electronics

MDPI

Article

Mission-Conditioned Path Planning with Transformer Variational Autoencoder

Kyoungho Lee †, Eunji Im † and Kyunghoon Cho *

Department of Information and Telecommunication Engineering, Incheon National University, Incheon 22012, Republic of Korea; kno022603@inu.ac.kr (K.L.); twintwin0243@inu.ac.kr (E.I.)

* Correspondence: ckh0923@inu.ac.kr

† These authors contributed equally to this work.

Abstract: This paper introduces a novel deep learning framework for robotic path planning that addresses two primary challenges: integrating mission specifications defined through Linear Temporal Logic (LTL) and enhancing trajectory quality via cost function integration within the configuration space. Our approach utilizes a Conditional Variational Autoencoder (CVAE) to efficiently encode optimal trajectory distributions, which are subsequently processed by a Transformer network. This network leverages mission-specific information from LTL formulas to generate control sequences, ensuring adherence to LTL specifications and the generation of near-optimal trajectories. Additionally, our framework incorporates an anchor control set—a curated collection of plausible control values. At each timestep, the proposed method selects and refines a control from this set, enabling precise adjustments to achieve desired outcomes. Comparative analysis and rigorous simulation testing demonstrate that our method outperforms both traditional sampling-based and other deep-learning-based path-planning techniques in terms of computational efficiency, trajectory optimality, and mission success rates.

Keywords: deep-learning-based control synthesis; formal methods; mission-based path planning

Citation: Lee, K.; Im, E.; Cho, K. Mission-Conditioned Path Planning with Transformer Variational Autoencoder. *Electronics* **2024**, *13*, 2437. https://doi.org/10.3390/electronics13132437

Academic Editors: Luis Gracia, Juan Ernesto Solanes Galbis and Jaime Valls Miro

Received: 10 May 2024
Revised: 18 June 2024
Accepted: 19 June 2024
Published: 21 June 2024

1. Introduction

Path planning is a cornerstone of robotics, evolving from simple two-dimensional navigation to addressing more complex systems such as robot manipulators [1–3] and challenging scenarios [4–6]. This evolution underscores the necessity for sophisticated path-planning algorithms capable of navigating both the physical aspects of environments and the intricate requirements of diverse tasks.

Translating mission specifications, often articulated in human language, into computational models presents a significant challenge in path planning. Formal methods such as Linear Temporal Logic (LTL), Computation Tree Logic (CTL), and μ-calculus are pivotal in this area. LTL, in particular, is favored for its flexibility and expressive power in defining complex missions [7–9], offering a structured yet adaptable framework for encoding mission objectives.

Additionally, the quest for trajectories that balance cost-effectiveness with computational efficiency is critical. For instance, in environments with variable communication strengths, it is crucial to find low-cost paths that minimize exposure to areas with poor connectivity. Traditional methods like Rapidly Exploring Random Tree Star (RRT*) [10] are effective but can be computationally demanding, especially under numerous constraints.

The integration of deep learning into path planning offers a promising alternative, excelling in deriving optimal paths directly from data, thus mitigating the computational drawbacks of conventional methods. These techniques have broadened their utility across various domains, enhancing control strategies for robot manipulators [1] and addressing

complex challenges in autonomous vehicle navigation [11,12]. The versatility and computational efficiency of deep learning approaches continue to propel advancements in the field of robotics.

This paper introduces a novel deep learning framework for robotic path planning that seamlessly integrates Linear Temporal Logic (LTL) formulas for mission specification with advanced trajectory optimization techniques. Our model employs a Conditional Variational Autoencoder (CVAE) and a Transformer network to innovatively generate control sequences that adhere to LTL specifications while optimizing cost efficiency. This integration marks a significant advancement in the fusion of deep learning with formal methods for path planning.

Key contributions of our approach include:

1. Application of the Transformer Network:We utilize the Transformer network to interpret LTL formulas and generate control sequences [13]. This allows for the effective handling of complex mission specifications.
2. Conditional Variational Autoencoder (CVAE): The CVAE is employed to navigate complex trajectory manifolds [14], providing the capability to generate diverse and feasible paths that meet the mission requirements.
3. Anchor Control Set: Our framework includes an anchor control set—a curated collection of plausible control values. At each timestep, the method selects and finely adjusts a control from this set, ensuring precise trajectory modifications to achieve desired objectives.
4. Incorporation of a Gaussian Mixture Model (GMM): The integration of a GMM to refine outputs enhances our framework's capacity to handle uncertainties, thereby improving both the precision and reliability of path planning under LTL constraints.

These contributions collectively advance efficient robotic path planning by providing near-optimal solutions that satisfy given LTL formulas.

As illustrated in Figure 1, our method synthesizes a control sequence distribution, enhanced by a GMM, for a given test scenario that adheres to the LTL formula $\phi = \Diamond(a \wedge \Diamond(b \wedge (\Diamond c)))$. This formula requires sequentially visiting regions a, b, and c. The figure displays trajectories sampled from the output control sequence distribution generated by the proposed approach. It is notable that these trajectories navigate through low-cost areas (depicted in blue) while avoiding obstacles and fulfilling the specified LTL requirements.

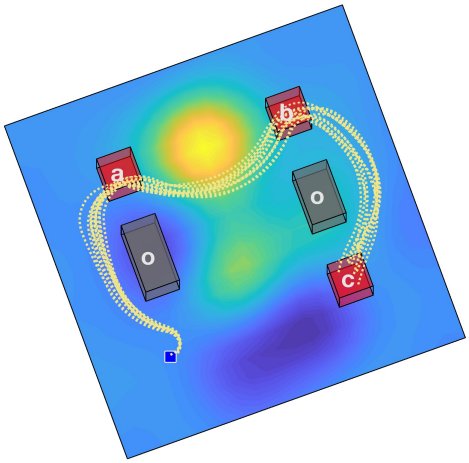


Figure 1. Illustrative example of trajectories generated using the proposed method in a test scenario. The mission, specified by the Linear Temporal Logic (LTL) formula $\phi = \Diamond(a \wedge \Diamond(b \wedge (\Diamond c)))$, requires sequential visits to regions a, b, and c.

Our contributions establish new benchmarks for cost efficiency and computational performance in robotic path planning. The effectiveness and superiority of our model

compared to existing deep-learning-based strategies are demonstrated through rigorous comparative simulations, showcasing its potential to significantly influence the field.

2. Related Work

Path planning is a foundational element of robotics, requiring a balance between low-cost trajectories, complex dynamics, and precise mission specifications. The literature offers a diverse range of strategies addressing these challenges with varying degrees of success.

Finite Deterministic Systems: Research in finite deterministic systems has explored optimal controls with varied cost functions, such as minimax for bottleneck path problems [15] and weighted averages for cyclic paths [16]. However, these approaches often struggle in continuous path-planning scenarios due to limitations in integrating robot dynamics and the necessity for high-resolution discretization.

Sampling-based Motion Planning: Sampling-based methods, such as Rapidly Exploring Random Tree (RRT) [17], have addressed the integration of temporal logic and complex dynamics. The Rapidly Exploring Random Graph (RRG) [18] and Rapidly Exploring Random Tree Star (RRT*) [19] demonstrate utility in optimizing motion planning but face scalability and efficiency challenges as complexity increases.

Multi-layered Frameworks: Multi-layered frameworks that blend discrete abstractions with automata for co-safe LTL formulas [20–22] guide trajectory formation using sampling-based methods. Despite advancements, these approaches often rely heavily on geometric decomposition, which limits their computational efficiency.

Optimization Methods: Optimization techniques, especially those utilizing mixed-integer programming, aim to achieve optimal paths under LTL constraints [23,24]. Although effective, these methods encounter scalability issues when dealing with complex LTL formulas and a growing number of obstacles. The cross-entropy-based planning algorithm [25] enhances efficiency but also struggles with extensive LTL formulas.

Learning from Demonstration (LfD): LfD has increasingly integrated temporal logic to enhance autonomous behaviors, employing strategies such as Monte Carlo Tree Search (MCTS) adjusted with STL robustness values to enhance constraint satisfaction [26]. This integration illustrates LfD's potential in continuous control scenarios [27], with significant developments in blending formal task specifications within LfD skills using STL and black-box optimization for skill adaptation [28].

