IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2024, Yokohama, Japan.
Cite as T-ASE paper.

Learning-based Risk-Bounded Path Planning Under Environmental Uncertainty

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

***Abstract*—Building a general and efficient path planning framework in uncertain nonconvex environments is challenging due to the safety constraints and complex configuration. Traditional avenues usually involve convexifying obstacles and presume Gaussian distribution, which are not universal. Meanwhile, the fast convergence of high-quality solutions is not guaranteed. Therefore, we develop a novel neural risk-bounded path planner to quickly find near-optimal solutions that have an acceptable collision probability in the complex environments. Firstly, we retrieve the nonconvex obstacles with arbitrary probabilistic uncertainties in the form of a deterministic point cloud map. A neural network sampler encodes it into a latent embedding and is trained with sufficient expert demonstrations, predicting states in the potential subspace. We construct a neural cost estimator to select the best informed state from those samples. Then, we recursively use the simple yet effective neural networks to march toward the start and goal bidirectionally. The collision risk of the intermediate connections is verified based on sum-of-squares optimization. Simulation results show that our approach significantly saves time and resources in finding comparable solutions over the state-of-the-art methods in the seen and unseen challenging environments.**

***Note to Practitioners*—More and more robots are deployed in unstructured environments, such as forests and subterranean caves. However, uncertainty in the environment situational awareness usually causes accidents. To quickly generate safe paths without over-conservation in uncertain complex environments, we propose a neural risk-bounded sampling-based path planner. Conventional methods consume lots of computation time and resources to generate satisfactory results. Our learning-based risk-bounded path planning framework can efficiently find paths with a guaranteed risk tolerance avoiding uncertain nonconvex static obstacles. It imitates the expert to generate informed states in a subspace that potentially contains the optimal solution. In practice, we need to formulate the observed uncertain obstacle at a grid map into the polynomial containing random variables and determine their probability distributions.**

***Index Terms*—Sampling-based algorithm, deep learning methods, risk-bounded path planning.**

This work was supported in part by Hong Kong RGC CRF grant C4063-18G, Shenzhen Outstanding Scientific and Technological Innovation Talents Training Project under Grant RCBS20221008093305007, and National Natural Science Foundation of China grant #62103181. *(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).

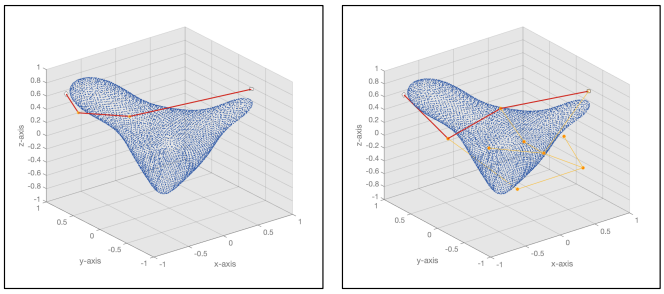
J. Wang is with Shenzhen Key Laboratory of Robotics Perception and Intelligence and the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China, and also with Jiaxing Research Institute, Southern University of Science and Technology, Jiaxing, 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 at Southern University of Science and Technology in Shenzhen, China. He is a Professor Emeritus in the Department of Electronic Engineering at The Chinese University of Hong Kong in Hong Kong and was a Professor in the Department of Electrical and Computer Engineering at the University of Alberta in Canada (e-mail: max.meng@ieee.org).

I. Introduction

AUTONOMOUS robots have been increasingly deployed in unstructured and uncertain environments such as subterranean [1], planetary [2], and human-robot coexisting [3] scenarios over the past few years. Such environments are highly risky and challenging, forcing the robot to plan trajectories with safety guarantees. Even if stressing the importance to safety is necessary, it is also essential to make the robots not operate too conservatively. Taking uncertainty into consideration by a probabilistic framework has been proven to overcome the inherent disadvantages of set-bounded uncertainty models such as over-conservatism and inefficiency [4]. In this framework, the probability of colliding with the uncertain obstacles cannot exceed a predefined value [5], which maintains feasibility in uncertain complex environments.

Most methods to tackle planning under uncertainty can be basically categorized into optimization-based [6] and sampling-based planning (SBP) [7] approaches. The optimization-based methods, e.g., convex optimization formulated as semidefinite programs (SDP) to estimate the probability of constraint violation [8], are often limited to small-size state-spaces. The SBP algorithms, such as probabilistic roadmap (PRM) [9] and rapidly-exploring random tree (RRT) [10], are popular because they provide probabilistic completeness in high-dimensional C-spaces. Specifically, they always find a solution, if one exists, as the number of iterations goes to infinity. Their advanced variants also guarantee asymptotic optimality, namely the cost of the generated path decreases gradually to the optimum as the number of iterations goes to infinity [11]. Compared with PRM, the RRT-based algorithm does not need a prebuilt roadmap of the entire space, which makes it widely used in the real world. RRT-based algorithms can quickly find an initial solution; however, they cannot guarantee the high quality of the solution due to their random exploration. Moreover, the generated path fails to quickly converge to the optimum through the advanced variant because of the randomness as well. Informed trees were proposed as heuristically biased solvers to overcome these drawbacks [12]. However, they are sensitive to environment complexity. More recently, deep neural network (DNN) based SBP methods,



(a) 3D NR-RRT (b) RRT-SOS

Fig. 1: Our 3D NR-RRT algorithm can efficiently compute a comparable solution (red) in an uncertain nonconvex 3D environment compared with RRT-SOS [15]. Our method draw fewer but critical samples (yellow), drastically reducing the computational expense of finding high-quality solutions.

such as MPNet [13] and 3D Neural RRT* [14], have gained great improvements in finding collision-free paths in 3D fully known environments.

