlessly blending detailed mission specifications with the inherent unpredictability and variability of practical scenarios, underscoring the ongoing quest for more adaptable and robust path planning solutions.

III. PRELIMINARIES

In this section, we lay the foundational concepts and notations essential for understanding our approach to path planning under LTL specifications. Establishing a clear framework at the outset is crucial for comprehensively presenting the system model, dynamics, and the specific temporal logic used to articulate the desired properties of the paths. The forthcoming subsections will introduce the mathematical formulations that underpin our system's model, delineate the dynamics governing the system, and detail the principles of LTL that are instrumental in defining and evaluating the trajectory objectives. This preliminary groundwork is indispensable for navigating the complex landscape of autonomous systems and their operational criteria, setting the stage for our proposed method's detailed exploration.

A. SYSTEM MODEL

To establish a foundation for our system model, we first introduce essential notations:

- $\mathcal{X} \subset \mathbb{R}^n$: The system's state space.
- $\mathcal{X}_{obs} \subset \mathbb{R}^n$: Space occupied by obstacles.
- $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$: Free space not occupied by obstacles.
- $\mathcal{U} \subset \mathbb{R}^m$: Set of feasible controls.
- $\mathcal{W} \subset \mathbb{R}^{n_w}$: Workspace in which the system operates.
- $h : \mathcal{X} \to \mathcal{W}$: Mapping function from the state space to the workspace.

The system's dynamics are described by the following equation:

$$\dot{\mathrm{x}}_t = f(\mathrm{x}_t, \mathrm{u}_t), \tag{1}$$

where $x_t \in \mathcal{X}_{free}$ represents the system's state, $u_t \in \mathcal{U}$ denotes the control input, and f is a function that is continuously differentiable.

Given a control signal u over a time interval $[0, T]$, the trajectory $x(x_0, u)$ originates from state x_0. The state of the system along this trajectory at any time $t \in [0, T]$ is denoted by $x(x_0, u, t)$.

For discrete analysis, the trajectory $x(x_0, u)$ is discretized at time increments $\Delta t \in \mathbb{R}^+$, expressed as:

$$x_{\Delta t}(x_0, u) = \{x(x_0, u, i\Delta t)\}_{i=0}^{i_f}, \tag{2}$$

where $i_f \in \mathbb{N}$ represents the final time step and is determined based on the specific requirements of the trajectory analysis. This selection ensures that the discrete representation captures the essential dynamics of the trajectory over the desired analysis period, balancing computational efficiency with the accuracy of the simulation.

B. LINEAR TEMPORAL LOGIC (LTL)

LTL is a framework for formulating properties in a linear timeline [34]. It comprises atomic propositions (APs), Boolean operators, and temporal operators. An atomic proposition is a statement that is either true or false. Key LTL operators include $\bigcirc$ (next), U (until), $\Box$ (always), $\Diamond$ (eventually), and $\Rightarrow$ (implication). The construction of LTL formulas follows a specific grammar detailed in [35].

In our representation, $\Pi = \{\pi_0, \pi_1, \ldots, \pi_N\}$ represents all atomic propositions. An LTL trace, denoted as σ, is a sequence of atomic propositions. LTL evaluations typically involve infinite traces, with Σ^ω symbolizing all possible infinite traces derived from $\Sigma = 2^\Pi$. A trace σ is said to satisfy a formula ϕ if it is denoted as $\sigma \vDash \phi$.

For the purposes of this paper, our focus is on finite-time path planning using syntactically co-safe LTL formulas (sc-LTL) [36]. A sc-LTL formula ϕ has the property that any infinite trace satisfying ϕ also has a finite segment adhering to ϕ. All temporal logic formulas in this paper are in sc-LTL format.

1) AUTOMATON REPRESENTATION

Given a set of atomic propositions Π and a syntactically co-safe LTL formula ϕ, a nondeterministic finite automaton (NFA) can be constructed [37]. For example, for $\phi = \Diamond(a \wedge \Diamond(b \wedge \Diamond(c)))$, the resulting NFA is illustrated in Figure 2. An NFA can be converted to a deterministic finite automaton (DFA), which is more computationally convenient. A DFA is represented as $\mathcal{A}_\phi = (Q, \Sigma, \delta, q_{init}, Q_{acc})$, with each component defined as follows:

• Q	: Set of states
• $\Sigma = 2^\Pi$	: Alphabet
• $\delta : Q \times \Sigma \to Q$	: Transition function
• $q_{init} \subseteq Q$	: Initial states
• $Q_{acc} \subseteq Q$	: Accepting states

A trace σ of a DFA is accepted if its prefix reaches the accepting states, i.e., $\sigma_i \cap Q_{acc} \neq \emptyset$. Therefore, a trace satisfies the sc-LTL formula ($\sigma \vDash \phi$) if it is accepted by the corresponding DFA $\mathcal{A}_\phi$.

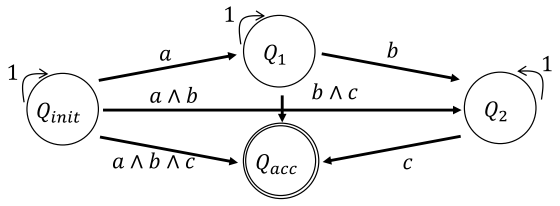


FIGURE 2. An NFA corresponding to the sc-LTL formula $\phi = \Diamond(a \wedge \Diamond(b \wedge \Diamond(c)))$. The diagram shows four states and the input alphabets for transition relations.

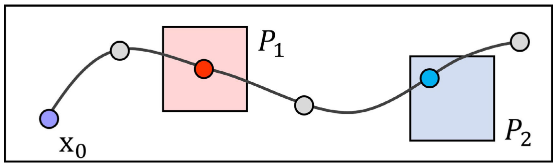


FIGURE 3. A trace defined over a discretized trajectory: For given $x_{\Delta t}(x_0, u) = x_0, x_1, \ldots, x_5$, its trace is a sequence with 6 elements $\{\pi_0, \neg\pi_1, \neg\pi_2\}, \{\pi_0, \neg\pi_1, \neg\pi_2\}, \{\neg\pi_0, \pi_1, \neg\pi_2\}$, $\{\pi_0, \neg\pi_1, \neg\pi_2\}, \{\neg\pi_0, \neg\pi_1, \pi_2\}, \{\pi_0, \neg\pi_1, \neg\pi_2\}$.

2) LTL SEMANTICS OVER TRAJECTORIES

In this work, we define regions of interest as $P = \{P_1, \ldots, P_n\}$ within the workspace $\mathcal{W}$. Each atomic proposition, π_j, from the set Π, is directly associated with a corresponding region of interest P_j. We introduce a labeling function, $L : \mathcal{W} \to 2^\Pi$, that maps each point in the workspace to a set of atomic propositions that are valid in that point. For given $\pi_i \in \Pi$, $\neg\pi_i$ holds true for $\{w \in \mathcal{W} \,|\, \pi_i \notin L(w)\}$. In particular, π_0 stays true in all workspace except regions of interest and obstacles.