Trajectory Forecasting: Recent advances in trajectory forecasting have utilized deep learning to predict future movements based on past data, aligning closely with LfD principles. This research employs models such as Gaussian Mixture Models (GMMs) and Variational Autoencoders with Transformer architectures to produce action-aware predictions [29,30]. These approaches push the envelope toward models that seamlessly integrate global intentions with local movement strategies for improved adaptability and accuracy [31].

3. Preliminaries

This section introduces the foundational concepts and notations critical for understanding our approach to path planning under LTL specifications. Establishing a clear framework is essential for a comprehensive presentation of the system model, dynamics, and temporal logic that articulates the desired path properties. We will outline the mathematical formulations that underpin our system's model, explain the dynamics governing the system, and detail the principles of LTL that are crucial for defining and evaluating trajectory objectives. This groundwork is vital for understanding the complexities of autonomous systems and their operational criteria, preparing the ground for a detailed exploration of our proposed method.

3.1. 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:

$$\dot{x}_t = f(x_t, u_t), \tag{1}$$

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

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

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

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

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

3.2. Linear Temporal Logic (LTL)

LTL is a formalism used to express properties over linear time [32]. It utilizes atomic propositions (APs), Boolean operators, and temporal operators. An atomic proposition is a simple statement that is either true or false. Essential LTL operators include $\bigcirc$ (next), U (until), $\Box$ (always), $\Diamond$ (eventually), and $\Rightarrow$ (implication). The structure of LTL formulas adheres to a grammar outlined in [33].

In our framework, $\Pi = \{\pi_0, \pi_1, \ldots \pi_N\}$ denotes the set of all atomic propositions. An LTL trace, represented as σ, is a sequence of atomic propositions. LTL typically deals with infinite traces, with Σ^ω representing all possible infinite traces originating from $\Sigma = 2^\Pi$. A trace σ satisfies a formula ϕ if it is expressed as $\sigma \vDash \phi$.

For this study, we focus on finite-time path planning using syntactically co-safe LTL (sc-LTL) formulas [34], which are particularly suited for finite scenarios. A sc-LTL formula ϕ ensures that any infinite trace satisfying ϕ also has a finite prefix that satisfies ϕ. All temporal logic formulas discussed in this paper adhere to the sc-LTL format.

3.2.1. Automaton Representation

Given a set of atomic propositions Π and a syntactically co-safe LTL formula ϕ, a non-deterministic finite automaton (NFA) can be constructed [35]. For instance, for the formula $\phi = \Diamond(a \wedge \Diamond(b \wedge \Diamond(c)))$, an example of the resulting NFA is depicted in Figure 2. This NFA can be converted into a deterministic finite automaton (DFA), which is more suitable for computational processes. A DFA is described by the tuple $\mathcal{A}_\phi = (Q, \Sigma, \delta, q_{init}, Q_{acc})$, where:

- Q : Set of states
- $\Sigma = 2^\Pi$: Alphabet, where each letter is a set of propositions
- $\delta : Q \times \Sigma \to Q$: Transition function
- $q_{init} \subseteq Q$: Initial state(s)
- $Q_{acc} \subseteq Q$: Accepting states

A trace σ from a DFA is accepted if, at any point, it leads to one of the accepting states (i.e., $\sigma_i \cap Q_{acc} \neq \emptyset$). Thus, a trace satisfies the sc-LTL formula ϕ (denoted as $\sigma \vDash \phi$) if it is accepted by the DFA $\mathcal{A}_\phi$.

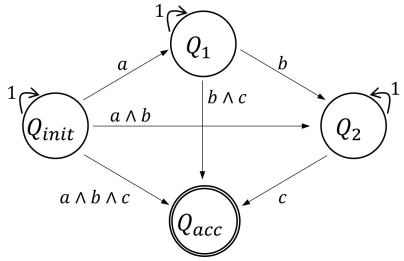


Figure 2. Example NFA for the sc-LTL formula $\phi = \Diamond(a \wedge \Diamond(b \wedge \Diamond(c)))$. The diagram illustrates four states and the transitions based on the input alphabets.

3.2.2. LTL Semantics over Trajectories

In this work, we define regions of interest within the workspace, $\mathcal{W}$, as $P = \{P_1, \ldots P_n\}$. These regions of interest are specified by the user. Each atomic proposition, π_j, from the set Π, corresponds to a specific region of interest, P_j. We employ a labeling function, $L : \mathcal{W} \rightarrow 2^{\Pi}$, to map each point in the workspace to a set of atomic propositions that are valid at that location. For any $\pi_i \in \Pi$, the negation $\neg\pi_i$ holds true for all points $\{w \in \mathcal{W} \mid \pi_i \notin L(w)\}$. Notably, π_0 remains true in all areas of the workspace except for the defined regions of interest and obstacles.

For a discretized trajectory, represented as $\mathbf{x}_{\Delta t}(x_0, \mathbf{u}) = \mathrm{x}_0, \mathrm{x}_1, \ldots \mathrm{x}_m$, which originates from x_0 and follows the control inputs $\mathbf{u}$ at each time step Δt, the trajectory trace can be defined as follows [20]:

$$trace(\mathbf{x}_{\Delta t}(\mathrm{x}_0, \mathbf{u})) = L(h(\mathrm{x}_0)), L(h(\mathrm{x}_1)), \ldots L(h(\mathrm{x}_m)). \tag{3}$$

This trace captures the sequence of atomic propositions valid at each point along the trajectory, reflecting the dynamic interaction with the workspace.

Figure 3 illustrates an example of a trajectory and its associated trace. This visual representation aids in understanding how the discrete segments of a trajectory map to their corresponding traces. For a given trajectory trace $trace(\mathbf{x}_{\Delta t}(\mathrm{x}_0, \mathbf{u})) = \tau_0, \tau_1, \ldots \tau_m$, we define the automaton state sequence $\mathcal{A}_\phi(trace(\mathbf{x}_{\Delta t}(\mathrm{x}_0, \mathbf{u}))) = q_0, q_1, \ldots q_m$ with each q_k specified as:

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

A trajectory $\mathbf{x}_{\Delta t}(\mathrm{x}_0, \mathbf{u})$ complies with the LTL formula ϕ, denoted by $\mathbf{x}(\mathrm{x}_0, \mathbf{u}) \vDash_{\Delta t} \phi$, if the automaton sequence reaches a subset of the accepting states Q_{acc}.

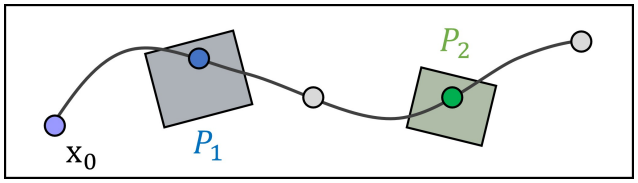


Figure 3. A trace defined over a discretized trajectory: For given $\mathbf{x}_{\Delta t}(\mathrm{x}_0, \mathbf{u}) = \mathrm{x}_0, \mathrm{x}_1, \ldots \mathrm{x}_5$, its trace is a sequence with 6 elements $\{\pi_0, \neg\pi_1, \neg\pi_2\}, \{\pi_0, \neg\pi_1, \neg\pi_2\}, \{\neg\pi_0, \pi_1, \neg\pi_2\}$, $\{\pi_0, \neg\pi_1, \neg\pi_2\}$, $\{\neg\pi_0, \neg\pi_1, \pi_2\}, \{\pi_0, \neg\pi_1, \neg\pi_2\}$.

4. Proposed Method

Our approach primarily focuses on optimizing the accumulated cost $J(\mathrm{x}_0, \mathbf{u})$, which is the line integral of a cost function c over a trajectory, mathematically expressed as:

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

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

This paper introduces a novel deep learning framework for robotic path planning that significantly advances the synthesis of near-optimal control sequences. Our method is designed to meet specific mission requirements, adhere to system dynamics (as defined in Equation (1)), and optimize cost efficiency (as outlined in Equation (5)).

At the core of our innovative approach is the integration of a Conditional Variational Autoencoder (CVAE) with Transformer networks, creating an end-to-end solution that represents a significant leap forward in path-planning research. The CVAE plays a crucial role in learning the distribution within the latent space of optimal control sequences, enabling the generation of sequences that satisfy LTL constraints while minimizing costs.

