Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.078696

ARTICLE

IPN-RRT*: Neural-Guided RRT* for Optimal Path Planning Using an Improved Point-Cloud Network

Zhengshun Fei[1,*], Qiao Sun[1], Chuang Yang[2], Siranee Nuchitprasitchai[3], Yongping Zheng[1] and Xinjian Xiang[1,*]

[1]School of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou, China
[2]Zhejiang Hanchine AI Tech. Co., Ltd., Hangzhou, China
[3]Faculty of Information Technology and Digital Innovation, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand

*Corresponding Authors: Zhengshun Fei. Email: zsfei@zju.edu.cn; Xinjian Xiang. Email: 188002@zust.edu.cn

Received: 06 January 2026; Accepted: 25 May 2026; Published: 23 July 2026

ABSTRACT: Path planning is a critical component for enabling autonomous navigation in mobile robots. Sampling-based planners are widely adopted due to their strong generality, yet they rely heavily on uniform sampling, which often leads to unstable performance and high computational cost in complex environments. To address this issue, recent studies feed free-space point clouds into neural networks to infer a set of guidance states near the optimal path, thereby enabling non-uniform sampling; however, the accuracy of the guidance set becomes a key bottleneck for further improvement. In this paper, we propose an improved point-cloud neural RRT* framework, termed IPN-RRT*, which achieves fast near-optimal planning via a high-precision guidance state set. Specifically, we develop an Improved PointNeXt-based neural sampling network (IPN) that enhances the geometric representation of free-space point clouds using high-dimensional sinusoidal positional encoding (HPE_{SIN}), and further improves cross-scene feature discriminability and robustness through a gated covariance-enhanced channel attention module (GCECA). These designs substantially improve the quality of the guidance state set and accelerate convergence toward better solutions during planning. Extensive experiments on large-scale datasets built from complex random maps demonstrate that IPN significantly outperforms existing point-cloud prediction models in region prediction accuracy. Moreover, across diverse challenging environments, IPN-RRT* finds near-optimal paths with fewer nodes and shorter runtime, while preserving probabilistic completeness and asymptotic optimality.

KEYWORDS: Optimal path planning; sampling-based algorithm; point-cloud neural networks; guidance state set

1 Introduction

With the widespread deployment of mobile robots in industrial, commercial, and daily-life scenarios, including warehouse automated guided vehicles (AGVs) and service robots in shopping malls, path planning, as a core component of autonomous navigation, is increasingly challenged by complex and dynamic environments. Optimal path planning aims to find a collision-free path from the start to the goal in the configuration space while minimizing a predefined cost function. Grid-based methods such as A* [1] and Dijkstra's [2] algorithm are widely used in static environments due to their completeness and optimality; however, their computational cost grows exponentially with map size and state-space dimensionality, limiting their applicability in dynamic or high-dimensional spaces. The artificial potential field (APF) [3] method guides motion by constructing attractive and repulsive fields and is computationally efficient, but it is prone to

local minima. Optimization-based methods, such as trajectory smoothing and trajectory generation, excel at producing controllable paths, but are sensitive to parameter choices and environmental variations, which limits robustness. Ant colony optimization [4] and genetic algorithms [5] are representative bio-inspired algorithms; their iterative search enables integration with other methods and supports self-organization and self-learning capabilities, offering strong search performance in path planning [6]. However, they often converge slowly and may become trapped in local optima. In addition, learning-based methods such as deep reinforcement learning [7] have also attracted attention because they can learn policies that predict feasible actions. However, these methods often lack interpretability, and their black-box nature makes it difficult to explain and anticipate their decisions.

Sampling-based motion planning algorithms have attracted considerable attention in recent years due to their efficiency in high-dimensional state spaces and under complex constraints. Probabilistic Roadmap (PRM) [8] constructs a roadmap by randomly sampling the state space and connecting feasible samples, which is efficient in static environments; however, its reliance on preprocessing limits its applicability in changing environments. In contrast, Rapidly-exploring Random Tree (RRT) [9] incrementally grows a tree from the start toward the goal, thereby covering large regions of the state space without requiring a predefined roadmap. This property makes RRT well suited to navigation in complex and dynamic environments and provides a foundation for a wide range of extensions and improvements in motion planning research. Nevertheless, RRT-type methods are only probabilistically complete and typically cannot guarantee path optimality; the resulting paths may be unnecessarily long or insufficiently smooth. To enhance their performance, substantial efforts have been devoted to improving RRT variants. Karaman and Frazzoli [10] proposed RRT* to address the shortcomings of RRT, enabling random samples to converge toward an optimal solution at the expense of higher computational cost. Nasir et al. [11] introduced RRT*-smart, an informed-sampling variant that incorporates informed sampling and path-optimization techniques, achieving certain improvements over RRT* in both solution quality and computation time, albeit with limited adaptability to different environments. Informed RRT* (IRRT*) [12] is a representative variant that augments RRT* with a heuristic search mechanism; after an initial feasible solution is found, IRRT* can converge toward the optimum more quickly. Moreover, Batch Informed Trees (BIT*) [13] and its enhanced version, Advanced BIT* (ABIT*) [14], further exploit heuristic functions to guide exploration of the state space, thereby reducing computation time. This allows them, in some scenarios, to obtain an initial solution sooner. However, the heuristic sampling region in these methods can, under certain conditions, become large, and sampling within that region remains random.

In recent years, machine learning, particularly neural networks, has been introduced into the path-planning domain to predict feasible paths or explorable regions, thereby overcoming limitations of conventional sampling-based planners. Neural-network-based approaches can exploit learned environmental knowledge to enable more intelligent planning. Neural RRT* (NRRT*) [15] integrates machine learning with sampling-based methods: it employs a neural network to segment potentially high-quality regions in the state space, so that RRT* performs sampling only within the resulting smaller subregions. Compared with algorithms relying on heuristic search, this substantially shrinks the sampling domain, accelerating both the discovery of an initial path and the convergence toward the optimal solution. Zhang et al. [16] leveraged generative adversarial networks (GANs) to construct heuristic maps for exploration, steering the search toward more promising areas of the environment. RPNN-RRT* [17] guides non-uniform sampling via high-precision region prediction, markedly improving planning efficiency and success rates in complex environments and thereby enabling rapid identification of an optimal goal-reaching path. Overall, neural-network-driven methods can concentrate more samples in the vicinity of the optimal path, which in turn speeds up convergence.

However, grid-based neural networks have been widely adopted to encode the search space [18], which requires discretizing the continuous configuration space and makes performance highly sensitive to map resolution. Recent studies have also explored applying graph neural networks (GNNs) to random geometric graphs constructed from samples in the configuration space, and then selecting edges on the graph to form an approximately optimal path [19,20]. Other work has instead employed PointNet to encode obstacle point clouds, but it models only the internal information of obstacles, leading to limited efficiency when searching for feasible paths in free space [21]. Building on this line of research, Neural Informed RRT* [22] and NAMR-RRT [23] uses PointNet++ as the point-cloud prediction network; nevertheless, in more complex environments or over longer distances, it can still suffer from broken predicted paths.

This study aims to improve the accuracy of the guidance state set in neural-network-driven methods, thereby further enhancing sampling efficiency and reducing the computational time required for optimal path planning. Our main contributions are as follows:

1. An improved PointNeXt-based point-cloud prediction network, termed IPN, is proposed for guidance-state-set generation. The proposed IPN directly processes free-state point clouds and improves the prediction accuracy of path-relevant regions, providing a high-quality guidance state set for subsequent non-uniform sampling.
2. A two-stage HPE_{SIN} fusion mechanism and the GCECA module are designed to improve free-space point-cloud representation. The former embeds high-dimensional sinusoidal positional encoding into the Set Abstraction layers and InvResMLP blocks, while the latter enhances channel discriminability through covariance statistics and gated modulation.
3. A neural-guided optimal path-planning method, IPN-RRT*, is developed by integrating IPN with RRT*. The predicted guidance state set is used to guide non-uniform sampling, reducing invalid exploration and improving planning efficiency while retaining the probabilistic completeness and asymptotic optimality of RRT*.