Considering a discretized trajectory, denoted as $x_{\Delta t}(x_0, u) = x_0, x_1, \ldots, x_m$, which starts from x_0 and follows the control inputs u with a time step Δt, we can express its trace as [20]:

$$trace(x_{\Delta t}(x_0, u)) = L(h(x_0)), L(h(x_1)), \ldots, L(h(x_m)). \tag{3}$$

An illustrative example of a trajectory and its associated trace is provided in Figure 3. This representation allows us to map the trajectory's discrete segments to their corresponding traces. Given a trajectory trace $trace(x_{\Delta t}(x_0, u)) = \tau_0, \tau_1, \ldots, \tau_m$, we define the automaton states set $\mathcal{A}_\phi(trace(x_{\Delta t}(x_0, u))) = q_0, q_1, \ldots, q_m$ with q_k being:

$$q_k = \begin{cases} \delta(q_{init}, \tau_0) & \text{if } k = 0 \\ \delta(q_{k-1}, \tau_k) & \text{if } k \geq 1. \end{cases} \tag{4}$$

A trajectory $x_{\Delta t}(x_0, u)$ is considered to comply with the LTL formula ϕ, denoted by $x(x_0, u) \vDash_{\Delta t} \phi$, if the automaton reaches a subset of the accepting states Q_{acc}.

IV. PROPOSED METHOD

In our approach, the focal point is optimizing the accumulated cost $J(x_0, u)$, which represents the line integral of a cost function c over a trajectory. The cost is mathematically described as:

$$J(\mathrm{x}_0, u) = \frac{1}{T}\int_0^T c(x(\mathrm{x}_0, u, t))dt, \tag{5}$$

where $c : \mathcal{X} \rightarrow \mathbb{R}^+$ is a bounded and continuous cost function, $\mathbf{u}$ is the control signal from $t = 0$ to $t = T$, and x_0 denotes the initial state. The mission tasks are outlined using a syntactically co-safe LTL formula, with each atomic proposition linked to a designated region of interest.

In this paper, we introduce a novel approach to robotic path planning that significantly advances the synthesis of near-optimal control sequences. Our method uniquely conforms to specific mission requirements, adheres to system dynamics (as defined in Equation 1), and optimizes cost-efficiency criteria (outlined in Equation 5).

Central to our innovative methodology is a deep learning framework that marries a CVAE with Transformer networks, creating an end-to-end solution unparalleled in current path planning research. The CVAE is pivotal in learning the distribution within the latent space of optimal control sequences, a cornerstone for producing sequences that not only satisfy LTL constraints but also aim at minimizing costs.

Our approach utilizes convolutional neural networks (CNNs) to transform environmental inputs—such as cost maps, regions of interest, and obstacle configurations—into an image-like representation, optimizing the processing of spatial information in a novel way.

The key characteristics of our research is the introduction of two pioneering concepts: the anchor control sequence and the utilization of Gaussian Mixture Model (GMM) components. This modeling technique, involving a GMM, offers a novel method to accurately represent the array of potential control sequences. This technique is instrumental in optimizing these sequences through precise modeling of multimodal distributions, thereby enabling detailed and effective trajectory planning in complex environments.

Initiating with the Transformer's decoder generating an anchor control sequence, the GMM refines this sequence to incorporate minor uncertainties. This synergy, enhanced by the latent distribution learning, markedly boosts the precision and reliability of control sequence predictions. Our structured approach innovatively addresses uncertainties, thereby elevating the robustness of the path planning framework.

We dedicate subsequent sections to an in-depth exploration of our methodology, particularly focusing on the novel implementation of anchor controls within the GMM framework and its significant implications. Through this comprehensive elucidation, we aim to provide readers with a clear understanding of the groundbreaking advancements our path planning strategy introduces to the field.

A. DATA COMPONENTS

This section describes the input configurations used in our proposed deep learning framework, specifically designed to facilitate the interpretation of LTL formulas. Regions of interest within the operational environment are denoted alphabetically, starting with a, to simplify the association of LTL formulas with spatial regions.

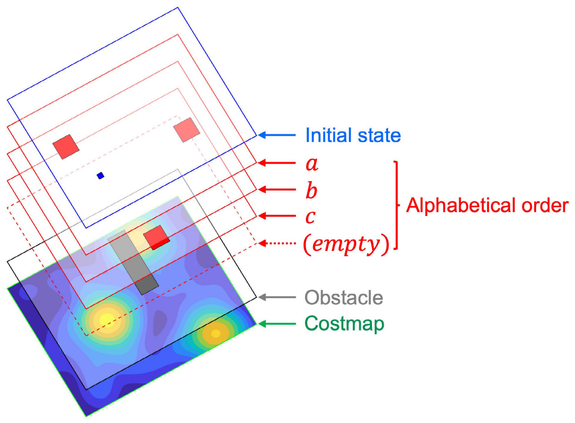


FIGURE 4. Depiction of the state image X layers for regions of interest $\{a, b, c\}$.

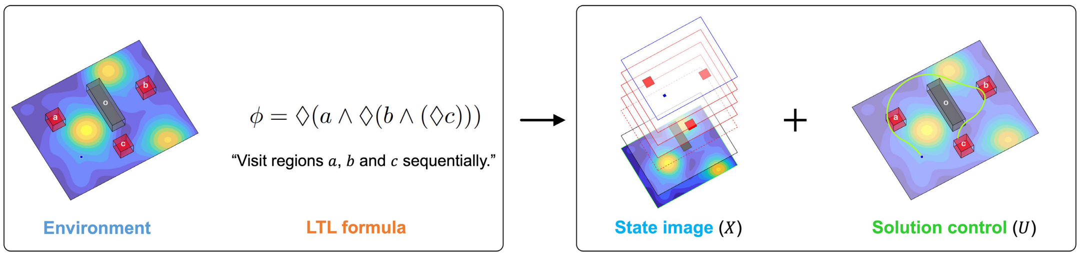
Our framework relies on two primary data components: the state image X and the solution control sequence U. The state image X is composed of multiple channels, with each channel representing different environmental features in a format that the neural network can process. As illustrated in Figure 4, the channels for the regions of interest, labeled $\{a, b, c\}$, are stacked to provide a comprehensive environmental context. The layering starts with the costmap at the base, followed by obstacle and regions of interest layers, and concludes with the layer representing the initial position.

The computation of control sequences is informed by methodologies outlined in prior research [23], which are in harmony with the dynamics of our system as delineated in Equation 1 and the requirements of co-safe temporal logic specifications. Specifically, the approach we have adopted from [23] aims to discover low-cost trajectories that fulfill the given co-safe temporal logic specifications. This is achieved through a evaluation of LTL semantics applied to the generated trajectories, ensuring compliance with the necessary specifications.

Our strategy for data generation is intricately tailored to meet the environmental constraints and LTL objectives pertinent to our study. The process is visually represented in Figure 5, illustrating the transformation of environmental parameters and LTL formulas into the resultant output control sequences. In instances where the length of generated sequences does not meet the maximum designated length, we employ dummy control values as placeholders. This practice ensures uniformity in sequence length, facilitating a straightforward training process by standardizing the input data across various scenarios.