Dealing with the environmental uncertainty in planning is complicated. Axelrod *et al.* [16] developed a theory of shadows for uncertain obstacles with a Gaussian distribution and proposed a variant of the RRT method to find provably safe paths. Although many SBP approaches have shown their performance in uncertain environments while considering stochastic dynamical constraints, they usually assume either convex obstacles [17] or Gaussian uncertainties [18], [19]. Chance constraints in RRT are guaranteed under the worst-case probability of violating constraints in the moment-based ambiguity set [20]. Jasour *et al.* also utilize the moment-based techniques to describe probabilistic obstacles that follow any probability distribution on a deterministic information map [21]. The authors then proposed RRT-SOS [15], which grows the search tree on this transformed map while verifying collision risk by the sum of squares (SOS) methods. They identify the upper bound of the probability of constraint violation by using *Cantelli's inequality* defined for scalar random variables. RRT-SOS guarantees that the probability of collision for every edge of the solution is below a given risk tolerance; however, its computational performance is unsatisfactory, which is an inherent drawback of RRT. The authors recently solved the similar planning problem with additional stochastic dynamical constraint [22]. Nevertheless, there is still room to promote the planning efficiency. In this article, we develop a simple yet effective DNN-aided risk-bounded path planner in uncertain nonconvex environments, which extends our previous work neural risk-aware RRT (NR-RRT) [23]. For a given risk-bounded path planning problem, we use the moment-based techniques and transform the uncertain environment into a point cloud-based risk contours map for training. The obstacles with arbitrary probabilistic locations, sizes, and geometries are all retrieved in the form of the deterministic map obeying a given risk tolerance. After having experience in similar uncertain environments, our learning-based sampler can iteratively generate the promising states. At each iteration, we select the best state based on the costs to goal predicted by a neural cost estimator. Our method integrates the informed sampler, neural cost estimator, and an efficient risk assessor into a bidirectional SBP manner to avoid exhaustive search and rapidly lay out a risk-bounded near-optimal solution.

The remainder of this article is structured as follows. Section II reviews the literature related to path planning under environmental uncertainty. In Section III, we define the risk-bounded path planning problem and introduce some preliminaries. We present the details of our learning-based risk-bounded path planner in Section IV. Section V shows the experiments that compare our method with the state-of-the-art. In Section VI, we draw some conclusions and discuss the limitation of our method and the future directions.

II. Related Work

Some existing planning methods address robot motion [24]–[27] and localization [28] uncertainties as well as disturbances [29]. The environmental uncertainty is also nonnegligible because the imperfect sensing capabilities often make it hard for a robot to capture the exact or complete information about its surroundings [30]. The robot tries to find paths whose collision probabilities do not exceed a given threshold, which is formulated as a chance-constrained path planning problem [5]. Safe-RRT [16] was proposed for path planning under environmental uncertainty. It gains the computational advantages of SBP methods and is robust via verifying safety w.r.t. the geometric equivalent of a confidence interval around the uncertain obstacles. One approach to find safe paths is through a Monte Carlo-based motion planner, where the sampler also estimates the likelihood of collision [7]. However, it is difficult to bound the probability of path collision analytically, and drawing lots of uncertainty samples is computationally demanding [31]. An alternative for optimal motion planning with a guarantee of safety is encapsulating the obstacle's space as constraints and then using nonlinear programming to solve the obtained optimization problem. The representation of the obstacles matters in the constraint evaluation. Castillo *et al.* bound every obstacle by a suitable differentiable convex surface, such as a sphere or an ellipsoid, so that a tractable nonlinear problem can be formed [32]. However, formulating the nonlinear chance constraints like this often requires additional processing, such as linearization, which may result in over-conservatism and suboptimal performance. The authors further developed a tight quadratic bound for an obstacle analytically, which benefits from both strategies above [33]. In [30], the authors retrieve the obstacle uncertainties in the form of a probabilistic map and perform tube-based SBP on it to obtain safety-guaranteed trajectories. These methods often require convex obstacles or convexify nonconvex obstacles first. However, the common successive convex approximation for tackling nonconvex collision avoidance constraints might be computationally complex, infeasible, and overly conservative in robotic applications [34]. An emerging class of risk-bounded path planning methods are able to address nonconvex non-Gaussian probabilistic obstacles [15], [22]. They are, however, computationally prohibitive because of abundant sampling nodes or iterations required.

More recently, learning-based methods have seen an explosion of interest in the planning community and have been applied to aid or accelerate the planning process [35]. Works about deep reinforcement learning (RL) usually model the sequential planning under uncertainty as a Markov decision process (MDP) or partially observable MDP [36]. They try to directly learn either policies for safe operation or cost functions to adjust the designed risk tolerance. In [37], several RL agents are firstly trained offline with noisy measurements on the small-scale maps. A global motion planner using PRM then selects the best policy to build safe long-range roadmaps with collision-free edges. Fan *et al.* proposed an uncertainty-aware predictor to account for environmental uncertainties and trained the policy network to enable resilient operation [24]. However, the RL-based methods often require an exhausting tuning procedure and sometimes need to be re-trained when adjusting to a new risk level.

