Received 27 February 2024, accepted 27 April 2024, date of publication 3 May 2024, date of current version 10 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3396733

RESEARCH ARTICLE

UaMPNet: Uncertainty-Aware Motion Planning Network for Manipulator Motion Planning

EUNHOO LEE AND HYUNSEOK YANG
Department of Mechanical Engineering, Yonsei University, Seodaemun-gu, Seoul 03722, South Korea
Corresponding author: Hyunseok Yang (hsyang@yonsei.ac.kr)

ABSTRACT In this study, we propose the Uncertainty-aware Motion Planning Network (UaMPNet) to address the challenges of learning-based motion planning in out-of-distribution scenarios, such as novel environments, with a primary focus on enhancing motion planning performance. UaMPNet comprises a feature extraction network and an uncertainty-aware sampling network. The feature extraction network is constructed as a variational auto encoder characterized by a normalizing flow. It not only extracts features from complex 3D point cloud data but also models that serve as a multimodal distribution, enabling fine-grained clustering of environments with similar characteristics. Additionally, Uncertainty-aware Sampling Network, leveraging evidential learning, provides both predictions and uncertainties, allowing the adjustment of sampling ranges based on the uncertainty associated with predictions. This promotes guided sampling and exploration within limited regions in new environments. We integrate UaMPNet with the rapidly exploring random trees (RRT)-connect algorithm, creating a learning-based motion planning algorithm capable of both exploration within limited ranges and exploitation toward the goal area in new environments. Evaluating the proposed algorithm's motion planning performance in novel environments, including both simple 3D spaces and intricate office environments with a 7-DoF Franka Emika Panda robot, we demonstrate its superior performance compared with that of state-of-the-art learning-based motion planning algorithms.

INDEX TERMS Learning-based motion planning, evidential learning, normalizing flow, out-of-distribution.

I. INTRODUCTION

With the advancement in robotics technology, the application areas of high-dimensional manipulators have expanded from traditional industrial environments such as product assembly and welding [1], [2] to fields such as the medical sector and coffee shops [3], [4]. To effectively use manipulators in such a wide range of applications, achieving fast and cost-minimal motion planning, considering factors such as the length of the path or the operational time of the robot, is essential.

Various motion planning approaches, including sampling-based [5], [6], search-based [7], [8], [9], and trajectory optimization-based methods [10], [11] have been proposed to address this challenge. Sampling-based motion planning, such as the rapidly-exploring random tree star (RRT*) [12] algorithm, is widely used in robotics due to its probabilistic completeness, scalability, and asymptotically optimal properties in the context of optimal motion planning. However, as the dimensionality increases, the space complexity grows, leading to difficulties in effective exploration and a decrease in convergence speed—the so-called "curse of dimensions." Additionally, in situations where limitations such as obstacles in the workspace exist, the efficiency of traditional which follows a random sampling approach, is compromised. Many samples are discarded because of the presence of obstacles, and exploration in 'challenging regions' with closely packed obstacles becomes difficult within a reasonable time.

To address these issues, various sampling-based motion planning algorithms [13], [14] have been proposed, and more recently, with the advancement of artificial intelligence, various learning-based motion planning algorithms have been proposed in such contexts to overcome the inefficiency of random sampling and achieve motion planning within a reasonable time [15], [16], [17], [18], [19], [20].

In [16] and [17], a method based on manifold learning is proposed using Conditional Variational auto encoder (CVAE) [21], which uses previous motion planning data

The associate editor coordinating the review of this manuscript and approving it for publication was Guilin Yang.

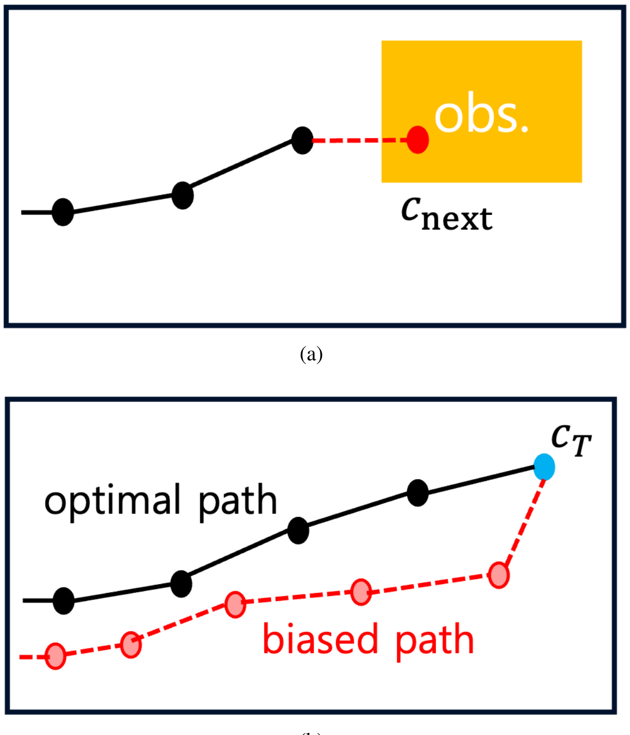

FIGURE 1. The problems with learning-based motion planning algorithms in untrained environments. In (a), the sampling process becomes stuck, while in (b), biased and noisy pathways are generated.

and environmental information to execute biased sampling in restricted regions containing optimal pathways. Moreover, in [18], Motion Planning Networks, a regression model, was introduced to enhance motion planning speed. This model takes the current state, goal state, and 3D point cloud for environment as input, and outputs the next state. It guides the sampling process, i.e., exploitation, facilitating rapid motion planning in a learned environment. In [22], an improvement in generalization performance for similar environments was achieved by utilizing a variational auto encoder (VAE) [23] to extract features from the environment's 3D point cloud and constructing a linearly interpolatable latent space. However, these learning-based methods still face challenges in generalizing to out-of-distribution settings, such as unseen environments and unseen initial and goal configurations, resulting in potential inaccuracies in predictions, increased collisions with obstacles and a degradation in the optimality of generated pathways [20], [24] as illustrated in Fig.1.

The uncertainty-aware motion planning network (UaMPNet) is proposed in this study to address the challenges of learning-based motion planning in untrained environments. UaMPNet consists of a feature extraction network that extracts features from 3D point cloud data of obstacles and an uncertainty-aware sampling network that predicts the next sample's position. The feature extraction network is trained to extract features from 3D point cloud representing various environments using a VAE. Unlike the Gaussian-VAE used in [22], our VAE is characterized by a normalizing flow to model the latent space distribution as a multimodal distribution, improving the representation power for various environments and clustering performance for environments with similar features [25].

Furthermore, to improve UaMPNet's generalization performance to out-of-distribution settings, we used deep evidential regression [26] for the Uncertainty-aware Sampling Network. Given the environment's features, current, and goal states, this network outputs both the prediction for the next state and the uncertainty associated with the prediction. This enables UaMPNet to guide the sampling process continuously based on uncertainty, preventing getting stuck in situations similar to Fig.1(a) while navigating around obstacles. Furthermore, for Fig.1(b), we integrated UaMPNet with the RRT-connect algorithm [6], creating the UaMPNet-RRT-connect algorithm. This integration enables exploitation towards the goal area in obstacle-free space, leveraging the CONNECT heuristic to reduce path bias and noise, thereby resulting in rapidly discover smoother pathways.

We evaluate the performance of our proposed algorithm in motion planning for point-mass robots in simple 3D environments and 7-degree-of-freedom (DoF) Franka Emika Panda manipulators, demonstrating its superior performance with minimal training data compared to the state-of-the-art Points-Guided Sampling Network and PG-RRT [22].

Contributions to this paper include:

- UaMPNet efficiently limits exploration ranges based on uncertainty and guides the sampling process towards the goal area continually by producing both predictions and uncertainty, addressing inefficiencies in untrained environments.
- We modeled the feature extraction network as a multimodal distribution by characterizing its latent space with a normalizing flow. This method enhances the representational capacity of point cloud data and the clustering ability of environments with similar characteristics.
- We propose UaMPNet-RRT-connect, capable of both exploration within limited ranges and exploitation toward the goal area, achieving fast and accurate motion planning and demonstrate its generalization performance for new and complex environments.

The rest of this study is organized as follows: Section II reviews learning-based motion planning methods. Section III describes the notation used in this study and the key concepts of the proposed method. Section IV details the proposed method, whereas Section V presents various experimental results. In Section VI, we discuss the advantages and disadvantages of the proposed algorithm. Finally, Section VII concludes this study.

II. RELATED WORK

Sampling-based motion planning algorithms, such as rapidly exploring random trees (RRT), probabilistic roadmaps, and their variations, have been generally used to solve motion planning problems in high-dimensional and complex environments because of their features, such as probabilistic completeness (the probability of finding a valid path is 1), computational efficiency, and scalability, which contribute to numerous robotics tasks [27], [28], [29]. Based on sufficient samples, several variations have demonstrated the ability to find not only feasible pathways but also asymptotically optimal pathways [12]. However, in high-dimensional spaces, they suffer from slower convergence speed because of the need for extensive exploration to locate asymptotically optimal pathways. To address this, research has explored techniques such as restricting the exploration range [13] and biased sampling toward the goal area [30], [31].

Recent advancements in artificial intelligence have introduced learning-based methods to overcome the slow convergence speed of conventional methods in high-dimensional spaces and rapidly locate optimal and suboptimal pathways. In [16] and [17], CVAE were trained using previous optimal motion planning data and obstacle poses or voxel representations of the environment. These models discovered subspaces in the state space containing optimal trajectories. During the motion planning phase, these learning-based models were then combined with traditional sampling-based motion planning, enabling fast exploration in the learned subspace, thus quickly finding suboptimal pathways. Additionally, research has been conducted to reduce exploration using learned data and guide the sampling process toward the objective, significantly enhancing motion planning speed. OracleNet, proposed in [32], used a recurrent neural network (RNN) structure to guide the sampling process, demonstrating rapid pathway generation in both simple 2D environments and high-dimensional robotic motion planning. Motion Planning Networks [19], a regression model, achieved good generalization performance for similar environments by training with 3D point cloud data. Motion Planning Networks demonstrated promising motion planning performance when combined with bidirectional iterative planning algorithms. Points-Guided Sampling Network was proposed in [22] to enhance generalization performance in similar environments. Points-Guided Sampling Network extracted features from 3D point clouds, performed clustering on point clouds with similar features, and modeled a linearly interpolatable latent space using a VAE-based feature extraction net (PointNet [33] encoder and AtlasNet [34] decoder) and multimodal sampling net. Despite their outstanding performance, especially in supervised learning scenarios, these motion planning algorithms may face limitations when generalizing to out-of-distribution settings [20]. This means that they can only locate short pathways in environments similar to those used for training. For completely unseen environments, performance may be significantly degraded without the use of random sampling or exploration. This is because deep neural networks (DNNs) may learn only the mapping function from inputs to predictions during the training process, or they may learn only the inherent uncertainty in the data from a maximum probability perspective [26]. To address this, various methods for learning the uncertainty in network predictions for out-of-distribution settings have been proposed [35] using dropout [36] as a Bayesian approximation, probabilistically modeling the neural network parameters to make predictions stochastically. [37] proposed an ensemble-based uncertainty modeling method, reducing the structural constraints of the model proposed in [35], and performing parallel calculations to improve prediction speeds and uncertainty prediction performance. Furthermore, [26] and [38] explicitly modeled prediction uncertainty in regression problems using evidential deep learning.