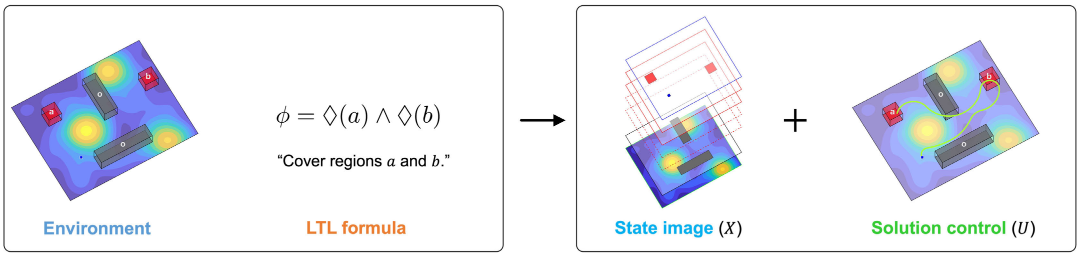
B. PROPOSED ARCHITECTURE

The architecture for the training phase of our proposed deep learning network is depicted in Figure 6. This comprehensive end-to-end network facilitates the entire process



$\phi = \lozenge(a \wedge \lozenge(b \wedge (\lozenge c)))$
"Visit regions a, b and c sequentially."
Environment
LTL formula
State image (X)
Solution control (U)

(a) $\phi = \lozenge(a \wedge \lozenge(b \wedge \lozenge(c)))$



$\phi = \lozenge(a) \wedge \lozenge(b)$
"Cover regions a and b."
Environment
LTL formula
State image (X)
Solution control (U)

(b) $\phi = \lozenge(a) \wedge \lozenge(b)$

FIGURE 5. Illustration of the data generation methodology for distinct LTL formulas, showcasing the synthesis of state images and the resultant control sequences.

from inputting the environmental image and LTL formula to generating the output control sequence.

Below is a glossary of key symbols used within the architecture, as depicted across Figures 6, 8, and 7:

- h_X : The encoded representation of the environmental state.
- h_ϕ : The encoded representation of the LTL formula, encapsulating mission objectives.
- h_U : The encoded control sequence, reflecting.
- ν : Parameters defining the CVAE encoder.
- θ : Parameters of the control decoder.
- z : A latent sample drawn from the encoded space.
- U_a : The anchor control sequence.
- $\alpha_U^i, \mu_U^i, \Sigma_U^i$: Components of the GMM, facilitating the modeling of control sequence distributions with respect to uncertainty.

In the encoding stage, the network processes the input data through specialized components: a CNN encodes the state image, while Transformer encoders [13] are employed for the control sequence and LTL formulas. The LTL formulas undergo an encoding process where each character is converted into an embedded representation, utilizing a predefined alphabet and operator symbols set. The encoded outputs, denoted as h_X for the state image X and initial state x_0, h_ϕ for the LTL formula ϕ, and h_U for the solution control sequence U, are then integrated within the network to facilitate the learning process.

One key feature of our proposed network is the utilization of a CVAE model, chosen for its proficiency in navigating the complexities of high-dimensional spaces and its adaptability to various input configurations. The CVAE plays a crucial role in our model by facilitating the generation of output control sequences through a process of latent space exploration. During the learning stage, the CVAE learns a probability distribution that represents the potential control sequences, conditioned on encoded state image features h_X and LTL formula features h_ϕ.

Our CVAE model is structured around three key parameterized functions: the recognition model $q_\nu(Z|h_X, h_\phi, h_U)$, which approximates the latent variable Z's distribution given the input features and the control sequence; the prior model $p_\theta(Z|h_X, h_\phi)$, representing the latent variable's distribution independent of the control sequence; and the generation model $p_\theta(U|z, h_X, h_\phi)$, which predicts the control sequence from a latent sample z. The parameters θ and ν denote the model-specific parameters, with Z as the latent space and z as a specific sample within that space.

The models function as follows:

- The recognition model $q_\nu(Z|h_X, h_\phi, h_U)$ is defined as a Gaussian distribution $\mathcal{N}(\mu_\nu(h_X, h_\phi, h_U), \Sigma_\nu(h_X, h_\phi, h_U))$, with μ_ν and Σ_ν representing the mean and covariance determined by the network.
- The prior model $p_\theta(Z|h_X, h_\phi)$ is assumed to be a standard Gaussian $\mathcal{N}(0, I)$, simplifying the latent space structure.
- The generation model $p_\theta(U|z, h_X, h_\phi)$ computes the likelihood of each control sequence element u_i conditional on the latent sample z and the encoded inputs, represented as the product of conditional probabilities over the sequence length N_u.

A sample z drawn from the recognition model is input into the decoder, generating the predicted control sequence $\hat{U} = u_0, \ldots, u_{N_u}$, where N_u denotes the sequence length. This

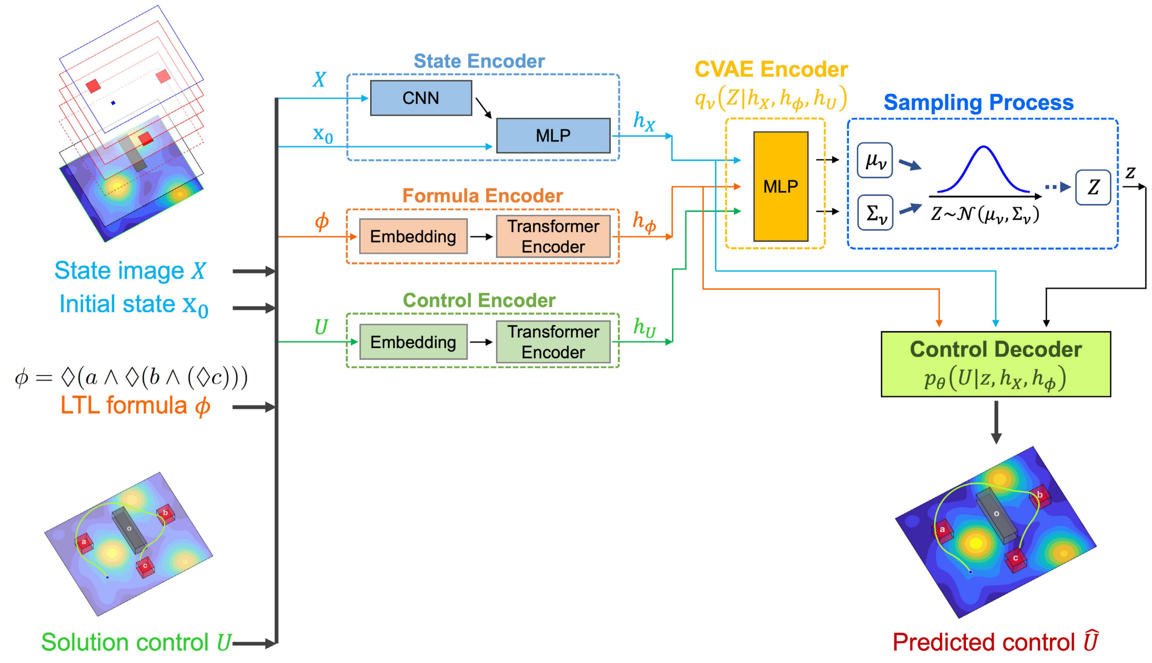


FIGURE 6. Training process of the proposed end-to-end deep learning network. This diagram illustrates the network's training process, highlighting the data flow from the multi-channeled state image X, initial state x_0, LTL formula ϕ, and solution control sequence U to the predicted control sequence $\hat{U}$.

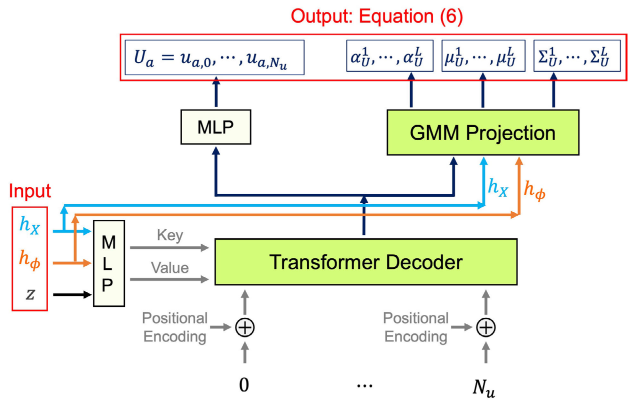


FIGURE 7. Schematic of the control decoder architecture, illustrating the process of generating the control sequence from the latent sample z and the encoded information h_X, h_ϕ.

process encapsulates the network's capability to synthesize control sequences that align with the specified conditions and objectives.