Deep learning has also gained popularity in solving path planning problems owing to the powerful function approximation capability of DNNs [38]. It is widely used to speed up the convergence of SBP by learning biased sampling heuristics [39] or embedding a neural network-based sampler in SBP [13] to guide the sampling process. Ichter *et al.* proposed to build the SBP graph and perform collision checking in the learned embedded spaces [38]. In deterministic 3D spaces, 3D generative adversarial network and 3D convolutional neural network (3D CNN) models are developed to predict the promising probability distribution of the optimal solution [14]. However, 3D CNN is computationally inefficient since lots of redundant voxels inside the obstacles must be considered. Cubic representation is usually constrained by its resolution because of sparse data and the heavy computation burden of 3D convolution [40]. In our previous work [23], we proposed NR-RRT, a fast and near-optimal path planner based on the CNN model for uncertain nonconvex 2D environments. It requires no data parametrization or handcrafted local features. We now take a step to broaden its scope and improve its efficiency to realize the fast risk-bounded SBP in uncertain nonconvex environments.

This article provides the following main contributions:

- A simple yet powerful DNN architecture for informed sampling in uncertain nonconvex environments is proposed. Besides, we develop a general and high performance risk-bounded geometric path planning framework to quickly find the near-optimal solutions satisfying the probabilistic safety constraints. It is general because the uncertainties of the probabilistic convex or nonconvex obstacles can be arbitrary. As an advanced version of NR-RRT that effectively generates risk-bounded 2D paths [23], the algorithm presented here, 3D NR-RRT, additionally build a neural cost estimator to enable more efficient path finding in uncertain nonconvex environments;
- We conduct case studies in uncertain nonconvex environments that include seen and unseen layouts, and comparison experiments with the state-of-the-art SBP and optimization-based algorithms [15] in terms of the computation time, resources, and path qualities.

III. Preliminaries

In Section III-A, we first formulate the problem of risk-bounded path planning in uncertain environments. Then, we introduce an advanced mapping method and the baseline risk-bounded SBP approach in Section III-B.

A. Risk-Bounded Robot Path Planning Under Environmental Uncertainty

The objective of risk-bounded path planning under environmental uncertainty is to find a path that consists of a set of configurations connecting the start and goal while the collision probability of the found path is bounded. Let $\mathcal{E} \subset \mathbb{R}^d$, $d = \{2, 3\}$, be the surrounding environment where the robot operates (also called workspace). $\mathcal{E}_{\text{obs}_i}(\omega_i)$, $i = 1, \ldots, n_o$, represent static uncertain obstacles in the workspace, where $\omega_i \in \mathbb{R}^{n_\omega}$ are probabilistic parameters with provided probability distributions. The obstacles are described as polynomials in coordinate positions $\boldsymbol{p} \in \mathcal{E}$:

$$\mathcal{E}_{\text{obs}_i}(\omega_i) = \{\boldsymbol{p} \in \mathcal{E} : \mathcal{P}_i(\boldsymbol{p}, \omega_i) \geq 0\}, i = 1, \ldots, n_o, \quad (1)$$

where $\mathcal{P}_i : \mathbb{R}^{d+n_\omega} \to \mathbb{R}$ are the given polynomials. Eq. (1) can describe convex and nonconvex probabilistic obstacles with uncertain size, location, or geometry. For instance, a spherical uncertain obstacle is $\mathcal{P} \geq 0 : \{(x, y, z) : \omega^2 - x^2 - y^2 - z^2 \geq 0, \omega \sim \mathcal{U}(1, 2)\}$. Its radius ω is uncertain and follows a uniform distribution. Notably, the type of the probability distribution depends on the obstacle situations. $\mathcal{E}_{\text{safe}} = \{\boldsymbol{p} \in \mathcal{E} : \text{Prob}(\boldsymbol{p} \in \mathcal{E}_{\text{obs}_i}(\omega_i)) \leq \Delta, i = 1, \ldots, n_o\}$ denotes the relatively safe environment, where Prob$(\cdot)$ is the probability of collision, and $\Delta \in [0, 1]$ is a predefined risk level.

Let $\mathcal{X} \subset \mathbb{R}^n$ be robot configuration-space (C-space), where $n \in \mathbb{N}$. There exist obstacle $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ and risk-bounded $\mathcal{X}_{\text{safe}} = \mathcal{X} \backslash \mathcal{X}_{\text{obs}}$ configuration spaces. The common collision checker [41] that determines whether a configuration $\boldsymbol{x} \in \mathcal{X}_{\text{obs}}$ and collides with the fully known obstacles in $\mathcal{E}$ fails under uncertainty since the C-space mapped from the probabilistic $\mathcal{E}_{\text{obs}}$ is usually hard to know. For simplicity, we assume the robot considered in this article is a point mass, and $\boldsymbol{x} \in \mathcal{X}_{\text{safe}} \iff \boldsymbol{p} \in \mathcal{E}_{\text{safe}}$, where $\boldsymbol{p}$ is the corresponding position of $\boldsymbol{x}$. It is obvious that $\boldsymbol{p} \in \mathcal{E}_{\text{safe}}$ if $\boldsymbol{x} \in \mathcal{X}_{\text{safe}}$. On the other hand, we consider a robot's configuration $\boldsymbol{x}$ to be safe if the position in $\mathcal{E}$ is risk-bounded because the robot is regarded as a point. This assumption holds for a wide variety of robots, such as unmanned aerial vehicles, autonomous ground vehicles, and underwater robots [22].