The remainder of this paper is organized as follows. Section 2 introduces the fundamentals of RRT*. Section 3 presents the architecture of IPN and the algorithmic framework of IPN-RRT*. Section 4 describes the dataset details and evaluates prediction accuracy and optimal path-planning performance through experiments. Section 5 concludes the paper and outlines directions for future research.

2 Preliminaries

2.1 Problem Formulation

In a path-planning problem, let $X \subset \mathbb{R}^n$ denote the robot's state space, and let $X_{\text{obs}} \subset X$ represent the obstacle space. Accordingly, the collision-free free space available for sampling can be written as

$$X_{\text{free}} = \text{cl}(X \backslash X_{\text{obs}}), \tag{1}$$

where $\text{cl}(\bullet)$ denotes the closure of a set. Let the initial state be $x_{\text{init}} \in X_{\text{free}}$, and define the goal region by

$$x_{\text{goal}} \in X_{\text{goal}} = \{x \in X_{\text{free}} \parallel \|x - x_{\text{goal}}\| < radius\}, \tag{2}$$

where the goal region X_{goal} is an open subset of X_{free}. A path-planning problem can thus be defined as the triple $(X_{\text{free}}, x_{\text{init}}, X_{\text{goal}})$: find a feasible path $\sigma : [0,1] \to X_{\text{free}}$, where σ is a continuous state trajectory satisfying $\sigma(0) = x_{\text{init}}$ and $\sigma(1) \in X_{\text{goal}}$.

The objective of optimal path planning is to find a path σ^* that minimizes the total cost under a given cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$, while ensuring that the trajectory starts from x_{init} and that all states along the path

remain in the free space when reaching the specified goal state x_{goal}. This can be formulated as

$$\sigma^* = \arg\min_{\sigma \in \Sigma} c(\sigma) \tag{3}$$

$$\text{s.t.} \quad \sigma(0) = x_{\text{init}},\ \sigma(1) = x_{\text{goal}},\ \forall s \in [0,1],\ \sigma(s) \in X_{\text{free}}. \tag{4}$$

2.2 RRT and Sampling Strategies*

RRT rapidly explores the state space by incrementally expanding a tree through random sampling. Starting from the initial state, it repeatedly samples a random state, connects it to the nearest existing node, and adds the resulting collision-free segment to the tree. This process continues until the goal is reached, after which a path is obtained by backtracking along the tree. However, RRT guarantees feasibility only, and the resulting paths are typically jagged and not optimized. RRT* is an asymptotically optimal variant of RRT designed for optimal motion planning. The key idea of RRT* is to account for path cost during tree growth. For each new node, RRT* constructs a set of neighboring vertices, compares candidate connection costs, and selects the best connection. Two operations are central to RRT*: *ChooseParent* and *Rewire*. The *ChooseParent* operation updates the tree structure by selecting, from a neighborhood of candidate vertices, the vertex that yields the minimum cost-to-come as the parent of the newly generated vertex, ensuring that the new vertex is connected to the tree with a locally minimal cost. The *Rewire* operation subsequently refines the existing tree by examining nearby vertices and attempting to reconnect them through the new vertex whenever doing so reduces their path costs. As iterations continue, RRT* progressively improves the solution cost and ultimately produces a smoother path that approaches an optimal solution. In contrast, the path produced by RRT is often jagged and lacks an inherent mechanism for cost optimization, whereas RRT* can gradually reduce the path cost due to its asymptotic optimality.

Because uniform sampling tends to be inefficient in complex environments, researchers have proposed non-uniform sampling strategies. The fundamental motivation is that more samples should be generated in regions that are more important for planning. In many cases, such important regions are induced by rejection-sampling rules, which still require a certain amount of uniform sampling in the early search stage to establish an exploration baseline. With the development of deep learning, region prediction has increasingly been formulated as a segmentation problem. Various segmentation models have been trained on large collections of planning instances to predict these key regions. Neural-network-driven non-uniform sampling methods have demonstrated notable improvements in reducing computation time and increasing sampling efficiency. The proposed IPN-RRT* further follows this direction by using a point-cloud segmentation network to process the free-state point cloud of the map and to predict sampling points in the path-relevant region, enabling rapid exploration of the space. Unlike image-based approaches that rely on pixel-level processing, IPN-RRT* only needs to model the free-state point cloud representation.

2.3 Point-Cloud Neural Networks

Point-cloud neural networks can directly process unordered point sets without converting them into grid-based or image-like representations. In this paper, the free space in a 2D map is represented as a free-state point cloud, and a point-wise prediction network is used to infer the guidance state set for non-uniform sampling. Therefore, the feature extraction capability of the point-cloud network directly affects the quality of the predicted sampling region.

PointNet is one of the earliest neural networks designed for point-cloud processing. It maps each point independently into a high-dimensional feature space through shared multilayer perceptrons and then aggregates global features using a symmetric pooling function, thereby ensuring permutation invariance for

unordered point sets. However, PointNet mainly relies on global feature aggregation and has limited ability to model local geometric structures among neighboring points.

To enhance local geometric modeling, PointNet++ introduces a hierarchical feature-learning strategy. It constructs local neighborhoods through sampling and grouping operations and extracts local features at multiple levels. Compared with PointNet, PointNet++ can better capture local spatial structures. However, when applied to sparse free-space point clouds for path-region prediction, the predicted guidance regions may still suffer from fragmentation or discontinuity.

PointNeXt further improves local feature aggregation and residual MLP-based feature refinement based on the hierarchical architecture of PointNet++. It provides stronger feature representation capability while preserving the ability to process unordered point sets. Since free-space point clouds in path-planning tasks contain limited semantic information and mainly rely on geometric relationships among points, PointNeXt is adopted as the backbone network in this paper. On this basis, a two-stage HPE_{SIN} fusion mechanism and the GCECA module are further introduced to improve spatial representation and feature discriminability for guidance state set prediction.

3 Methodology

This section presents the overall framework of IPN-RRT*, as illustrated in Fig. 1. We focus on the proposed methodological improvements. Specifically, we enhance the PointNeXt-based neural network and develop a new network, termed IPN, to improve point-cloud segmentation performance (mIoU) in the neighborhood of the optimal path. This yields a more accurate guidance state set, which is then seamlessly integrated with RRT* to form the proposed IPN-RRT* method, enabling efficient and rapid convergence toward high-quality solutions.

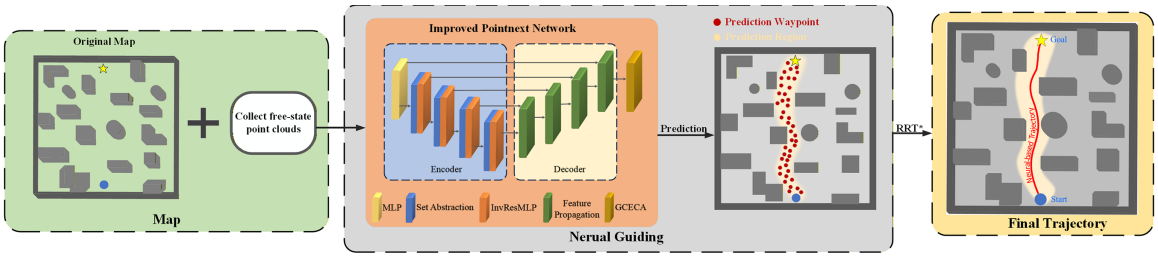


Figure 1: Overview of the IPN-RRT* framework.

3.1 Embedding High-Dimensional Sinusoidal Positional Encoding

Path planning methods based on image segmentation are inherently constrained by pixel resolution. To reduce this dependence, we first parse the map to generate a free-space point cloud, and then feed it into a point-cloud segmentation network to produce a guidance state set. However, the neural networks adopted in existing approaches have inherent limitations when applied to such low-dimensional and sparse 2D point clouds with weak geometric priors. Their feature abstraction primarily relies on relative coordinate differences and lacks explicit modeling of absolute positions and global alignment in the normalized coordinate system, which can result in feature ambiguity, path drift, or local discontinuities. To address these issues, we introduce a high-dimensional sinusoidal positional encoding (HPE_{SIN}) into the backbone and adopt a two-stage HPE_{SIN} fusion strategy to explicitly characterize inter-point geometric relations. This design restores absolute positional information while keeping the scale consistent, thereby enhancing spatial sensitivity and topology preservation, as illustrated in Fig. 2.