The architecture of the proposed control decoder model, depicted in Figure 7, which generates the output control sequence. Given a latent sample z and the encoded information h_X, h_ϕ, the control decoder synthesizes the control sequence for a set duration in a single step, adopting a non-autoregressive model for efficiency and coherence.

In the proposed model, a Transformer decoder is utilized where time information, encoded as sinusoidal positional encodings, serves as the query. Simultaneously, the latent vector, combined with the encoded state image and LTL formula features, acts as the key and value in the decoder. This configuration enables the decoder to produce an anchor control sequence $U_a = u_{a,0}, \cdots, u_{a,N_u}$, along with GMM components.

The generation of the control sequence distribution employs a probabilistic model, described by the equation below [31], which projects input features to a parameterized space of control sequences:

$$p_\theta(U|z, h_X, h_\phi) = \sum_{l=1}^{L} \alpha_U^l \mathcal{N}\big(U|U_a + \mu_U^l, \Sigma_U^l\big). \quad (6)$$

In this formulation, α_U^l, μ_U^l, and Σ_U^l represent the mixture coefficients, means, and covariances of the GMM, respectively, derived from the control decoder. The parameter L specifies the number of mixture components, encapsulating the model's capacity to represent complex control sequence distributions.

This approach integrates the latent variable z, and historical data h_X and h_ϕ, through the GMM to delineate a probabilistic space where the potential control sequences are distributed. The anchor control sequence U_a serves as a reference from which the distribution is centered, facilitating the identification of feasible control sequences by contextualizing them within the operational domain. Rather than predetermining the outcome, U_a and the GMM enable the prediction of control sequences that are adaptable to varying conditions and uncertainties inherent in environments.

To ensure clarity and maintain focus on the model's predictive capability, we present the parameters of the GMM directly, without delving into the underlying functional dependencies on θ. This direct presentation underscores the model's utility in forecasting control sequences that are both feasible and optimized, based on the computed probability distribution.

C. DEFINING THE LOSS FUNCTION DURING THE TRAINING PHASE

The training of our CVAE is governed by the Evidence Lower Bound (ELBO) loss function, initially formulated as:

$$\mathbb{E}_{q_\nu(Z|h_X,h_\phi,h_U)}[\log p_\theta(U|z,h_X,h_\phi)] - \mathcal{D}_{KL}\left(q_\nu(Z|h_X,h_\phi,h_U)||p_\theta(Z|h_X,h_\phi)\right). \quad (7)$$

We adapt the ELBO function to a form more suited to our application's specific requirements. The customized loss function is defined as:

$$-\sum_{i=0}^{N_u}\log\left(p_\theta(u_i|z,h_X,h_\phi)\right) + \lambda\cdot\mathcal{D}_{KL}\Big(\mathcal{N}\left(\mu_\nu(h_X,h_\phi,h_U),\,\Sigma_\nu(h_X,h_\phi,h_U)\right)||\mathcal{N}(0,I)\Big), \quad (8)$$

where u_i represents an element of the control sequence U, and λ is a scaling factor introduced to balance the terms. The $\mathcal{D}_{KL}$ (Kullback-Leibler divergence) measures how one probability distribution diverges from a second probability distribution. We set $\lambda = 1$, optimizing parameters ν and θ by minimizing this loss function. The first term of Equation 8, leveraging Equation 6, is detailed as follows:

$$\log\left(p_\theta(u_i|z,h_X,h_\phi)\right) = \sum_{l=1}^{L}\left[\log\alpha_U^l + \log\mathcal{N}\left(u_i|u_{a,i}+\mu_U^l,\,\Sigma_U^l\right)\right], \quad (9)$$

with $u_{a,i}$ being an element of the anchor control sequence U_a. This formulation embodies our methodology for estimating the probability distribution of control sequences, which is pivotal in ensuring that the model accommodates the diversity of potential solutions and manages uncertainties with efficacy.

D. TEST PHASE

The test phase of our deep learning architecture is illustrated in Figure 8. During this phase, the control decoder operates by accepting a latent sample z, which is drawn from the prior distribution. It then deterministically transforms this sample into a predicted control sequence $\hat{U}$. Importantly, the generation of the control sequence continues until one of the following conditions is met: the sequence fulfills the specified LTL constraints, encounters a collision with obstacles, or exceeds the boundaries defined by the cost map.

This process effectively incorporates encoded state information alongside the LTL specifications, thereby enabling the network to produce control sequences that not only adhere to the required temporal logic constraints but also optimize for cost efficiency. The integration of these elements ensures that the resulting trajectories are viable within the operational environment, balancing adherence to LTL specifications with practical navigational requirements.

V. EXPERIMENTAL RESULTS

This section presents the outcomes from a series of simulations and experiments utilizing the Dubins car model as the dynamic model. The model follows the kinematic equations:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \omega, \quad (10)$$

where (x, y) denotes the car's position, θ the heading, and v and ω the linear and angular velocities, respectively. It's important to clarify that the symbol θ is also utilized elsewhere in this manuscript to denote different concepts, distinct from its use here as the vehicle's heading.

Using Gaussian process regression, we generated costmaps for training, ensuring that each map contained no more than four regions of interest and no more than eight obstacles. Employing the dynamic model defined by Equation 10, we computed near-optimal control sequences in accordance with the method described in [23]. The resulting dataset comprised 750 costmaps, with each facilitating the generation of 1200 control sequences, thereby capturing a wide array of environmental scenarios.

To enhance the robustness and generalizability of our model, we introduced variability in the data generation phase. This was achieved by randomly varying the starting positions, the placements of regions of interest, obstacle configurations, and the assignments of LTL formulas for each data instance. Such a strategy aims to simulate a diverse array of potential operational scenarios, preparing the model to handle a wide range of conditions effectively.

For the network input, we standardized images to 128×128 pixels, aligning with the constraints imposed by our GPU hardware's memory capacity. This dimensionality strikes a balance between retaining necessary environmental details and maintaining computational feasibility. The network underwent training over 300 epochs to ensure adequate learning depth, with a batch size of 32 selected to optimize the balance between memory usage and convergence stability. An initial learning rate of 1e−3 and a weight decay of 1e−5 were empirically determined to provide a suitable compromise between training speed and the minimization of overfitting risks. Furthermore, the decision to use eight components ($L = 8$) in the GMM was based on experimental validation, which indicated this as a reasonable number for capturing the complexity of control sequence distributions within our model framework.