In our study, we used evidential deep learning to explicitly model uncertainty to ensure the generalization performance of deep learning-based motion planning methods in out-of-distribution settings (unseen environments). This uncertainty was used to appropriately adjust the sampling range and perform motion planning. This method addresses the challenges posed by unseen environments and enhances the reliability of the motion planning process.

III. PROBLEM FORMULATION

In this section, we elaborate on the motion planning problem and the notation used for the proposed method. Let us denote the set of all possible configurations of a robot with d degrees of freedom (DoF) (C-space) as $\mathcal{C} \in \mathbb{R}^d$. Here, the C-space consists of obstacle space ($\mathcal{C}_{\text{obs}}$) and obstacle-free space ($\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$). Similarly, the robot's workspace is structured to the C-space, with obstacle space ($\mathcal{X}_{\text{obs}}$) and obstacle-free space ($\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$). The motion planning problem is defined as finding a collision-free trajectory $\boldsymbol{\tau} = \{c_i | c_i \in \mathcal{C}_{\text{free}}, i = [0, T]\}$ such that $c_0 = c_{\text{init}}, c_T = c_{\text{goal}}$, given $\{c_{\text{init}}, c_{\text{goal}}, \mathcal{C}_{\text{obs}}\}$.

In this study, we propose replacing the traditional sampling-based motion planning's random sampling method with UaMPNet to guide the sampling process during motion planning. The following are the key elements of the proposed method:

First, the features of the 3D point cloud data ($\mathcal{P} \in \mathbb{R}^{3\times n}$) for the environment are denoted as $\mathbf{z} \sim p(\mathbf{z}|\mathcal{P})$, where n represents the total number of points per frame in the point cloud data, obtained using an RGB-D camera concerning the robot base frame.

Second, the distribution for the next sampling position is denoted as $c_{\text{next}} \sim p(c_{\text{next}}|c_c, c_T, \mathbf{z})$, dependent on the current and target states and the environmental features.

Finally, the DNN uncertainty is represented in two ways [39]. The first is aleatoric uncertainty, defined as the uncertainty on the data or noise and the second is epistemic uncertainty, which is defined as the uncertainty about the prediction. In this study, epistemic uncertainty is used to adjust the sampling range for new environments, whereas aleatoric uncertainty is used to account for data noise during the sampling process.

IV. METHODS

In this section, we introduce UaMPNet and describe the motion planning method that uses it. UaMPNet comprises a feature extraction network for extracting point cloud features

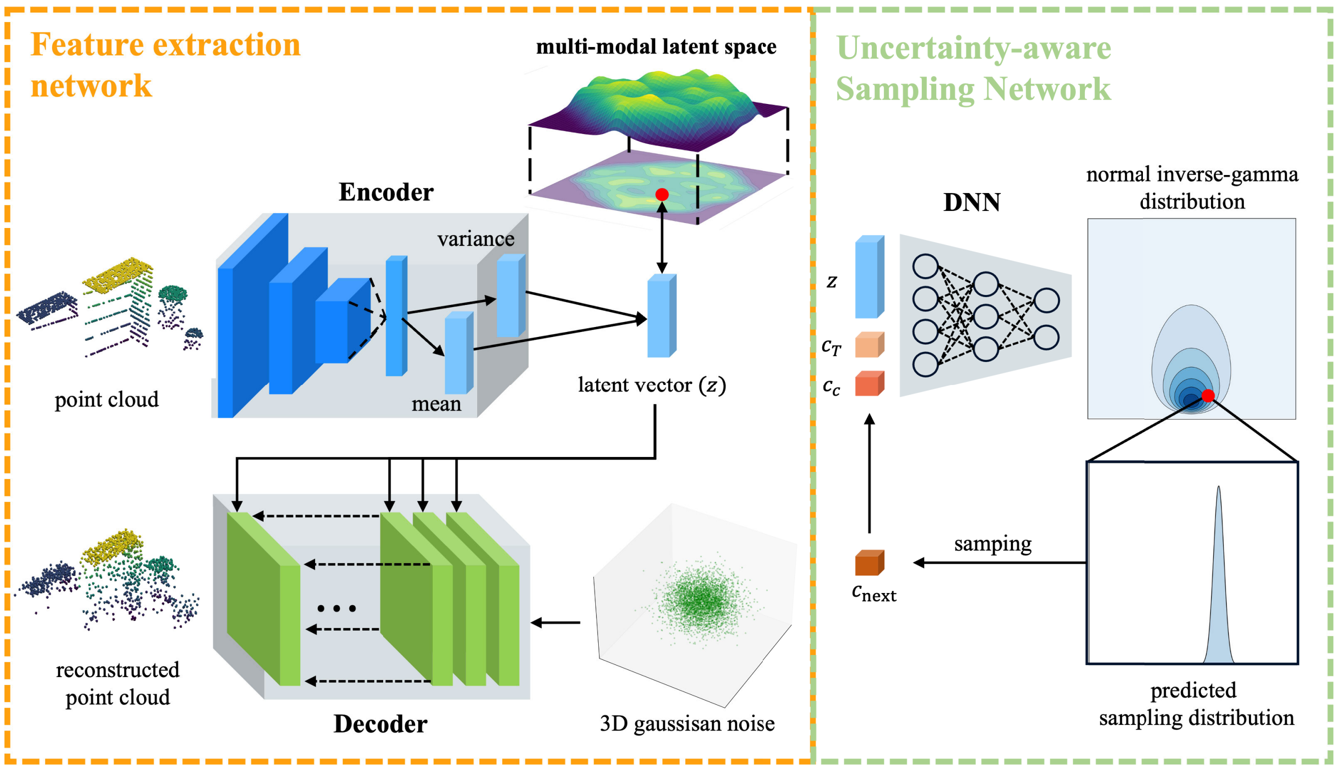
Feature extraction network
multi-modal latent space
Encoder
variance
mean
point cloud
latent vector (z)
reconstructed point cloud
Decoder
3D gaussisan noise
Uncertainty-aware Sampling Network
DNN
normal inverse-gamma distribution
z
c_T
c_c
c_{next}
samping
predicted sampling distribution

FIGURE 2. The overall schematic diagram of the proposed UaMPNet consists of a Feature Extraction Network, comprising an encoder and a decoder, and an Uncertainty-aware Sampling Network with a DNN architecture.

and a Uncertainty-aware Sampling Network for predicting the next sampling position based on uncertainties. The overall architecture of the proposed model and the sampling process for the next sample position are shown in Fig.2.

A. UNCERTAINTY-AWARE SAMPLING NETWORK

The Uncertainty-aware Sampling Network aims to guide the robot to the objective by predicting the next sampling position based on the current state, target state, and environmental features. It also aims to use the learned network as effectively as possible in new environments by considering uncertainty and restricting sampling to a limited area. In the context of regression model learning, the distribution for the next sampling position of one of the robot's joints is modeled, considering the robot's joint limits, as expressed in (1).

$$p(c_{\text{next}}|c_c, c_T, \mathbf{z}) = \mathcal{N}_{\text{trunc}}(\bar{\mu}, \bar{\sigma}, a, b). \tag{1}$$

Here, $\mathcal{N}_{\text{trunc}}(\cdot)$ denotes a truncated normal distribution [40], and $\bar{\mu}, \bar{\sigma}$ denotes the mean and standard deviation of the parent general normal probability density function (PDF), whereas a and b denotes the truncated lower and upper bounds. By assuming the distribution as above, we ensure that the next sampling position is sampled within a reasonable sampling range that takes into account the network's predictions and the robot's joint limits. To explicitly model uncertainty, the mean and variance of the distribution in (1) are assumed to be random variables following a truncated normal and inverse gamma distribution, respectively:

$$\bar{\mu}_i \sim \mathcal{N}_{\text{trunc}}\left(\gamma_i, \sqrt{\sigma_i^2 \nu_i^{-1}}, a_i, b_i\right),$$
$$\bar{\sigma}_i \sim \Gamma^{-1}(\alpha_i, \beta_i), \tag{2}$$

where $\Gamma(\cdot)$ denotes the gamma function, i represents an index that distinguishes each sample in the dataset and $\gamma \in \mathbb{R}$, $\nu_i > 0$, $\alpha_i > 1$, $\beta_i > 0$.