Our methodology leverages convolutional neural networks (CNNs) to transform environmental inputs—such as cost maps, regions of interest, and obstacle configurations—into an image-like format, thus optimizing the processing of spatial information.

A distinctive feature of our research is the introduction of an anchor control set. Instead of directly generating a control sequence, our method selects an appropriate control from the anchor control set at each timestep. This selection and subsequent refinement are facilitated by a Gaussian Mixture Model (GMM), which effectively accounts for environmental uncertainties. This approach enables more precise control predictions and enhances the system's robustness against dynamic and unpredictable conditions.

The process begins with the Transformer's decoder generating an initial anchor control value, which is then refined by the GMM to incorporate minor uncertainties. This integration, bolstered by learning from the latent distribution, significantly improves the precision and reliability of control sequence predictions. Our structured approach innovatively addresses uncertainties, thereby enhancing the robustness of our path-planning framework.

In summary, the proposed method offers several key advantages:

1. End-to-end approach: By employing a Transformer network to encode LTL formulas, our method eliminates the need for the discretization processes and graph representations typically required in previous studies. This streamlined approach simplifies the encoding and handling of complex specifications directly within the network architecture.
2. Step-by-step uncertainty consideration: Our approach meticulously addresses uncertainties within the path-planning process. The latent space is designed to account for major uncertainties, while the selection of controls from the anchor control set and their subsequent refinement through the GMM framework effectively manage minor uncertainties, enhancing the robustness and reliability of the trajectory planning.

Subsequent sections will delve deeper into our methodology, particularly focusing on the innovative implementation of anchor controls within the GMM framework and its profound implications. Through this detailed exploration, we aim to provide a clear understanding of the significant advancements our strategy introduces to the field of path planning.

4.1. Data Components

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

Our framework utilizes two primary data components: the state image X and the solution control sequence U. The state image X consists of multiple layers, each representing different environmental features in a format readily processed by the neural network. As illustrated in Figure 4, these layers include the costmap, obstacles, regions of interest, and the initial position, with each layer stacked to provide a comprehensive environmental context.

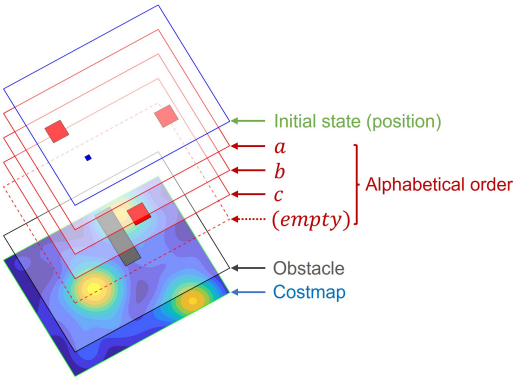


Figure 4. Configuration of the state image X for regions of interest $\{a, b, c\}$. Layers are sequentially arranged to depict the costmap, obstacles, regions of interest, and the initial state.

The generation of control sequences is guided by methodologies from our prior research [36], which align with our system dynamics as defined in Equation (1) and the specifications of co-safe temporal logic. Specifically, the adopted approach focuses on identifying low-cost trajectories that comply with given co-safe temporal logic specifications, with LTL semantics evaluated against the trajectories to ensure adherence to necessary specifications.

Our data generation strategy is specifically tailored to accommodate environmental constraints and LTL objectives relevant to our study. This strategy is depicted in Figure 5, illustrating how environmental parameters and LTL formulas are transformed into output control sequences. When the length of generated sequences falls short of the maximum designated length, dummy control values are utilized as placeholders to maintain uniform sequence length, aiding in standardizing the training process across different scenarios.

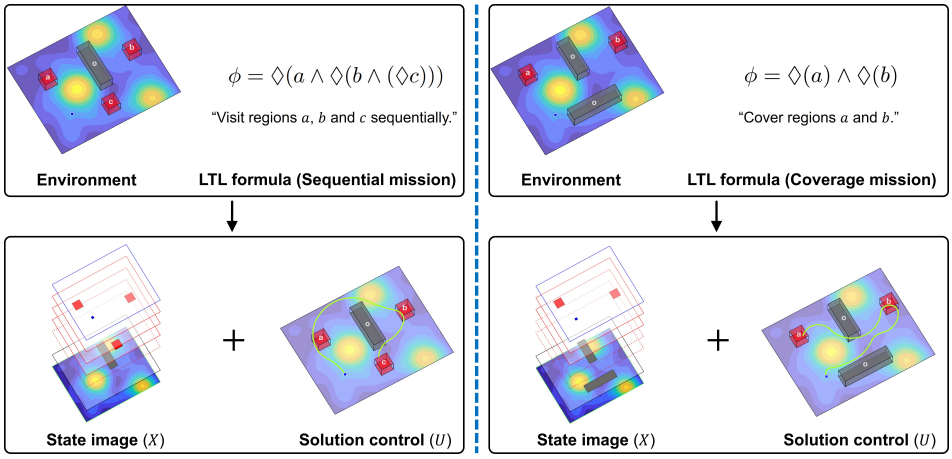


Figure 5. Visual representation of the data generation methodology, highlighting the integration of state images and control sequences derived from distinct LTL formulas.

4.2. Anchor Control Set

In our proposed model, we utilize an anchor control set $\mathbf{A} = \{\mathbf{a}^k\}_{k=1}^K$, consisting of a predefined, fixed set of control sequences. Each anchor control, $\mathbf{a}^k$, is a sequence of control inputs $[\mathrm{u}_0^k, \mathrm{u}_1^k, \ldots \mathrm{u}_{H_A-1}^k]$ that spans a designated time horizon H_A. It is important to note that this time horizon H_A is not necessarily equivalent to the time horizon of the final solution's control sequence.

The use of longer sequences for each anchor control, rather than single timestep controls, is motivated by the difficulty of capturing significant behavioral trends with overly granular, single-step data. In the control generation stage of our network, which will be elaborated on in subsequent sections, controls are selected in bundles of H_A rather

than individually per timestep. This approach simplifies the decoder's prediction tasks compared to methods that operate on a per-timestep basis, thereby enhancing both the efficiency and efficacy of our model.

To construct the anchor control set, we derived it from a dataset using the k-means clustering algorithm. The distance between two anchor controls, necessary for the clustering process, is calculated as follows [29]:

$$d(\mathbf{a}^i, \mathbf{a}^j) = \sum_{t=0}^{H_A} \|\mathbf{u}_t^i - \mathbf{u}_t^j\|^2. \tag{6}$$

This metric ensures that the controls within our set are optimally spaced to effectively cover diverse potential scenarios.

4.3. Proposed Architecture

The architecture of our proposed deep learning network is depicted in Figure 6. This comprehensive end-to-end system processes inputs from the environmental image and LTL formula to the final generation of the control sequence.

- Data components (1st row): The input layer consists of the multi-channeled state image X, the initial state x_0, the LTL formula ϕ, and the solution control sequence U. These components establish the context and objectives for path planning.
- Training phase (2nd row): During training, the network utilizes the solution control sequence U from the dataset to train the Conditional Variational Autoencoder (CVAE). This enables the network to effectively learn the latent distribution, facilitating accurate prediction of the control sequence $\hat{U}$ that adheres to the LTL specifications.
- Testing phase (3rd row): In the testing phase, the architecture demonstrates how latent samples are converted into predicted control sequences, validating the trained model's efficacy in real-world scenarios.