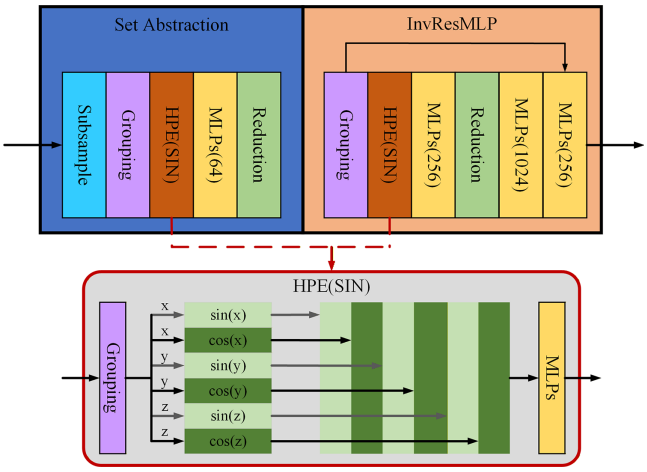


Figure 2: Overview of HPEsin and two-stage fusion.

3.1.1 Principle of High-Dimensional Sinusoidal Positional Encoding (HPE_{SIN})

Positional information is the most critical attribute of a point, as it encodes the geometric details of the scene. For the collected free-state point cloud, to fully exploit positional cues, we introduce positional encoding (PE). The concept of PE originates from the Transformer literature (Vaswani et al., 2017). In the point cloud context, PE maps the point coordinate $p_m = [p_x^m, p_y^m, p_z^m]^\top \in \mathbb{R}^{1\times 3}$ into the space of its corresponding feature $f_m \in \mathbb{R}^{1\times c}$, thereby embedding geometric information. To enhance the model's spatial awareness, the original Transformer adopts sinusoidal positional encoding (Sinusoidal Positional Encoding), which performs a fixed coordinate mapping using sine and cosine functions with different frequencies. The mathematical formulation is given as follows:

$$PE_{SIN} = \begin{cases} (p_m, 6i+0) = \sin\left(\dfrac{100 p_m^x}{1000^{6i/c}}\right), \\ (p_m, 6i+1) = \cos\left(\dfrac{100 p_m^x}{1000^{6i/c}}\right), \\ (p_m, 6i+2) = \sin\left(\dfrac{100 p_m^y}{1000^{6i/c}}\right), \\ (p_m, 6i+3) = \cos\left(\dfrac{100 p_m^y}{1000^{6i/c}}\right), \\ (p_m, 6i+4) = \sin\left(\dfrac{100 p_m^z}{1000^{6i/c}}\right), \\ (p_m, 6i+5) = \cos\left(\dfrac{100 p_m^z}{1000^{6i/c}}\right). \end{cases} \tag{5}$$

Here, $i = c/6$ is the index of the subgroup PE vector. The sine and cosine functions constrain the output range to $[-1, 1]$. However, the low-dimensional representation of PE_{SIN} is insufficient to effectively characterize the complex relative geometric relationships in unstructured point clouds, and its parameters are fixed, preventing adaptive learning tailored to task requirements. In environments with irregular point distributions or complex path geometries, sinusoidal basis functions with fixed frequencies struggle to fully capture nonlinear spatial variations, and they are particularly inadequate in regions with high curvature or frequent boundary changes.

To overcome this issue, we adopt a high-dimensional sinusoidal positional encoding, HPE_{SIN}. The core idea of HPE_{SIN} is to project the low-dimensional coordinates of each point into a high-dimensional latent space through a learnable nonlinear mapping, enabling the network to learn local geometric structures and global topological patterns simultaneously at the feature level. The mathematical formulation of HPE_{SIN} is given as follows:

$$HPE_{SIN}(p_m) = \theta_{(\lfloor c/6 \rfloor \times 6),\, c}\left(PE_{SIN}(p_m)\right) \tag{6}$$

Here, θ denotes an MLP-based transformation, and the subscripts indicate the channel dimensions of its input and output. HPE_{SIN} uses sinusoidal functions to expand the channel dimension to $(\lfloor c/6 \rfloor \times 6)$; the resulting high-dimensional vector is then aligned to the feature dimension via an MLP. With this multi-frequency, multi-dimensional combination, HPE_{SIN} can simultaneously encode low-frequency global geometric structures and high-frequency local spatial variations, exhibiting stronger geometric discriminability in high-curvature regions such as path turns and narrow passages.

Since the planning environments considered in this paper are 2D maps, each point is represented as $p_m = [p_x^m, p_y^m, 0]^T$ in our implementation. The z coordinate is fixed to zero and serves only as a placeholder dimension to maintain compatibility with general point-cloud network inputs. Therefore, the effective positional encoding in the 2D experiments is mainly derived from the x and y coordinates.

3.1.2 Two-Stage HPE_{SIN} Fusion Mechanism

To fully exploit geometric cues across different stages of feature extraction, we introduce HPE_{SIN} into two key components of PointNeXt, i.e., the Set Abstraction (SA) layers and the Inverted Residual MLP (InvResMLP) blocks, forming a two-stage high-dimensional positional encoding fusion mechanism. This design provides periodic local geometric awareness while maintaining global semantic consistency.

In the SA layer, PointNeXt selects center points via farthest point sampling (FPS) and groups their neighbors within a local radius. Unlike conventional designs, PointNeXt performs radius normalization on relative coordinates after grouping and then applies an MLP for feature aggregation. Following this workflow, we embed HPE_{SIN} into the radius-normalized relative coordinates so that each neighbor point carries a periodic geometric embedding before aggregation. The formulation is given as follows:

$$\hat{\Delta p}_{ij} = \frac{p_j^l - p_i^l}{r_l}, \tag{7}$$

$$h_{ij} = HPE_{SIN}\left(\hat{\Delta p}_{ij}\right), \tag{8}$$

$$x_i^{l+1} = R_{j:(i,j)\in\mathcal{N}}\left\{\delta_\theta\left(x_j^l; \hat{\Delta p}_{ij}; h_{ij}\right)\right\}. \tag{9}$$

where p_i^l and p_j^l are the coordinates of the center point and its neighbor point in the l-th layer, and r_l is the spherical neighborhood query radius used in the l-th layer; $\hat{\Delta p}_{ij}$ denotes the radius-normalized relative coordinates; h_{ij} represents the relative coordinates processed by high-dimensional sinusoidal positional encoding; x_j^l is the input feature of the neighbor point j in the l-th layer; x_i^{l+1} is the output feature of the center point i after aggregation; δ_θ denotes the shared MLP; R is the reduction layer (e.g., max pooling) used to aggregate features from its neighborhood to point i, and the neighborhood is denoted as $\{j : (i, j) \in \mathcal{N}\}$. With this embedding, local features can explicitly account for the geometric periodicity of points during aggregation, thereby strengthening the modeling of spatial relationships within the neighborhood.

In PointNeXt, each SA layer is followed by several InvResMLP blocks, which further refine features using separable MLPs and an inverted residual structure, while residual connections maintain feature continuity and stable gradient propagation. Similar to the local-neighborhood processing in the SA layer, we also introduce HPE_{SIN} into InvResMLP; however, it is designed to play a more prominent role in two stages, namely set-level context modeling and point-level refinement. Before grouping and Reduction, we first apply HPE_{SIN} to point coordinates, mapping them into a high-dimensional space aligned with the feature dimension, and concatenate them with the original features as the input to the MLP so that high-dimensional geometric cues can be leveraged when modeling local contextual relationships. After Reduction, the updated features are further refined via a residual MLP, where the geometric encoding injected in the previous stage is stably preserved to retain the path direction and local shape information.

