






NR-RRT: Neural Risk-Aware Near-Optimal Path Planning in Uncertain Nonconvex Environments

Fei Meng , Liangliang Chen , Han Ma , Jiankun Wang* , Senior Member, IEEE,
Max Q.-H. Meng* , Fellow, IEEE

Abstract—Balancing the trade-off between safety and efficiency is of significant importance for path planning under uncertainty. Many risk-aware path planners have been developed to explicitly limit the probability of collision to an acceptable bound in uncertain environments. However, convex obstacles or Gaussian uncertainties are usually assumed to make the problem tractable in the existing method. These assumptions limit the generalization and application of path planners in real-world implementations. In this article, we propose to apply deep learning methods to the sampling-based planner, developing a novel risk bounded near-optimal path planning algorithm named neural risk-aware RRT (NR-RRT). Specifically, a deterministic risk contours map is maintained by perceiving the probabilistic nonconvex obstacles, and a neural network sampler is proposed to predict the next most-promising safe state. Furthermore, the recursive divide-and-conquer planning and bidirectional search strategies are used to accelerate the convergence to a near-optimal solution with guaranteed bounded risk. Worst-case theoretical guarantees can also be proven owing to a standby safety guaranteed planner utilizing a uniform sampling distribution. Simulation experiments demonstrate that the proposed algorithm outperforms the state-of-the-art remarkably for finding risk bounded low-cost paths in seen and unseen environments with uncertainty and nonconvex constraints.

Note to Practitioners—This article is motivated by developing an efficient risk-aware path planner that can quickly find risk bounded solutions in uncertain nonconvex environments for practical applications, such as surgical navigation and delivery in crowded squares. Sampling-based planning approaches such as rapidly-exploring random tree (RRT) and its variants are popular for their good performance in exploring the state space. However, it is quite time-consuming to look for risk bounded

paths in uncertain environments, especially under nonconvex and non-Gaussian constraints. The initial paths are often of poor quality as well. Therefore, we propose the NR-RRT algorithm to rapidly find near-optimal solutions with guaranteed bounded risk. It utilizes an informed bidirectional search strategy after having past experiences in those challenging environments. NR-RRT can be applied in not only seen but also unseen uncertain scenarios. However, the algorithm cannot handle entirely unseen environments that contain new or additional obstacles. In future research, we will address the problem of planning under robot model uncertainty.

Index Terms—Planning under uncertainty, Sampling-based path planning, Learning from demonstration.

I. INTRODUCTION

ROBOTS are often required to navigate safely under sensing uncertainties in many practical applications [1]. In such circumstances, robots need to plan safe trajectories prior to execution where the probability of collision with uncertain obstacles is limited to a user-specified bound [2]. To this end, a desirable risk-aware path planning approach should have the following critical features: 1) completeness and optimality guarantee—a path will be eventually found if one exists, and its cost is the lowest, 2) adaptation to environment constraints—the algorithm can adapt to diverse uncertain environments where the obstacles can be convex or not and have arbitrary probabilistic uncertainties, and effectively generate paths that satisfy any user-preferred risk level, 3) computational efficiency—time and memory consumed to find a solution should be as less as possible. At present, sampling-based motion planners (SBMPs), such as probabilistic roadmap (PRM) [3] and rapidly-exploring random tree (RRT) [4], have become prominent because they can quickly find a solution and guarantee probabilistic completeness [5]. However, the pre-construction of a roadmap makes the PRM-based approaches impractical for online planning. In contrast, RRT algorithm iteratively builds a rapidly-exploring tree to connect the robot's collision-free states. The primary solution is obtained by querying the constructed graph after the initial and goal states are contained in the tree. Despite RRT and its variants guarantee to find a solution, they often fail to find the shortest one. RRT* [7], an optimal variant of RRT, was proposed to additionally keep the cost of the solution decreasing until being the lowest, i.e., it guarantees asymptotic optimality. Unfortunately, it not only needs to assume perfect measurement information but also takes a large amount of time to find the optimal solution.

This work was supported in part by the Hong Kong RGC TRS grant T42-409/18-R, Hong Kong RGC CRF grant C4063-18G, Hong Kong Health and Medical Research Fund (HMRP) under Grant 06171066, and Hong Kong RGC GRF grants #14211420 awarded to Max Q.-H. Meng. (Corresponding authors: Jiankun Wang and Max Q.-H. Meng.)

F. Meng and H. Ma are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: {feimeng, hanma}@link.cuhk.edu.hk).

L. Chen is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: liangliang.chen@gatech.edu).

J. Wang is with Shenzhen Key Laboratory of Robotics Perception and Intelligence, and the Department of Electronic and Electrical Engineering of the Southern University of Science and Technology, Shenzhen, China (e-mail: wangjk@sustech.edu.cn).

Max Q.-H. Meng is with Shenzhen Key Laboratory of Robotics Perception and Intelligence, and the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China, on leave from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute of The Chinese University of Hong Kong, Shenzhen 518057, China (e-mail: max.meng@ieee.org).

Color versions of one or more of the figures in this article are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier

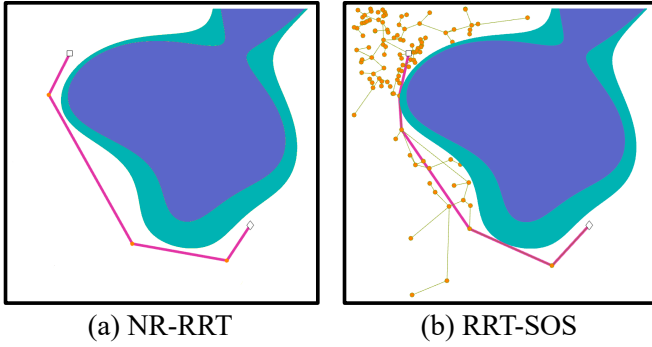


Fig. 1: NR-RRT can find comparable risk bounded near-optimal path solutions in uncertain nonconvex environments with much higher computation efficiency after exploiting learned distributions compared to RRT-SOS [6].

Recently, some references have extended the SBMPs to systems with noisy observations to obtain risk bounded trajectories. For example, Axelrod et al. [8] modified RRT by verifying safety certificates regarding polytopes with Gaussian distributed faces to generate safe trajectories in uncertain circumstances. However, most existing sampling-based risk-aware planning methods are limited to either convex obstacles or Gaussian uncertainties [8]–[11]. Although the probabilistic location, size, and geometry of uncertain nonconvex obstacles can be explicitly described according to the prior knowledge of environments, dealing with complex and nonconvex constraints is still intractable [6]. Jasour et al. [12] transformed the uncertain environment into the deterministic information map named risk contours map through moment-based methods, no matter what probabilistic distributions over the uncertain parameters of the obstacles are. The map’s safe regions also vary with different predefined risk levels. Then, they proposed RRT-SOS, which constructs an exploring-tree graph in the risk contours map. By introducing the sum of squares (SOS) techniques, the algorithm can provide safety guarantees for the edges of a tree without the need for time discretization [6]. However, this risk-aware path planning algorithm suffers from a heavy computation burden.

The excellent performance achieved by combining traditional SBMPs and machine learning techniques has caught the attention of the motion planning community. In our previous work [13], an efficient learning-based optimal path planner was developed, the Neural RRT*, which significantly reduces the planning time. It guides the sampling process of RRT* according to a nonuniform probability distribution that is learned from the expert paths generated by the A* algorithm [14]. In addition, Qureshi et al. [15] proposed the deep neural network-based bidirectional iterative motion planning method called MPNet to generate collision-free paths in real-time. By taking advantage of latent space encoded from the workspace, MPNet can explicitly generate the deeply informed samples for the SBMPs in both seen and unseen scenarios of certainty. Adding lazy states contraction and bidirectional search strategies additionally contributes to high-performance planning. However, developing an efficient learning-based risk-aware

path planner in uncertain nonconvex environments is still an open problem.

To fill this gap, we propose a neural risk-aware path planning method, neural risk-aware RRT (NR-RRT), to find risk bounded near-optimal solutions in uncertain nonconvex environments through learning from RRT-SOS algorithm. The pipeline of our algorithm is illustrated in Fig. 2. A risk-aware path planning problem is formulated given an uncertain nonconvex environment, a user-specified risk level, and the start and goal states. To capture the information of diverse uncertain nonconvex obstacles, we record their probabilistic locations, sizes, geometries, and the demand of risk level in a risk contours map. Then, we train a neural network sampler consisting of an encoder and inference network. The encoder embeds the features of the risk contours map. At the same time, the inference network learns from abundant expert demonstration paths generated by RRT-SOS to predict the risk bounded node. The sampler will iteratively output the next informed state that is probably to be contained in the resulting near-optimal solution. Next, a risk-aware bidirectional neural planning strategy utilizes the sampler to extend two trees incrementally and employs a risk assessor to verify every edge’s safety. This strategy will be recursively called until a fixed number of iterations. Once the trees are connected, the risk bounded near-optimal path is found. In case it cannot be found within a limited period, the algorithm calls a standby path planner that guarantees completeness to continue.