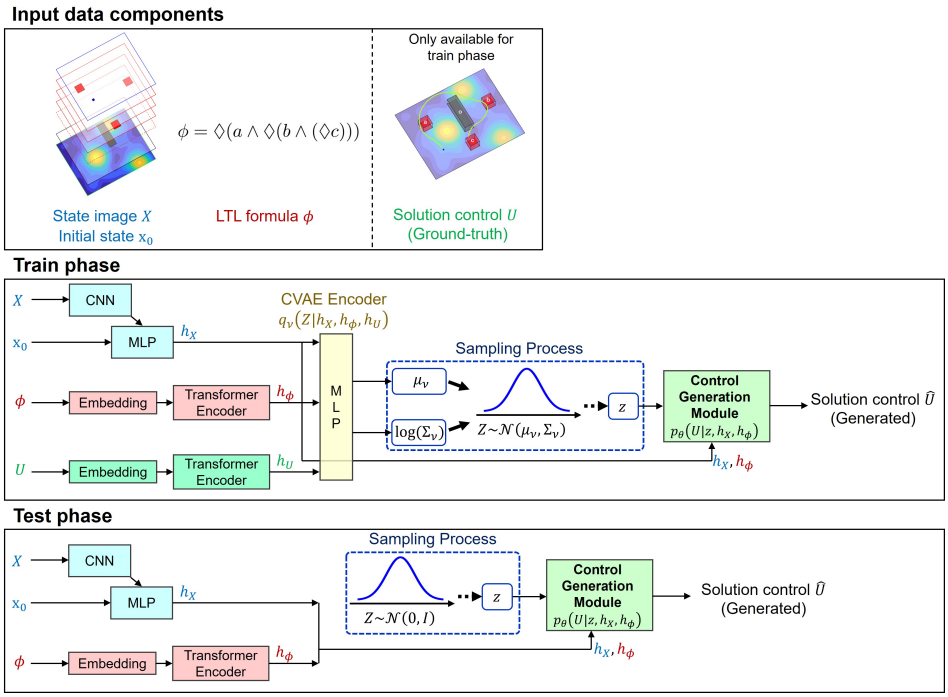


Figure 6. Overview of the proposed end-to-end deep learning network, illustrating the flow from input data components through the training and testing phases to the output control sequences.

This network not only streamlines the transition from input to output but also ensures that all elements, from environmental conditions to control strategies, are integrated within

a unified framework. By detailing each phase of the network, we provide a clear pathway from data ingestion to practical application, highlighting the network's capacity to adapt and respond to complex path-planning demands.

In the encoding stage, the network processes the input data using specialized components. A CNN encodes the state image, while Transformer encoders [13] handle the encoding of control sequences and LTL formulas. The LTL formulas are encoded by transforming each character into an embedded representation using a predefined set of alphabet and operator symbols. The resulting encoded outputs—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 to facilitate the learning process.

A key feature of our network is the implementation of a CVAE, chosen for its efficacy in handling high-dimensional spaces and versatility with various input configurations. The CVAE is instrumental in generating output control sequences by exploring the latent space. During training, it learns a probability distribution representing potential control sequences, conditioned on the encoded state image h_X and LTL formula features h_ϕ.

Our CVAE model includes three crucial parameterized functions:

- The recognition model $q_\nu(Z|h_X, h_\phi, h_U)$ approximates the distribution of the latent variable Z based on the input features and the control sequence. It is modeled as a Gaussian distribution, $\mathcal{N}(\mu_\nu(h_X, h_\phi, h_U), \Sigma_\nu(h_X, h_\phi, h_U))$, where μ_ν and Σ_ν represent the mean and covariance determined by the network.
- The prior model $p_\theta(Z|h_X, h_\phi)$ assumes a standard Gaussian distribution, $\mathcal{N}(0, I)$, simplifying the structure of the latent space.
- The generation model $p_\theta(U|z, h_X, h_\phi)$ calculates the likelihood of each control sequence element u_i based on the latent sample z and the encoded inputs, computed as the product of conditional probabilities over the sequence's length $N_A \times H_A$. This model is implemented in the Control Generation Module depicted in Figure 7.

A sample z drawn from the recognition model is fed into the decoder, which then generates the predicted control sequence. The length of this control sequence is $N_A \times H_A$, reflecting the selection of N_A anchor controls, each with a time length of H_A.

4.4. Control Generation Module

The architecture of our proposed control generation module is depicted in Figure 7. This module efficiently synthesizes the entire control sequence distribution from the latent sample z and encoded information h_X, h_ϕ in a single step, utilizing a non-autoregressive model to enhance efficiency and coherence. The GMM projection networks share parameters across all calculations, ensuring consistency.

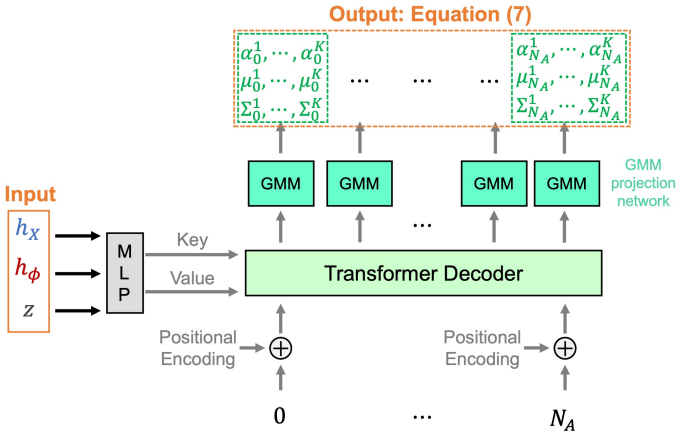


Figure 7. Schematic of the control decoder architecture, illustrating how the distribution of the output control sequence is generated from the latent sample z and encoded information h_X, h_ϕ.

In this model, a Transformer decoder utilizes sinusoidal positional encodings to integrate time information as the query, while the latent vector, combined with the encoded state image and LTL formula features, acts as the key and value. This configuration, coupled with a GMM mixture module, enables the Transformer decoder to generate the output control sequence distribution as defined by the following equation:

$$p_\theta(U|z,h_X,h_\phi) = \prod_{t=0}^{N_A} \sum_{k=1}^{K} \alpha_t^k \mathcal{N}\left(\tilde{u}_t|\mathbf{a}^k + \mu_t^k, \Sigma_t^k\right). \tag{7}$$

In this formulation:

- α_t^k, μ_t^k, and Σ_t^k are the mixture coefficients, means, and covariances of the GMM, respectively, produced by the GMM projection network.
- $\mathbf{a}^k$ represents the anchor control from the set $\mathbf{A}$.
- K is the number of mixture components, equal to the number of anchor controls, demonstrating the model's capability to represent complex distributions of control sequences.
- Each $\tilde{u}_t$ spans the time horizon H_A.

This approach uses the GMM to integrate the latent variable z, historical state image data h_X, and LTL formula features h_ϕ, creating a probabilistic space where potential control sequences are distributed around the anchor control $\mathbf{a}^k$. This serves as a central reference point, enabling the identification of feasible control sequences within the operational domain. Unlike static models, $\mathbf{a}^k$ and the GMM adapt predictions to varying conditions and uncertainties inherent in dynamic environments, enhancing the model's flexibility.

To maintain focus on the model's predictive capability and avoid unnecessary complexity, we present the GMM parameters directly without delving into the detailed functional dependencies on θ. This straightforward presentation underscores the model's utility in forecasting control sequences that are both feasible and optimized according to the computed probability distributions.

4.5. Loss Function for Training Phase

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

$$\mathbb{E}_{q_\nu(Z|h_X,h_\phi,h_U)}[\log p_\theta(U|z,h_X,h_\phi)] - \mathcal{D}_{KL}\left(q_\nu(z|h_X,h_\phi,h_U)||p_\theta(z|h_X,h_\phi)\right). \tag{8}$$

To better suit our application's specific requirements, we adapt the ELBO function and define the loss function as follows:

$$-\sum_{t=0}^{N_A} \log\left(p_\theta(\tilde{u}_t|z,h_X,h_\phi)\right) + \lambda \cdot \mathcal{D}_{KL}\Big(\mathcal{N}\left(\mu_\nu(h_X,h_\phi,h_U), \Sigma_\nu(h_X,h_\phi,h_U)\right)||\mathcal{N}(0,I)\Big), \tag{9}$$

where $\tilde{u}_t$ represents an element of the control sequence U, N_A is the number of anchor controls, and λ is a scaling factor used to balance the terms. The Kullback-Leibler divergence ($\mathcal{D}_{KL}$) measures how one probability distribution diverges from a second, reference probability distribution. We set $\lambda = 1$, optimizing parameters ν and θ by minimizing this loss function.