With HPE_{SIN} fused in both the SA layers and InvResMLP blocks, the PointNeXt backbone can better exploit absolute and relative positional information when processing low-dimensional, sparse free-space point clouds with weak geometric priors: low-frequency components stabilize the global path trend and topological tendency, while high-frequency components capture local turning angles, curvature, and boundary details, thereby yielding stronger spatial consistency and topology-preserving capability in the path prediction task.

3.2 Gated CoVariance-Enhanced Channel Attention

Although the two-stage high-dimensional positional encoding fusion mechanism substantially improves geometric sensitivity and spatial consistency, path prediction on free-space point clouds still faces a key challenge: the samples are sparse, the structures vary significantly across scenes, and generalization remains limited. Obstacle layouts and path shapes differ markedly among environments, while the sampled point clouds contain only 2D coordinates and lack salient semantic cues such as texture and color. As a result, the model may struggle to learn sufficiently discriminative representations during training, leading to path deviations, discontinuities, or distortions in the predictions. To address this issue, we design a Gated Covariance-Enhanced Channel Attention (GCECA) module and place it after the decoder feature output. GCECA integrates conventional channel attention with second-order covariance-based attention, and further employs a gating mechanism to strengthen feature discriminability. The architecture is illustrated in Fig. 3.

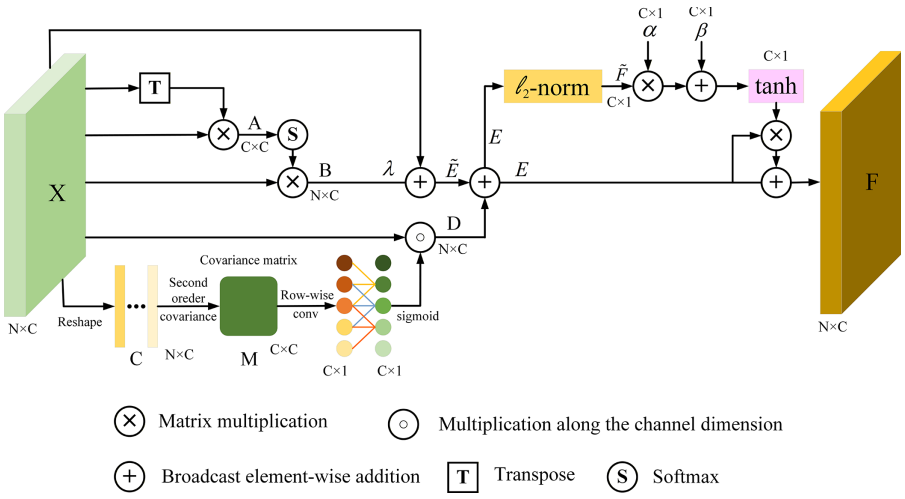


Figure 3: Framework of gated covariance-enhanced channel attention.

For the channel-attention operation, the feature map is updated by exploiting inter-channel dependencies, thereby improving the semantic feature representation and yielding a more expressive embedding. Given an input feature $X \in \mathbb{R}^{N\times C}$, where N is the number of points and C is the channel dimension, we directly compute the correlations among all channels via X^T and X, and then apply Softmax to obtain the channel-attention map $A \in \mathbb{R}^{C\times C}$:

$$a_{ij} = \frac{\exp(X_i \cdot X_j)}{\sum_{i=1}^{C} \exp(X_i \cdot X_j)} \tag{10}$$

where C is the channel dimension; i and j denote the i-th and j-th channels, respectively; and $\cdot$ denotes matrix multiplication. Then, we perform matrix multiplication between X and A to obtain the feature map $B \in \mathbb{R}^{N\times C}$:

$$B = X \cdot A. \tag{11}$$

Next, we scale B by a learnable weight parameter λ and add it to X element-wise to obtain the preliminary channel-attention output $\tilde{E} \in \mathbb{R}^{N\times C}$:

$$\tilde{E} = \lambda B + X. \tag{12}$$

For the second-order statistical covariance-attention operation, we reshape the original feature map X into $C \in \mathbb{R}^{N\times C}$ and compute the correlations between channel pairs to obtain a second-order covariance matrix $M \in \mathbb{R}^{C\times C}$. This covariance matrix has a clear physical meaning: its i-th row represents the correlations between the i-th channel and all channels in the point cloud. Specifically, M_{ij} denotes the correlation between the i-th and j-th channels:

$$M = C^T \cdot \bar{I} \cdot C, \tag{13}$$

$$\bar{I} = \frac{1}{N}\left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T\right), \tag{14}$$

where I is the $N \times N$ identity matrix, $\mathbf{1}$ is an all-ones vector, and T denotes matrix transpose. The i-th row of matrix C corresponds to the i-th point, and the j-th column corresponds to the j-th feature across all points. The matrix M forms a covariance-descriptor representation of point cloud X, encoding the co-occurrence relationships among multiple features in a second-order statistical form. Since the channel dimension spans multiple features, it is able to characterize complex structured objects.

To exploit covariance-based structural information, we employ element-wise operations to capture channel dependencies. Specifically, we aggregate each row into a $C \times 1$ weight vector, and use a one-dimensional convolution with kernel length h to capture correlations among adjacent channels. Finally, we multiply each channel of the input feature by the corresponding element of the weight vector, formulated as

$$D = \mathrm{sigmoid}\left(C1D_h\left(GroupC \times 1(M)\right)\right) \odot X \tag{15}$$

where $C1D_h(\cdot)$ denotes a one-dimensional convolution with length h (default $h = 5$), $GroupC \times 1(\cdot)$ denotes a grouped convolution with kernel size $C \times 1$, and $\odot$ denotes element-wise multiplication along the channel dimension. Each channel is enhanced or suppressed through the sigmoid operation according to its weight. The sigmoid function is adopted here because its output is constrained to $[0, 1]$, which makes it suitable for generating non-negative channel-wise attention weights. In this way, the covariance-enhanced branch can

suppress redundant channels and retain informative ones without reversing the sign of the original feature responses, thereby introducing second-order covariance information in a stable manner.

Finally, we initialize $\lambda = 0.1$ and perform element-wise addition between D and $\tilde{E}$ to generate the covariance-enhanced channel-attention feature map $E \in \mathbb{R}^{N\times C}$:

$$E = \tilde{E} + D = \lambda B + X + D. \tag{16}$$

For the gating operation, we apply channel-wise ℓ_2-norm embedding to the fused feature E to adaptively modulate the activation strength of each channel, yielding the global response intensity for each channel:

$$\tilde{F} = \sqrt{\sum_{n=1}^{N} E^2 + \varepsilon}. \tag{17}$$

We then construct a residual-form gating signal for E as

$$F = E\left[1 + \tanh(\alpha\tilde{F} + \beta)\right], \tag{18}$$

where α and β are learnable, channel-wise parameters. The residual form encourages the module to behave close to an identity mapping at initialization, which contributes to stable optimization. In this gate, tanh is adopted to provide bounded bidirectional modulation. Since the output of tanh lies in $[-1,1]$, the residual factor $1+\tanh(\alpha\tilde{F}+\beta)$ modulates the feature responses around the identity mapping, allowing weak or redundant channel responses to be attenuated and informative ones to be amplified. This differs from the sigmoid function used in the covariance-enhanced attention branch, where sigmoid generates non-negative channel-wise weights for stable attention masking. Therefore, sigmoid and tanh are selected according to their different functional roles: the former is used for non-negative covariance-based channel weighting, while the latter is used for bounded residual feature modulation. This design enables GCECA to enhance feature discriminability and robustness while maintaining stable optimization.

In GCECA, we integrate second-order covariance statistics to explicitly capture inter-channel correlation patterns, thereby improving the discriminability of learned features. The module further adopts a channel-wise ℓ_2 embedding together with a residual tanh-based gating function to perform lightweight channel re-scaling. With almost no additional learnable parameters, it adaptively strengthens informative channels while suppressing redundant activations, resulting in more semantically consistent and robust point-cloud features.