We conducted three distinct sets of simulation experiments to comprehensively evaluate the proposed method's effectiveness and versatility. The first set utilized generated costmaps to test the method in controlled environments, focusing on its ability to navigate based on cost efficiency. The second set applied real-world data, incorporating a

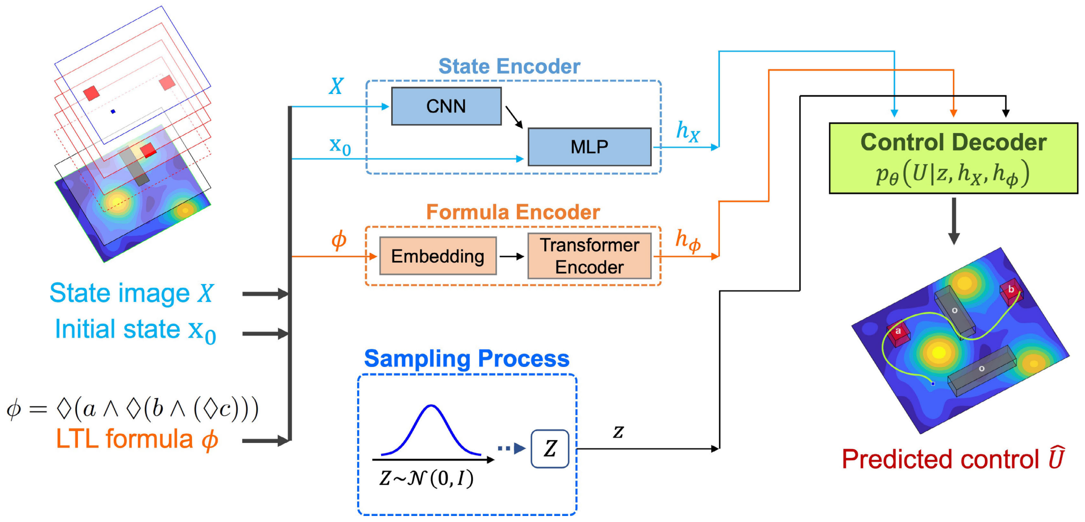


FIGURE 8. **Test phase architecture of the proposed system, demonstrating the process of converting latent samples into the predicted control sequences.**

traffic accident density map from Helsinki, to demonstrate the method's applicability and performance in real-world scenarios. The third set of experiments was designed to showcase the proposed method's capability to generate near-optimal solutions.

A. THE GENERATED COSTMAP

Figure 9 showcases the test costmap, synthesized using the same methods as the training costmaps. The costmap displays regions of interest marked with red boxes labeled with alphabetic identifiers, while obstacles are indicated by grey boxes.

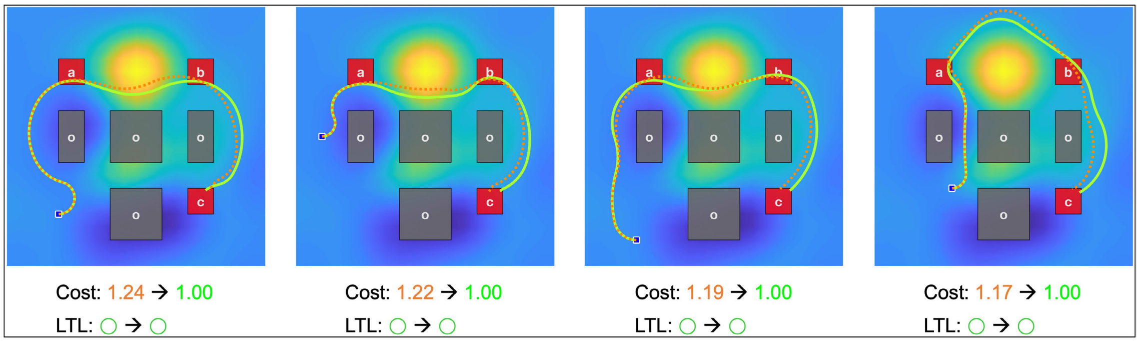
In our experimental setup, we aimed to assess the robustness and effectiveness of our system in executing a variety of LTL missions. These missions were categorized into sequential missions (ϕ_1 and ϕ_2) and coverage missions (ϕ_3 and ϕ_4). A key aspect of this evaluation involved monitoring for potential failure modes, such as collisions with obstacles or deviations from the designated costmap areas. Such events were critical for determining the mission's success or incompleteness. Specifically, a mission was deemed incomplete if the system either exceeded the maximum allowed sequence length or encountered a collision, thereby failing to meet the mission criteria. To visually represent these instances, collisions observed during the experiments are highlighted with red dotted circles in Figure 9 (subsections (b) and (c)). This inclusion serves to illustrate the system's interaction with complex environments and the challenges posed by obstacle avoidance and mission adherence.

Each row in the figure corresponds to a set of four subfigures, with each subfigure varying the initial position. Two control sequence solutions are displayed for comparison in each subfigure: the anchor control sequence U_a from the control decoder module is shown with an orange line, while the green line represents the final sampled solution from the GMM of the control decoder module. The cost of the final solution is normalized to one, with the cost of the anchor control sequence expressed in relation to this value.

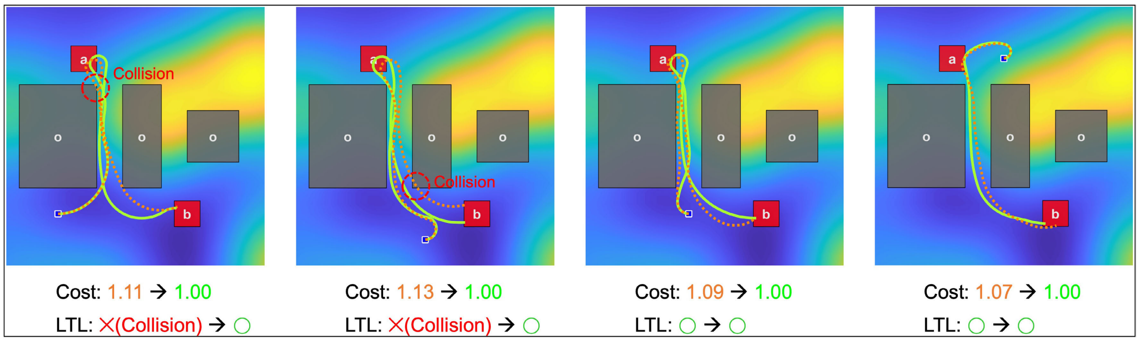
The solutions were generated to traverse low-cost areas on the costmap, depicted in blue, striving to complete the LTL missions effectively. For coverage missions, as seen in Figures 9(c) and 9(d), the solution paths differ in the order they visit regions of interest, varying according to the starting position. It is observed that the GMM projection in the control decoder module enhances the solution quality, in terms of cost and LTL mission fulfillment, when compared to the anchor control sequence.

Figure 10 presents solution trajectories generated by the proposed network for an LTL mission specified by $\phi = \Diamond(a) \wedge \Diamond(b) \wedge \Diamond(c)$, which mandates visiting regions of interest a, b, and c at least once. Each subfigure corresponds to trajectories initiated from different starting positions, illustrating the adaptability of the network to various initial states. In this figure, trajectories colored identically are derived from the same latent sample value. The latent distribution encapsulates the sequence in which regions of interest are visited, ensuring LTL satisfaction, while the GMM component of the control decoder models uncertainty within this framework. For instance, the green trajectories in subfigure (a) represent a sequence of visiting b, followed by a, then c. In contrast, the orange trajectories in subfigure (b) depict a sequence of visiting a, b, and then c. This variation demonstrates the network's capability to generate diverse solutions that adhere to the given LTL mission while accounting for uncertainty.