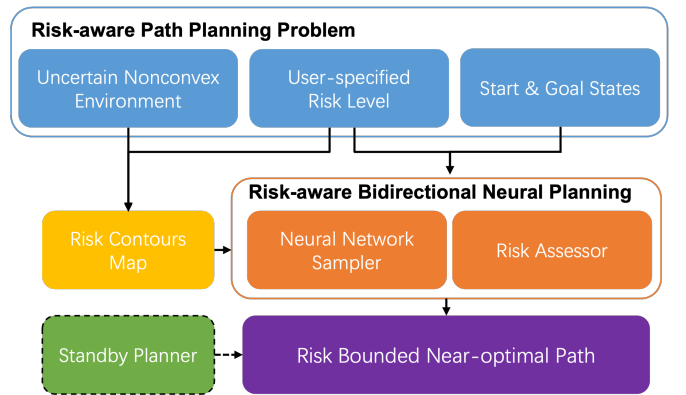


Fig. 2: Block diagram illustrating the pipeline of NR-RRT algorithm.

The main contribution of this article is threefold. Firstly, we propose a neural network sampler to iteratively generate the risk bounded and informed states in uncertain nonconvex environments. Secondly, we propose a computationally efficient risk-aware path planning approach, NR-RRT, which can find the risk-bound near-optimal path solutions rapidly. Thirdly, our method can generalize to unseen uncertain nonconvex scenarios different from those in the training dataset. Compared with [15] and our previous method [13], NR-RRT has improved in three aspects: 1) not only considers risk level, clearance called in [13] but also takes explicit probabilistic uncertainties of obstacles into account, 2) can handle path planning problems in both uncertain convex and nonconvex environments, 3) guarantees a bounded probability of colli-

sion in such environments. In addition, NR-RRT can obtain comparable results with significantly reduced computational costs compared to RRT-SOS in [6].

The remainder of the article is organized as follows. We review the main piece of planning methods that are related to this topic in Section II. We formulate the risk-aware path planning problem in uncertain nonconvex scenarios and introduce preliminary in Section III, and then present the details of NR-RRT algorithm in Section IV. We conduct simulation experiments and analyze the results in Section V. Section VI proves worst-cases theoretical guarantees for NR-RRT and discuss its asymptotic optimality. Finally, we draw conclusions and present consideration of future works in Section VII.

II. RELATED WORK

A risk-aware path planning algorithm aims to find a feasible path connecting given start and goal states, if one exists, in an uncertain environment; meanwhile, the probability of collision with uncertain obstacles for the solution is no larger than a user-specified bound [2]. To this end, SBMPs have been extended to look for such solutions under measurement uncertainty [11], [16], [17]. For instance, Janson et al. [16] proposed a Monte Carlo-based motion planner to find safe trajectories in which the probability of collision is estimated by sampling. However, there are no analytical bounds of the collision risk of the trajectory. An alternative for planning under uncertainty is the chance-constraint strategy [18]–[20]. Instead of maximizing the probability of success, it aims to compute a feasible solution in which the waypoints or segmented trajectories satisfy a minimum probability of collision constraint. Johnson et al. [19] developed a motion planner that was capable of reducing the computation complexity in high-dimensional and obstacle-filled spaces. Furthermore, performance improvements can be gained by combining stochastic optimal control with traditional SBMPs [9], [21]–[23]. The linear-quadratic Gaussian motion planning algorithm (LQGM) [22] incorporated a local controller into the extensions procedure of RRT to deal with noisy sensing. Its extended version environment-guided RRT [23] estimated path quality and guided sampling toward safer parts of the state-space. Planning under uncertainty can also be modeled as a Markov decision process (MDP) [24] if the fully observable system states exist, otherwise as a partially observable Markov decision process (POMDP) [25]–[27]. For example, Van et al. [25] proposed the belief iterative LQG to obtain a time-varying affine feedback controller by approximating the quadratic value function over a belief space. It has all feasible probability distributions over the robot state space. However, the methods above were built upon strong assumptions, e.g., convex obstacles and sensor noise sampled from a Gaussian distribution, which are limited in more general cases such as nonconvex obstacles and non-Gaussian measurement noises. Note that complex and nonconvex constraints are often imposed in planning problems due to the safety reason [28]. A new arisen branch of the planning method can address non-Gaussian uncertainties [29], nonconvex planning problem [2], or combination of them [6] when sensing uncertainty dominates. However, the computa-

tional costs of these methods are still prohibitive because they need plenty of samples or multiple iterations to find a solution.

In recent years, reinforcement learning (RL) and deep learning (DL) have emerged as promising motion planning tools [30]. The advanced version of RL, called deep RL (DRL), is shown capable of effectively planning under uncertainty through incorporating traditional RL with deep neural networks [31], [32]. Faust et al. [33] trained multiple RL agents on the smallest map in simulation with sensing noise to learn robust near point-to-point navigation policies. Then, the PRM-based global planner uses the best policy to construct roadmaps in the robot's C-space where all configurations along the path are safe, realizing a long-range motion planner. To learn resilient actions for navigation in unseen uncertain environments, Fan et al. [34] presented an uncertainty-aware predictor and policy network to gain the uncertain information of circumstances and learn the desirable behaviors, respectively. Although these methods can generalize to new environments or tasks, they may suffer from weak rewards or require discretization of the state space.

The function approximation ability of deep neural networks can be employed to speed up the convergence of SBMPs to the optimal solution [35], [36]. One main category is to learn bias sampling heuristic to guide the sampling process, namely, to learn the promising probability distribution of the optimal solution [13], [37], while another is to use neural networks to embed a planner [15], [36]. For example, Zhang et al. [38] proposed a generative adversarial network model to compute the most promising area on a map such that the sampling process can be guided. A method that constructs the SBMP graph and conducts the collision-checking procedure both in learned latent spaces was proposed in [36]. Although these neural planners have achieved remarkable performance in deterministic convex environments, realizing rapid path planning in uncertain nonconvex scenarios remains challenging.

III. PRELIMINARY

A. Problem Definition

Let $\mathcal{X} \in \mathbb{R}^{n_x}$ be an uncertain environment containing safe space $\mathcal{X}_{\text{safe}} \subset \mathcal{X}$ and static uncertain obstacles $\mathcal{X}_{\text{obs}_i}(\omega_i) \subset \mathcal{X}$, $i = 1, \dots, n_o$, where $\omega_i \in \mathbb{R}^{n_\omega}$ represent probabilistic parameters with given probability distributions. All obstacles can be convex or nonconvex, and each obstacle may have an uncertain size, location, or geometry. We represent them in terms of polynomials in $\mathbf{x} \in \mathcal{X}$ as below:

$$\mathcal{X}_{\text{obs}_i}(\omega_i) = \{\mathbf{x} \in \mathcal{X} : \mathcal{P}_i(\mathbf{x}, \omega_i) \geq 0\}, i = 1, \dots, n_o, \quad (1)$$

where $\mathcal{P}_i : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ denotes the known polynomial uncertain obstacles.

Take an ellipse-shaped uncertain convex obstacle as an example, and a nonconvex obstacle is similar. The polynomial of it is $\mathcal{P} : \{(x, y) : \omega^2 - x^2/2 - y^2 \geq 0, \omega \sim \mathcal{N}(0, 1)\}$. Its major and minor axes are subjected to a Gaussian distribution of random variable ω . Note that the type and dimension of the probability distribution should depend on the realistic situation.

Given a safe initial state \mathbf{x}_{init} , goal state \mathbf{x}_{goal} , and an uncertain environment, we intend to find the shortest and risk bounded path $\pi = [\mathbf{x}_{\text{init}}, \dots, \mathbf{x}_{\text{goal}}] : [0, T] \rightarrow \mathcal{X}_{\text{safe}}$ such that $\pi(0) = \mathbf{x}_{\text{init}}$ and $\pi(T) \in \mathcal{X}_{\text{target}}(\mathbf{x}_{\text{goal}})$, where $\mathcal{X}_{\text{target}}(\mathbf{x}_{\text{goal}}) = \{\mathbf{x} \in \mathcal{X}_{\text{safe}} \mid \|\mathbf{x} - \mathbf{x}_{\text{goal}}\| < r\}$ is the target region commonly used in practice. Meanwhile, the probability of collision with uncertain obstacles is no greater than Δ , where $\Delta \in [0, 1]$ is a predefined constant representing the acceptable risk level.