*3.3 IPN-RRT**

In the standard RRT* algorithm, the random sample x_{rand} is typically drawn from the free space X_{free} according to a uniform distribution, followed by nearest-neighbor search, state extension, parent selection, and rewiring. Consequently, the search efficiency of RRT* is strongly affected by the sampling distribution. When the state space is large or the near-optimal path region occupies only a small portion of X_{free}, uniform sampling may produce numerous ineffective expansions, increasing the number of vertices and slowing convergence.

To improve sampling efficiency, the free space is represented as a point cloud:

$$X_{\text{input}} = \{x_1, x_2, \ldots, x_N\} \subset X_{\text{free}}. \tag{19}$$

The point-cloud density is chosen such that each point has a sufficient number of neighbors within the neighborhood radius associated with the extension step size η, thereby maintaining the continuity of the subsequent search process.

After coordinate normalization, the processed point cloud $\bar{X}_{\text{input}}$ is fed into the proposed IPN network. The network predicts the probability that each point belongs to the path-relevant region:

$$p_1, p_2, \ldots, p_N = f(\bar{X}_{\text{input}}), \tag{20}$$

where $f(\cdot)$ denotes the nonlinear mapping implemented by the IPN network, which incorporates high-dimensional sinusoidal positional encoding and gated covariance-enhanced channel attention. Based on a thresholding rule, the guidance state set is defined as

$$X_{\text{guide}} = \{x_i \mid p_i > 0.5\}. \tag{21}$$

The guidance state set provides an estimate of the spatial distribution of the near-optimal path region. High-dimensional sinusoidal positional encoding enhances the representation of absolute and relative positional information, while gated covariance-enhanced channel attention improves channel discriminability. As a result, the predicted X_{guide} exhibits improved spatial connectivity and structural consistency.

IPN-RRT* then adopts a hybrid sampling strategy to generate x_{rand}. A random number $\text{Rand}() \in (0,1)$ is first generated. If $\text{Rand}() > 0.5$, x_{rand} is uniformly sampled from X_{guide}; otherwise, it is uniformly sampled from X_{free} following the standard RRT* sampling strategy.

This guidance sampling mechanism reshapes the spatial distribution of random samples. Compared with uniform sampling over the entire free space, the path-relevant region is assigned a higher sampling probability, increasing the expansion frequency of the search tree in key corridor regions. During nearest-neighbor search and state extension, the tree is more likely to grow toward the predicted path-relevant region, enabling valid vertices to form connected passages with fewer samples. At the same time, retaining a proportion of global uniform sampling prevents the planner from relying entirely on the predicted guidance states and preserves global exploration capability.

In the rewiring stage, the denser vertex distribution in the path-relevant region enables more sufficient local cost optimization, allowing the tree to approach a near-optimal path more efficiently. Thus, without modifying the basic RRT* framework, the hybrid sampling strategy changes the probability distribution of random samples and converts the path-relevant region into a high-sampling-density subspace. By retaining the original RRT* sampling branch with non-zero probability, IPN-RRT* preserves the theoretical properties of RRT* while guiding the search toward more promising regions, reducing ineffective expansions and improving path-planning efficiency.

4 Experiment

In this section, we describe the dataset and training details, followed by performance comparisons between IPN and other point-cloud networks. The segmentation performance of the guidance state set is evaluated using the mIoU metric. Finally, we assess the planning algorithms from three perspectives: the number of vertices, computation time, and success rate. We then conduct a comparative analysis of IPN-RRT* against other sampling-region prediction methods based on point-cloud networks.

4.1 Dataset and Training Details

To evaluate the effectiveness of the proposed IPN in free-space point-cloud region prediction, a dataset containing 10,000 randomly generated 2D environment maps was constructed. Each map has a resolution

of 224 × 224 pixels, and the obstacle layouts were randomly generated to ensure structural diversity across different scenarios. For each map, the A* algorithm was applied in the pixel space to generate a reference optimal path. The path step size was set to one pixel, and a safety clearance of three pixels was adopted to obtain pixel-level optimal-path labels. Then, 2048 points were uniformly sampled from the free space of each map to form a 2D point cloud. For each sampled point, a binary label was assigned according to whether the point was located within a 10-pixel radius of the reference optimal path. Points inside this region were labeled as 1, while the remaining points were labeled as 0. In this way, the dataset for the guidance-state prediction task was established.

To ensure the objectivity of the experimental evaluation and assess the generalization ability of the model, the dataset was divided into training, validation, and test sets at a ratio of 8:1:1. The test scenarios were not involved in model training and were used only to independently evaluate the prediction performance in unseen environments. All models were trained offline using the PyTorch framework on a workstation equipped with two NVIDIA RTX 3090 GPUs. The Adam optimizer was adopted with an initial learning rate of 0.001, a batch size of 16, and a total of 150 training epochs. During training, the cross-entropy loss was used to supervise the predicted probabilities with the ground-truth labels. After training, the model with the best validation performance was selected for subsequent testing and path-planning experiments.

4.2 Comparison of the Guidance State Set

In this section, we compare the proposed improved method with the point-cloud networks used in existing approaches, namely PointNet and PointNet++. Since our method is developed by enhancing PointNeXt, we also include PointNeXt as a baseline for comparison. All models are trained using the cross-entropy loss.

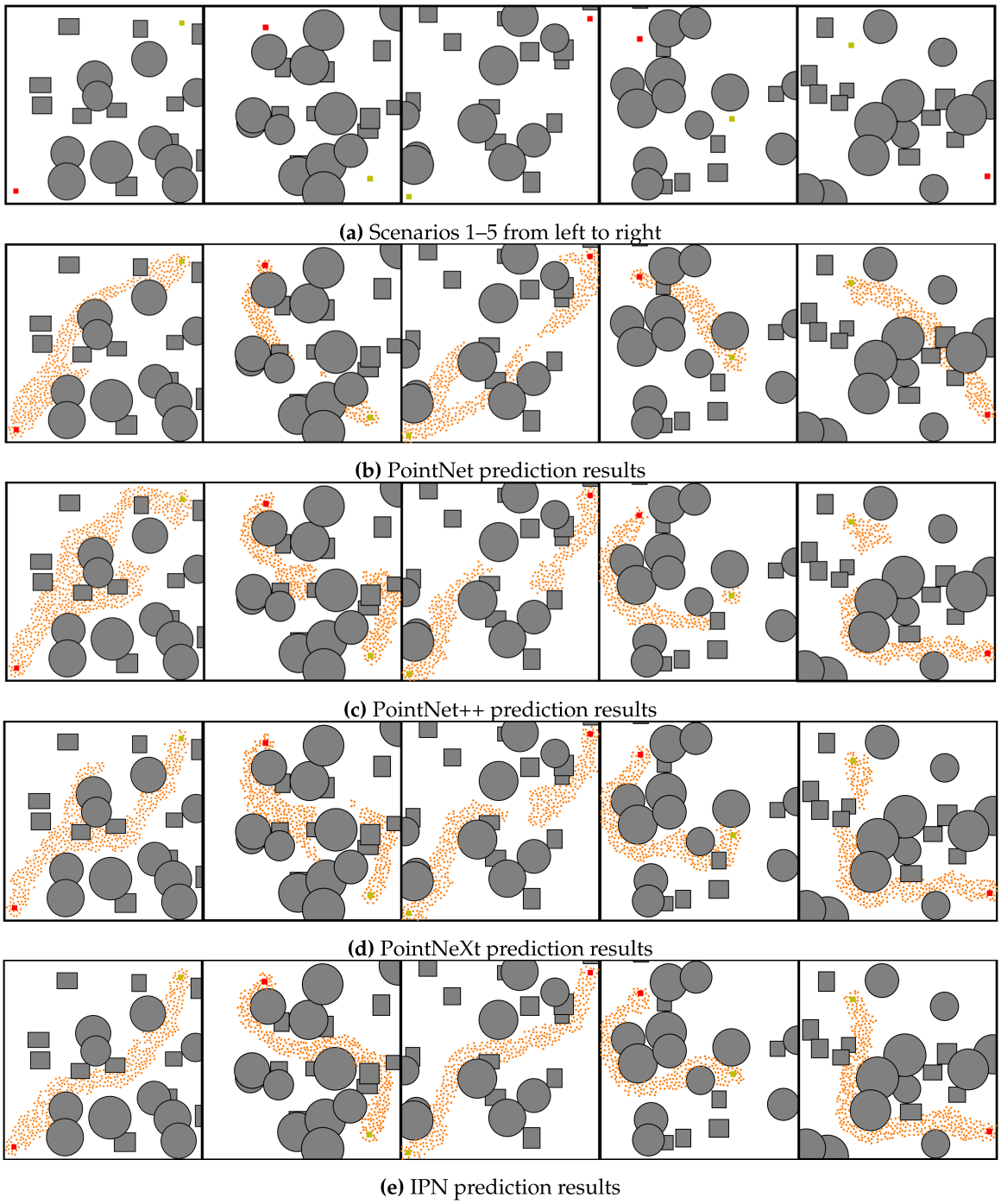
The comparative results are reported in Table 1. The table presents both the implementation results of our method against other models and the ablation study results for different improvement modules. The results of IPN are highlighted in bold and significantly outperform those of the other point-cloud networks. In particular, the added HPE_{SIN} and GCECA substantially improve PointNeXt's ability to learn point-cloud features of guidance points in the path neighborhood. HPE_{SIN} enhances the spatial perception of points and strengthens the capability of learning to discriminate feasible path regions. GCECA improves feature discriminability and yields a more informative semantic representation.