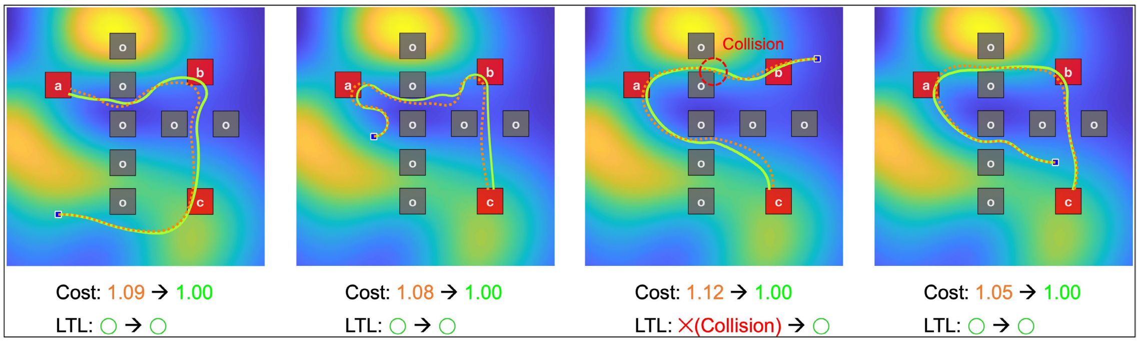
Performance evaluation of the developed path planning approach was carried out through comparative experiments. These experiments aimed to quantify trajectory cost and mission success rates across a variety of scenarios characterized by differing lengths of sequential LTL formulas ($|\phi|$) and obstacle counts (n_{obs}). Here, the length of the sequential LTL formula corresponds to the number of regions of interest it specifies.



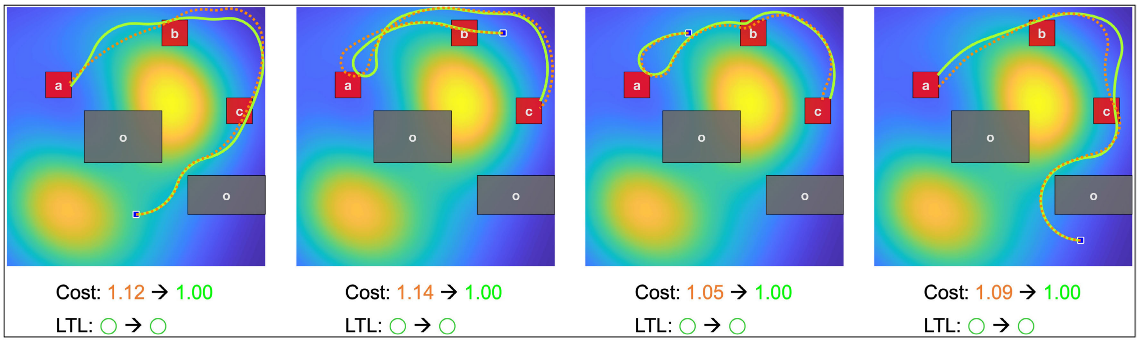
(a) $\phi_1 = \lozenge(a \wedge \lozenge(b \wedge (\lozenge c)))$



(b) $\phi_2 = \lozenge(a \wedge \lozenge(b))$



(c) $\phi_3 = \lozenge(a) \wedge \lozenge(b) \wedge \lozenge(c)$



(d) $\phi_4 = \lozenge(a) \wedge \lozenge(b) \wedge \lozenge(c)$

FIGURE 9. Comparative results generated by the proposed control sequence generation method for different LTL formulas, denoted as ϕ_i. In each subfigure, two trajectories are shown: the anchor control sequence U_a is depicted by an orange line, representing a baseline solution from the control decoder module, and the green line indicates the optimized sampled solution derived from the GMM of the control decoder module. The costmaps illustrate cost gradients, with blue zones signifying lower costs and yellow indicating higher costs.

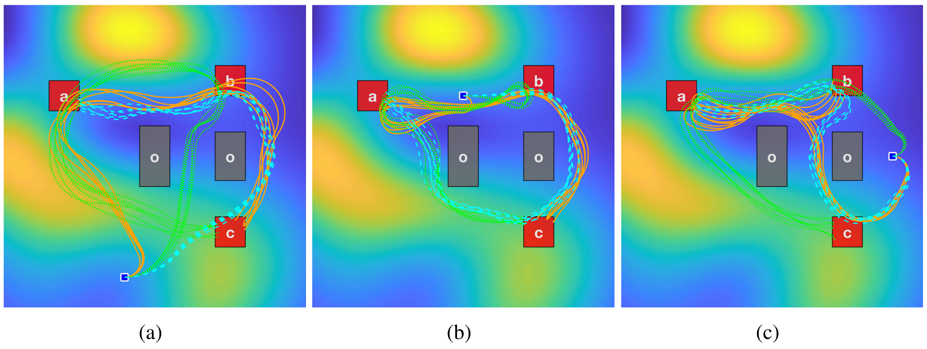


FIGURE 10. Solution trajectories for an LTL mission $\phi = \Diamond(a) \wedge \Diamond(b) \wedge \Diamond(c)$, requiring at least one visit to each region of interest *a*, *b*, and *c*. Trajectories sharing the same color originate from the same latent sample value, illustrating the network's approach to fulfilling the mission criteria from different starting points.

The experimental design included 500 trials per scenario, encompassing variables such as costmap configurations, region of interest placements, obstacle locations, initial positions, and LTL formulas. To ensure a rigorous assessment, these elements were systematically varied within each trial, providing a comprehensive evaluation of the approach's robustness. Trials lacking feasible solution paths were excluded to maintain the experiment's integrity.

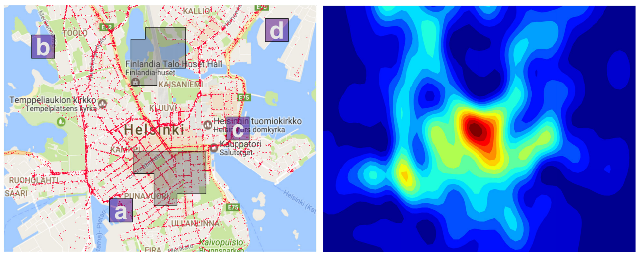
The method's performance was benchmarked against several established deep learning networks:

- MLP: A fundamental Multilayer Perceptron architecture.
- Seq2Seq-LSTM: A sequence-to-sequence model utilizing LSTM networks [38].
- TCN: A Temporal Convolutional Network [39].
- TFN: A Transformer network model [13].
- CVAE: A Conditional Variational Autoencoder approach [14].

These models were trained to learn the mapping from initial conditions and LTL formulas to control sequences, with a CNN feature extractor handling image-like inputs. The LTL formula ϕ was encoded as an input sequence using the same embedding technique applied in the proposed method. Additionally, LBPP-LTL [23], a sampling-based path planning algorithm known for its longer computation times but also for guaranteeing asymptotic optimality, was included as a benchmark for cost performance despite its computational intensity.

The results, summarized in Table 1, present the average trajectory cost and mission success rate for each method in the evaluated scenarios. The LBPP-LTL method serves as a baseline with its trajectory cost normalized to 1, providing a standard for comparison despite its computational demands.

The experimental findings indicate that the proposed method outperforms other deep learning-based path planning techniques in terms of cost efficiency and success rate for missions defined by LTL. This superior performance can be primarily attributed to two novel aspects of the proposed method: (1) the adoption of advanced transformer networks



(a) Traffic accident map of Helsinki. (b) Traffic accident density map.