Further, the risk-aware optimal path planning is defined mathematically as follows:

$$\min_{\pi: [0, T] \rightarrow \mathcal{X}_{\text{safe}}} \int_0^T \|\dot{\mathbf{x}}(t)\|_2^2 dt \quad (2)$$

$$\text{s.t. } \mathbf{x}(0) = \mathbf{x}_{\text{init}}, \mathbf{x}(T) \in \mathcal{X}_{\text{target}}, \quad (3)$$

$$\text{Prob}(\mathbf{x}(t) \in \mathcal{X}_{\text{obs}_i}(\omega_i)) \leq \Delta, \forall t \in [0, T] \mid_{i=1}^{n_o}, \quad (4)$$

where (2) denotes the cost function regarding the length of trajectory π measured by \mathcal{L}_2 norm, and (4) are the collision risk constraints in terms of probability for the trajectory π .

B. Risk Contours Map

We introduce a cutting-edge mapping method, called risk contours map [12], to describe the probabilistic information of uncertain obstacles. This map depicts the relatively safe region in the uncertain environment where the probability of collision with the uncertain obstacles is no greater than a given risk level Δ . Specifically, considering one uncertain obstacle $\mathcal{X}_{\text{obs}}(\omega)$ in (1), we define the Δ -risk contour \mathcal{C}_r^Δ as the following set of states:

$$\mathcal{C}_r^\Delta := \{\mathbf{x} \in \mathcal{X} : \text{Prob}(\mathbf{x} \in \mathcal{X}_{\text{obs}}(\omega) \leq \Delta)\}. \quad (5)$$

Then, to transform the original probabilistic optimization problem (2) into a solvable one, a deterministic constraint in terms of the set of safe states is approximated as follows [6]:

$$\hat{\mathcal{C}}_r^\Delta = \left\{ \mathbf{x} \in \mathcal{X} : \frac{\mathbb{E}[\mathcal{P}^2(\mathbf{x}, \omega)] - \mathbb{E}[\mathcal{P}(\mathbf{x}, \omega)]^2}{\mathbb{E}[\mathcal{P}^2(\mathbf{x}, \omega)]} \leq \Delta, \mathbb{E}[\mathcal{P}(\mathbf{x}, \omega)] \leq 0 \right\} \quad (6)$$

where $\mathcal{P}(\mathbf{x}, \omega)$ is the known polynomial of the uncertain obstacle $\mathcal{X}_{\text{obs}}(\omega)$. $\mathbb{E}[\mathcal{P}(\mathbf{x}, \omega)]$ and $\mathbb{E}[\mathcal{P}^2(\mathbf{x}, \omega)]$ are polynomials in terms of states \mathbf{x} and the moments of probabilistic parameter ω . In other words, the coefficients of \mathbf{x} in them are computed by using the moment of ω and the coefficients of $\mathcal{P}(\mathbf{x}, \omega)$.

Note that the set $\hat{\mathcal{C}}_r^\Delta$ is a rational polynomial-based inner approximation of the original risk contour \mathcal{C}_r^Δ . Thus, we have a deterministic constraint in terms of the set of risk bounded states $\mathbf{x} \in \hat{\mathcal{C}}_r^\Delta, \forall t \in [0, T]$. The resulting path π comprising of these states is guaranteed to have a no greater than Δ chance of collisions. For more details, the readers are referred to [6]. Finally, we have a deterministic polynomial optimization problem with risk contours as follows:

$$\min_{\pi: [0, T] \rightarrow \mathcal{X}_{\text{safe}}} \int_0^T \|\dot{\mathbf{x}}(t)\|_2^2 dt \quad (7)$$

$$\text{s.t. } \mathbf{x}(0) = \mathbf{x}_{\text{init}}, \mathbf{x}(T) \in \mathcal{X}_{\text{target}}, \quad (8)$$

$$\mathbf{x}(t) \in \hat{\mathcal{C}}_{r_i}^\Delta, \forall t \in [0, T] \mid_{i=1}^{n_o}. \quad (9)$$

C. RRT-SOS

RRT algorithm is known for rapid space exploration and can always find a solution, if one exists, as the number of samples keeps increasing. It first performs sampling from the state space according to some sampling distribution, given a start state as the root vertex of a searching tree. Then, it selects the nearest vertex x_{nearest} and tries to connect it through an extension function. If the connection is collision-free after performing collision detection, x_{new} adjusted from x_{sample} will be a new vertex and added to the vertex set of the searching tree. The pair $(x_{\text{nearest}}, x_{\text{new}})$ will also be included in the edge set. Repeating the procedures above until a state in the target region $\mathcal{X}_{\text{target}}$ is contained in the tree, and then the algorithm returns the solution. However, the collision detection needs to be modified since we have probabilistic polynomial information $\mathcal{P}_i(\mathbf{x}, \omega_i)$ rather than certain polynomial information $\mathcal{P}_i(\mathbf{x})$ of the obstacles in uncertain environments. We need to make sure the probability of collision for the connected path is bounded by Δ instead of zero. In other words, the paths should be risk bounded rather than purely collision-free due to environmental uncertainty.

For this purpose, Jasour et al. [6] proposed RRT-SOS algorithm, which uses the safety sum of squares (SOS) condition as follows to check the collision risk of the trajectory between x_{nearest} and x_{new} in an uncertain environment. $\mathcal{S} = \{\mathbf{x} : g_k(\mathbf{x}) \geq 0, k = 1, \dots, l\}$ is defined as the feasible point set of (7), where $g_k(\cdot) \geq 0, k = 1, \dots, l$ are polynomial constraints of all the inner approximations of risk contours $\hat{\mathcal{C}}_{r_i}^\Delta, i = 1, \dots, n_o$. The trajectory $\mathbf{x}(t)$ between two vertices satisfies the constraint (9) over the time interval $[t_1, t_2]$, iff polynomials $g_k(\mathbf{x}(t)), k = 1, \dots, l$ has the SOS form below:

$$g_k(\mathbf{x}(t)) = \pi_{0k}(t) + \pi_{1k}(t)(t - t_1) + \pi_{2k}(t)(t_2 - t) \mid_{k=1}^l, \quad (10)$$

where $\pi_{0k}(t), \pi_{1k}(t)$, and $\pi_{2k}(t)$ are SOS polynomials with suitable degrees.

The authors in [6] also initialized a straight line between the start and target vertices to adaptively sample in its neighborhood during the expansion, which relieved the computation burden to some extent. After finding a primary path that connects the start and goal states, a PRM graph whose nodes and edges satisfy the SOS condition in (10) is constructed for the solution. Then, the Dijkstra algorithm, a shortest path-finding method, is performed to reduce the path length of the feasible solution by querying the generated graph.

IV. NR-RRT: A NEURAL RISK-AWARE PATH PLANNER

In this section, we first describe how to process the uncertain environment to facilitate the training of our neural networks in Section IV-A. The architecture and other details of the proposed neural network sampler are introduced in Sections IV-B. NR-RRT algorithm is presented in Section IV-C.

A. Pre-processing

2-D images are commonly used to represent the 2-D space. An image is usually converted into an occupancy grid map in which zero/one indicates free/occupied space in definite environments. However, it is insufficient to present the space

information of uncertain obstacles using this kind of map because we cannot be sure whether some grids are occupied or not. To solve this problem, we construct a risk contours map that replaces the probabilistic grids with deterministic ones in the image. Specifically, by computing the inner approximation of static risk contour in (6), we obtain continuous risk levels in $[0, 1]$ for the coordinates in the risk contours map instead of $\{0, 1\}$ used for grid-based maps.