Table 1: Comparison of guidance state set segmentation performance (mIOU).

Point-cloud region prediction model	mIOU (%)
PointNet	65.14
PointNet++	74.33
PointNeXt	75.78
PointNeXt (HPE_{sin})	78.84
PointNeXt (GCECA)	78.56
IPN	79.06

As shown in the Fig. 4, we present qualitative results from several scenarios, none of which are included in the training dataset. Across all scenarios, PointNet fails to correctly learn the point-cloud features of the path region, and the predicted guidance state set contains only coarse directional information from the start to the goal. Compared with PointNet++, PointNeXt achieves a slight improvement in region prediction; however, it still exhibits fragmented regions, and some areas are incorrectly predicted. Low-quality or

misleading regions may fail to cover the optimal path, which is fatal for non-uniform sampling. In contrast, the connected regions returned by our IPN consistently and accurately cover the optimal path and can effectively learn the spatial relationships among points in the point cloud.



(a) Scenarios 1–5 from left to right

(b) PointNet prediction results

(c) PointNet++ prediction results

(d) PointNeXt prediction results

(e) IPN prediction results

Figure 4: Visualization results of the region prediction experiment on the 224 × 224 test map. Orange: guidance state set; gray: obstacles; red: start; yellow: goal.

4.3 Comparison of Optimal Path Planning

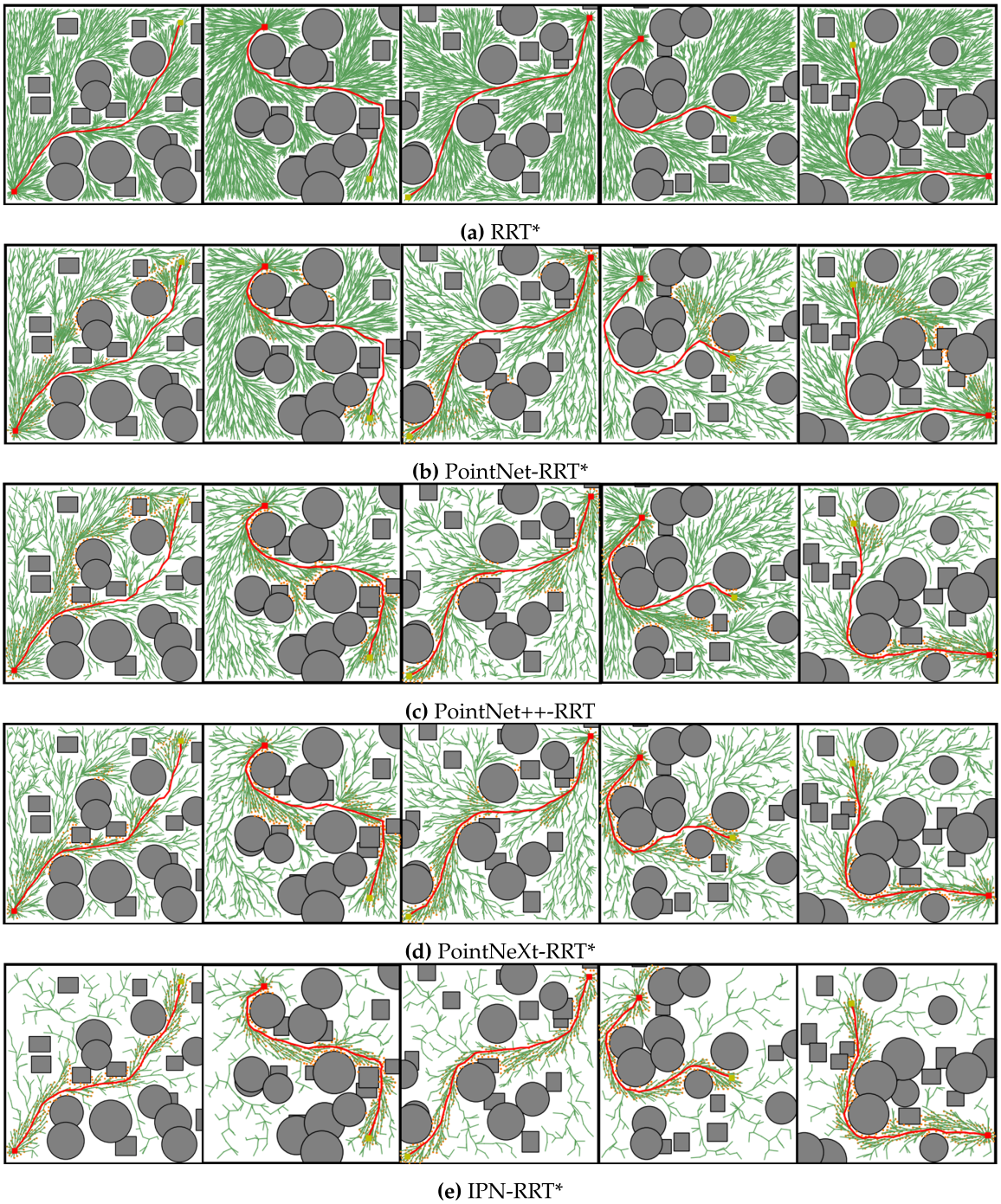
To better demonstrate that IPN can more effectively guide RRT* for path planning, we run RRT* on the predicted regions produced by different point-cloud methods. Each experiment is conducted independently for 50 runs. All RRT* variants grow the tree on 224 × 224 2D random maps with a step length of 10. If, within 10,000 samples, the difference between the cost of the generated path and the optimal path length is no more than 2%, the algorithm terminates and returns the current solution.

In addition to the average computation time, Table 2 reports the number of vertices and the success rate, with the best results highlighted in bold. From the results over the five scenarios in the table, IPN-RRT* achieves the overall best performance in terms of vertex count, planning time, and success rate. Regardless of changes in scenario complexity, IPN-RRT* consistently yields the lowest number of vertices and the shortest planning time, and attains a success rate of 98%–100% across all scenarios, demonstrating stable and outstanding overall performance.

Table 2: Comparison results of different point-cloud-guided RRT* variants for optimal path planning.