FIGURE 11. Visual representation of Helsinki's traffic landscape: (a) Traffic accidents are marked with red dots, regions of interest are alphabetically labeled, and gray areas denote obstacles. (b) A traffic accident density map, where blue indicates low-density (low-cost) zones and red indicates high-density (high-cost) zones.

for accurate sequence prediction, and (2) the effective incorporation of diversity and uncertainty into the path planning process through latent space modeling paired with GMMs. Although the LBPP-LTL algorithm demonstrates superior trajectory cost and LTL mission success rates, its practicality is moderated by the requirement for extensive computational resources. These results highlight the capability of the proposed method to reliably approximate optimal solutions with reduced computational demands.

B. THE HELSINKI TRAFFIC ACCIDENT MAP

This section presents an examination of autonomous navigation for traffic surveillance within a designated area of Helsinki, as depicted in Figure 11. Four regions of interest are demarcated with alphabetic labels, while obstacles are represented as gray rectangles.

To facilitate path planning, a synthesized traffic accident density map was created using Gaussian process regression based on historical traffic accident data [40]. This map categorizes areas with high accident density as high-cost and those with lower density as low-cost, influencing the path planning algorithm's cost assessments.

The study defines four distinct scenarios, each with a unique mission profile. Scenarios 1 and 2 focus on sequential

TABLE 1. Comparative Performance of Path Planning Approaches on Scenarios with Sequential LTL Missions. The table reports trajectory costs and LTL mission success rates, categorized by the length of LTL missions (|ϕ|) and the number of obstacles (n_{obs}). Metrics are normalized to the LBPP-LTL benchmark (with normalized trajectory cost of 1 and mission success rate of 100%). The proposed method demonstrates robust efficacy across the scenarios.

	$\|\phi\| = 2, n_{obs} = 1$	$\|\phi\| = 3, n_{obs} = 1$	$\|\phi\| = 2, n_{obs} = 2$	$\|\phi\| = 3, n_{obs} = 2$	$\|\phi\| = 3, n_{obs} = 3$
Trajectory cost (relative)					
MLP	1.395	1.423	1.411	1.462	1.496
Seq2Seq-LSTM	1.181	1.191	1.185	1.285	1.314
TCN [39]	1.186	1.199	1.190	1.292	1.305
TFN [13]	1.091	1.112	1.104	1.133	1.179
CVAE [14]	1.295	1.303	1.299	1.324	1.397
Proposed	**1.087**	**1.095**	**1.093**	**1.123**	**1.168**
LBPP-LTL [23]	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
LTL mission success rate					
MLP	92.0%	91.6%	90.4%	89.0%	88.6%
Seq2Seq-LSTM	96.8%	95.6%	93.8%	92.4%	91.2%
TCN [39]	96.2%	95.0%	93.2%	91.8%	90.8%
TFN [13]	**97.6%**	96.2%	95.2%	94.8%	92.6%
CVAE [14]	94.8%	93.8%	93.2%	92.6%	90.8%
Proposed	97.2%	**96.8%**	**96.0%**	**95.2%**	**93.2%**
LBPP-LTL [23]	**100%**	**100%**	**100%**	**100%**	**100%**

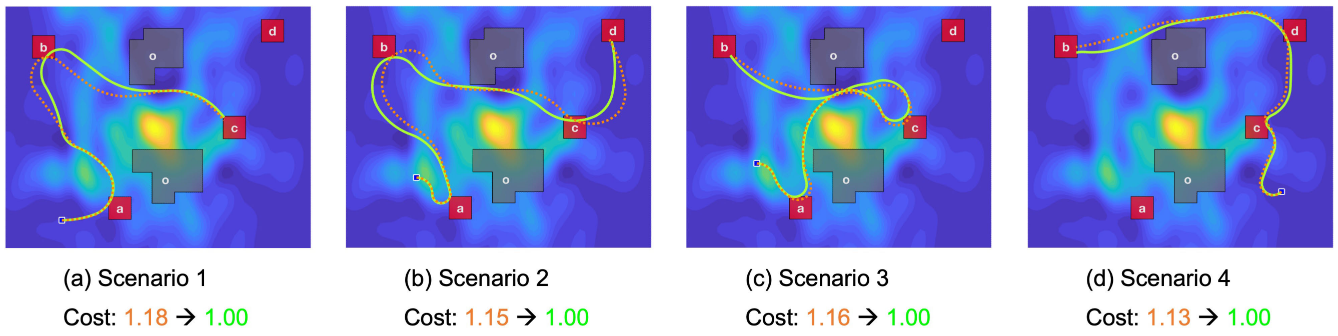


FIGURE 12. Solution paths on the Helsinki traffic scenario map, with the initial anchor control sequences presented in orange and the optimized GMM-influenced solutions in green. The accompanying annotations indicate cost improvements, with all cost metrics normalized relative to the cost of the final solution.

navigation missions, while Scenarios 3 and 4 are based on area coverage. The corresponding LTL formulas defining the mission objectives are as follows:

$$\text{Scenario 1} : \phi_1 = \Diamond(a \wedge \Diamond(b \wedge (\Diamond c))).$$
$$\text{Scenario 2} : \phi_2 = \Diamond(a \wedge \Diamond(b \wedge \Diamond(c \wedge (\Diamond d)))).$$
$$\text{Scenario 3} : \phi_3 = \Diamond(a) \wedge \Diamond(b) \wedge \Diamond(c).$$
$$\text{Scenario 4} : \phi_4 = \Diamond(b) \wedge \Diamond(c) \wedge \Diamond(d).$$

These LTL formulas articulate the mission goals for the autonomous agent. For example, ϕ_1 stipulates that the agent must visit regions a, b, and c in any order, whereas ϕ_2 adds region d to the sequence. The formulas for Scenarios 3 and 4 imply the agent is required to visit specified regions without dictating a particular sequence, focusing on coverage.

Figure 12 illustrates the trajectories computed by the proposed algorithm for various scenarios within the Helsinki traffic framework. Echoing the format seen in Figure 9, each subfigure contrasts two control sequences: the provisional anchor control sequence U_a derived from the control decoder module is delineated by an orange line, while the refined trajectory, influenced by the GMM within the same module, is traced in green. In line with previous experiments, trajectory costs are normalized against the cost of the final, refined solution, which is established as a baseline metric of 1. Annotations accompanying the visual data highlight the cost metric adjustments, emphasizing the algorithm's ability to improve cost efficiency through its refinement process. These graphical depictions reinforce the algorithm's adeptness at optimizing trajectories, passing lower-cost regions and meeting the specified LTL mission objectives efficiently.

C. OPTIMALITY ANALYSIS

To assess the performance of our proposed algorithm, we conducted experiments focused on optimality in path planning (as illustrated in Figure 13). The employed costmap, represented in Figure 13, features a concentric low-cost area (depicted in blue), which strategically challenges the algorithm to navigate efficiently.

We positioned the regions of interest (a, b, and c) adjacent to the upper boundary of the low-cost region to encourage

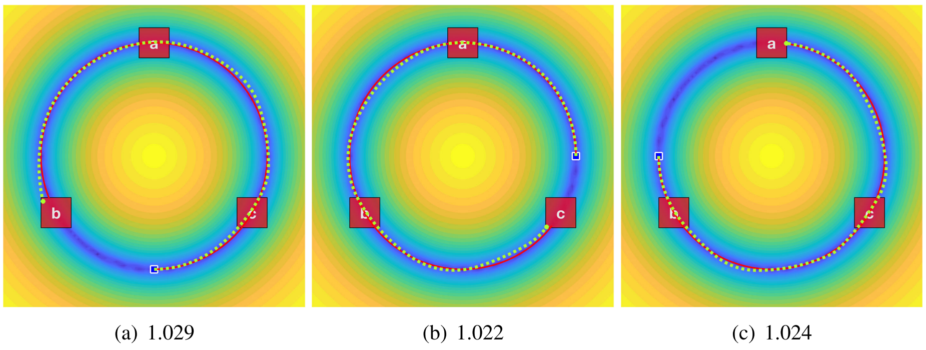


FIGURE 13. Solution trajectories for an LTL mission $\phi = \lozenge(a) \wedge \lozenge(b) \wedge \lozenge(c)$, mandating a visit to each region of interest *a*, *b*, and *c*. Each subfigure illustrates the generated solution (dashed green line) alongside the optimal solution (solid red line), starting from different initial positions.

the algorithm to find the most cost-effective path that still satisfies the mission requirements. The specific LTL mission formula we used was $\phi = \lozenge(a) \wedge \lozenge(b) \wedge \lozenge(c)$, necessitating at least one visit to each region of interest.