We take the 2-D ellipse-shaped uncertain obstacle in Section III-A as an example again. Given the polynomial $\mathcal{P}(\mathbf{x}, \omega)$ and the distribution of uncertain parameter $\omega \sim \mathcal{N}(0, 1)$, we can calculate the first and second moments as

$$\begin{aligned}\mathbb{E}[\mathcal{P}(\mathbf{x}, \omega)] &= \mathbb{E}[\omega^2 - 0.5x^2 - y^2] \\ &= -0.5x^2 - y^2 + \mathbb{E}[\omega^2] = -0.5x^2 - y^2 + 1, \\ \mathbb{E}[\mathcal{P}^2(\mathbf{x}, \omega)] &= \mathbb{E}[(\omega^2 - 0.5x^2 - y^2)^2] \\ &= 0.25x^4 + y^4 + x^2y^2 - \mathbb{E}[\omega^2](x^2 + 2y^2) + \mathbb{E}[\omega^4] \\ &= 0.25x^4 + y^4 + x^2y^2 - x^2 - 2y^2 + 3.\end{aligned}\quad (11)$$

We substitute the moments (11) into the deterministic constraint $\hat{\mathcal{C}}_r^\Delta$ in (6), obtaining two inequalities without the uncertain parameter but symbolic variables, x and y . By substituting the coordinates (x, y) in the environment into the inequalities, we can split the risk contours map into three parts: safe zone, dangerous zone, and risk zone, as shown in Fig. 3. More precisely, the safe zone (white) means the inequalities

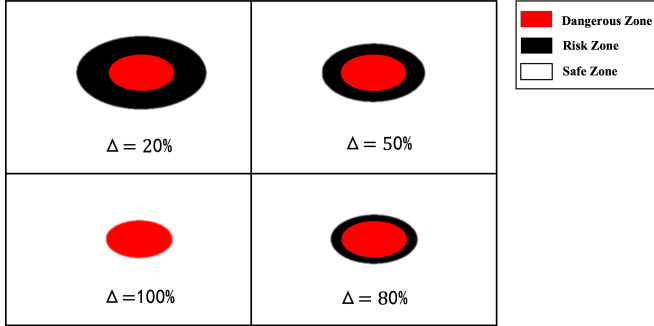


Fig. 3: The risk contours maps with different Δ for an ellipse-shaped uncertain obstacle. Δ is a predefined acceptable probability of collision. The dangerous zone (red) is fixed. In contrast, the risk zone (black) shrinks as the risk level Δ increases, as shown in the subfigures clockwise from the left-top, indicating that the farther away from the dangerous zone, the smaller probability of collision.

in (6) hold for all coordinates in it, while the dangerous zone (red) and risk zone (black) indicate $\mathbb{E}[\mathcal{P}(\mathbf{x}, \omega)] \leq 0$ and $\frac{\mathbb{E}[\mathcal{P}^2(\mathbf{x}, \omega)] - \mathbb{E}[\mathcal{P}(\mathbf{x}, \omega)]^2}{\mathbb{E}[\mathcal{P}^2(\mathbf{x}, \omega)]} \leq \Delta$ do not hold, respectively. In other words, only states in the safe zone have a no greater than Δ chance of collisions.

Furthermore, the image has a size of $W \times H \times C$, corresponding to width, height, and the number of channels, respectively, and every pixel in it has a value from 0 to 255 in each channel. As a result, pixels in the image-based risk contours map have three kinds of values corresponding to three colors of the zones, i.e., the values of pixels are $(0, 0, 0)$ in the risk zone (black), $(255, 255, 255)$ in the safe zone (white),

and $(255, 0, 0)$ in the dangerous zone (red). Note that we do not require discretization of the 2-D environment, and the positions of pixels in the image are not equal to the coordinate values in the risk contours map, which means our method will not suffer from the exponentially increased computation cost brought by the increased map size. Therefore, we record the continuous probabilistic observations of uncertain obstacles in image I . We also normalize the path data generated by RRT-SOS with respect to the boundaries of the risk contours map to facilitate the training.

B. Neural Network Sampler Architecture

We design an end-to-end neural network sampler (NNS) consisting of an observation encoder network \mathcal{E} with weights θ_e and an inference network \mathcal{I} with weights θ_i , which is inspired by MPNet [15]. The architecture of our proposed neural networks is shown in Fig. 4.

Firstly, the encoder \mathcal{E} embeds the observation of obstacles in images I into latent features Z ,

$$\mathcal{E}(I; \theta_e) \rightarrow Z. \quad (12)$$

The fully connected networks treat pixels far apart and those close together in the same way. It ignores the local features and is not suitable for image data processing. By contrast, the convolutional neural networks (CNNs) have impressive features such as local connectivity, shared weights, and pooling, leading to their better generalization with lower memory and fewer free parameters [39]. We employ a CNN as our encoder to automatically extract latent features Z from images I . It has two 2D convolutional layers with the kernel size 3×3 . Each is connected with batch normalization (BN), an activation function, the rectified linear unit (ReLU), and a downsampling layer, max pooling, with filters of size 2×2 . The input and output channels of the first and second layers are $(3, 32)$ and $(32, 128)$, respectively. Every convolutional layer takes one image-based map and outputs a low-level feature map. The ReLU allows the CNN to be trained faster, and the max-pooling layer makes it more robust to variations in the shapes of uncertain obstacles. The image is then abstracted to the latent features Z with a size of 64 through two fully connected layers (FCLs).

After that, the latent-space embedding Z concatenated with the current state $\mathbf{x}_t \in \mathcal{X}_{\text{safe}}$, goal state $\mathbf{x}_{\text{goal}} \in \mathcal{X}_{\text{safe}}$, and risk level Δ are fed into the inference network \mathcal{I} , and it outputs the next state $\hat{\mathbf{x}}_{t+1}$ towards the goal state \mathbf{x}_{goal} ,

$$\mathcal{I}(Z, \mathbf{x}_t, \mathbf{x}_{\text{goal}}, \Delta; \theta_i) \rightarrow \hat{\mathbf{x}}_{t+1}. \quad (13)$$

The inference network \mathcal{I} comprises six FCLs, and each of them has Dropouts [40] to let the network \mathcal{I} play a stochastic behavior. This way increases the probability of exploring the whole state space rather than being trapped in several informed samples, which is better to maintain the completeness guarantee.

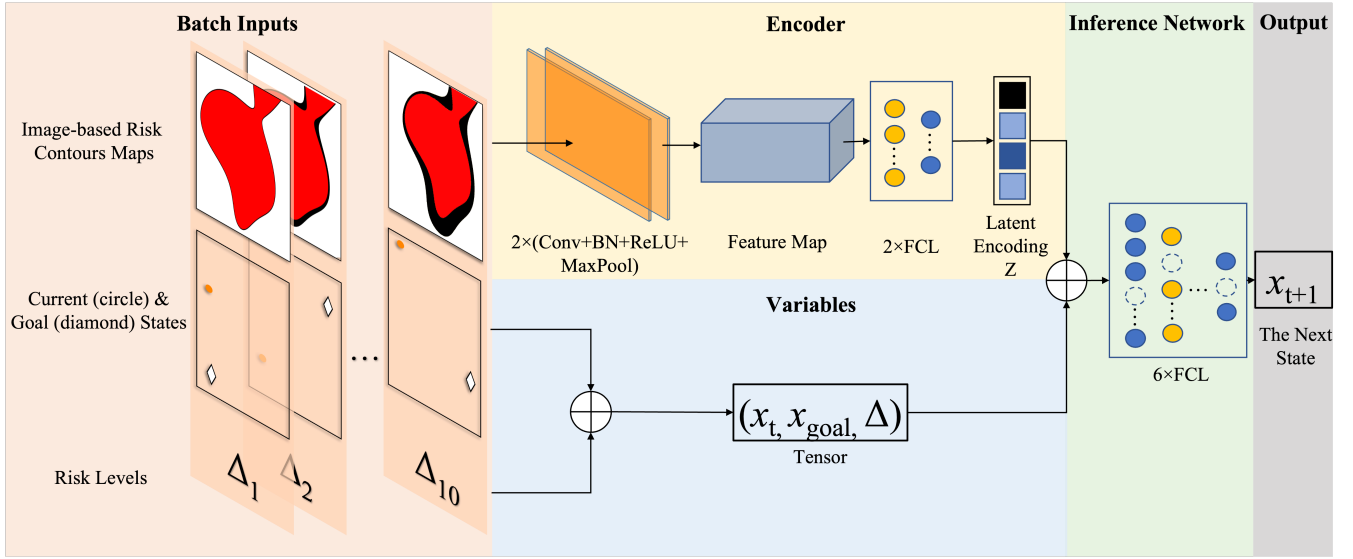


Fig. 4: The overall structure of the neural network sampler for informed sampling.

We use a mean-squared-error (MSE) loss between the predicted state \hat{x}_{t+1} and the label next state x_{t+1} for both the encoder \mathcal{E} and inference network \mathcal{I} during training, i.e.,

$$l(\theta_e, \theta_i) = \frac{1}{N_a} \sum_p \sum_{q=0}^{L_p-1} \|\hat{x}_{p,q+1} - x_{p,q+1}\|^2, \quad (14)$$

where N_a is a positive averaging constant, \hat{N} is the total number of training paths in an environment, and L_p is the length of p -th path, respectively.

Therefore, the NNS is recursively called to make the newly generated state \hat{x}_{t+1} become the robot's current state x_t in the next time step, leading the searching tree to expand incrementally.

C. Online Path Planning

This section introduces the key components of NR-RRT, a bidirectional neural risk-aware path planning algorithm. The procedures of NR-RRT are summarized in Algorithm 1.