The first term of Equation (9), which leverages Equation (7), is detailed as follows:

$$\log\left(p_\theta(\tilde{u}_t|z,h_X,h_\phi)\right) = \sum_{t=0}^{N_A}\sum_{k=1}^{K} \mathbf{1}(k=\hat{k}_t)\left[\log\alpha_t^k + \log\mathcal{N}\left(\tilde{u}_t|\mathbf{a}^k + \mu_t^k, \Sigma_t^k\right)\right]. \tag{10}$$

This expression represents a time-sequence extension of the standard GMM likelihood fitting [37]. The notation $\mathbf{1}(\cdot)$ is the indicator function, and $\hat{k}_t$ is the index of the anchor control that most closely matches the ground-truth control, measured using the $l2$-norm distance. This hard assignment of ground-truth anchor controls circumvents the intractability of direct GMM likelihood fitting, avoids the need for an expectation-maximization procedure,

and allows practitioners to precisely control the design of anchor controls as discussed in [29]. This formulation underlines our methodology for estimating the probability distribution of control sequences, essential for ensuring that the model effectively handles the diversity of potential solutions and manages uncertainties.

4.6. Test Phase

During the test phase, the control decoder operates by processing a latent sample z, drawn from the prior distribution. This sample is deterministically transformed into a control sequence distribution as defined in Equation (7). From this distribution, the solution control sequence is generated.

The generation of the control sequence is a continuous process that persists until one of the following conditions is met: the sequence satisfies the specified LTL constraints, a collision with an obstacle occurs, or the sequence moves beyond the boundaries set by the cost map.

5. Experimental Results

This section details the outcomes from a series of simulations and experiments using the Dubins car model, which adheres to the following kinematic equations:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \omega, \tag{11}$$

where (x, y) represents the car's position, θ the heading angle, and v and ω the linear and angular velocities, respectively. It is important to note that the symbol θ is used elsewhere in this manuscript to denote different concepts unrelated to its usage here as the vehicle's heading.

We conducted three distinct sets of simulation experiments to comprehensively evaluate the proposed method's effectiveness and versatility. The first set involved generated costmaps to test the method's performance in controlled environments, focusing on its ability to navigate based on cost efficiency. The second set utilized real-world data, incorporating a traffic accident density map from Helsinki, to demonstrate the method's applicability and performance in real scenarios. The third set involved autonomous driving contexts, where the method demonstrated stable control sequence generation amidst the dynamic movement of surrounding vehicles.

The datasets for learning, including costmaps, were created as follows:

- First and second experiments: Gaussian process regression was employed to generate training costmaps, each containing no more than four regions of interest and no more than eight obstacles. Using the dynamic model defined in Equation (1), near-optimal control sequences were computed based on the method described in [36]. The resulting dataset comprised 800 costmaps, each facilitating the generation of 1500 control sequences, thereby capturing a wide range of environmental scenarios. To enhance the robustness and generalizability of our model, variability was introduced in the data generation phase by randomly varying the starting positions, placements of regions of interest, obstacle configurations, and assignments of LTL formulas for each dataset instance. This approach was designed to simulate a diverse set of potential operational scenarios, thereby preparing the model to handle a wide range of conditions effectively.
- Last experiment: In the final autonomous driving experiment, the costmap was generated from the highD dataset [38]. High costs were assigned to areas where other vehicles were present and to out-of-track areas. In this experiment, three types of LTL formulas were used: lane change to the left, lane keeping, and lane change to the right. Each data instance consists of a costmap, an LTL formula, and a control sequence. The LTL formula was selected as the closest match from the dataset based on the control sequence. For example, if the vehicle changes lanes to the left in the data, an

LTL formula related to "lane change to the left" is selected. The dataset comprised 1,000,000 instances. Please refer to the relevant subsection for a detailed explanation.

For network input, images were standardized to 128×128 pixels to comply with the memory capacity limitations of our GPU hardware. This resolution strikes a balance between retaining essential environmental details and maintaining computational feasibility. The network was trained over 300 epochs to ensure adequate learning depth, with a batch size of 32 to optimize the balance between memory usage and convergence stability. An initial learning rate of 1×10^{-3} and a weight decay of 1×10^{-5} were empirically set to provide an optimal compromise between training speed and minimizing the risk of overfitting.

In the experimental setup, the number of anchor controls K was defined as 20, and the anchor control set $\mathbf{A}$ was established using the entire training data.

Simulation experiments were conducted on an AMD R7-7700 processor with an RTX 4080 Super GPU, and the proposed network was implemented using PyTorch (version 2.2.1) [39].

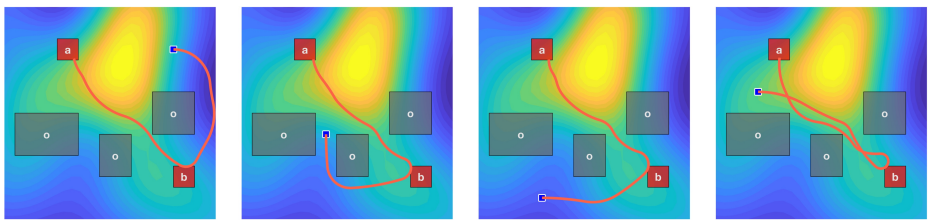
The results for each experiment are described in the subsequent subsections.

5.1. The Generated Costmap

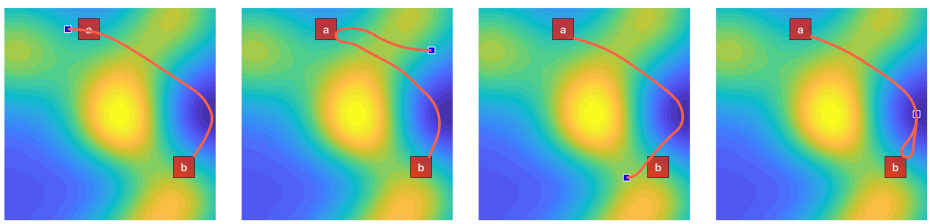
Figure 8 showcases the test costmap, synthesized using the same methods as those for the training costmaps. The costmap features regions of interest, marked with red boxes labeled with alphabetic identifiers, and obstacles are indicated by gray boxes.

In our experimental setup, we aimed to assess the robustness and effectiveness of our system across various LTL missions. These missions were categorized into sequential missions (ϕ_{toy1} and ϕ_{toy4}) and coverage missions (ϕ_{toy2} and ϕ_{toy3}). 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. Each row in the figure corresponds to a set of four subfigures, each varying by the initial position. Trajectories generated from the output distribution (as defined in Equation (7)) are displayed as red lines in each subfigure.

The proposed method strategically generates control sequences that traverse low-cost areas, depicted in blue, to effectively complete the LTL missions. Notably, these sequences successfully avoid collisions even in environments with obstacles. For coverage missions, as shown in Figure 8b,c, the solution paths differ in the order they visit regions of interest, which varies according to the starting position. For example, in Figure 8c, the sequences in each subfigure visit the regions in various orders, such as "c->b->a", "a->b->c", "a->b->c", and "b->c->a".

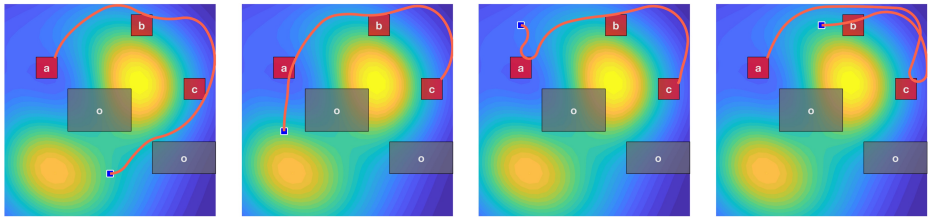


(a) $\phi_{toy1} = \Diamond(a \wedge \Diamond(b))$, "Visit regions a and b sequentially."

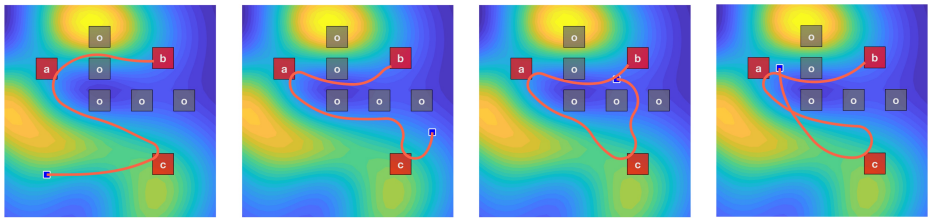


(b) $\phi_{toy2} = \Diamond(a) \wedge \Diamond(b)$, "Cover regions a and b."