In the Bayesian framework, our objective is to locate the true posterior, $p(\bar{\mu}, \bar{\sigma}|c_{\text{next}})$. This is achieved by substituting the normal inverse-gamma distribution for the prior distribution in the Bayesian framework, which is a joint distribution for unknown mean and variance and serves as a Gaussian conjugate prior [26]. A DNN is used to approximate the substituted distribution, and the normal inverse-gamma distribution is expressed as follows:

$$p(\boldsymbol{\theta}_i|\mathbf{m}_i)$$
$$= \frac{\beta_i^{\alpha_i}\sqrt{\nu_i}}{\Gamma(\alpha_i)\sqrt{2\pi\sigma_i^2}}\left(\frac{1}{\sigma_i^2}\right)^{\alpha_i+1} \exp\left\{-\frac{2\beta_i+\nu_i(\gamma_i-\mu_i)^2}{2\sigma_i^2}\right\}. \tag{3}$$

Here, $\boldsymbol{\theta}_i = [\bar{\mu}_i, \bar{\sigma}_i]$ denotes the parameters of the truncated normal distribution for sampling $c_{\text{next},i}$,, whereas $\mathbf{m}_i = [\gamma_i, \nu_i, \alpha_i, \beta_i]$ denotes the parameters of the normal inverse-gamma distribution and the DNN output. Assuming that the prior distribution is an normal inverse-gamma distribution, the probability density function for $c_{\text{next},i}$, given that $\mathbf{m}_i$, is calculated analytically as a student t distribution with $2\alpha_i$ DoF:

$$p(c_{\text{next},i}|\mathbf{m}_i) = \text{St}\left(c_{\text{next},i}; \gamma_i, \frac{\beta_i(1+\nu_i)}{\nu_i\alpha_i}, 2\alpha_i\right). \quad (4)$$

As shown from the equation above, the network's prediction is $\mathbb{E}[\bar{\mu}_i] = \gamma_i$, and the corresponding aleatoric uncertainty u_{al} and epistemic uncertainty u_{ep} are defined according to [38] as follows:

$$u_{\text{al}} = \sqrt{\frac{\beta_i(1+\nu_i)}{\alpha_i\nu_i}}, \quad u_{\text{ep}} = \frac{1}{\sqrt{\nu_i}}. \quad (5)$$

The network's prediction represents the initial prediction for the next sampling position at each step, with epistemic uncertainty guiding the resampling process within a defined range depending on prediction uncertainty. The noise in the data used for network training is reflected in aleatoric uncertainty.

Algorithm 1 describes the process for obtaining the next sampling position, and this process can be broadly divided into three steps.

First, calculate aleatoric and epistemic uncertainty from the DNN output, which takes the current and target state and point cloud features as input.

Second, from the truncated normal distribution parameterized sample a new prediction ($\bar{\mu}$) with prediction, epistemic uncertainty, and joint limits. This step represents obtaining a new mean for the final sampling within the defined range, adjusting the sampling distribution based on the calculated epistemic uncertainty. As shown in Fig.3, the sampling range to sample $\bar{\mu}$ is determined by the extent to which the input values to Uncertainty-aware Sampling Network deviate from the training dataset, leveraging the calculated epistemic uncertainty. This process aims to use the trained network's predictions as much as possible while smoothly adjusting the $\bar{\mu}$ sampling distribution from a sharp normal distribution shape to a uniform distribution, as shown in Fig.3.

Finally, the sample c_{next} from the truncated normal distribution is characterized by the new prediction ($\bar{\mu}$) and aleatoric uncertainty. If the obtained sample, $c_{\text{next}} \in \mathcal{C}_{\text{obs}}$, the process is repeated until $c_{\text{next}} \in \mathcal{C}_{\text{free}}$.

We used the objective function proposed in [38] for network training, composed of the model fitting and regularization loss. The objective function is expressed as follows:

$$\begin{aligned}\mathcal{L}_i(\boldsymbol{\omega}) &= \mathcal{L}_i^{\text{NLL}}(\boldsymbol{\omega}) + \boldsymbol{\lambda}\mathcal{L}_i^{\text{R}}(\boldsymbol{\omega}) \\ &= \log\sigma_i^2 + (1+\boldsymbol{\lambda}\nu_i)\frac{(c_{next,i}-\gamma_i)^2}{\sigma_i^2}. \end{aligned} \quad (6)$$

In the above equations, $\boldsymbol{\omega}$ represents the weights of the neural network, and $\boldsymbol{\lambda}$ is the regularization coefficient.

B. FEATURE EXTRACTION NETWORK

To extract key features from the environmental point cloud for integration into motion planning, we designed a VAE-based

Algorithm 1 Uncertainty-Aware Sampling Net

```
1 Function Uncertainty-aware Sampling
    Network($\mathcal{T}^a_{end}$, $\mathcal{T}^b_{end}$, $\mathbf{z}$):
2   Initialize $a$, $b$
3   $\gamma, \nu, \alpha, \beta \leftarrow$ DNN($\mathcal{T}^a_{\text{end}}, \mathcal{T}^b_{\text{end}}, \mathbf{z}$)
4   $u_{\text{al}}, u_{\text{ep}} \leftarrow$ CalculateUncertainties($\nu, \alpha, \beta$)
5   $\bar{\mu} \leftarrow \bar{\mu} \sim \mathcal{N}_{\text{trunc}}(\gamma, u_{\text{ep}}, a, b)$
6   $c_{next} \leftarrow c_{next} \sim \mathcal{N}_{\text{trunc}}(\bar{\mu}, u_{\text{al}}, a, b)$
7   if $c_{next} \in \mathcal{C}_{free}$ then
8       return $c_{next}$
9   return Uncertainty-aware Sampling
      Network($\mathcal{T}^a_{end}$, $\mathcal{T}^b_{end}$, $\mathbf{z}$)
```

feature extraction network. The VAE comprises an encoder and decoder, mapping the point cloud to a feature space $\mathbf{z} \in \mathbb{R}^m$, where m denotes the size of the latent space dimension, and reconstructing the point cloud from the embedded latent vector, respectively. The embedded latent vector corresponding to the input point cloud data is utilized as the feature of the point cloud.

In this study, the PointNet [33] architecture was employed as the encoder for the feature extraction network. PointNet is well-suited for tasks involving 3D unordered point sets, offering permutation-invariant feature extraction, making it widely used in applications like classification and semantic segmentation. Additionally, the diffusion probabilistic models [41] were employed as the decoder, which reconstructs point clouds from 3D Gaussian noise through the reverse diffusion process. This model was trained by maximizing the log-likelihood for each point, considering points of the point cloud as a sample of a distribution.

Furthermore, we modeled the latent space of the VAE as a multimodal distribution parameterized by normalizing flow, as expressed in (8). Each mode represents various features of point clouds in the latent space. Normalizing flow, based on the change of variable theorem, sequentially transforms simple distributions, such as normal distributions, into complex distributions, enhancing the expressive power for representing features of point clouds with different positions, shapes, and sizes. This also significantly improves clustering performance for environments with similar features compared to modeling methods using single-mode distributions like gaussian distributions.

$$p(\mathbf{z}) = p_{\mathbf{w}}(\mathbf{w}) \cdot \left|\det\frac{\partial F_\delta}{\partial \mathbf{w}}\right|^{-1}, \quad \text{where} \quad \mathbf{w} = F_\delta^{-1}(\mathbf{z}). \quad (7)$$

Here, $p(\mathbf{z})$ is the probability distribution for the latent variable, serving as the prior distribution approximated by the encoder during VAE training. F_δ represents an affine coupling layer consisting of a trainable bijector, and $p_{\mathbf{w}}(\mathbf{w})$ is the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

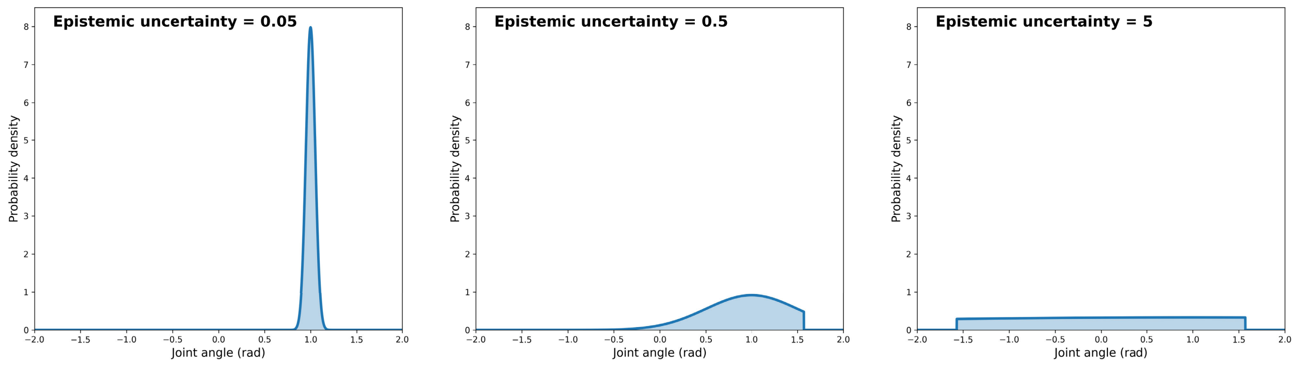

FIGURE 3. An example of a truncated normal distribution for sampling new predictions with different epistemic uncertainties. All graphs have a mean of 1, and the epistemic uncertainties (EU) are 0.05, 0.5, and 5 from left to right. The truncated boundaries are [-1.57, 1.57]. Depending on the predicted epistemic uncertainty, the sampling range is appropriately adjusted, ranging from a sharp Gaussian distribution to a shape resembling a uniform distribution.

For network training, we utilized the objective function proposed by [41], expressed as follows:

$$L(\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\delta}) = \mathbb{E}_q \left[\sum_{t=2}^{T} \sum_{i=1}^{N} D_{KL} \left(q(x_i^{(t-1)} | x_i^{(t)}, x_i^{(0)}) \parallel \right.\right.$$
$$\left. \times \; p_{\boldsymbol{\psi}}(x_{i-1}^{(t-1)} | x_i^t, \mathbf{z}) \right) - \sum_{i=1}^{N} \log p_{\boldsymbol{\psi}}(x_i^{(0)} | x_i^{(1)}, \mathbf{z})$$
$$\left. + D_{KL} \left(q_{\boldsymbol{\phi}}(\mathbf{z}|X^{(0)}) \parallel p_{\mathbf{w}}(\mathbf{w}) \cdot \left| \det \frac{\partial F_\delta}{\partial \mathbf{w}} \right|^{-1} \right) \right]. \tag{8}$$

Here, x represents a point in the point cloud, z is the latent vector, t and i denote the number of diffusion steps and the number of points in a single point cloud, respectively. $\boldsymbol{\psi}$, $\boldsymbol{\phi}$, and $\boldsymbol{\delta}$ represent the parameters of each network, and $X^{(0)}$ denotes the input point cloud. The last KL divergence term in (8) ensures that the latent vector output from the encoder follows a multimodal distribution parameterized by normalizing flow.