Scenario	Metric	RRT*	PointNet-RRT*	PointNet++-RRT*	PointNeXt-RRT*	IPN-RRT*
Scenario 1	Vertices	3723	2448	1773	1393	1245
	Time (s)	27.76	31.22	16.57	12.38	9.49
	Success rate	78	24	72	74	100
Scenario 2	Vertices	3767	3489	2602	2504	1634
	Time (s)	21.61	46.35	33.91	31.48	14.67
	Success rate	56	58	78	86	98
Scenario 3	Vertices	3390	2448	1638	1650	1244
	Time (s)	18.64	30.53	18.42	14.55	8.01
	Success rate	100	96	98	100	100
Scenario 4	Vertices	3512	3214	2753	2263	1445
	Time (s)	24.14	59.15	45.92	30.98	15.11
	Success rate	78	64	90	98	100
Scenario 5	Vertices	2859	3127	1714	2195	1012
	Time (s)	19.82	58.60	21.20	35.49	6.15
	Success rate	100	92	100	100	100

Fig. 5 illustrates the vertex distribution and explored regions. We observe that the accuracy and connectivity of the guidance state set play a critical role in non-uniform sampling: once the guidance set drifts or becomes fragmented, the sampling process is likely to be pulled toward invalid areas or even wrong directions, which in turn causes the search tree to inflate and increases the failure rate. In contrast, the guidance set generated by IPN more stably covers the neighborhood of the optimal path, allowing samples to concentrate on key corridors leading to the goal. As a result, the algorithm can converge to a near-optimal solution faster with fewer vertex expansions, experimentally validating the effectiveness of the proposed IPN in guiding RRT*.



(a) RRT*

(b) PointNet-RRT*

(c) PointNet++-RRT

(d) PointNeXt-RRT*

(e) IPN-RRT*

Figure 5: Illustration of vertex distribution and explored regions in a 256 × 256 environment. The endpoints of the green branches indicate the vertices, and the dark red curve denotes the final path.

In addition to the aforementioned RRT* variants guided by different point-cloud networks, recent studies have also incorporated neural networks into the Informed RRT* framework to constrain the sampling region and further improve convergence efficiency. Representative methods include NIRRT* and its extended variants. In these approaches, the sampling distribution within the heuristic ellipsoidal subspace is reconstructed through a learning-based mechanism, such that the search process is more concentrated

in potentially feasible regions. As a result, convergence can be accelerated while asymptotic optimality is preserved.

In NIRRT*, the point cloud is first constrained by NeuralFocus within the ellipsoidal subset of Informed RRT*, and is then fed into PointNet++ to infer fine-grained guidance states. Based on this framework, an extended version, namely NIRRT*-PNG, is developed to enhance the representation capability for complex environmental structures through region prediction. To alleviate the fragmentation of the guidance set generated by PointNet++, NIRRT*-PNG(C) is further proposed, where (C) denotes the introduction of the NeuralConnect strategy. In NeuralConnect, disconnected guidance points are completed through segmented point-cloud completion and iteratively filled using bidirectional BFS, thereby forming a continuous and sampleable guidance point set between the start and goal states. This improves the connectivity and reachability of the sampling space.

Since the above methods also belong to the paradigm of learning-driven sampling distribution optimization, they are conceptually similar to the proposed IPN-RRT*. Therefore, they are selected as comparative methods to evaluate the performance differences among different guidance mechanisms under a unified planning framework. To demonstrate the advantages of IPN-RRT*, direct comparisons are conducted with NIRRT* and its extended variants. The first three environments are selected, and each experiment is independently repeated 20 times in each environment. If a path is generated within 3000 samples and the difference between its cost and the optimal path length is no greater than 2%, the algorithm is terminated early and the current solution is returned. The comparative results are presented in Table 3.

Table 3: Comparison results of IRRT*, NIRRT*-PNG, NIRRT*-PNG(C), and IPN-RRT* for optimal path planning.

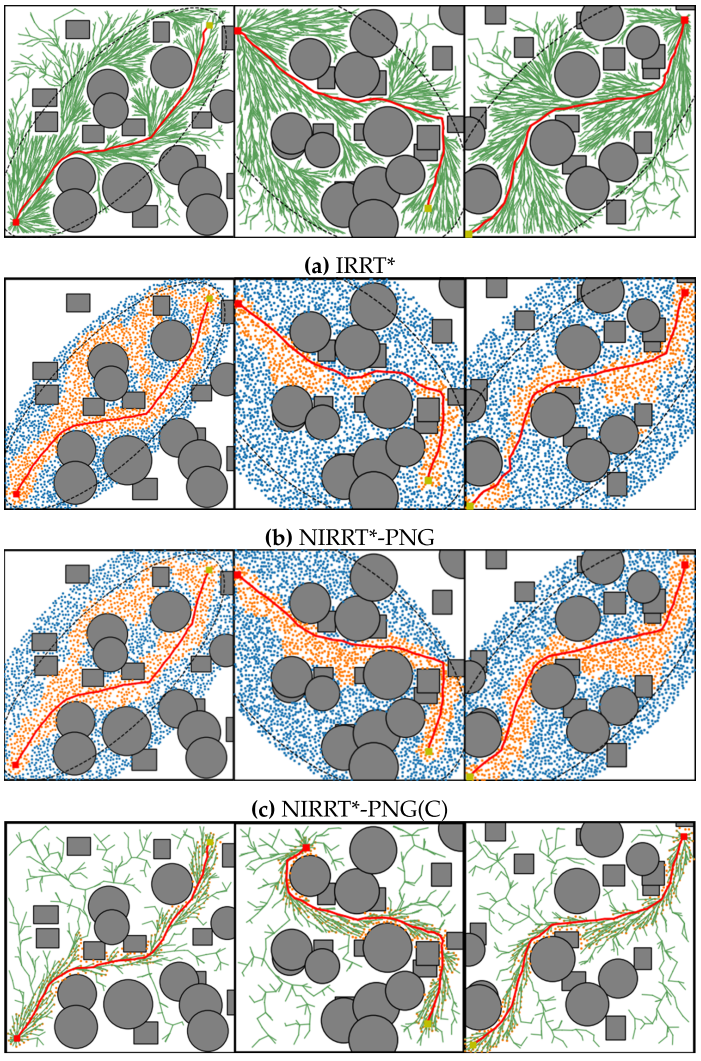
Scenario	Metric	IRRT*	NIRRT*-PNG	NIRRT*-PNG(C)	IPN-RRT*
Scenario 1	Vertices	2360	1425	1523	1202
	Time (s)	11.81	9.45	11.62	8.82
	Success rate	80	70	90	100
Scenario 2	Vertices	2704	1675	1451	1445
	Time (s)	12.67	12.47	11.68	10.04
	Success rate	55	50	70	65
Scenario 3	Vertices	2471	1500	1338	1228
	Time (s)	10.47	10.92	8.08	8.04
	Success rate	65	100	100	95

It can be observed from the statistical results in Table 3 that the compared algorithms show clear differences in search scale, planning efficiency, and success rate across the three test environments. Overall, IPN-RRT* achieves more stable advantages in node scale and planning efficiency, while maintaining a relatively high success rate in most scenarios.

In terms of node scale, IPN-RRT* generates the fewest or nearly the fewest search nodes in all test scenarios, reducing the number of nodes by approximately 40%–50% compared with IRRT*. This indicates that the guiding state set effectively concentrates the sampling distribution around potentially feasible path regions, thereby reducing invalid expansions. By contrast, although NIRRT*-PNG and NIRRT*-PNG(C) constrain the sampling region using neural networks, redundant nodes are still generated in some scenarios, suggesting that their predicted regions may contain local deviations. With respect to planning time, the overall trend is consistent with the variation in node scale. IPN-RRT* achieves a reduction of approximately 20%–30% compared with IRRT*, indicating that the search tree can approach feasible path regions more rapidly under the guidance of the predicted states. In comparison, NIRRT*-PNG still explores a

relatively broad predicted region in some environments, while NIRRT*-PNG(C), despite improving regional connectivity, may enlarge the search space due to connectivity completion. In terms of success rate, IPN-RRT* maintains a high success rate in most scenarios, showing better stability than IRRT* and performance comparable to NIRRT*-PNG(C). Although NIRRT*-PNG(C) may achieve higher success rates in some complex environments due to its connectivity completion mechanism, its node scale and planning time are still slightly higher than those of IPN-RRT*.