Figure 8. *Cont.*



(c) $\phi_{toy3} = \Diamond(a) \wedge \Diamond(b) \wedge \Diamond(c)$, "Cover regions a, b and c."

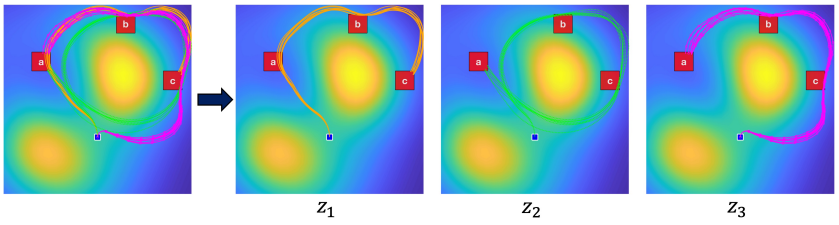


(d) $\phi_{toy4} = \Diamond(a \wedge \Diamond(b \wedge (\Diamond c)))$, "Visit regions a, b and c sequentially."

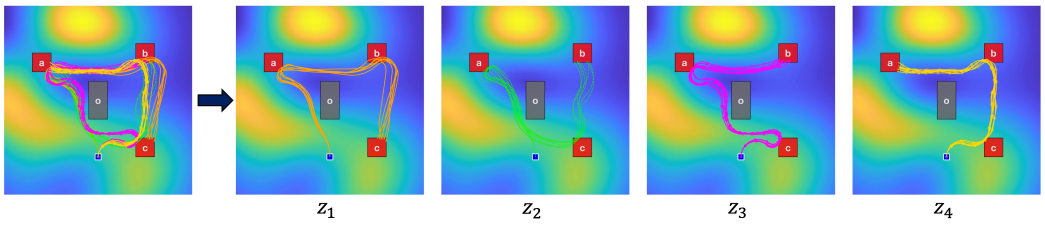
Figure 8. Solution trajectories generated by the proposed method for different LTL formulas, denoted as ϕ_{toyi}, shown in each subfigure.

Figure 9 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. In this figure, trajectories colored identically originate from the same latent sample value. The latent distribution governs the sequence in which the regions of interest are visited, ensuring compliance with the LTL mission, while the GMM component of the control decoder models the uncertainty within this sequence.

For example, in subfigure (b), the orange trajectories represent a sequence visiting a, followed by b, then c. Conversely, the green trajectories depict a sequence visiting a, c, and then b. This variability illustrates the network's ability to generate diverse solutions that not only adhere to the given LTL mission but also effectively account for uncertainties.



(a)



(b)

Figure 9. 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 are derived from the same latent sample value.

Performance evaluation of the developed path-planning approach was conducted through comparative experiments. These experiments aimed to quantify trajectory cost and mission success rates across various scenarios characterized by different lengths of

sequential LTL formulas ($|\phi|$) and obstacle counts (n_{obs}). The length of a sequential LTL formula corresponds to the number of specified regions of interest.

The experimental design included 250 trials per scenario, which featured varying costmap configurations, region of interest placements, obstacle locations, initial positions, and LTL formulas. To ensure a comprehensive evaluation of the approach's robustness, these elements were systematically varied within each trial. Trials lacking feasible solution paths were excluded to maintain the integrity of the experiment.

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

- MLP: A basic Multilayer Perceptron architecture.
- Seq2Seq-LSTM: A sequence-to-sequence model using LSTM networks [40].
- TCN: A Temporal Convolutional Network [41].
- TFN: A Transformer network model [13].

These models were trained to map 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 employed in the proposed method. Additionally, LBPP-LTL [36], a sampling-based path-planning algorithm noted for its longer computation times but guaranteed 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 model 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.

Table 1. Comparative Performance of Path Planning Approaches on Scenarios with Sequential LTL Missions. This table details trajectory costs and LTL mission success rates, categorized by the length of the LTL missions ($|\phi|$) and the number of obstacles (n_{obs}). Metrics are normalized against the LBPP-LTL benchmark, which is set with a normalized trajectory cost of 1 and a mission success rate of 100%.

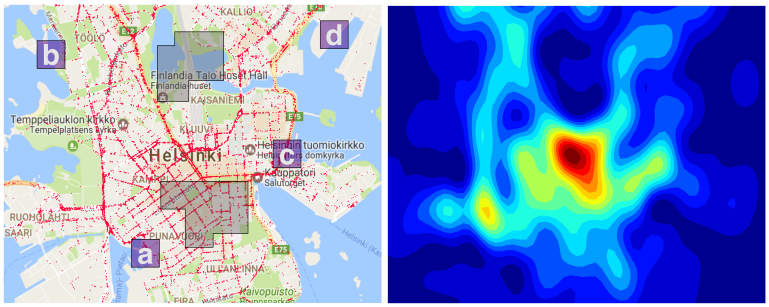
	$\|\phi\| = 2, n_{obs} = 1$	$\|\phi\| = 2, n_{obs} = 3$	$\|\phi\| = 3, n_{obs} = 2$	$\|\phi\| = 3, n_{obs} = 5$
Trajectory cost (relative)				
MLP	1.391	1.401	1.417	1.445
Seq2Seq-LSTM	1.180	1.186	1.190	1.209
TCN [41]	1.187	1.193	1.198	1.215
TFN [13]	1.075	1.097	1.102	1.114
Proposed	1.081	1.086	1.093	1.107
LBPP-LTL [36]	1.000	1.000	1.000	1.000
LTL mission success rate				
MLP	92.4%	90.4%	88.8%	87.2%
Seq2Seq-LSTM	96.4%	94.4%	93.6%	92.8%
TCN [41]	96.0%	93.6%	92.8%	91.6%
TFN [13]	98.0%	95.6%	95.2%	94.0%
LBPP-LTL [36]	100%	100%	100%	100%

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 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.

5.2. The Helsinki Traffic Accident Map

This section examines autonomous navigation for traffic surveillance within a designated area of Helsinki, as depicted in Figure 10. The map identifies four regions of interest with alphabetic labels and obstacles as gray regions.

For path planning, a traffic accident density map was synthesized using Gaussian process regression based on historical traffic accident data [42]. This map distinguishes areas with higher accident density as high cost and those with lower density as low cost, affecting the path-planning algorithm's cost evaluations.



(**a**)Traffic accident locations in Helsinki. (**b**)Traffic accident density map.

Figure 10. Visual representation of Helsinki's traffic landscape: (**a**) Traffic accidents marked with red dots, regions of interest alphabetically labeled, and obstacles shown in gray. (**b**) Traffic accident density map, where blue denotes low-density (low-cost) areas and red signifies high-density (high-cost) zones.

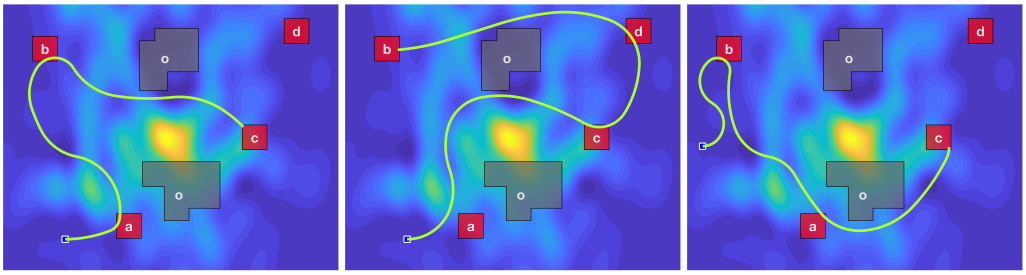
The study outlines three distinct scenarios, each associated with a unique mission profile defined by Linear Temporal Logic (LTL) formulas:

$$\begin{aligned}
&\text{Scenario1}: \phi_{h1} = \Diamond(a \wedge \Diamond(b \wedge (\Diamond c))).\\
&\text{Scenario2}: \phi_{h2} = \Diamond(b) \wedge \Diamond(c) \wedge \Diamond(d).\\
&\text{Scenario3}: \phi_{h3} = \Diamond(b \wedge ((w \vee a)\mathrm{U}(a \wedge ((w \vee b)\mathrm{U}(c))))).
\end{aligned}$$

ϕ_{h1} and ϕ_{h2} are designed for sequential and coverage missions across three distinct regions, respectively. ϕ_{h3} specifies a strict sequential mission where w represents the workspace adjacent to the regions of interest.