C. UAMPNET WITH RRT-CONNECT

We integrated our proposed sampling method, UaMPNet, with the sampling-based motion planning algorithm RRT-connect [6]. RRT-connect is an algorithm that biases sampling towards the goal region using a connect heuristic when collision-free paths are identified, as seen in Algorithm 2. By employing this algorithm, we utilize UaMPNet to guide and explore within a constrained range in obstacle-rich regions, while leveraging the connect heuristic for exploitation in obstacle-free areas.

Algorithm 3 outlines the details of the motion planning approach using our proposed method. Two trees are initialized with the initial and goal configurations, respectively, and the algorithm is iteratively executed for a predefined sampling limit N. Here, ϵ is a fixed incremental distance used for tree expansion. If the two trees connect and successfully generate a path, it is returned, otherwise, nothing is returned upon planning failure.

The functions and their roles in the motion planning are as follows:

- **Encoder**: The encoder of the trained feature extraction network processes the 3D point cloud to extract corresponding features. This is executed only once for a given environment, and the decoder is not used in the motion planning process.
- **Uncertainty − aware Sampling Network**: Given the most recently added nodes from both trees and environmental features, this function outputs the position of the next sample. The specific sampling process is detailed in Algorithm 1.
- **Extend**: Expands a tree given the tree, a sample, and the incremental distance if expansion is feasible.
- **NearestNeighbor**: Returns the closest node in the tree to the given sample.
- **Steer**: Returns the node located ϵ distance away from $c_{nearest}$ in the direction of the given sample.
- **CollisionFree**: Checks for collision between two nodes. If no collision occurs, it connects the two nodes and extends the tree.
- **Connect**: The core part of the RRT-connect algorithm. It continuously extends the tree in the direction of the given sample until a collision with obstacles occurs.
- **Swap**: Swaps the two trees.

V. EXPERIMENTS AND RESULTS

In this paper, we conducted a total of three experiments to validate the uncertainty prediction performance and motion planning capabilities of the proposed UaMPNet. In experiment 1, we focused on evaluating the performance of epistemic uncertainty prediction during environmental changes and motion planning processes in a given environment. Experiment 2 involved utilizing UaMPNet for motion planning in a simple 3D environment for point-mass robots and a more complex environment resembling an office setup for a 7-DoF Franka Emilka Panda robot.

Algorithm 2 Extend

```
1  Function Extend($\mathcal{T}, c, \epsilon$):
2      $c_{nearest} \leftarrow$ NearestNeighbor($c, \mathcal{T}$)
3      $c_{new} \leftarrow$ Steer($c_{nearest}, c, \epsilon$)
4      if CollisionFree($c_{nearest}, c_{new}$) then
5          $\mathcal{T} \leftarrow \mathcal{T} \cup \{c_{new}\}$
6          if $c_{new} = c$ then
7              return Reached
8          else
9              return Advanced
10     return Trapped
```

Algorithm 3 Connect

```
1  Function Connect ($\mathcal{T}, c, \epsilon$):
2      repeat
3          $S \leftarrow$ Extend($\mathcal{T}, c, \epsilon$)
4      until $S \neq$ Advanced;
5      return $S$
```

Algorithm 4 UaMPNet With RRT-Connect

```
1  Input: Initial state $c_0$, goal state $c_T$, point cloud $\mathcal{P}$
2  Output: $\boldsymbol{\tau} = \{c_i | c_i \in \mathcal{C}_{free}, i = [0, T]\}$
3  Initialize $N, \epsilon, \mathcal{T}^a \leftarrow \{c_0\}, \mathcal{T}^b \leftarrow \{c_T\}$
4  $\mathbf{z} \leftarrow$ Encoder($\mathcal{P}$)
5  for $i = 0, \ldots, N$ do
6      $c_{next} \leftarrow$Uncertainty-aware Sampling
         Network($\mathcal{T}^a_{end}, \mathcal{T}^b_{end}, \mathbf{z}$)
7      if Extend($\mathcal{T}^a, c_{next}, \epsilon$) $\neq$ Trapped then
8          if Connect ($\mathcal{T}^b, c_{new}$) $=$ Reached then
9              $\boldsymbol{\tau} \leftarrow \mathcal{T}^a \cup \mathcal{T}^b$
10             return $\boldsymbol{\tau}$
11     Swap($\mathcal{T}^a, \mathcal{T}^b$)
12 return $\varnothing$
```

We compared the motion planning performance against the state-of-the-art algorithm PG-RRT [22], which utilizes Points-Guided Sampling Network. Lastly, in Experiment 3, we conducted a study to assess motion planning performance based on different feature extraction methods. We compared a single-mode latent space modeling approach with our proposed method using Panda robot motion planning.

For the evaluation of motion planning performance, we considered metrics such as success rate, cost (length of the path in the configuration space of the robot), and average planning time.

A. EXPERIMENTAL SETUP AND DATA COLLECTION

1) 3D ENVIRONMENTS WITH A POINT-MASS ROBOT

We configured the normalizing flow to perform 14 transformations for the modeling of multimodal latent space, and the feature extraction network was designed to undergo 100 diffusion processes. The size of the latent vector (environment feature) extracted through this process was set to 256. Uncertainty-aware Sampling Network was built with a DNN architecture consisting of 5 hidden layers, applying batch normalization to all layers but the final layer.

To obtain a point cloud for obstacles, in this case, a dataset was constructed by sampling points on the obstacle surfaces using mathematical representations of the obstacles with the origin as the reference point. In other words, for the sake of experiment simplification, it was assumed that a fully scanned point cloud for the given environment could be obtained. We acquired 5000 frames of point clouds for environments with eight obstacles of different sizes randomly positioned and environments with nine obstacles (including one additional obstacle randomly added), respectively, totaling 10,000 frames. This dataset was used for training the feature extraction network. Also, previously selected from the pool of 5,000 environments with eight obstacles, 50 environments were randomly chosen. For each selected environment, pre-motion planning data for 100 different initial and goal configurations of a point-mass robot were obtained using the RRT* algorithm.

2) GAZEBO SIMULATION ENVIRONMENTS WITH A PANDA ROBOT

The neural network structure used in the Gazebo simulation experiment with the Panda robot was configured similarly to the structure used in the previous 3D environment. The Gazebo simulation environment was set up with a total of 5 obstacles, including a file holder, books, cola, a bear, and a tissue box, placed on a table. To obtain point cloud data for obstacles, we created environments with 4 obstacles (no book) and 6 obstacles (two books) additionally. For each environment, we captured point clouds consisting of different 5000 frames using a Kinect RGB-D camera mounted above, with respect to the base frame of the Panda robot, in the Gazebo simulation. In this case, we did not assume obtaining a perfect point cloud for obstacles, and the method of acquiring point clouds using an actual sensor was the same. Additionally, obstacles were randomly positioned in each environment.

The dataset, consisting of a total of 15,000 frames, was divided into training, validation, and test sets with a ratio of 8:1.5:0.5 to train the feature extraction network. We selected 70 different environments out of those with 5 obstacles for the training of the feature extraction network. For Uncertainty-aware Sampling Network training, we obtained pre-motion planning data for 100 different initial and goal configurations for each selected environment, using the Moveit! and OMPL interface with the RRT* algorithm.

All networks were trained on a system equipped with an Intel i9-12900K CPU, 32GB RAM, and Nvidia 1080 GPU.

B. UNCERTAINTY ESTIMATION

The purpose of this experiment can be broadly divided into two main objectives. Firstly, to observe the difference in

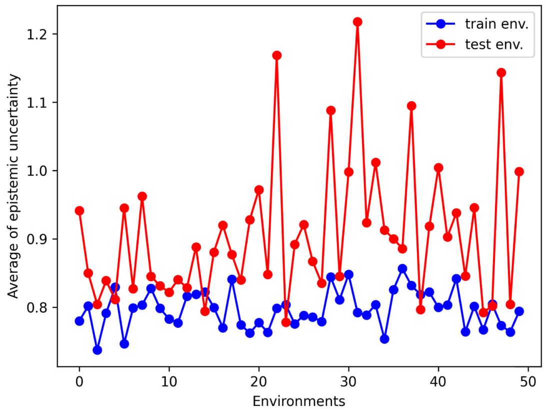

FIGURE 4. The average predicted epistemic uncertainty during the motion planning process in the selected office environment with the Panda robot for the train and test scenarios. The blue color represents the results for the training environment (seen-$\mathcal{X}_{\text{obs}}$), while the red color represents the results for the test environment (unseen-$\mathcal{X}_{\text{obs}}$).

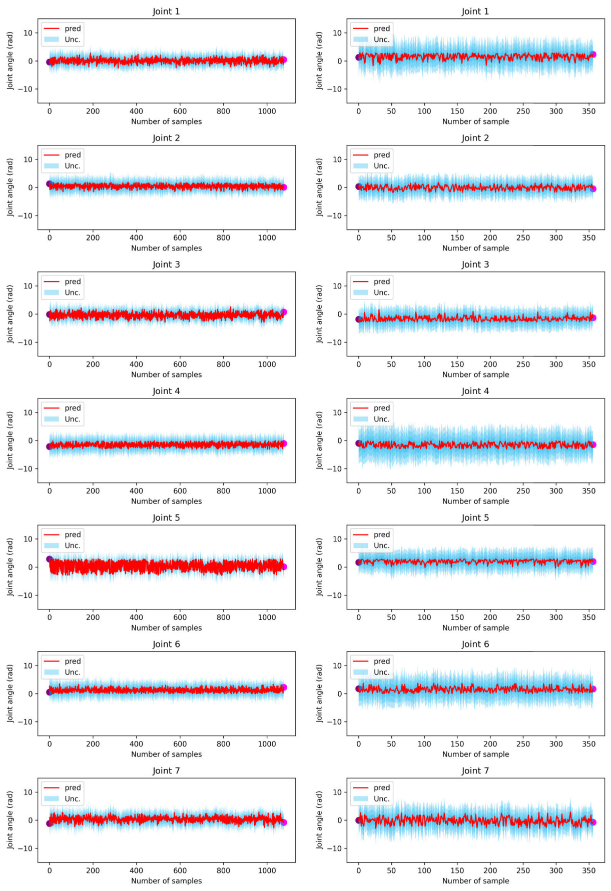

FIGURE 5. Predicted samples and epistemic uncertainty during the motion planning process of the 7-DoF Panda robot. The left column represents results for one of the environments used in Fig.4 (train scenario), while the right column represents results for one of the test environments. Each row corresponds to a joint of the Panda robot. In the figures, the red color represents the predicted samples, the cyan color represents epistemic uncertainty, the purple dots denote the initial state of the motion planning, and the magenta dots indicate the goal state of the motion planning.

predicted epistemic uncertainty during the motion planning process between the trained environment and untrained environments belonging to the out-of-distribution setting. Here, the out-of-distribution setting refers to data not utilized in the training of the regression model Uncertainty-aware Sampling Network. For instance, this includes scenarios where the motion planning problem introduces changes in obstacle positions or adds obstacles, leading to the extraction of features ($\mathbf{z}$) from the feature extraction network that were not used in Uncertainty-aware Sampling Network training, or cases where current and goal configurations not utilized in Uncertainty-aware Sampling Network training are provided. Secondly, to investigate whether the prediction of epistemic uncertainty is sensitive to changes in the current configuration (c_c) during the motion planning process in a given environment.