Now, the neural network sampler in Section IV-B can prompt the tree to expand efficiently. The algorithm still needs a *Risk Assessor* (RA) to verify that the risk of collision is bounded during the expansion. RA is used to verify a vertex or edge whether lies in the safe zone, and it returns True if so or False if not. Given a vertex, we substitute its states and predefined risk level Δ into the inner approximation of risk contour $\hat{\mathcal{C}}_r^\Delta$. As we did to decide which area is the safe zone, the vertex is risk bounded if the inequalities in (6) hold. On the other hand, an edge between two vertices is a straight line segment and can be rewritten as a linear polynomial trajectory such as $\{(x, y), x = a_1 t + b_1, y = a_2 t + b_2\}$, given the two endpoint states. The sum of squares (SOS) form as (10) will be verified for the edge using the Spotless package [41]. As a result, the collision probability along this edge is smaller than or equal to Δ if it satisfies the SOS condition. Moreover, we use a hybrid replanning strategy [15] to speed up the algorithm further and guarantee its completeness.

In principle, given the original planning problem, NR-RRT performs bidirectional neural planning at first (described later) and outputs a coarse global path. Then, if any edge between two contiguous vertices lies in the risk or dangerous zone, we call *Replan* to replan this edge. This function takes the non-connectable nodes as a start and goal pair and utilizes the bidirectional neural planning again to recursively generate a local path for them. If there is no solution available yet, in case some problems are hard, we employ the standby planner that exhibits probability completeness to find it. Since the primary solution is usually not optimal, we implement the shortcutting process, *Lazy States Contraction* (LSC) [42], to eliminate the useless vertices in it. With *Replan* and LSC, the algorithm keeps finding possible critical connectable vertices and removing the unconnectable nodes.

As shown in Algorithm 1, NR-RRT takes the start and goal states, x_{init} and x_{goal} , uncertain obstacles $\mathcal{X}_{\text{obs}_i}$, and user-specified risk level Δ as inputs. The encoder \mathcal{E} encodes the image-based risk contours map I into a latent feature map Z (Line 1). We initialize two trees, i.e., $\mathcal{T}_f = (V_f, E_f)$ to grow forwards from start to goal and $\mathcal{T}_b = (V_b, E_b)$ to grow backwards from goal to start, respectively, and an empty path solution π (Line 2). Each tree comprises a vertex set $V \subset \mathcal{X}_{\text{safe}}$ and an edge set $E \subseteq V \times V$. We expand trees in an alternating manner and repeat this manner for a fixed number of iterations N . We take the expansion step for forward path π_f (Lines 5-14) as an example here. Our method uses NNS to generate a sample x_{sample} , given the latent encoding Z , and the top nodes, x_f^{end} and x_b^{end} . Then, x_{sample} is verified through RA, and if the collision risk is bounded, it will be adjusted to x_{new} and added to the vertex set V_f . It and its nearest vertex in the tree x_{nearest} are inputted into *Steer* (Line 9). In *Steer*, we connect two vertices with a straight line and perform RA to assess the collision risk for their edge. If risk bounded, it is added to the edge set E_f . After each expansion step, the algorithm tries to connect both trees by calling *Steer* again (Line 11). Once the connection is successful, we obtain a

coarse tree \mathcal{T} containing the vertices from \mathcal{T}_f and \mathcal{T}_b , and have a primary path solution $\pi = \{x_{\text{init}}, \dots, x_{\text{goal}}\}$ by querying the tree \mathcal{T} . If the connection fails, we swap \mathcal{T}_f with \mathcal{T}_b and keep expanding (Line 15). If a solution π is found, we refine it with *LCS*. The *LCS* function will return critical vertices regarding the optimal path. *RA* is then utilized to check the collision risk of every edge in the global path. Suppose there is a local path crossing either the dangerous or risk zone. In that case, we input the failed global path into *Replan*, followed by *LSC*, to generate a near-optimal solution until the number of iterations reaches the threshold N_j (Lines 20-24). In case the strategy using bidirectional neural planning fails after N_j attempts, the algorithm employs the standby planner, RRT-SOS, which samples from a uniform sampler. By adjusting the non-connectable nodes as new start and goal states, RRT-SOS attempts to find a path solution while maintaining the theoretical probabilistic completeness (Lines 30-32). Eventually, the resulting path is optimized through *LSC* and the collision risk of it is verified through *RA* again (Lines 35-36).

V. SIMULATION EXPERIMENTS

In this section, we conduct several simulation experiments to demonstrate NR-RRT's distinguishing performance and generality. We describe the collection of data and the implementation details of our neural network sampler in Section V-A. Then, we compare our algorithm with RRT-SOS in uncertain environments with different acceptable risk levels in Section V-B. In Section V-C, we further demonstrate that NR-RRT outperforms RRT-SOS in seen and unseen cluttered uncertain environments.

A. Implementation Details

All experiments are conducted on the same system with $3.60 \text{ GHz} \times 8$ Intel Core i9-9900KF processor, 64GB RAM. Neural network models were trained with the PyTorch Python API on NVIDIA RTX 2080Ti. The Adam optimizer [43] is utilized with default parameters for training. To ensure that our experiment is fair, we load the trained neural network model across the same CPUs as used in RRT-SOS algorithm. The neural network only takes about 40ms to predict the next state, which is a benefit to expanding the searching tree.

RRT-SOS algorithm is utilized to obtain expert demonstration paths. Every expert path comprises several 2D points associated with the cost-to-go values with regard to their next points. The 90% of data is used for training, and the others become the validation dataset. Besides, experiment maps are obtained by processing the risk contours map into images with the size of 256×256 pixels. The batch size is 64, and the learning rate is fixed at $\eta = 0.0004$.

B. Path Planning in Uncertain Environments with Different Risk Levels

Although NR-RRT can handle diverse uncertainties, i.e., location, size, and geometry with various probability distributions, by constructing a risk contours map, we need to further

Algorithm 1: NR-RRT ($x_{\text{init}}, x_{\text{goal}}, \mathcal{X}_{\text{obs}_i}, \Delta$)

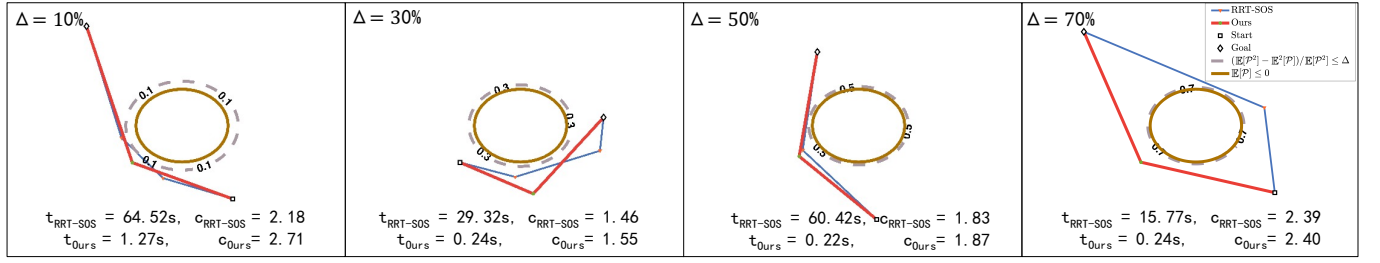
```

1  $\mathcal{E}(I(\mathcal{X}_{\text{obs}_i})) \rightarrow Z$ ;
2  $\mathcal{T}_f = (V_f, E_f), \mathcal{T}_b = (V_b, E_b), \pi \leftarrow \emptyset$ ;
3  $V_f \leftarrow \{x_{\text{init}}\}, V_b \leftarrow \{x_{\text{goal}}\}$ ;
4 for  $i = 0, \dots, N$  do
5    $x_{\text{sample}} \leftarrow NNS(Z, V_f^{\text{end}}, V_b^{\text{end}})$ ;
6   while  $RA(x_{\text{sample}}, \Delta) = \text{False}$  do
7      $x_{\text{sample}} \leftarrow NNS(Z, V_f^{\text{end}}, V_b^{\text{end}})$ ;
8    $V_f \leftarrow V_f \cup x_{\text{new}}$ ;
9   if  $Steer(x_{\text{nearest}}, x_{\text{new}})$  then
10      $E_f \leftarrow E_f \cup (x_{\text{nearest}}, x_{\text{new}})$ ;
11   if  $Steer(V_f^{\text{end}}, V_b^{\text{end}})$  then
12      $\mathcal{T} \leftarrow \text{concatenate}(\mathcal{T}_f, \mathcal{T}_b)$ ;
13      $\pi = \{x_{\text{init}}, \dots, x_{\text{goal}}\}$ ;
14     Break;
15   SWAP( $\mathcal{T}_f, \mathcal{T}_b$ );
16  $\pi \leftarrow LCS(\pi)$ ;
17 if  $RA(\pi, \Delta)$  then
18   return  $\pi$ ;
19 else
20   for  $j = 0, \dots, N_j$  do
21      $\pi \leftarrow Replan(\pi, Z)$ ;
22      $\pi \leftarrow LCS(\pi)$ ;
23     if  $RA(\pi, \Delta)$  then
24       return  $\pi$ ;
25    $\pi_{\text{new}} \leftarrow \emptyset$ ;
26   for  $i = 0, \dots, size(\pi) - 1$  do
27     if  $Steer(\pi_i, \pi_{i+1})$  then
28        $\pi_{\text{new}} \leftarrow \pi_{\text{new}} \cup \{\pi_i, \pi_{i+1}\}$ 
29   else
30      $\pi_{\text{local}} \leftarrow RRT-SOS(\pi_i, \pi_{i+1}, \mathcal{X}_{\text{obs}_i})$ ;
31     if  $\pi_{\text{local}}$  is not None then
32        $\pi_{\text{new}} \leftarrow \pi_{\text{local}} \cup \pi_{\text{new}}$ ;
33   else
34      $\pi_{\text{new}} = \emptyset$ 
35    $\pi \leftarrow LCS(\pi_{\text{new}})$ ;
36   if  $RA(\pi, \Delta)$  then
37     return  $\pi$ ;
38 return  $\pi = \emptyset$ ;

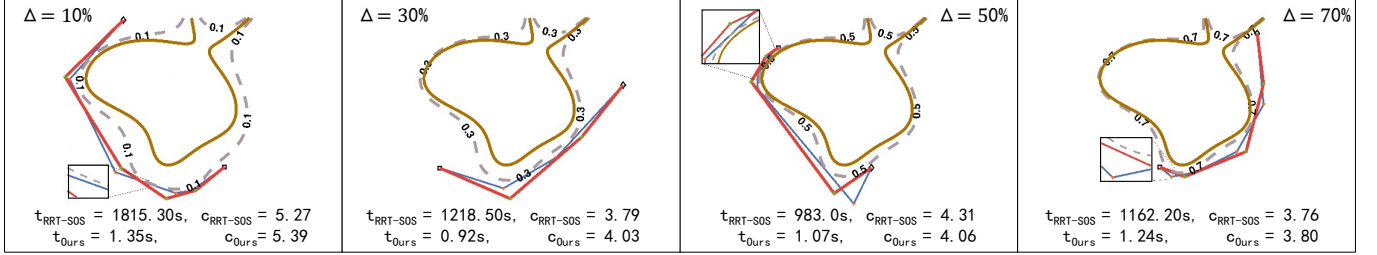
```