These formulas define the autonomous agent's mission objectives. Specifically, ϕ_{h1} mandates the agent to sequentially visit regions a, b, and c. ϕ_{h2} requires coverage of regions b, c, and d, ensuring each is visited at least once. ϕ_{h3} dictates a strict sequence for visiting regions b, a, and then c, focusing on a precise navigational order.

Figure 11 illustrates the trajectories computed by the proposed algorithm for these scenarios within the Helsinki traffic framework. The proposed method successfully generates low-cost paths that adhere to the specified LTL missions, demonstrating its effectiveness.



(**a**) ϕ_{h1} (**b**) ϕ_{h2} (**c**) ϕ_{h3}

Figure 11. Solution paths on the Helsinki traffic scenario map, demonstrating the successful completion of designated LTL missions.

5.3. Application to an Autonomous Driving Environment

In the dynamic landscape of autonomous driving, reactive planning is paramount for safe navigation. The solution of the proposed network was utilized as the output of the reactive planner. During the evaluation, three Linear Temporal Logic (LTL) formulas were employed (Figure 12):

- $\phi_{left} = \Diamond(R_{left}) \wedge \Box(R_{track}) \wedge \Box(\neg R_{other})$;
- $\phi_{center} = \Diamond(R_{center}) \wedge \Box(R_{track}) \wedge \Box(\neg R_{other})$;
- $\phi_{right} = \Diamond(R_{right}) \wedge \Box(R_{track}) \wedge \Box(\neg R_{other})$.

Here, R_{left}, R_{center}, and R_{right} denote temporal goal regions, R_{track} denotes the region inside the track, and R_{other} denotes regions occupied by other vehicles. The formula $\phi_{left} = \Diamond(R_{left}) \wedge \Box(R_{track}) \wedge \Box(\neg R_{other})$ signifies the goal to "reach the region R_{left} while remaining within the track region ($\Box(R_{track})$) and avoiding collisions with other vehicles ($\Box(\neg R_{other})$)."

For each situation, one of the three LTL formulas was selected, and a control sequence corresponding to the chosen LTL formula was generated. Generally, ϕ_{center} is selected, with the lane-changing formulas (ϕ_{left}, ϕ_{right}) being chosen only in specific times.

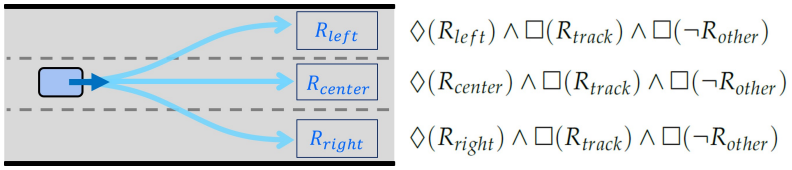


Figure 12. Three LTL formulas for the autonomous driving experiment.

Figure 13 presents snapshots from the experiment conducted using the highD dataset [38]. The decision to change lanes was made at specific points in time, allowing up to two lane changes per trial. In the figure, the ego vehicle is marked in blue, with its trajectory indicated by a blue line. Each subfigure contains three snapshots from a single trial, with lane change points indicated by arrows.

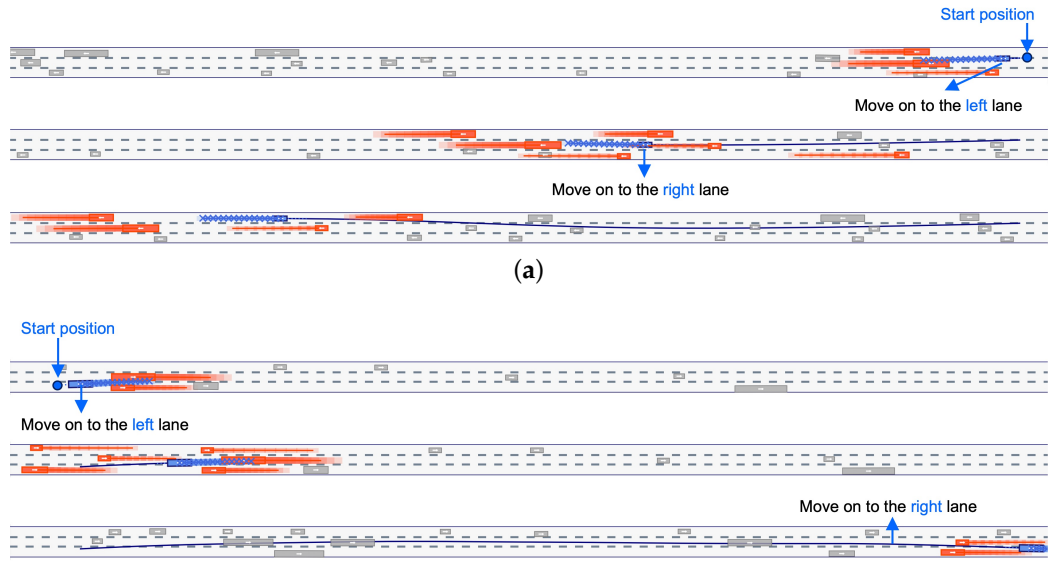
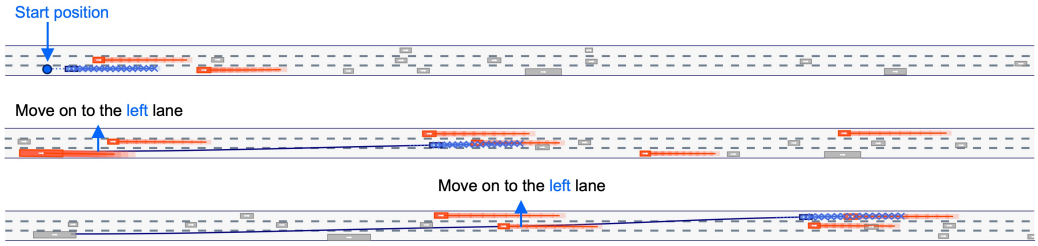


Figure 13. *Cont.*



(**c**)

Figure 13. Snapshots of the autonomous driving experiment. The ego vehicle is marked in blue, with its trajectory indicated by a blue line. Lane change points are highlighted by arrows.

Comparison experiments with other deep learning methods were conducted. We selected a test dataset from the highD dataset, which includes 60 tracks. The test dataset specifically included track IDs 10, 20, 30, 40, and 50, which were not used during the training stage. For each track, 200 trials were conducted. Each trial involved controlling a different vehicle, ensuring fairness by using the same vehicle for each method. The success metric was defined as the controlled vehicle reaching the end of the lane without incidents (collisions or going off track). Table 2 summarizes the results. The proposed method exhibited superior safety compared to other deep learning methods.

Table 2. Success ratio (percentage) in the highD dataset.

	Track ID: 10	**Track ID: 20**	**Track ID: 30**	**Track ID: 40**	**Track ID: 50**
MLP	90.0	89.5	86.5	85.5	88.0
Seq2Seq-LSTM	91.0	90.0	89.0	88.0	89.5
TCN [41]	90.5	89.5	87.5	86.5	88.5
TFN [13]	93.5	92.0	91.0	90.0	92.5
Proposed	94.5	92.5	91.5	91.0	93.0

6. Conclusions

This study presents a pioneering path-planning approach that effectively integrates co-safe Linear Temporal Logic (LTL) specifications with an end-to-end deep learning architecture. Our method stands out for its ability to generate near-optimal control sequences by combining a Transformer encoder, which is informed by LTL requirements, with a Variational Autoencoder (VAE) enhanced by Gaussian Mixture Model (GMM) components. This architecture adeptly handles the complexities of path planning by accommodating a diverse range of tasks and managing inherent uncertainties within these processes.