Figure 13 showcases the results of the algorithm's path planning with each subfigure depicting the generated solution (illustrated by a dashed green line) against the optimal solution (depicted in red), originating from varied initial states. The relative cost of the generated solutions is indicated beneath each subfigure, with the cost of the optimal solution normalized to 1. The outcomes indicate that the paths generated by our algorithm not only traverse the low-cost areas but also closely approximate the optimal solution.

VI. CONCLUSION

This study presented an innovative path planning approach that effectively integrates co-safe LTL specifications with an end-to-end deep learning architecture. Our method stands out by generating near-optimal control sequences through the synergy of a transformer encoder informed by LTL requirements and a variational autoencoder enhanced with GMM components. This architecture navigates the complexities of path planning by embracing both the diversity of tasks and the inherent uncertainties. Empirical evaluations highlight our approach's advantages over similar deep learning strategies. Its adaptability and scalability confirm the method's suitability for a wide array of systems, enhancing path planning processes significantly.

Looking ahead, we aim to apply our methodology to more challenging high-dimensional path planning problems, particularly those that include additional logical constraints and intricate operational contexts, like multi-joint robotic manipulations. Focusing on these areas will likely yield further valuable insights into robotics and automation, enhancing both the sophistication and efficiency of path planning.

The fusion of deep learning with logical frameworks in our study signifies a notable advancement in robotic path planning, setting the stage for more complex and efficient mission executions in the future.

ACKNOWLEDGMENT

(Chaeeun Yang and Sojeong Yoon are co-first authors.)

REFERENCES

[1] È. Pairet, C. Chamzas, Y. Petillot, and L. E. Kavraki, "Path planning for manipulation using experience-driven random trees," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3295–3302, Apr. 2021.

[2] F. Lamiraux and J. Mirabel, "Prehensile manipulation planning: Modeling, algorithms and implementation," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2370–2388, Aug. 2022.

[3] K. Xu, H. Yu, R. Huang, D. Guo, Y. Wang, and R. Xiong, "Efficient object manipulation to an arbitrary goal pose: Learning-based anytime prioritized planning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 7277–7283.

[4] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 902–911, Aug. 2009.

[5] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path planning in dynamic environments using generative RNNs and Monte Carlo tree search," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 10263–10269.

[6] J. Lin, T. Zhou, D. Zhu, J. Liu, and M. Q.-H. Meng, "Search-based online trajectory planning for car-like robots in highly dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 8151–8157.

[7] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005.

[8] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc. 48h IEEE Conf. Decision Control, 28th Chin. Control Conf.*, Dec. 2009, pp. 2222–2229.

[9] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 3227–3232.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[11] H. Wang, P. Cai, Y. Sun, L. Wang, and M. Liu, "Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 13731–13737.

[12] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao, "ST-p3: End-to-end vision-based autonomous driving via spatial–temporal feature learning," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2022, pp. 533–549.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–7.

[14] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015.

[15] S. L. Smith, J. Tumová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Robot. Res.*, vol. 30, no. 14, pp. 1695–1708, Oct. 2011.

[16] E. Wolff, U. Topcu, and R. Murray, "Optimal control with weighted average costs and temporal logic specifications," in *Proc. Robot., Sci. Syst.*, Jul. 2012.

[17] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 735–742.

[19] V. Varricchio, P. Chaudhari, and E. Frazzoli, "Sampling-based algorithms for optimal motion planning using process algebra specifications," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 5326–5332.

[20] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2689–2696.

[21] E. Plaku, "Planning in discrete and continuous spaces: From LTL tasks to robot motions," in *Proc. Towards Auto. Robotic Syst.*, Aug. 2012, pp. 331–342.

[22] J. McMahon and E. Plaku, "Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 3726–3733.

[23] K. Cho, "Learning-based path planning under Co-safe temporal logic specifications," *IEEE Access*, vol. 11, pp. 25865–25878, 2023.

[24] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. 47th IEEE Conf. Decis. Control*, Dec. 2008.

[25] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based control of nonlinear systems with linear temporal logic specifications," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 5319–5325.

[26] S. C. Livingston, E. M. Wolff, and R. M. Murray, "Cross-entropy temporal logic motion planning," in *Proc. 18th Int. Conf. Hybrid Syst., Comput. Control*, Apr. 2015.

[27] K. Cho, "A hierarchical learning approach to autonomous driving using rule specifications," *IEEE Access*, vol. 10, pp. 74815–74824, 2022.

[28] J. J. Aloor, J. Patrikar, P. Kapoor, J. Oh, and S. Scherer, "Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 1320–1326.

[29] Y. Wang, N. Figueroa, S. Li, A. Shah, and J. Shah, "Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations," 2022, *arXiv:2206.04632*.

[30] A. Dhonthi, P. Schillinger, L. Rozo, and D. Nardi, "Optimizing demonstrated robot manipulation skills for temporal logic constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 1255–1262.

[31] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," 2019, *arXiv:1910.05449*.

[32] M. Petrovich, M. J. Black, and G. Varol, "Action-conditioned 3D human motion synthesis with transformer VAE," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10965–10975.

[33] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022.

[34] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA, USA: MIT Press, 2005.

[35] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[36] A. P. Sistla, "Safety, liveness and fairness in temporal logic," *Formal Aspects Comput.*, vol. 6, no. 5, pp. 495–511, Sep. 1994.

[37] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods Syst. Design*, vol. 19, no. 3, pp. 291–314, Nov. 2001.

[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014.

[39] S. Bai, J. Zico Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[40] (2011). *Traffic Accidents in Helsinki*. Accessed: Mar. 23, 2024. [Online]. Available: http://www.hri.fi/en/dataset/liikenneonnettomuudet-helsingissa

CHAEEUN YANG is currently pursuing the bachelor's degree with the Department of Information and Telecommunication Engineering, Incheon National University, Incheon, South Korea. With a keen interest in the intersection of technology and automation, her research endeavors center on intelligent systems and autonomous driving, particularly in the application of machine learning techniques to enhance vehicular autonomy.

SOJEONG YOON is currently pursuing the bachelor's degree with the Department of Information and Telecommunication Engineering, Incheon National University, Incheon, South Korea. Her academic pursuits are in the realm of digital media, with a specific focus on character animation. She is dedicated to exploring and advancing the application of machine learning algorithms to create more lifelike and dynamic character animations.

KYUNGHOON CHO received the B.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea, in 2014 and 2020, respectively. He is currently an Assistant Professor with the Department of Information and Telecommunication Engineering, Incheon National University, Incheon, South Korea. His primary research interests include optimal control, intelligent systems, and autonomous navigation.