For this purpose, we selected 50 seen and 50 unseen environments randomly and then performed motion planning for random initial and goal configurations. The seen environment involved the random selection of 50 out of the 70 environments mentioned in the previous V-A-2 for the training of Uncertainty-aware Sampling Network, as described. The unseen environment, denoted as unseen-$\mathcal{X}_{\text{obs}}$, differs from seen-$\mathcal{X}_{\text{obs}}$ not only by changing the object's position as in [22] but also by increasing the complexity and difference from the trained environment. We achieved this by adding a book (An additional obstacle) to the environment, resulting in a total of six randomly positioned obstacles.

The dataset mentioned in Section V-A-II was used for UaMPNet training, and the regularization coefficient $\boldsymbol{\lambda}$ in (6) was set to 0.01. The average of the predicted total epistemic uncertainty during motion planning in the selected seen and unseen environments is shown in Fig.4. In general, the unseen environment had higher uncertainty, indicating a broader sampling range when sampling in a new environment. Furthermore, the relatively consistent uncertainty in predictions for both seen and unseen environments implies that UaMPNet uses learned information effectively during motion planning in new environments.

Fig.5 shows the sampled joint states and corresponding epistemic uncertainty during motion planning in one seen and one unseen environment used in Fig.4. We observed that the predicted epistemic uncertainty during the motion planning process in the unseen environment is consistently higher at almost every sampling step. Furthermore, we noted that the sampling range changes at each sampling step due to variations in $[c_c, c_T]$, indicating that UaMPNet adequately learns uncertainty in predictions for new inputs.

C. MOTION PLANNING WITH UAMPNET

1) MOTION PLANNING IN 3D ENVIRONMENTS

We compared our proposed method with PG-RRT using motion planning for a point-mass robot in simple 3D environments. The Points-Guided Sampling Network of PG-RRT was trained using the same dataset as the one mentioned in V-A-1 for the training of UaMPNet.

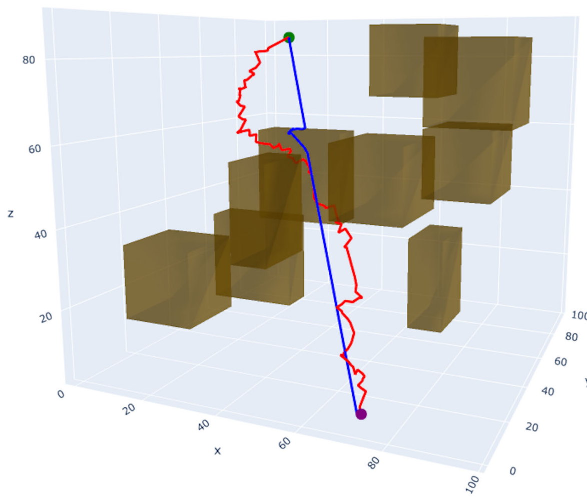

FIGURE 6. An example of motion planning in a simple 3D environment with a point mass robot. The blue path represents the result of UaMPNet-RRT-connect, while the red path corresponds to PG-RRT. The purple and green dots indicate the initial and goal states of the motion planning, respectively, and the brown boxes represent obstacles. The length of the generated paths is 150.95 for UaMPNet-RRT-connect and 221.49 for PG-RRT.

The objective of this experiment was to quickly locate short pathways for untrained new environments. We selected 50 environments in a 3D setting with nine randomly positioned obstacles of different sizes to demonstrate the generalization ability of our proposed algorithm for unseen new environments. In detail, the set of 50 different environments with 9 obstacles chosen for testing were utilized in the training of the feature extraction network. However, these environments were not employed in the training of Uncertainty-aware Sampling Network. Consequently, when conducting motion planning in the given test environments, environmental features not utilized in the training of Uncertainty-aware Sampling Network are input to Uncertainty-aware Sampling Network during the motion planning process. Fig.6 shows examples of selected 3D environment pathways created using UaMPNet-RRT-connect and PG-RRT. The motion planning parameter and incremental distance ϵ, was set to 1 for both methods, and the motion planning termination condition was set to 5000 samples. In untrained new environments, UaMPNet-RRT-connect creates smooth pathways through exploitation in obstacle-free areas and quickly locates appropriate pathways within limited exploration ranges in regions with obstacles. In contrast, PG-RRT generates noisy and biased pathways because of inaccurate predictions. Furthermore, for motion planning performance evaluation, we performed motion planning 100 times for a single initial and goal configuration in each environment. Fig.7 shows the success rates for each environment, average path length (cost), and average planning time for the pathways generated by the two algorithms. Here, we observed that our proposed method generates faster and shorter pathways in almost all

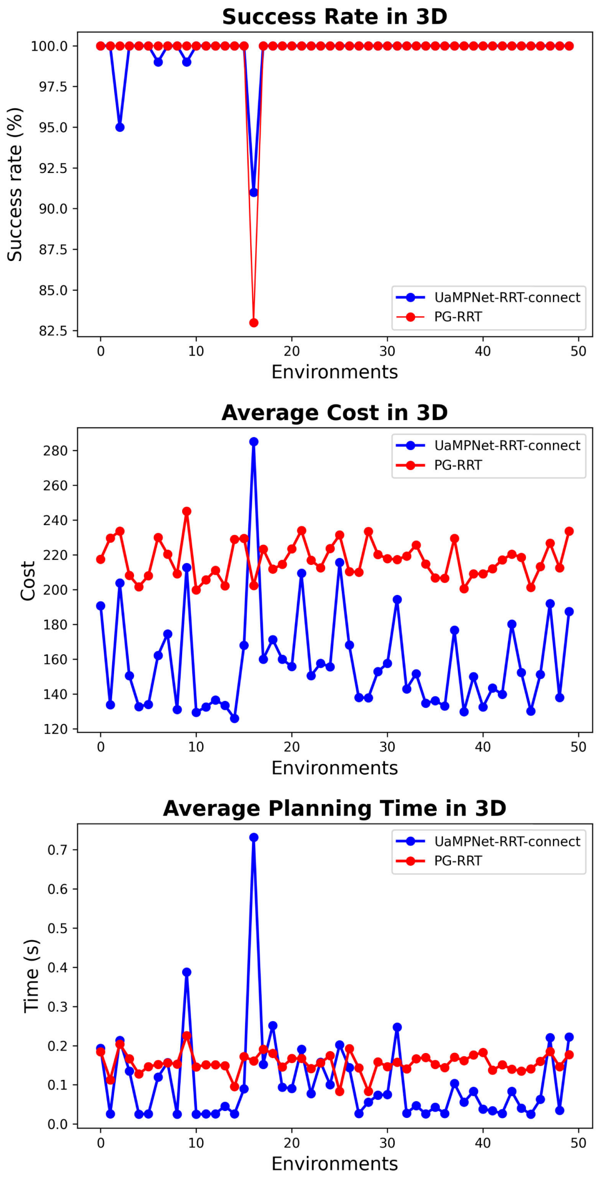

FIGURE 7. Comparison of motion planning performance between UaMPNet-RRT-connect and PG-RRT for a point-mass robot in a simple 3D environment. average success rates, average costs (path length), and average planning times for each of the 50 environments, with 100 motion planning attempts per environment, are presented from top to bottom.

environments. Furthermore, both algorithms showed similar success rates in all 3D environments for motion planning, with UaMPNet-RRT-connect at 99.68% (4984/5000) and PG-RRT at 99.66% (4985/5000).

2) MOTION PLANNING IN OFFICE ENVIRONMENT

Next, we conducted motion planning experiments for the Franka Emika Panda robot in an office environment. To demonstrate the algorithm's generalization ability for unseen environments, we used six randomly positioned obstacles, similar to the V-B unseen environment configuration, and 100 different environments, ten times more than in previous studies [19], [22]. We selected one initial and goal