prove its generality to different risk levels. Thus, the objective of the following experiments is to find a near-optimal risk bounded path between a start and goal states in an uncertain environment with different risk levels Δ .

We create two uncertain environments; one is convex and the other is nonconvex. The circle-shaped convex obstacle shown in Fig. 5(a) is denoted as $\mathcal{X}_{\text{obs}_1}(\omega) = \{(x, y) : \omega^2 - x^2 - y^2 \geq 0\}$, where $(x, y) \in [-1, 1] \times [-1, 1]$, and the radius ω has a uniform probability distribution $\omega \sim \mathcal{U}(0.3, 0.4)$. Note that this setup is the same as the illustrative example 3 in



(a) Planning Results in Uncertain Convex Environments with Different Risk Levels



(b) Planning Results in Uncertain Nonconvex Environments with Different Risk Levels

Fig. 5: Figs. (a) and (b) are the comparisons of the planning results in uncertain convex and nonconvex environments with different risk levels, respectively. Given a risk-aware convex or nonconvex planning problem with some risk levels Δ , NR-RRT (red) and RRT-SOS (blue) can find risk bounded low-cost path solutions that entirely do not lie in the dangerous and risk zone (brown and gray areas). NR-RRT uses extremely less time (t) than RRT-SOS takes to plan paths with comparable cost (c).

[6] for comparison. $\mathcal{X}_{\text{obs}_2}(\omega) = \{(x, y) : -0.35x^5 - x^4y - 0.5x^4 + 0.2x^3y^2 - 0.5x^3y + 0.31x^3 - 0.5x^2y^3 + 0.2x^2y^2 + 1.7x^2y + 0.26x^2 + 0.7xy^4 - 0.1xy^3 - 1.5xy^2 - 0.1xy + 0.1x + 0.02y^5 - 0.1y^4 - 0.04y^3 - 0.1y^2 + 0.28y + 0.89 - 0.7\omega\}$ represents the heart-shaped nonconvex obstacle with uncertain parameter $\omega \sim \text{Beta}(9, 0.5)$, as shown in Fig. 5(b), where $(x, y) \in [-2, 2] \times [-2, 2]$. We set ten risk levels $\Delta = \{10\%, 20\%, \dots, 100\%\}$ for each environment. After choosing the risk level one by one for each environment, we randomly generate 1000 different valid start and goal pairs $(\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ for training, and randomly sample 200 pairs for testing. Hence, we have 10000 training data and 2000 test data for every environment.

The run-time to find the near-optimal solutions and their lengths are compared between two methods: RRT-SOS [6] and NR-RRT. Table I reports the total mean planning times and path lengths with standard deviations when the near-optimal trajectories are founded in uncertain environments with ten risk levels. We can see that our algorithm can generate near-optimal solutions with significantly less computation time than using RRT-SOS whether in the uncertain convex or nonconvex environment. The medians of planning times are 27.12s and 0.25s by RRT-SOS and NR-RRT in the convex environment, respectively, while 1667.85s by RRT-SOS and 0.93s by NR-RRT in the nonconvex environment. NR-RRT's smaller standard deviations and medians of the computation time imply it has a better stability. Note that RA takes about 0.09 seconds to verify the safety of every edge, which demonstrates that NR-RRT can be used in real-world applications. Our method sometimes generates slightly shorter paths than the demonstrations generated by RRT-SOS because the expert paths are not always optimal, which will be discussed in

Section VI. Therefore, no matter the risk level specified for an encountered uncertain obstacle, NR-RRT achieves a better performance than RRT-SOS by a large margin, thanks to the informative state sampling and bidirectional search strategy.

C. Path Planning in Uncertain Cluttered Nonconvex Environments

In this section, we evaluate the performance of NR-RRT in seen and unseen uncertain cluttered nonconvex environments. Twelve environments are created by random placement of seven uncertain obstacles. Five are convex, i.e., three are circle-shaped and two are ellipse-shaped, and two are nonconvex and calabash-shaped. The acceptable chance of collisions is set as $\Delta = 10\%$. We select ten of them as the seen environments for training and the remaining two as unseen scenarios only for the testing. The range of the states is $[x, y] \in [-5, 5] \times [-5, 5]$. We randomly sample 1000 different valid start and goal pairs in each seen environment. As a result, we have 10000 training data belonging to ten seen environments. On the other hand, we randomly create 200 start and goal pairs in every seen environment and 1000 pairs in every unseen environment. Consequently, there are a total of 4000 data for the testing. Half of them are different pairs in the seen environments, while the others are new pairs in the unseen environments. Figs. 6 (a) and (b) show the planned trajectories in one seen and one unseen environment, respectively.

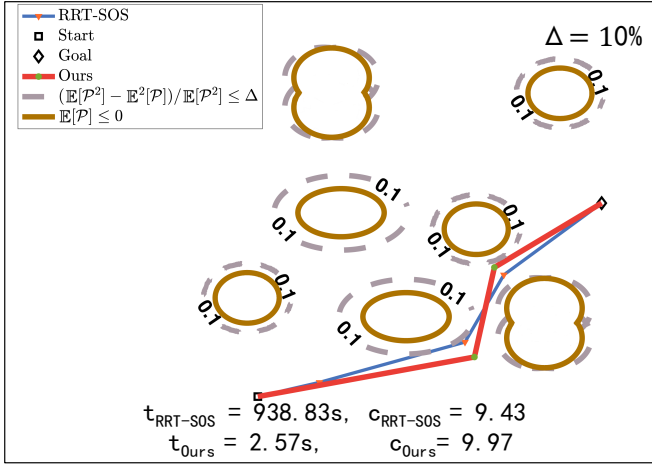
Table II presents the comparisons of the total mean planning times and path lengths with standard deviations between NR-RRT and RRT-SOS [6] when the near-optimal trajectories are founded in the ten seen and two unseen cluttered environments. It can be seen that compared to RRT-SOS, NR-RRT

TABLE I: Comparisons of the Total Mean Planning Times and Path Lengths with Standard Deviations in Uncertain Convex and Nonconvex Environments With Ten Risk Levels