The path planning results in Fig. 6 further illustrate the differences in search behavior. IRRT* produces many scattered branches, indicating global diffusion of the search tree. NIRRT*-PNG guides nodes toward potential path regions but still exhibits regional expansion, while NIRRT*-PNG(C) forms a more continuous sampleable region with a relatively large search space. In contrast, IPN-RRT* concentrates its search nodes mainly within the feasible corridor between the start and goal states, resulting in a more compact search tree, fewer redundant branches, and a more continuous and stable final path.



(a) IRRT*

(b) NIRRT*-PNG

(c) NIRRT*-PNG(C)

(d) IPN-RRT*

Figure 6: Path results of different methods in the 256 × 256 environment.

Therefore, IPN-RRT* enables sampling to focus more accurately on the neighborhood of potentially feasible paths through guiding state prediction, thereby reducing the search node scale and improving planning efficiency while maintaining a high planning success rate.

5 Conclusion

This study proposes the IPN-RRT* framework to improve the sampling efficiency of RRT* through a high-quality guidance state set predicted from free-state point clouds. The motivation is that the segmentation accuracy and connectivity of neural-network-guided sampling regions remain critical factors affecting the performance of non-uniform sampling-based path planning. To address this issue, an improved PointNeXt-based point-cloud network, termed IPN, is developed by introducing high-dimensional sinusoidal positional encoding and gated covariance-enhanced channel attention. The former enhances the representation of absolute and relative geometric relationships, while the latter strengthens higher-order inter-channel correlations and improves feature discriminability in path-relevant regions.

Based on the guidance state set generated by IPN, IPN-RRT* reshapes the sampling distribution of RRT* by assigning higher sampling probability to regions that are more likely to contain near-optimal paths, while retaining a certain proportion of global uniform sampling. This design reduces ineffective exploration without changing the basic expansion, parent selection, and rewiring procedures of RRT*. Experimental results in simulated 2D static environments show that IPN achieves higher guidance-region prediction accuracy than the compared point-cloud networks. When integrated with RRT*, the proposed method reduces the number of sampled vertices and planning time while maintaining a high success rate, demonstrating the effectiveness of free-state point-cloud prediction for neural-guided sampling-based path planning.

Nevertheless, the current evaluation is mainly conducted in simulated 2D static environments. Real-world deployment remains to be further investigated, as practical robotic applications may involve sensor noise, localization uncertainty, map-building errors, dynamic obstacles, and calibration inaccuracies. In addition, the performance of IPN-RRT* in larger-scale, highly cluttered, dynamic, and higher-dimensional environments still requires further study. Future work will validate the proposed framework on physical robotic platforms using LiDAR- or camera-based perception data. Adaptive sampling ratios, online guidance-state updating, and dynamic-obstacle-aware prediction mechanisms will also be explored to improve the robustness and generalization ability of IPN-RRT* in more complex planning scenarios. Further integration with trajectory optimization and model predictive control will be considered to enhance its applicability to real robotic navigation tasks.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Key Research and Development Program of Zhejiang Province (No. 2024C01071), the Research Project of Zhejiang Provincial Department of Education (No. Y202249418), the National Natural Science Foundation of China (No. 62303419), and the Zhejiang Provincial Natural Science Foundation of China (No. LQ24F030024).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Zhengshun Fei, Qiao Sun; algorithm design and theoretical analysis: Zhengshun Fei, Qiao Sun, Siranee Nuchitprasitchai; data collection and result analysis: Chuang Yang; research resources and experimental support: Yongping Zheng; project supervision and funding acquisition: Xinjian Xiang; draft manuscript preparation: Zhengshun Fei, Qiao Sun. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and/or analyzed during the current study are available from the corresponding authors on reasonable request.

Ethics Approval: Not applicable.

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

References

1. Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern. 1968;4(2):100–7. doi:10.1109/tssc.1968.300136.
2. Dijkstra EW. A note on two problems in connexion with graphs. Numer Math. 1959;1(1):269–71. doi:10.1007/bf01386390.
3. Agirrebeitia J, Avilés R, De Bustos IF, Ajuria G. A new APF strategy for path planning in environments with obstacles. Mech Mach Theory. 2005;40(6):645–58. doi:10.1016/j.mechmachtheory.2005.01.006.
4. Zhou T, Wei W. Mobile robot path planning based on an improved ACO algorithm and path optimization. Multimed Tools Appl. 2025;84(12):10899–922. doi:10.1007/s11042-024-19370-x.
5. Ab Wahab MN, Nazir A, Khalil A, Ho WJ, Akbar MF, Noor MHM, et al. Improved genetic algorithm for mobile robot path planning in static environments. Expert Syst Appl. 2024;249(3):123762. doi:10.1016/j.eswa.2024.123762.
6. Essaadoui A, Baba Y, Hamed O, Hamlich M, Guemimi C, Kebch AE. Adaptive particle swarm and ant colony optimization path planning for autonomous robot navigation. J Robot Control. 2025;6(4):2064–76. doi:10.18196/jrc.v6i4.26853.
7. Zhang Y, Zhao W, Wang J, Yuan Y. Recent progress, challenges and future prospects of applied deep reinforcement learning: a practical perspective in path planning. Neurocomputing. 2024;608:128423.
8. Gang L, Wang J. PRM path planning optimization algorithm research. Wseas Trans Syst Control. 2016;11(7):81–6.
9. LaValle S. Rapidly-exploring random trees: a new tool for path planning. Ames, IA, USA: Iowa State University; 1998.
10. Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. Int J Robot Res. 2011;30(7):846–94. doi:10.15607/rss.2010.vi.034.
11. Nasir J, Islam F, Malik U, Ayaz Y, Hasan O, Khan M, et al. RRT*-SMART: a rapid convergence implementation of RRT. Int J Adv Robot Syst. 2013;10(7):299.
12. Gammell JD, Srinivasa SS, Barfoot TD. Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway, NJ, USA: IEEE; 2014. p. 2997–3004.
13. Gammell JD, Srinivasa SS, Barfoot TD. Batch informed trees (BIT*): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). Piscataway, NJ, USA: IEEE; 2015. p. 3067–74.
14. Strub MP, Gammell JD. Advanced BIT*(ABIT*): sampling-based planning with advanced graph-search techniques. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). Piscataway, NJ, USA: IEEE; 2020. p. 130–6.
15. Wang J, Chi W, Li C, Wang C, Meng MQH. Neural RRT*: learning-based optimal path planning. IEEE Trans Autom Sci Eng. 2020;17(4):1748–58.
16. Zhang T, Wang J, Meng MQH. Generative adversarial network based heuristics for sampling-based path planning. IEEE/CAA J Autom Sin. 2021;9(1):64–74. doi:10.1109/jas.2021.1004275.
17. Huang Y, Tsao CT, Shen T, Lee HH. Neural-network-driven method for optimal path planning via high-accuracy region prediction. Artif Life Robot. 2024;29(1):12–21. doi:10.1007/s10015-023-00915-6.
18. Ichter B, Schmerling E, Lee TWE, Faust A. Learned critical probabilistic roadmaps for robotic motion planning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). Piscataway, NJ, USA: IEEE; 2020. p. 9535–41.

19. Yu C, Gao S. Reducing collision checking for sampling-based motion planning using graph neural networks. Adv Neural Inf Process Syst. 2021;34:4274–89.
20. Feng M, Li S, Yin X. RRT* former: environment-aware sampling-based motion planning using transformer. arXiv:2511.15414. 2025.
21. Sugiura K, Matsutani H. P3Net: PointNet-based path planning on FPGA. In: 2022 International Conference on Field-Programmable Technology (ICFPT). Piscataway, NJ, USA: IEEE; 2022. p. 1–9.
22. Huang Z, Chen H, Pohovey J, Driggs-Campbell K. Neural informed RRT*: learning-based path planning with point cloud state representations under admissible ellipsoidal constraints. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). Piscataway, NJ, USA: IEEE; 2024. p. 8742–8.
23. Sun Z, Xia B, Xie P, Li X, Wang J. NAMR-RRT: neural adaptive motion planning for mobile robots in dynamic environments. IEEE Trans Autom Sci Eng. 2025;22:13087–100. doi:10.1109/tase.2025.3551464.