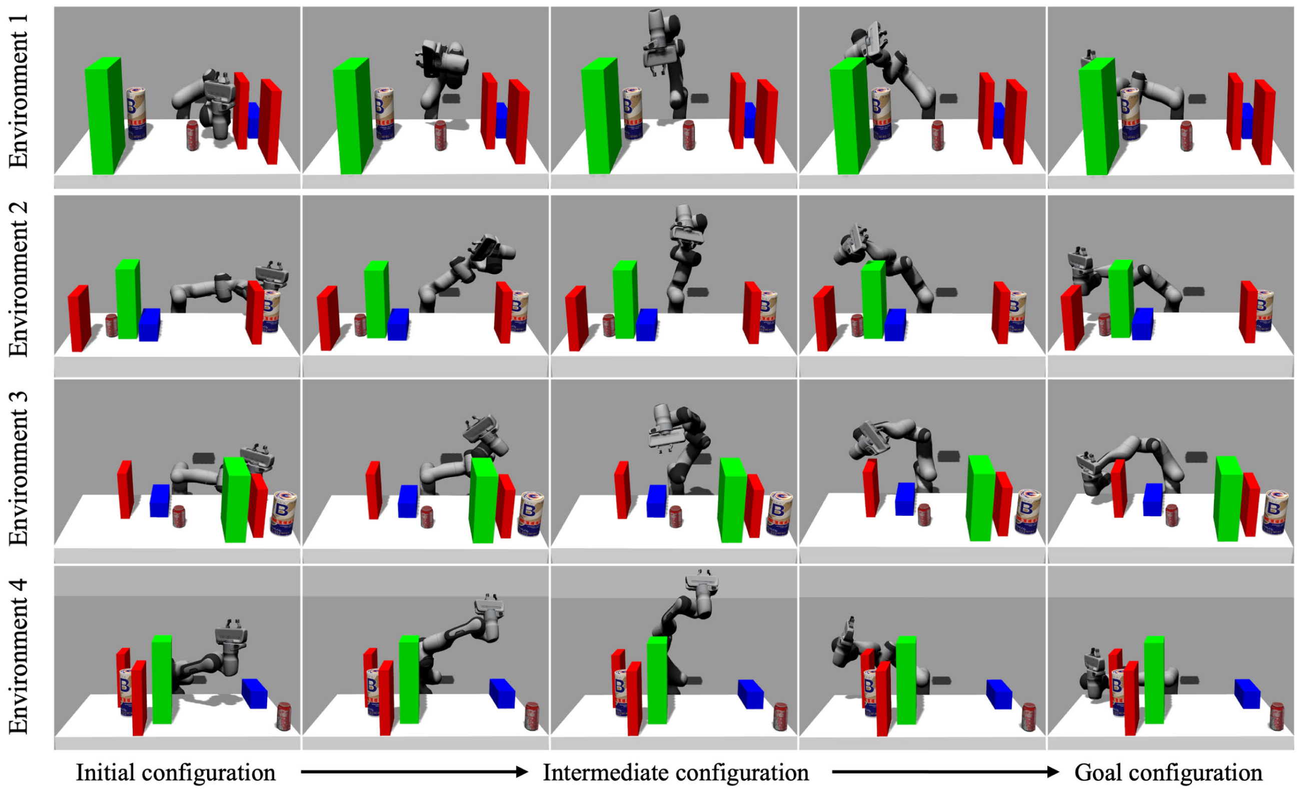

FIGURE 8. **Motion planning results for a Panda robot in an office environment using UaMPNet-RRT-connect. The images depict the motion planning process for four out of 100 test environments, with the far-left representing the initial state and the far-right indicating the goal state. The intermediate sections illustrate the robot's movements along the generated paths.**

configuration for each environment and performed motion planning 100 times for motion planning result evaluation. The incremental distance ϵ was set to 0.05 radians, consistent with the default value in the RRT-connect algorithm implemented in the OMPL interface, and the maximum sample count was set to 5000.

Fig.8 shows the motion planning process for four environments out of 100. Furthermore, Fig.9 shows the success rates, average path length (cost) in the robot's configuration space, and average planning time for the selected environments. In all Panda robot experiment, the cost was calculated as the sum of joint space trajectory lengths for the robot's all joints, measured in radians, using the same method as employed in [22]. Regarding the path cost, both algorithms showed similar values, with our proposed method having lower costs in 55 environments. However, in almost all environments (98/100), our proposed method showed a smaller average planning time. In other words, our proposed method can create paths in new environments faster. In fact, the average cost across all environments was 6.06 radians, 7.08 radians, and the average planning time across all environments was 1.45 seconds, 4.67 seconds, respectively. This indicates that the proposed algorithm exhibited lower values in all metrics. Furthermore, despite the limited sampling count, the success rates for both algorithms differed significantly in practically all environments. Overall, for motion planning, our proposed method demonstrated a success rate of 99.81% (9981/10000), while PG-RRT showed a success rate of 58.06% (5806/10000), confirming the overwhelming performance of our proposed method over PG-RRT in unseen environments. Moreover, in the case of PG-RRT, pathway generation failed for three environments, and the success rate of pathway generation was only 50% or lower for 41 of 100 environments. In other words, when using the same termination condition as in the previous 3D environments, it was evident that, with the increase in dimensions, the success rate drastically decreased in cases where incorrect direction is provided, despite the previously reported high success rate.

Secondly, to compare the difference between UaMPNet and Points-Guided Sampling Network, we constructed PG-RRT-connect by combining Points-Guided Sampling Network and RRT-connect algorithms. The experiment was conducted in the same environment as before, and the results are shown in Fig.10. The success rate for the entire PG-RRT-connect experiment was 99.36% (9936/10000), which was significantly higher than PG-RRT. This is because a considerable portion of the sampling process incorrectly guided by Points-Guided Sampling Network in new environments was replaced by the exploitation process of the RRT-connect

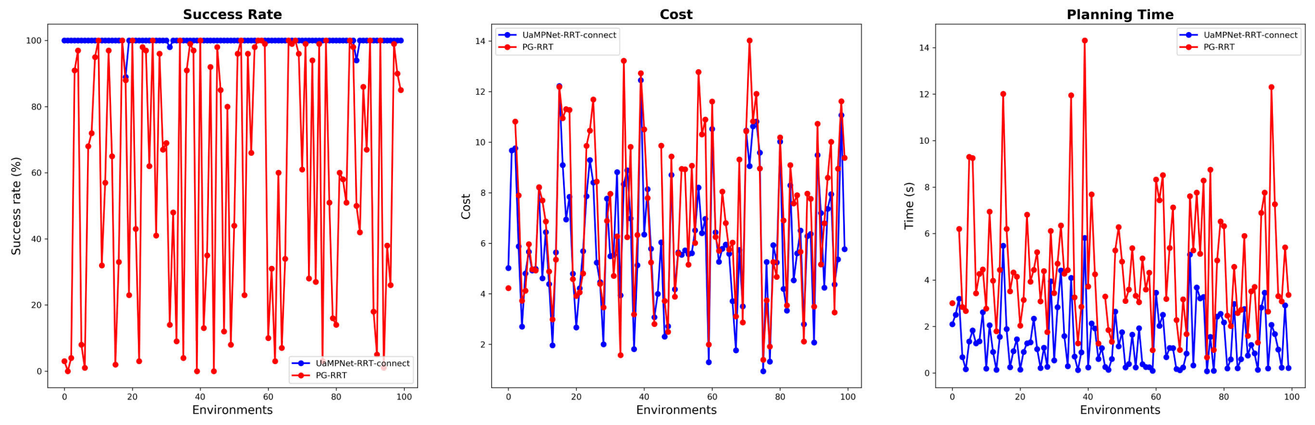

FIGURE 9. Comparison of motion planning results for a Panda robot in 100 unseen office environments. From left to right, the graphs illustrate the average success rate, average cost, and average planning time. Here, blue represents the results of UaMPNet-RRT-connect, while red corresponds to the results of PG-RRT.

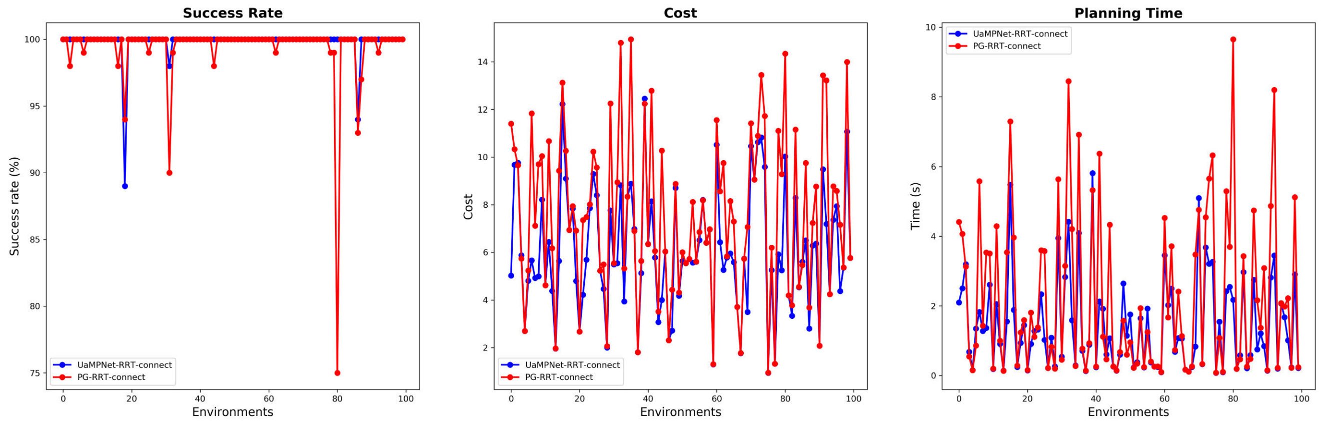

FIGURE 10. Comparison of motion planning results for a Panda robot in 100 unseen office environments based on the differences in the sampling network. From left to right, the graphs illustrate the average success rate, average cost, and average planning time. Here, blue represents the results of RRT-connect using UaMPNet (UaMPNet-RRT-connect), while red corresponds to the results of RRT-connect using Points-Guided Sampling Network (PG-RRT-connect).

algorithm, enabling path creation with less inaccurate sampling. Additionally, in the case of UaMPNet-RRT-connect, motion planning failed at least once in three environments, whereas in the case of PG-RRT-connect, motion planning failed in a total of fifteen environments. In other words, PG-RRT-connect demonstrated one or more failures in motion planning in 12 environments when performing motion planning, in contrast to the proposed algorithm. When comparing the created pathways, UaMPNet created pathways with lower costs in 82 out of 100 environments, and generated pathways faster in 72 environments in terms of pathway creation time. Furthermore, for PG-RRT-connect, the average cost across all environments was 7.53 radians, and the average planning time across all environments was 2.23 seconds. This indicates that PG-RRT-connect also exhibited inferior performance compared to the proposed algorithm.

Finally, we conducted experiments comparing the performance of the proposed network's sampling process guidance with the traditional RRT-connect algorithm. The experiments were carried out in both the trained environment and untrained environments selected from previous experiments. The network was trained in the same manner as before, and the experimental conditions were maintained consistent with previous settings.