Empirical evaluations demonstrate the significant advantages of our approach over existing deep learning strategies. In our experiments, the proposed method consistently outperformed other methods in terms of safety and efficiency, particularly in the context of autonomous driving scenarios using the highD dataset. Our approach achieved higher success rates in reaching the end of the lane without incidents, indicating its robustness and reliability. Furthermore, the method's adaptability and scalability were highlighted through various test cases involving both synthetic and real-world data.

The results confirm the method's suitability for a wide array of systems, significantly enhancing path-planning processes. By effectively addressing the challenges posed by complex environments and logical constraints, our approach offers a robust solution for diverse robotic applications.

Looking ahead, we aim to apply our methodology to more challenging high-dimensional path-planning problems, particularly those involving additional logical constraints and intricate operational contexts, such as multi-joint robotic manipulations. Expanding our focus to these areas is expected to yield further valuable insights into robotics and automation, enhancing the sophistication and efficiency of path-planning techniques.

The integration of deep learning with logical frameworks in our study represents a significant advancement in robotic path planning, paving the way for more complex and effective mission executions in the future.

Author Contributions: Conceptualization, K.L., E.I. and K.C.; methodology, K.L. and E.I.; validation, K.C.; data curation, K.L. and K.C.; writing—original draft preparation, K.L. and E.I.; writing—review and editing, K.C.; visualization, K.L. and E.I.; supervision, K.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Innovative Human Resource Development for Local Intellectualization program through the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2024-RS-2023-00259678).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Part of the dataset is available upon request from the authors, while another part is available in a publicly accessible repository.

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

References

1. Pairet, È.; Chamzas, C.; Petillot, Y.; Kavraki, L.E. Path planning for manipulation using experience-driven random trees. *IEEE Int. Conf. Robot. Autom.* **2021**, *6*, 3295–3302. [CrossRef]
2. Lamiraux, F.; Mirabel, J. Prehensile Manipulation Planning: Modeling, Algorithms and Implementation. *IEEE Trans. Robot.* **2021**, *38*, 2370–2388. [CrossRef]
3. Xu, K.; Yu, H.; Huang, R.; Guo, D.; Wang, Y.; Xiong, R. Efficient Object Manipulation to an Arbitrary Goal Pose: Learning-based Anytime Prioritized Planning. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 23–27 May 2022; pp. 7277–7283.
4. Belkhouche, F. Reactive path planning in a dynamic environment. *IEEE Trans. Robot.* **2009**, *25*, 902–911. [CrossRef]
5. Eiffert, S.; Kong, H.; Pirmarzdashti, N.; Sukkarieh, S. Path planning in dynamic environments using generative rnns and monte carlo tree search. In Proceedings of the IEEE International Conference on Robotics and Automation, Paris, France, 31 May–31 August 2020; pp. 10263–10269.
6. Lin, J.; Zhou, T.; Zhu, D.; Liu, J.; Meng, M.Q.H. Search-Based Online Trajectory Planning for Car-like Robots in Highly Dynamic Environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May–5 June 2021; pp. 8151–8157.
7. Fainekos, G.E.; Kress-Gazit, H.; Pappas, G.J. Temporal logic motion planning for mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005.
8. Karaman, S.; Frazzoli, E. Sampling-based motion planning with deterministic μ-calculus specifications. In Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, Shanghai, China, 15–18 December 2009; pp. 2222–2229.
9. Lahijanian, M.; Wasniewski, J.; Andersson, S.B.; Belta, C. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In Proceedings of the IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 3227–3232.
10. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
11. Wang, H.; Cai, P.; Sun, Y.; Wang, L.; Liu, M. Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation. In Proceedings of the IEEE International Conference on Robotics and Automation, Xian, China, 30 May–5 June 2021; pp. 13731–13737.
12. Hu, S.; Chen, L.; Wu, P.; Li, H.; Yan, J.; Tao, D. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 533–549.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
14. Sohn, K.; Lee, H.; Yan, X. Learning structured output representation using deep conditional generative models. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
15. Smith, S.L.; Tumova, J.; Belta, C.; Rus, D. Optimal path planning for surveillance with temporal logic constraints. *Int. J. Robot. Res.* **2011**, *30*, 1695–1708. [CrossRef]
16. Wolff, E.M.; Topcu, U.; Murray, R.M. Optimal Control with Weighted Average Costs and Temporal Logic Specifications. In Proceedings of the Robotics: Science and Systems, Sydney, NSW, Australia, 9–13 July 2012.
17. LaValle, S.M.; Kuffner, J.J. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]

18. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications. In Proceedings of the IEEE American Control Conference, Montreal, QC, Canada, 27–29 June 2012.
19. Varricchio, V.; Chaudhari, P.; Frazzoli, E. Sampling-based algorithms for optimal motion planning using process algebra specifications. In Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China, 31 May–7 June 2014.
20. Bhatia, A.; Kavraki, L.E.; Vardi, M.Y. Sampling-based motion planning with temporal goals. In Proceedings of the IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010.
21. Plaku, E. Planning in discrete and continuous spaces: From LTL tasks to robot motions. In Proceedings of the Towards Autonomous Robotic Systems, Bristol, UK, 20–23 August 2012.
22. McMahon, J.; Plaku, E. Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014.
23. Karaman, S.; Sanfelice, R.G.; Frazzoli, E. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In Proceedings of the IEEE Conference on Decision and Control, Cancun, Mexico, 9–11 December 2008.
24. Wolff, E.M.; Topcu, U.; Murray, R.M. Optimization-based control of nonlinear systems with linear temporal logic specifications. In Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China, 31 May–7 June 2014; pp. 5319–5325.
25. Livingston, S.C.; Wolff, E.M.; Murray, R.M. Cross-entropy temporal logic motion planning. In Proceedings of the 18th ACM International Conference on Hybrid Systems: Computation and Control, Seattle, WA, USA, 14–16 April 2015.
26. Aloor, J.J.; Patrikar, J.; Kapoor, P.; Oh, J.; Scherer, S. Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains. In Proceedings of the IEEE International Conference on Robotics and Automation, London, UK, 29 May–2 June 2023; pp. 1320–1326.
27. Wang, Y.; Figueroa, N.; Li, S.; Shah, A.; Shah, J. Temporal Logic Imitation: Learning Plan-Satisficing Motion Policies from Demonstrations. *arXiv* **2022**, arXiv:2206.04632.
28. Dhonthi, A.; Schillinger, P.; Rozo, L.; Nardi, D. Optimizing demonstrated robot manipulation skills for temporal logic constraints. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyoto, Japan, 23–27 October 2022; pp. 1255–1262.
29. Chai, Y.; Sapp, B.; Bansal, M.; Anguelov, D. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv* **2019**, arXiv:1910.05449.
30. Petrovich, M.; Black, M.J.; Varol, G. Action-conditioned 3D human motion synthesis with transformer VAE. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 10985–10995.
31. Shi, S.; Jiang, L.; Dai, D.; Schiele, B. Motion transformer with global intention localization and local movement refinement. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Volume 35, pp. 6531–6543.
32. Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; MIT Press: Cambridge, MA, USA, 2005.
33. Baier, C.; Katoen, J.P. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008.
34. Sistla, A.P. Safety, liveness and fairness in temporal logic. *Form. Asp. Comput.* **1994**, *6*, 495–511. [CrossRef]
35. Kupferman, O.; Vardi, M.Y. Model checking of safety properties. *Form. Methods Syst. Des.* **2001**, *19*, 291–314. [CrossRef]
36. Cho, K. Learning-based path planning under co-safe temporal logic specifications. *IEEE Access* **2023**, *11*, 25865–25878. [CrossRef]
37. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4.
38. Krajewski, R.; Bock, J.; Kloeker, L.; Eckstein, L. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. In Proceedings of the International Conference on Intelligent Transportation Systems, Maui, HI, USA, 4–7 November 2018; pp. 2118–2125. [CrossRef]
39. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Glasgow, UK, 2019; pp. 8024–8035.
40. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 1–9.
41. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
42. Traffic Accidents in Helsinki. 2011. Available online: http://www.hri.fi/en/dataset/liikenneonnettomuudet-helsingissa (accessed on 5 March 2024).