Let $\boldsymbol{x}_{\text{init}}$ and $\boldsymbol{x}_{\text{goal}}$ be the safe start and goal configurations, respectively. We aim to find the risk-bounded path as below:

$$\boldsymbol{\pi}(t) : [0, T] \to \mathcal{X}_{\text{safe}}, \boldsymbol{\pi}(0) = \boldsymbol{x}_{\text{init}}, \boldsymbol{\pi}(T) = \boldsymbol{x}_{\text{goal}}, \quad (2)$$

where $\boldsymbol{\pi}(t) = [\boldsymbol{x}_{\text{init}}, \ldots, \boldsymbol{x}_{\text{goal}}]$ denotes a risk-bounded path that embraces the configurations lying in $\mathcal{X}_{\text{safe}}$ and connecting $\boldsymbol{x}_{\text{init}}$ and $\boldsymbol{x}_{\text{goal}}$. According to the assumption mentioned above, the positions of the path $\boldsymbol{\psi}(t) = [\boldsymbol{p}_{\text{init}}, \ldots, \boldsymbol{p}_{\text{goal}}] : [0, T] \to$

$\mathcal{E}_{\text{safe}}$ given $\boldsymbol{\pi}(t) \to \mathcal{X}_{\text{safe}}$. Therefore, the risk-bounded optimal path planning problem is defined as:

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}\in\Sigma}{\operatorname{argmin}} \sum_{i=0}^{n-1} \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|_2^2 \tag{3}$$

$$\text{s.t. } \boldsymbol{\pi}(0) = \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}},\ \boldsymbol{\pi}(T) = \boldsymbol{x}_n = \boldsymbol{x}_{\text{goal}}, \tag{4}$$

$$\text{Prob}(\boldsymbol{\psi}(t) \in \mathcal{E}_{\text{obs}_i}(\omega_i)) \le \Delta,\ \forall t \in [0,T]\,|_{i=1}^{n_o}, \tag{5}$$

where the cost function (3) is the length of the path $\boldsymbol{\pi}(t)$ measured with the Euclidean distance, and Σ represents the set of all feasible paths. The probabilistic constraints representing the collision risk are indicated by (5).

B. Risk-Bounded Path Planning Using Risk Contours Map

RRT-based path planner: Sampling-based path planning methods can find paths quickly in high-dimensional C-space due to their strong exploration ability. RRT grows a searching tree rooted at the given start state $\boldsymbol{x}_{\text{init}}$ and uses a uniform sampler to choose a state $\boldsymbol{x}_{\text{sample}}$ randomly. If $\boldsymbol{x}_{\text{sample}} \in \mathcal{X}_{\text{safe}}$, the algorithm selects its nearest node on the tree $\boldsymbol{x}_{\text{nearest}}$ and attempts to extend $\boldsymbol{x}_{\text{nearest}}$ using a steering function. $\boldsymbol{x}_{\text{sample}}$ will be normalized as $\boldsymbol{x}_{\text{new}}$, and the tree includes it if the collision checking procedure for the extension passes. The edge set of the tree includes the pair $(\boldsymbol{x}_{\text{nearest}}, \boldsymbol{x}_{\text{new}})$ simultaneously. This process is repeated until a state close enough to the goal state is added to the tree, and RRT returns the final solution.

Risk contours map: Since the probabilistic uncertain obstacles $\mathcal{E}_{\text{obs}_i}(\omega_i)$ exist, the collision checking modules of the sampling-based risk-bounded path planners need to be modified. The environment cannot be simply divided into occupied and free space anymore due to the uncertain size, location, or geometry of $\mathcal{E}_{\text{obs}_i}(\omega_i)$. The obstacle augmentation method may result in over-conservative paths or even worse infeasibility. It is necessary to build a map to describe the probabilistic uncertain obstacles to facilitate the assessment of collision risk. Therefore, risk contours map was developed [21]. This map can show the continuous acceptable safe space $\mathcal{E}_{\text{safe}}$ in uncertain environments $\mathcal{E}$ under a designated risk level Δ. Specifically, a Δ-risk contour $\mathcal{C}_r^{\Delta}$ showing the acceptable safe workspace with respect to an obstacle is denoted as:

$$\mathcal{C}_r^{\Delta} \coloneqq \{\boldsymbol{p} \in \mathcal{E} : \text{Prob}\,(\boldsymbol{p} \in \mathcal{E}_{\text{obs}}(\omega) \le \Delta)\}\,. \tag{6}$$

We approximate the probabilistic constraint (6) as a deterministic constraint to make the problem tractable:

$$\widehat{\mathcal{C}}_r^{\Delta} = \left\{\boldsymbol{p}\in\mathcal{E}: \frac{\mathbb{E}[\mathcal{P}^2(\boldsymbol{p},\omega)] - \mathbb{E}[\mathcal{P}(\boldsymbol{p},\omega)]^2}{\mathbb{E}[\mathcal{P}^2(\boldsymbol{p},\omega)]} \le \Delta, \mathbb{E}[\mathcal{P}(\boldsymbol{p},\omega)] \le 0\right\}, \tag{7}$$

where $\mathbb{E}[\mathcal{P}(\boldsymbol{p},\omega)]$ and $\mathbb{E}[\mathcal{P}^2(\boldsymbol{p},\omega)]$ denote the first and second moments, respectively. The coefficients of these polynomials are determined by the moment of probabilistic parameter ω and the coefficients of obstacle polynomial $\mathcal{P}(\boldsymbol{p},\omega)$. The proof procedure as (7) is omitted for brevity, and the interested readers are referred to [15]. By substituting the moments into (7), we get two inequalities only with the variables about positions. Thus, the original problem (3) is transformed into the following risk-bounded optimal path planning problem with deterministic constraints:

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}\in\Sigma}{\operatorname{argmin}} \sum_{i=0}^{n-1} \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|_2^2 \tag{8}$$

$$\text{s.t. } \boldsymbol{\pi}(0) = \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}},\ \boldsymbol{\pi}(T) = \boldsymbol{x}_n = \boldsymbol{x}_{\text{goal}}, \tag{9}$$

$$\boldsymbol{\psi}(t) \in \widehat{\mathcal{C}}_{r_i}^{\Delta},\ \forall t \in [0,T]\,|_{i=1}^{n_o}. \tag{10}$$

RRT-SOS [15]: RRT-SOS conducts planning in the workspace. The C-space $\mathcal{X}$ shares the same space with the environment. The algorithm's sampler performs informed sampling near the straight line between $\boldsymbol{x}_{\text{init}} \in \mathcal{X}$ and $\boldsymbol{x}_{\text{goal}} \in \mathcal{X}$ and adaptively obtains samples far away. The collision checking module verifies whether the probability of collision for the connected path between $\boldsymbol{x}_{\text{nearest}}$ and $\boldsymbol{x}_{\text{new}}$ is continuously bounded by Δ on the risk contours map. If so, the edge is risk-bounded and can be added to the edge set of the searching tree. An SOS-based technique is utilized to check the collision risk [15]. $\mathcal{S} = \{\boldsymbol{x} : s_r(\boldsymbol{x}) \ge 0,\ r = 1,\ldots,q\}$ denotes the set of the safe states of solution $\boldsymbol{\pi}$, where $s_r(\cdot)$ are all of the polynomial items of $\widehat{\mathcal{C}}_R^{\Delta} \coloneqq \{\widehat{\mathcal{C}}_{r_i}^{\Delta},\ i = 1,\ldots,n_o\}$. The connected edge is risk-bounded over the time interval $[t_1,\ t_2]$, iff polynomials $s_r(\boldsymbol{x}(t))$, $r = 1,\ldots,q$, has the SOS representation as below:

$$s_r(\boldsymbol{x}(t)) = \sigma_{0r}(t) + \sigma_{1r}(t)(t-t_1) + \sigma_{2r}(t)(t_2-t)|_{r=1}^{q}, \tag{11}$$

where $\sigma_{0r}(t)$, $\sigma_{1r}(t)$, and $\sigma_{2r}(t)$ are SOS polynomials with proper degrees.

Once an initial risk-bounded solution is found, a corresponding PRM graph is built whose nodes and edges are already verified. RRT-SOS employs the well-known Dijkstra algorithm to keep reducing the path length by querying that graph until no shorter path can be found. However, since the number of nodes on the graph is fixed once a path is found, RRT-SOS cannot guarantee to output the optimal risk-bounded path.

IV. Algorithm

Section IV-A first describes the representation of nonconvex uncertain 3D environments. The details of deep neural networks and learning-based risk-bounded path planning framework are given in Sections IV-B and IV-C, respectively.

A. Point Cloud Representation of Uncertain Environments

To implement a learning-based path planner, we first need to encode the uncertain environment into a latent embedding. 3D environments can be projected into 2D images; however, some key information may be lost. Alternatively, 3D voxels are commonly used to represent a fully known 3D obstacle, where the occupied voxels are denoted as ones, and the free ones are zeros [14]. However, voxelization usually implies quantization errors and requires a significant amount of time and space to achieve high-quality results. By contrast, the point cloud is simple but effective in representing 3D geometry [40]. It is also able to process 2D geometry with the third dimension values set to zeros. Therefore, we represent uncertain obstacles in $\mathcal{E}$ as point clouds to realize an efficient neural network sampler (NNS) and neural cost estimator (NCE).

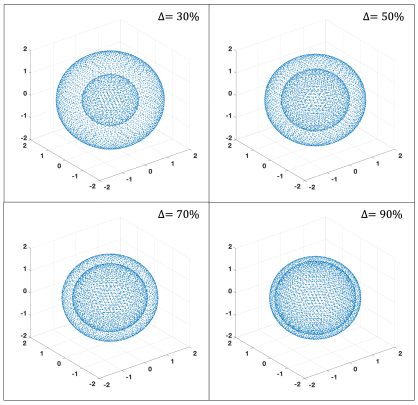


Fig. 2: Point cloud risk contours maps showing collision risk zone (the bigger blue sphere) where the collision probability is greater than Δ. The smaller Δ is, the broader risk zone is.