In the first experiment, we compared the performance of the proposed algorithm and the RRT-connect algorithm in the trained environment, as defined in V-A-2, with a total of 70 selected environments used for Uncertainty-aware Sampling Network training. One set of initial and goal configuration was chosen for each environment, and motion planning was iteratively performed 100 times for each environment with the selected configurations. Here, the initial and goal configurations for each environment were selected from the data used in network training. Fig.11 illustrates the experimental results in the trained environment. The proposed algorithm experienced only one motion planning failure in a single environment, confirming the effective learning of UaMPNet. In contrast, RRT-connect failed in

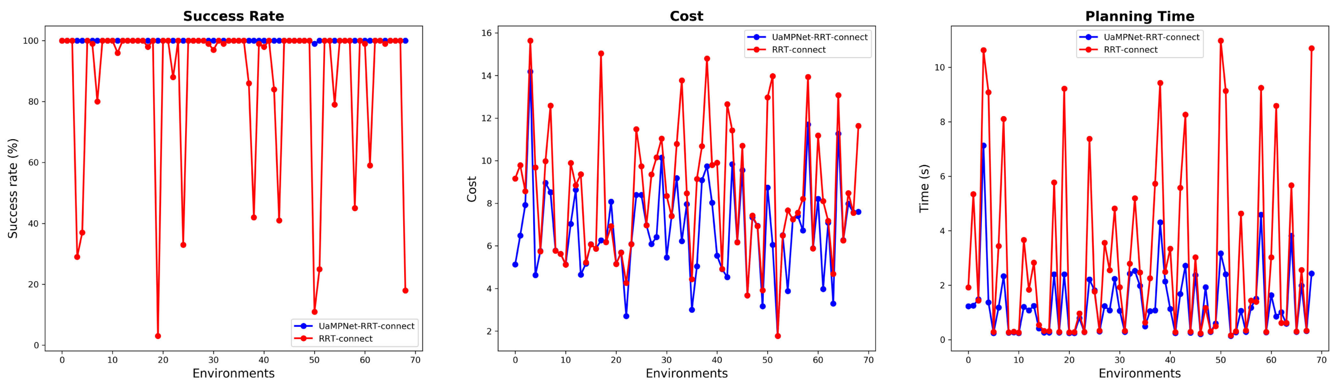

FIGURE 11. Comparison of motion planning results for a Panda robot in 70 seen office environments based on the differences in the motion planning algorithm. From left to right, the graphs illustrate the average success rate, average cost, and average planning time. Here, blue represents the results of UaMPNet-RRT-connect, while red corresponds to the results of traditional RRT-connect.

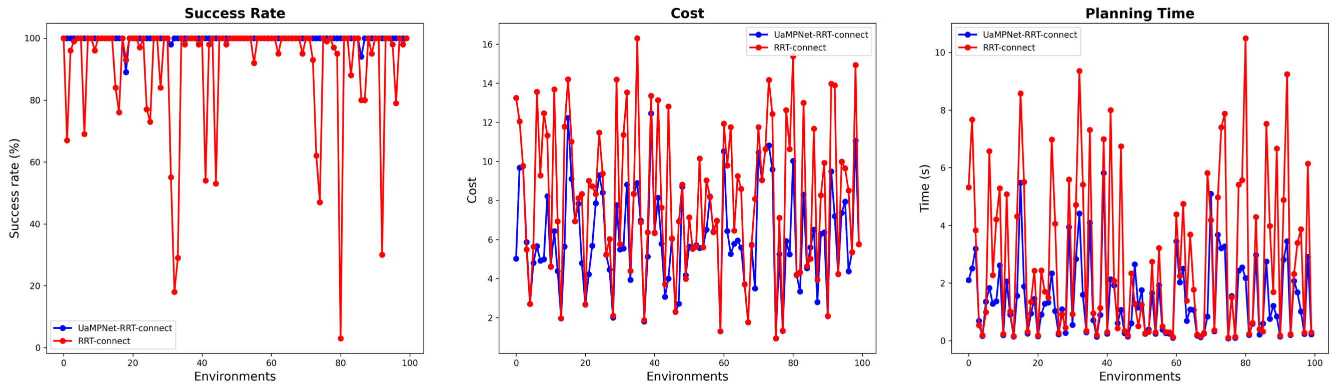

FIGURE 12. Comparison of motion planning results for a Panda robot in 100 unseen office environments based on the differences in the motion planning algorithm. From left to right, the graphs illustrate the average success rate, average cost, and average planning time. Here, blue represents the results of UaMPNet-RRT-connect, while red corresponds to the results of RRT-connect.

motion planning in more than one instance in 26 out of 70 environments. The success rates for all motion planning attempts were 99.99% (6999/7000) for the proposed algorithm and 87.76% (6143/7000) for RRT-connect. Additionally, in terms of cost, the proposed algorithm exhibited an average of 6.73 radians, while RRT-connect showed 8.55 radians. Regarding planning time, the proposed algorithm had an average of 1.35 seconds, whereas RRT-connect had an average of 3.11 seconds. Here, the average refers to the mean across all environments. These results highlight the advantage of using neural networks to guide the sampling process in motion planning, enabling faster and more efficient motion planning compared to using random sampling alone.

In the second experiment, the performance of the proposed algorithm and the RRT-connect algorithm was compared in untrained environments. The untrained environments and initial and goal configurations for motion planning in each environment were selected similarly to the experiments conducted to compare UaMPNet-RRT-connect and PG-RRT-connect. In other words, a total of 100 distinct environments, each containing six obstacles, were chosen. Motion planning was performed iteratively 100 times for each environment, and Fig.12 presents the results. The proposed algorithm achieved a success rate of 99.81% (9981/10000) in untrained environments, which was slightly lower than in trained environments, while the traditional RRT-connect achieved a slightly higher success rate of 91.38% (9138/10000). Additionally, RRT-connect experienced motion planning failures in more than one instance in 39 out of 100 environments. In terms of cost, the proposed algorithm exhibited an average of 6.06 radians across all environments, while RRT-connect showed 8.22 radians. Regarding planning time, the proposed algorithm had an average of 1.45 seconds, and RRT-connect had an average of 2.88 seconds. Here, the average refers to the mean of the results across all environments.

Through these results, we demonstrated the ability to overcome limitations in deep learning-based motion planning algorithms that may fail to provide suitable pathways in new environments without additional methods such as random sampling because of inaccurate predictions. We showed that these limitations can be addressed by leveraging UaMP-Net's uncertainty-based re-sampling process. Furthermore, we successfully alleviated the performance degradation caused by misguided sampling processes and highlighted the advantage of algorithmically incorporating exploitation alongside neural network guidance in performing motion planning in new environments, compared with relying solely

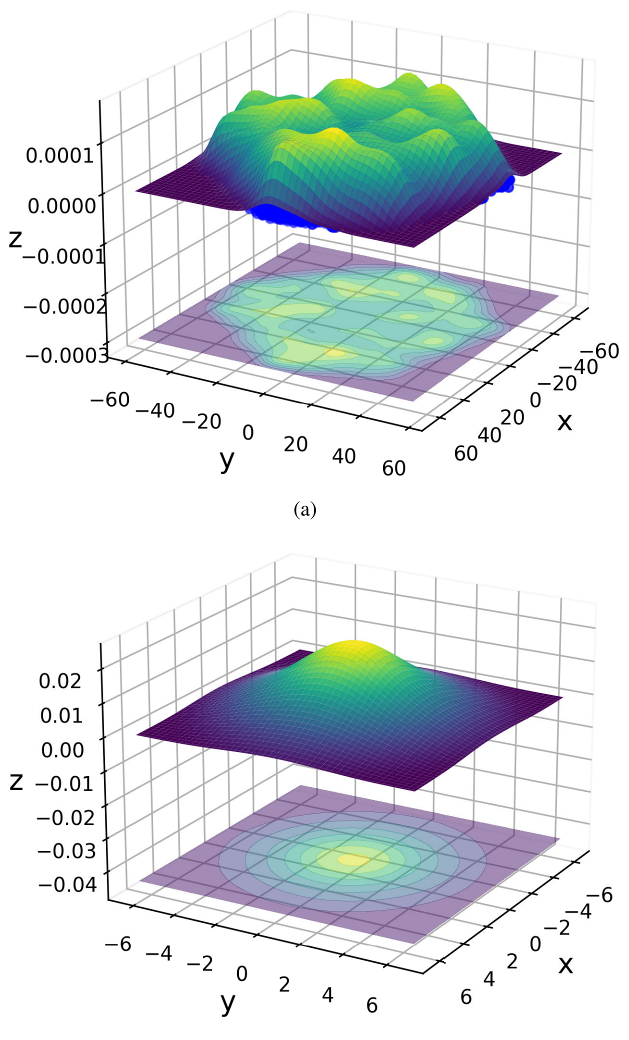

FIGURE 13. Visualizing the latent space distribution of the feature extraction network using t-SNE in a 2D space.(a) represents the outcome of modeling a multimodal distribution using a normalizing flow, while (b) depicts the latent space of a conventional VAE modeled with a Gaussian distribution.

on neural networks which are prone to making inaccurate predictions throughout the sampling process.

D. COMPARISON BASED ON FEATURE EXTRACTION METHODS

Finally, we conducted experiments to evaluate the motion planning generalization performance in new environments based on the difference in feature extraction methods. In this section, we compared our proposed multimodal distribution modeling method for latent space with the Gaussian distribution modeling method proposed in [22]. Fig.13 shows the 2D distribution of the original features $\mathbf{z} \in \mathbb{R}^{256}$ for both methods using t-distributed stochastic neighbor embedding (t-SNE).

To assess the generalization performance based on the difference in feature extraction methods, we constructed UaMPNet-M using the multimodal distribution modeling method and UaMPNet-G using the Gaussian distribution modeling method. We trained both networks on data collected from V-A-2 in the office environment and selected 100 unseen environments with six obstacles each for comparison of motion planning performance, denoted as V-C. Fig.14 shows the motion planning results obtained using both UaMPNet-M and UaMPNet-G. Comparing success rates, UaMPNet-M showed a slightly higher success rate at 99.81% (9981/10000) compared to 99.42% (9942/10000) for UaMPNet-G.

Analyzing the environments where motion planning failed, UaMPNet-M showed a tendency to fail repeatedly in specific environments, whereas UaMPNet-G showed a similar trend and also experienced slight failures in a broader range of environments. Furthermore, in terms of generated path length, UaMPNet-M located pathways with lower cost in 68/100 environments, and UaMPNet-M outperformed in 58/100 environments in terms of motion planning speed. In fact, when comparing the average cost and planning time across all environments, UaMPNet-M and UaMPNet-G exhibited costs of 6.06 radians and 6.78 radians, and planning times of 1.45 seconds and 1.79 seconds, respectively.

This demonstrates that the proposed feature extraction method performs more extensive clustering based on the similarity between environments (similar Uncertainty-aware Sampling Network inputs), ensuring better guidance performance for similar environments. For environments with less similarity to trained environments (where UaMPNet-G failed but UaMPNet-M succeeded), it predicts higher epistemic uncertainty, allowing sampling over a broader range to facilitate exploration using sampling-based motion planning algorithms to locate a possible path.