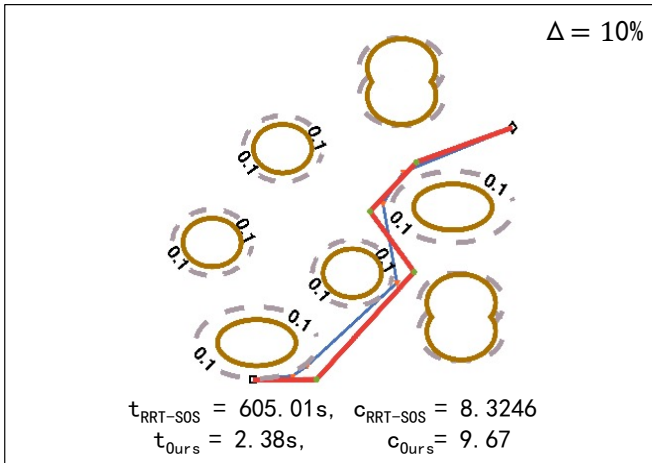
	Convex Env.		Nonconvex Env.	
	Time(s)	Length	Time(s)	Length
Ours	0.26 ± 0.11	2.17 ± 0.88	2.15 ± 2.58	4.65 ± 1.44
[6]	26.08 ± 42.40	1.60 ± 0.78	1614.60 ± 714.90	4.22 ± 1.23

TABLE II: Comparisons of the Total Mean Planning Times and Path Lengths with Standard Deviations in Seen and Unseen Uncertain Cluttered Nonconvex Environments

	Seen Cluttered Env.		Unseen Cluttered Env.	
	Time(s)	Length	Time(s)	Length
Ours	2.69 ± 0.28	7.40 ± 2.40	2.01 ± 0.16	6.82 ± 2.51
[6]	313.48 ± 669.53	6.49 ± 2.26	237.90 ± 518.50	6.31 ± 2.18



(a) Path solutions in a Seen Cluttered Scenario



(b) Path solutions in an Unseen Cluttered Scenario

Fig. 6: Comparisons of planning times (**t**) and path qualities (**c**) in seen (a) and unseen (b) uncertain cluttered nonconvex scenarios between NR-RRT (red) and RRT-SOS (blue) [6].

takes a much shorter amount of time to look for similar quality trajectories in the seen and unseen environments. Furthermore, Fig. 7 displays the box-plot of the planning times and path lengths that results from NR-RRT (for the sake of the large order of magnitudes, the box-plot for RRT-SOS is omitted). The green triangle of the boxes refers to the average value, the orange line in the middle of the boxes is the median, and the height of the boxes refers to variance. We can tell that NR-RRT achieves high performance on the two indexes from this figure. The effectiveness of NR-RRT is also obvious on unseen tasks. Besides, in the given finite time interval, RRT-SOS and NR-RRT success rates are 100%. The small variance of computation time indicates the robustness of NR-RRT in finding a near-optimal path in a short time.

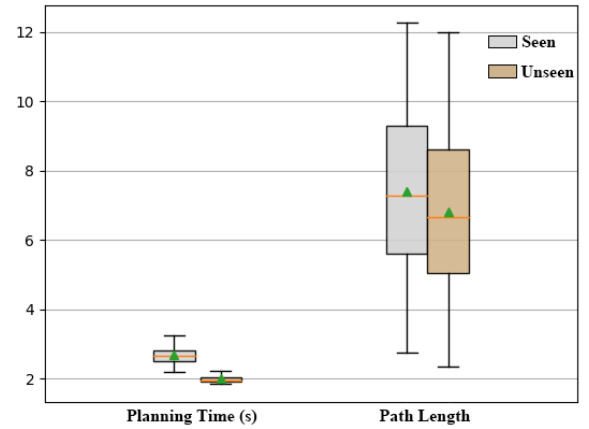


Fig. 7: The interquartile ranges of planning times and path solution lengths for NR-RRT in seen and unseen uncertain cluttered nonconvex environments.

VI. DISCUSSION

In this section, NR-RRT's probabilistic completeness and optimality are analyzed.

1) *Probabilistic Completeness*: We first discuss the probabilistic completeness of NR-RRT, proposed as below:

Proposition 1. *Given a risk-aware path planning problem $\{\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}}, \mathcal{X}_{\text{obs}}, \Delta\}$, and a risk assessor, NR-RRT can find a solution $\pi: [0, T]$, if one exists, such that $\pi_0 = \mathbf{x}_{\text{init}}$, $\pi_T \in \mathcal{X}_{\text{target}}$, and $\text{Prob}(\mathbf{x}(t) \in \mathcal{X}_{\text{obs}_i}(\omega_i)) \leq \Delta|_{i=1}^{n_o}$.*

Proposition 1 indicates that NR-RRT will find a risk bounded path, if one exists, where the probability of collision is no greater than Δ . Note that RRT-SOS is used as our standby SBMP that has *probabilistic completeness*. Based on the assumptions and Lemma as follows, we propose Theorem 1, which proves that the worst-case completeness guarantees of NR-RRT. Therefore, the algorithm has probabilistic completeness as its standby planner RRT-SOS.

Assumption 1. *The given start and goal states $(\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ are in safe space, i.e., $\mathbf{x}_{\text{init}} \in \mathcal{X}_{\text{safe}}$ and $\mathbf{x}_{\text{goal}} \in \mathcal{X}_{\text{target}} \subset \mathcal{X}_{\text{safe}}$. There exists at least one risk bounded path solution π that contains \mathbf{x}_{init} and \mathbf{x}_{goal} , and $\pi \not\subset \mathcal{X}_{\text{obs}}$ by using a planner to find a risk bounded solution.*

Lemma 1. [15] *In a certain environment, if the start and goal pairs $(\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ is feasible, and there exists a collision-free path that connects them, for a planning problem $\{\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}}, \mathcal{X}_{\text{obs}}\}$, the probability of finding a solution approaches one as the underlying RRT* will be allowed to perform until infinity if needed for a iterative and recursive learning-based path planner.*

Theorem 1. *If Assumption 1 holds, for a risk-aware path planning problem $\{\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}}, \mathcal{X}_{\text{obs}}, \Delta\}$, and a standby path planner whose sampler is with uniform distribution, NR-RRT will always find a risk bounded solution, if one exists. Because the planner that guarantees completeness is employed as a standby planner to run until infinity if NR-RRT fails to conclude a solution by using the neural network sampler for a fixed number of trials.*

Sketch of Proof: Lemma 1 indicates that a neural planning method exhibits probabilistic completeness, which is with an underlying oracle planner, e.g., RRT* that guarantees completeness if one solution and a competent collision checker exist in an environment of certainty. Similarly, assumption 1 states a condition that the risk-aware path planning problem is resolvable in our case. Namely, at least one solution can be found by a planner that guarantees probabilistic completeness. NR-RRT first obtains a coarse solution π that might contain non-connectable consecutive vertices. Then, the algorithm tries to connect them via replanning. These vertices lie in a safe space since each of them is evaluated by a risk assessor RA before adding it to the vertex set of the solution π . During hybrid replanning, we aim to refine the coarse solution π by performing neural replanning. Suppose there still exists any vertex that cannot be connected to conclude a final solution. In that case, we use RRT-SOS to connect the remaining non-connectable vertices afterward. The non-connectable vertices and the uncertain obstacles form a new risk-aware path planning problem and can be solved by RRT-SOS because of its probabilistic completeness. Therefore, if Assumption 1 holds for the non-connectable vertices, NR-RRT guarantees the convergence of a solution inherited from the standby planner RRT-SOS. NR-RRT with a standby planner RRT-SOS has probabilistic completeness in an uncertain environment.

2) *Optimality:* RRT-SOS is employed to generate expert demonstrations. It also performs in the hybrid replanning and as a baseline against our NR-RRT. However, the resulting paths using RRT-SOS are often non-optimal because the graph constructed by PRM only has a limited number of nodes in a map, and querying such a graph results in a non-global shortest solution. In other words, RRT-SOS does not guarantee asymptotic optimality. Since our resulting paths' quality cannot be the same as the training data, there is a situation where NR-RRT can find solutions with lower costs than the expert but non-optimal solutions. Note that the ability to learn informed sampling distribution is more important than learning the exact positions of the labeled data for us.