As introduced in Section III-B, all the approximated risk contours $\widehat{\mathcal{C}}_R^{\Delta} = \{\widehat{\mathcal{C}}_{r_i}^{\Delta}, i = 1, \ldots, n_o\}$ can explicitly evaluate whether a collision probability is smaller than the risk tolerance Δ. To generate point clouds for the risk contours of all obstacles, we compute the coordinates of points in $\mathcal{E}$ which make the equalities in the approximated contours $\widehat{\mathcal{C}}_R^{\Delta}$ hold. Once the 3D volume data of $\frac{\mathbb{E}[\mathcal{P}^2(\boldsymbol{x},\omega)]-\mathbb{E}[\mathcal{P}(\boldsymbol{x},\omega)]^2}{\mathbb{E}[\mathcal{P}^2(\boldsymbol{x},\omega)]}$ in each $\widehat{\mathcal{C}}_r^{\Delta}$ is equal to Δ, the isovalue surface data are extracted for the obstacles. The vertices, small faces, and 3D components of the normal vector data for the surfaces are computed simultaneously. The triangle point clouds corresponding to the small faces can be constructed as follows. Three boundary points $\boldsymbol{x}_1, \boldsymbol{x}_2$, and $\boldsymbol{x}_3$ of a single small face are selected and form a triangle mesh. We calculate the longest edge among them, and let us say $\boldsymbol{x}_1\boldsymbol{x}_2$. One point moves at a given step size from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$, and another point moves at another given step size from $\boldsymbol{x}_3$ to $\boldsymbol{x}_2$ simultaneously. An array of points are uniformly generated in the connected line between the two moving points. A triangle-shaped point cloud is created for the local small face step by step. As a result, the entire point clouds are formed by computing all of the small faces generated before. A deterministic point cloud map PM is finally built to represent all the uncertain obstacles with some Δ, whose boundaries are described by the $\widehat{\mathcal{C}}_R^{\Delta}$'s entries having Δ. For example, we construct the PM in Fig. 2 for the uncertain sphere in Section III-A. The risk zone (the bigger blue sphere) indicates the space where the probability of collision with the uncertain sphere is greater than Δ, while the probability of collision of the safe zone (outside the sphere) is no larger than Δ. It is noteworthy that our point cloud map does not suffer from the computational expense caused by a large size. Since it is constructed from continuous polynomials, we can adaptively change the step size when creating it.

B. Neural Network Sampler and Neural Cost Estimator for Informed Sampling in Uncertain Environments

To obtain the ground truth, we first employ RRT-SOS in every environment to obtain near-optimal solutions given abundant pairs of start and goal states. Each solution contains a set of states critical to a short path. We construct two simple yet effective DNNs, i.e., NNS and NCE to imitate RRT-SOS and generate informed critical states. Fig. 3 shows their detailed encoder-decoder architectures. NNS (top) comprises an obstacle encoder network (PCnet), inspired by PointNet [40], and an inference network (Inet), inspired by MPNet [13]. NCE (bottom) consists of the same encoder and a Cost-Inet for cost prediction. The inputs of NNS and NCE are the point cloud map PM, the current state $\boldsymbol{x}_t$, the goal $\boldsymbol{x}_{\text{goal}}$, and the specified risk tolerance Δ. NNS and NCE output the next promising state(s), and the corresponding cost to the goal for the state(s), respectively. The mean-squared-error loss is utilized for training the neural networks.

The PCnet has four key building blocks for extracting the latent embedding Z, including the transformation networks (T-Net), the CNNs, the max pooling layer, and the multi-layer perceptron (MLP). T-Net is a mini-network and performs certain convolution and full connection operations on the original sample data. It normalizes the rotation, translation and other adjustments of the data. More precisely, we use three 1D convolution layers $(64, 128, 1024)$ and select the maximum values of the column elements. Then, an MLP $(512, 256, 9)$ takes 1024-dimensional features, yielding a 3×3 transformation matrix for the T-Net. The matrix then multiplies the original point data $(k \times 3)$ to align it, which adjusts the point cloud's internal structure without changing its format. In other words, the invariability of the model for particular spatial transformation is ensured. A following 1D convolution layer (64) takes per point of the aligned data $(k \times 3)$ as input and outputs the latent encoding $(k \times 64)$. Next, another T-Net transforms the extracted encoding for the feature alignment. This T-Net has the same architecture as the last one, except its output dimension is 4096 (64×64). Two 1D convolution layers $(128, 1024)$ extract a higher-dimensional latent embedding $(k \times 1024)$, and it is passed through a max pooling layer that performs as a symmetric function to aggregate useful information from the data. Eventually, the extracted features of 1×1024 dimension are fed into the MLP $(512, 256, 64)$ to generate the latent feature Z. Each 1D convolution layer has kernel size of 1. The second layer of the MLP is additionally connected with Dropouts $(p = 0.4)$ [42]. Furthermore, the Inet, an MLP with the dimensions of $(512, 256, 128, 64, 32)$, takes the latent-space feature Z concatenated with the tensor comprised of $\boldsymbol{x}_t$, $\boldsymbol{x}_{\text{goal}}$, and Δ. Each layer of it is applied with a parametric ReLU with default parameters. We keep Dropout active during training and online planning. Using it in Inet results in a stochastic behavior to randomly generate informed samples constituting a feasible solution with high probability, and helps the planner recover from failures [13].