VI. DISCUSSION

In this section, we compare our proposed method to previous learning-based motion planning algorithms, highlighting their advantages and disadvantages. Previous learning-based motion planning algorithms often face a common challenge in untrained environments, where inaccurate predictions might hinder sampling processes. For example, Motion Planning Networks [19] addresses this by combining a neural sampler with a uniform sampler. However, using a uniform sampler can be inefficient and cause the state to drift away. Furthermore, after the initial path generation, additional methods, such as replanning are required. MπNet [20], attempts to reduce epistemic uncertainty by training the network with a large amount of data, but challenges in out-of-distribution settings persist. When sampling is performed more than a certain number of times near obstacle positions, Points-Guided Sampling Network proposes a method of doubling the variance of the predicted truncated normal distribution. This method requires continuous checking of collision sampling in a given state and significantly increases variation, potentially resulting in sampling over a wide range. Collisions are likely to occur frequently, especially in untrained environments, resulting in a high probability of performing collision

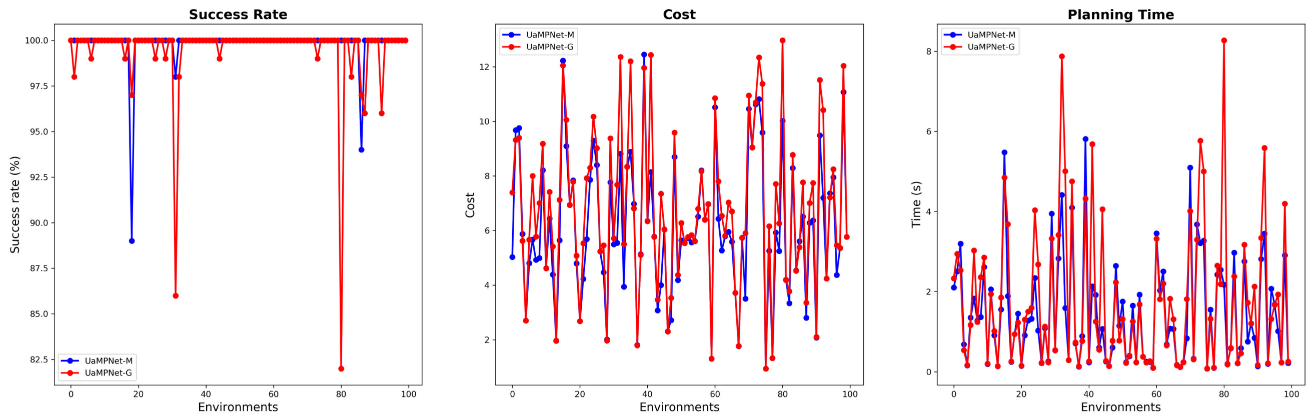

FIGURE 14. Motion planning results for 100 unseen office environments based on the difference in feature extraction methods. From left to right, the graphs represent the average success rate, average cost, and average planning time. Here, the blue color indicates the outcome of the multimodal distribution latent space modeling method, while the red color represents the results of the Gaussian distribution latent space modeling method.

sampling more than a certain number of times. Consequently, many samples are discarded, making the approach inefficient. In untrained environments, our proposed UaMPNet learns uncertainty about predictions by directing the sampling process within a specified range from the beginning without discarding samples until the variance increases. Therefore, we can generate pathways faster and locate shorter pathways by limiting the sampling range appropriately depending on the similarity between the given environment and the training dataset. However, we discovered that our proposed method performed similarly or slightly inferior to PG-RRT in trained environments or environments highly similar to the trained environment. There are three reasons for this: First, our proposed method suffers from the discontinuity problem, where slight environmental changes can result in significant changes in robot motion [22]. This is because of the deep neural network's prediction interpolation, resulting in degraded prediction performance in environments slightly different from the trained environment [42]. Second, the use of the connect heuristic contributes to this. The connect heuristic can generate pathways that approach obstacles, as shown in Fig.6. These paths may not be ideal, and may create longer pathways compared to using only the neural network in trained environments. Finally, the third concerns the regularization coefficient λ in (6). λ plays a role in mitigating the inflation of uncertainty by adjusting the fitness of the network [26]. In other words, the prediction performance of uncertainty in the network is sensitive to the value of λ, and accordingly, it can predict high uncertainty even in the learned environment, potentially disrupting the sampling guidance process. In this study, we experimentally determined the value of λ.

VII. CONCLUSION

The Uncertainty-aware Motion Planning Network (UaMPNet) was proposed in this study to address the issue of degraded motion planning performance in untrained, unfamiliar environments for learning-based motion planning algorithms. UaMPNet is a learning-based sampling network, consisting of a feature extraction network for extracting features and clustering similar environments from the environment's 3D point cloud. It also includes an uncertainty-aware sampling network that guides the sampling process, performs exploration within limited ranges based on the uncertainty of prediction and combines with the RRT-connect algorithm for motion planning. We compared the proposed with state-of-the-art learning-based motion planning in new environments using a 3D environment and an office environment with a Franka Emika Panda robot. The results demonstrated the superior performance of the proposed method. Additionally, we evaluated uncertainty prediction performance with environmental modifications and feature extraction methods, highlighting the high generalization ability of the proposed method for untrained environments. In the future, we will conduct research aimed at enabling neural networks to autonomously request additional data for environments with high uncertainty, leveraging learned uncertainties. Through continuous learning of the network using this approach, we aim to facilitate self-adaptation to new environments, allowing the network to find smooth and optimal paths in novel settings without relying on the connect heuristic. Additionally, we will explore research addressing the discontinuity issue while simultaneously incorporating the learning of uncertainties. This is intended to enhance the network's ability to adapt and navigate smoothly in new environments.

REFERENCES

[1] J. Zhong, T. Wang, and L. Cheng, "Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics," *Complex Intell. Syst.*, vol. 8, no. 3, pp. 1899–1912, Jun. 2022.

[2] R. Ahmad and P. Plapper, "Safe and automated assembly process using vision assisted robot manipulator," *Proc. CIRP*, vol. 41, pp. 771–776, Jan. 2016.

[3] F. Cursi, W. Bai, Eric. M. Yeatman, and P. Kormushev, "Adaptive kinematic model learning for macro-micro surgical manipulator control," in *Proc. Int. Conf. Adv. Robot. Mechatronics (ICARM)*, Jul. 2022, pp. 494–501.

[4] M. Gao, H. Zhou, Y. Yang, Z. Dong, and Z. He, "An intelligent master–slave collaborative robot system for cafeteria service," *Robot. Auto. Syst.*, vol. 154, Aug. 2022, Art. no. 104121.

[5] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1853

[6] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Automat. Symp.*, vol. 2, Apr. 2000, pp. 995–1001.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[8] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1994, pp. 3310–3317.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Mar. 1985, pp. 500–505.

[10] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 489–494.

[11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4569–4574.

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

[13] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 3067–3074.

[14] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*)): Fast asymptotically optimal path planning through adaptive heuristics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3191–3198.

[15] B. Burns and O. Brock, "Sampling-based motion planning using predictive models," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 3120–3125.

[16] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7087–7094.

[17] C. Gaebert and U. Thomas, "Learning-based adaptive sampling for manipulator motion planning," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2022, pp. 715–721.

[18] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 2118–2124.

[19] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 48–66, Feb. 2021.

[20] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *Proc. Conf. Robot Learn.*, 2023, pp. 967–977.

[21] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[22] E. Lyu, T. Liu, J. Wang, S. Song, and M. Q.-H. Meng, "Motion planning of manipulator by points-guided sampling network," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 821–831, Apr. 2023.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent.*, 2014, pp. 1–14.

[24] H. Ma, Z. Han, C. Zhang, H. Fu, J. T. Zhou, and Q. Hu, "Trustworthy multimodal regression with mixture of normal-inverse gamma distributions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 6881–6893.

[25] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 37, 2015, pp. 1530–1538.

[26] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 14927–14937.

[27] T. Rybus and K. Seweryn, "Application of rapidly-exploring random trees (RRT) algorithm for trajectory planning of free-floating space manipulator," in *Proc. 10th Int. Workshop Robot Motion Control (RoMoCo)*, Jul. 2015, pp. 91–96.

[28] C. Yuan, W. Zhang, G. Liu, X. Pan, and X. Liu, "A heuristic rapidly-exploring random trees method for manipulator motion planning," *IEEE Access*, vol. 8, pp. 900–910, 2020.

[29] C. Lau and K. Byl, "Smooth RRT-connect: An extension of RRT-connect for practical use in robots," in *Proc. IEEE Int. Conf. Technol. Practical Robot Appl. (TePRA)*, May 2015, pp. 1–7.

[30] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "RRT*-connect: Faster, asymptotically optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2015, pp. 1670–1677.

[31] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.

[32] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via Oracle imitation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 3965–3972.

[33] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.

[34] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mache approach to learning 3D surface generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 216–224.

[35] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[38] N. Meinert, J. Gawlikowski, and A. Lavin, "The unreasonable effectiveness of deep evidential regression," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 8, pp. 9134–9142.

[39] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[40] J. Burkardt, "The truncated normal distribution," Dept. Sci. Comput. Website, Florida State Univ., Tallahassee, FL, USA, pp. 1–35, 2014.

[41] S. Luo and W. Hu, "Diffusion probabilistic models for 3D point cloud generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2836–2844.

[42] G. Tang and K. Hauser, "Discontinuity-sensitive optimal control learning by mixture of experts," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 7892–7898.

EUNHOO LEE received the bachelor's degree from the Department of Mechanical Engineering, Gachon University, South Korea, in 2022. He is currently pursuing the master's degree in mechanical engineering with Yonsei University. His research interests include robotics, motion planning, and deep learning.

HYUNSEOK YANG received the bachelor's degree from the Department of Mechanical Engineering, Yonsei University, Seoul, in 1984, and the M.S. and Ph.D. degrees from the Department of Mechanical Engineering, Massachusetts Institute of Technology (MIT), in 1988 and 1993, respectively. He is currently a Professor with the Department of Mechanical Engineering, Yonsei University, and coordinates research on motion control and robotics.