In contrast, the number of samples in RRT* approaches infinity to obtain a graph covering the whole map area. For RRT*, the probability of finding the optimal solution, if one exists, approaches one as the number of iterations increases to infinity. It gets asymptotic optimality from *ChooseParent* and *Rewire* processes that incrementally update the tree connections such that the shortest path is ensured. Whenever RRT* attempts to connect $\mathbf{x}_{\text{nearest}}$ from the \mathbf{x}_{new} , it selects the best parent of \mathbf{x}_{new} in terms of the cost-to-come by searching the nodes in a certain radius in the *ChooseParent*. After adding a new edge about \mathbf{x}_{new} to the edge set, RRT* removes tree edges through \mathbf{x}_{new} with relatively higher costs, which is called *Rewire*. For more information, please refer to [7]. Because our algorithm can naturally inherit asymptotic optimality from the standby planner, RRT-SOS, if it has, as proved in [15], it is doable to modify the original RRT-SOS with these two processes to generate the optimal path. However, there is no need to do this since the ability to find the optimal solution is not our contribution. The resulting solutions found by NR-RRT have been near-optimal with little computational costs, which is more important in practice. On the contrary, those modifications will significantly increase the computational time but offer inconspicuous improvement for the path quality, resulting in a low input-output ratio.

VII. CONCLUSIONS AND FUTURE WORKS

This article provides a learning-based bidirectional risk-aware path planning algorithm, NR-RRT, for quickly finding a risk bounded near-optimal solution in seen and unseen uncertain nonconvex environments where the obstacles may have probabilistic locations, sizes, and geometries. The information of probabilistic nonconvex obstacles can be captured by constructing the risk contours map and further embedded in a deterministic latent feature map. A proposed neural network sampler predicts the most-promising safe vertices after learning the past experiences. An informed bidirectional-search sampling strategy accelerates the convergence to a solution. Our experiments show that our method can find comparable near-optimal solutions to the baseline but takes significantly less planning time in uncertain nonconvex environments.

In our future research directions, we will consider kinodynamic constraints (kinematics and differential constraints) [44] to make the planned trajectories easier to be tracked. Besides, it is interesting to investigate path planning for robots under motion uncertainties [10].

REFERENCES

- [1] F. S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, and N. Hawes, "Risk-aware motion planning in partially known environments," *arXiv preprint arXiv:2109.11287*, 2021.
- [2] M. da Silva Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, "Collision-free encoding for chance-constrained nonconvex path planning," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, 2019.
- [3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [4] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.
- [5] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
- [6] A. Jasour, W. Han, and B. Williams, "Convex risk bounded continuous-time trajectory planning in uncertain nonconvex environments," in *Robotics: Science and Systems*, 2021.
- [7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [8] B. Axelrod, L. P. Kaelbling, and T. Lozano-Pérez, "Provably safe robot navigation with obstacle uncertainty," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1760–1774, 2018.
- [9] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference*, 2010, p. 8160.
- [10] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.
- [11] Q. H. Ho, Z. N. Sunberg, and M. Lahijanian, "Gaussian belief trees for chance constrained asymptotically optimal motion planning," *arXiv preprint arXiv:2202.12407*, 2022.
- [12] A. M. Jasour and B. C. Williams, "Risk contours map for risk bounded motion planning under perception uncertainties," in *Robotics: Science and Systems*, 2019.
- [13] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.
- [14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [15] 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 Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.
- [16] L. Janson, E. Schmerling, and M. Pavone, "Monte carlo motion planning for robot trajectory optimization under uncertainty," in *Robotics Research*. Springer, 2018, pp. 343–361.
- [17] M. Cannon, "Chance-constrained optimization with tight confidence bounds," *arXiv preprint arXiv:1711.03747*, 2017.
- [18] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.
- [19] J. J. Johnson and M. C. Yip, "Chance-constrained motion planning using modeled distance-to-collision functions," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1582–1589.
- [20] W. Sun, J. van den Berg, and R. Alterovitz, "Stochastic extended lqr for optimization-based motion planning under uncertainty," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 437–447, 2016.
- [21] W. Sun, S. Patil, and R. Alterovitz, "High-frequency replanning under uncertainty using parallel sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.
- [22] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.
- [23] L. Jaillet, J. Hoffman, J. Van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg, "Eg-rrt: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2646–2652.
- [24] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty," in *Robotics: Science and systems*, 2007.
- [25] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.
- [26] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, "Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 558–573, 2021.
- [27] Y. Lee, P. Cai, and D. Hsu, "Magic: Learning macro-actions for online pomdp planning," *arXiv preprint arXiv:2011.03813*, 2020.
- [28] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 3. IEEE, 2002, pp. 1936–1941.
- [29] T. Summers, "Distributionally robust sampling-based motion planning under uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6518–6523.
- [30] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 302–314, 2021. [Online]. Available: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/csy2.12020>
- [31] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks: Learning generalizable representations for visuomotor control," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4732–4741.
- [32] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *Advances in neural information processing systems*, vol. 29, 2016.
- [33] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.
- [34] T. Fan, P. Long, W. Liu, J. Pan, R. Yang, and D. Manocha, "Learning resilient behaviors for navigation under uncertainty," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5299–5305.
- [35] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.
- [36] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.
- [37] H. Ma, C. Li, J. Liu, J. Wang, and M. Q. Meng, "Enhance connectivity of promising regions for sampling-based path planning," *CoRR*, vol. abs/2112.08106, 2021. [Online]. Available: <https://arxiv.org/abs/2112.08106>
- [38] T. Zhang, J. Wang, and M. Q.-H. Meng, "Generative adversarial network based heuristics for sampling-based path planning," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 1, pp. 64–74, 2021.
- [39] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [41] M. M. Tobenkin, F. Permenter, and A. Megretski, "Spotless polynomial and conic optimization," *View online*, 2013.
- [42] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 2493–2498.
- [43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [44] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.



Fei Meng received the B.Eng. in electrical engineering and automation, and the M.Eng. in control engineering from Harbin Institute of Technology, Weihai and Harbin, China, in 2016 and 2019, respectively. He is working toward the Ph.D. degree with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China.

His research interests include learning-based motion planning and control.



Liangliang Chen received the B.B.A. degree in business administration in 2017, the B.S. degree in automation in 2017, and M.Eng. degree in control engineering in 2019, all from Harbin Institute of Technology, Harbin, China. He is currently pursuing the Ph.D. degree in electrical and computer engineering at Georgia Institute of Technology, Atlanta, GA, USA.

His current research interests include deep reinforcement learning.



Han Ma received the B.E. degree in measurement, control technology and instrument from the Department of Precision Instrument of Tsinghua University, Beijing, China, in 2019. He is now working towards the Ph.D. degree in the Department of Electronic Engineering of The Chinese University of Hong Kong, Hong Kong SAR, China.

His research interests include path planning and machine learning in robotics.



Jiankun Wang received the B.E. degree in Automation from Shandong University, Jinan, China, in 2015, and the Ph.D. degree in Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2019. He is currently a Research Assistant Professor with the Department of Electronic and Electrical Engineering of the Southern University of Science and Technology, Shenzhen, China.

During his Ph.D. degree, he spent six months at Stanford University, CA, USA, as a Visiting Student

Scholar supervised by Prof. Oussama Khatib. His current research interests include motion planning and control, human robot interaction, and machine learning in robotics.



Max Q.-H. Meng received his Ph.D. degree in Electrical and Computer Engineering from the University of Victoria, Canada, in 1992. He is currently a Chair Professor and the Head of the Department of Electronic and Electrical Engineering at the Southern University of Science and Technology in Shenzhen, China, on leave from the Department of Electronic Engineering at the Chinese University of Hong Kong. He joined the Chinese University of Hong Kong in 2001 as a Professor and later the Chairman of Department of Electronic Engineering. He was

with the Department of Electrical and Computer Engineering at the University of Alberta in Canada, where he served as the Director of the ART (Advanced Robotics and Teleoperation) Lab and held the positions of Assistant Professor (1994), Associate Professor (1998), and Professor (2000), respectively. He is an Honorary Chair Professor at Harbin Institute of Technology and Zhejiang University, and also the Honorary Dean of the School of Control Science and Engineering at Shandong University, in China. His research interests include medical and service robotics, robotics perception and intelligence. He has published more than 750 journal and conference papers and book chapters and led more than 60 funded research projects to completion as Principal Investigator. Prof. Meng has been serving as the Editor-in-Chief and editorial board of a number of international journals, including the Editor-in-Chief of the Elsevier Journal of Biomimetic Intelligence and Robotics, and as the General Chair or Program Chair of many international conferences, including the General Chair of IROS 2005 and ICRA 2021, respectively. He served as an Associate VP for Conferences of the IEEE Robotics and Automation Society (2004-2007), Co-Chair of the Fellow Evaluation Committee and an elected member of the AdCom of IEEE RAS for two terms. He is a recipient of the IEEE Millennium Medal, a Fellow of IEEE, a Fellow of Hong Kong Institution of Engineers, and an Academician of the Canadian Academy of Engineering.