Considering there exists stochasticity in NNS, we let NNS process a batch of current states $\boldsymbol{B}_t$ and generate several promising states $\boldsymbol{B}_{t+1}$. Then, NCE predicts the costs to reach the goal for $\boldsymbol{B}_t$ to improve online planning performance. Note that we will prune the tree by eliminating the redundant nodes. Our method attempts to connect two nodes far away from each other. If a node has a high cost-to-goal value, there might exist some nodes in the area between the goal and itself, causing the long-distance connection fail with high probability. Thus,

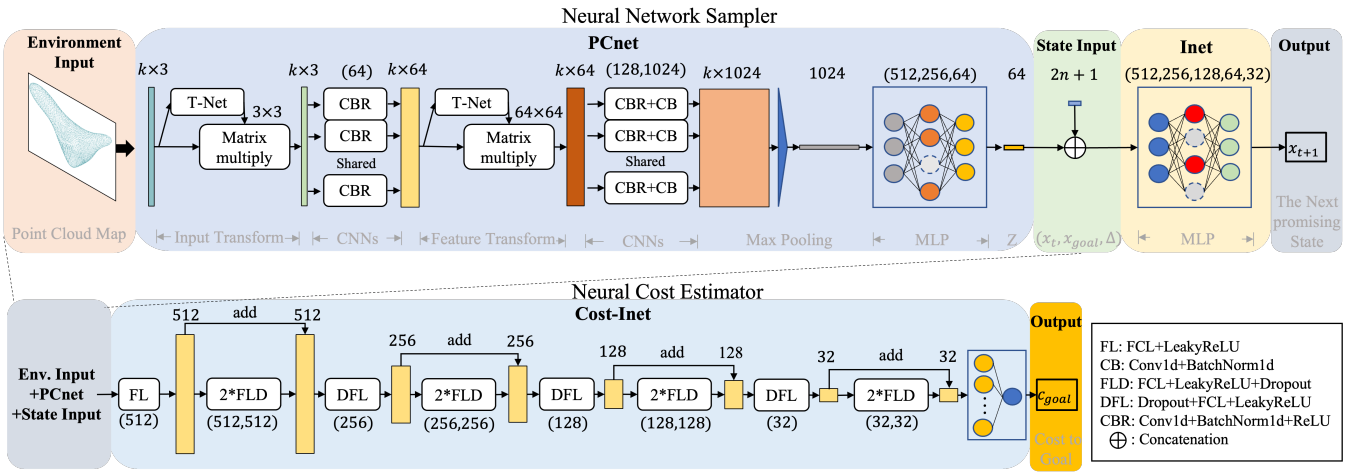


Fig. 3: The overall structures of the neural network sampler (top) and the neural cost estimator (bottom). The sampler is designed for iteratively generating the next most-promising state $\boldsymbol{x}_{t+1}$ in uncertain 3D environments, given the current state $\boldsymbol{x}_t$. The neural cost estimator aims to estimate the corresponding cost c_{goal} for $\boldsymbol{x}_t$ to reach the goal.

by using NCE to iteratively find the lowest cost node from $\boldsymbol{B}_{t+1}$, we select the critical states and save planning time.

C. 3D Neural Risk-Aware RRT Algorithm

Handcrafted heuristics are inadequate for achieving stable planning efficiency in diverse environments. To address the problem, we develop a bidirectional SBP method incorporating NNS and NCE for informed sampling, named 3D NR-RRT.

1) Algorithm Presentation: We employ a divide-and-conquer planning strategy [13] to improve planning performance. Our algorithm bidirectionally finds a coarse global path by using NNS and NCE. **NNS()** informatively expand the search tree, and **NCE()** helps to find a low-cost tree. The coarse path only has key nodes without collision risk assessment because some nodes will be eliminated by the shortcutting method, Lazy States Contraction (LSC) [43]. The connections between each pair of consecutive states are assessed by Risk Assessor (RA). **Replan()** plans a new risk-bounded path through performing bidirectional informed sampling again for any two consecutive non-connectable nodes. If it is hard to tackle some problems, we use RRT-SOS to make sure a risk-bounded path can be always found.

RA() shown in Algorithm 1 is designed to verify the collision risk of a path $\boldsymbol{\pi}$. Given the positions $\boldsymbol{p}$ of the configuration nodes $\boldsymbol{\pi}_i$, it deletes those lying in the risk zone if the inequalities in the all risk contours $\widehat{\mathcal{C}}_R^{\Delta}$ do not hold (Lines 3 and 4). Furthermore, the collision risk for every edge between two contiguous nodes is verified through **SOScondition()**. There are three steps to verify the SOS condition [15] by utilizing the Spotless toolbox [44]: 1) substitution—we derive the linear polynomial trajectory, such as $\{(x, y, z), x = a_1 t + b_1, y = a_2 t + b_2, z = a_3 t + b_3\}$, given the positions of two nodes as the endpoints of a line segment. The trajectory is substituted into each item of $\widehat{\mathcal{C}}_R^{\Delta}$, then polynomials with indeterminate time t like $\mathcal{P} = 0.6 - t + 0.5t^2 - 2.6t^3 + 1.3t^4 + 0.7t^5$ are obtained; 2) construction—we list semialgebraic constraints about time t, i.e. $h = t * (1 - t)$, and create many multiplier variables, $S = v_1 + v_2 t + v_3 t^2 + v_4 t^3 + v_5 t^4 + v_6 t^5$, where v_i are free variables; 3) check—Mosek solver [45] is used to minimize $p - Sh$ and it helps to determine whether the status of the expression is primal and dual feasible. If not, the line segment does not have the SOS form (11), and the collision probability along it is greater than Δ. The list F_{idx} records the indexes of the risk edges of current path $\boldsymbol{\pi}$. In the end, **RA()** outputs the indicator $succ$ showing whether the entire path is risk-bounded, the indexes of failed segments F_{idx}, and the global path $\boldsymbol{\pi}$ with safe nodes.

Algorithm 1: Risk Assessor (RA)

Input: $\boldsymbol{\pi}, \widehat{\mathcal{C}}_R^{\Delta}$
Output: $succ$, F_{idx}, $\boldsymbol{\pi}$

1 $F_{\text{idx}} \leftarrow \varnothing$;
2 **for** $i = 1, \ldots, size(\boldsymbol{\pi}) - 2$ **do**
3 **if** *not* $\widehat{\mathcal{C}}_R^{\Delta}(\boldsymbol{p}_i)$ **then**
4 $\boldsymbol{\pi} \leftarrow$delete $\boldsymbol{\pi}_i$;
5 **for** $i = 0, \ldots, size(\boldsymbol{\pi}) - 2$ **do**
6 **if** *not* SOScondition $(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$ **then**
7 $F_{\text{idx}} \leftarrow F_{\text{idx}} \cup \{i\}$;
8 **if** $F_{idx} = \varnothing$ **then**
9 $succ \leftarrow$True;
10 **else**
11 $succ \leftarrow$False;
12 **return** $succ$, F_{idx}, $\boldsymbol{\pi}$;

The proposed 3D NR-RRT is shown in Algorithm 2. The inputs are the pairs of start and goal configurations $\{\boldsymbol{x}_{\text{init}}, \boldsymbol{x}_{\text{goal}}\}$, the risk tolerance Δ, and all the inner approximation of risk contours $\widehat{\mathcal{C}}_R^{\Delta}$ and their corresponding point cloud map PM. We first utilize **PCnet()** to encode PM into an environment encoding Z. The path solution $\boldsymbol{\pi}$ is initialized with empty. A forward node set $\mathcal{V}_f$ containing the start state and a backward

node set $\mathcal{V}_b$ having the goal state are created, respectively. The update procedures of $\mathcal{V}_f$ and $\mathcal{V}_b$ are the same. Here, we take the update of $\mathcal{V}_f$ for example. A batch of samples $\boldsymbol{B}_{\text{sample}}$ is generated by **NNS()** given the batch of Z, Δ, and the top nodes in $\mathcal{V}_f$ and $\mathcal{V}_b$, i.e., $\boldsymbol{B}_Z$, $\boldsymbol{B}_\Delta$, $\boldsymbol{B}_f^{top}$ as the local starts, and $\boldsymbol{B}_b^{top}$ as the local goals. **NCE()** predicts the costs for $\boldsymbol{B}_{\text{sample}}$. The one $\boldsymbol{x}_{t+1}$ with the lowest cost is selected and added to $\mathcal{V}_f$. If the connection of the top nodes in $\mathcal{V}_f$ and $\mathcal{V}_b$ satisfies the SOS condition, a coarse global path $\boldsymbol{\pi}$ containing the samples from $\mathcal{V}_f$ and $\mathcal{V}_b$ is deemed to be found (Lines 7–10). If not, we swap $\mathcal{V}_f$ with $\mathcal{V}_b$ and continue to expand $\mathcal{V}_b$.

After getting $\boldsymbol{\pi}$, we employ **LSC()** to remove redundant nodes in $\mathcal{V}$. It selects critical nodes by trying to connect non-adjacent nodes of $\boldsymbol{\pi}$. If the shortcut line lies in the safe zone, the nodes originally locating between the endpoints are removed. The remaining nodes may become parts of the final solution. Then, we feed the current $\boldsymbol{\pi}$ and approximated risk contours $\widehat{\mathcal{C}}_R^\Delta$ into **RA()**. If the indicator $succ$ is False, the algorithm calls **Replan()**, **LSC()**, and **RA()** for the risk line segments. **Replan()** regards the non-connectable states as a new pair of start and goal. It employs the bidirectional informed SBP (Lines 3–12) to find local solutions for every pair. If a global feasible path solution still cannot be found after N_j attempts, we call RRT-SOS to plan local risk-bounded paths $\boldsymbol{\pi}_{\text{local}}$ for each risk line segment $\{\boldsymbol{\pi}_k, \boldsymbol{\pi}_{k+1}\}$ (Lines 24 and 25). RRT-SOS performs a uniform sampling, guaranteeing probabilistic completeness. Note that we enable it to sample in $\mathcal{X}$. It verifies the collision risks for the new local paths through SOScondition(), returning the final solution directly. Note that changing Δ during planning will not degrade our performance because NNS has learned how to generate safe nodes under different Δ and RA can assess different $\widehat{\mathcal{C}}_R^\Delta$.

2) Worst-Case Completeness and Optimality: For a resolvable risk-bounded path planning problem in uncertain environments, RRT-SOS algorithm sampling uniformly guarantees completeness, namely it always finds a risk-bounded path. In our 3D NR-RRT, **NNS()** draws samples randomly due to the Dropout layers, making a coarse path can be found at least for trials N. As the number of iterations of **Replan()** exceeds the threshold N_j, we immediately use the standby planner, i.e., RRT-SOS, to connect the non-connectable states. The states of a feasilbe solution in the environment will be sampled since the standby planner is allowed to run infinity. The probability that 3D NR-RRT will generate a risk-bounded path, if one exists, approaches one as the number of iterations of RRT-SOS approaches infinity, i.e.,

$$\lim_{n\to\infty} \text{Prob}(\boldsymbol{\pi}_n \cap \boldsymbol{x}_{\text{goal}} \neq \varnothing) = 1. \tag{12}$$

Therefore, employing RRT-SOS as the standby solver for corner cases provides the worst-case completeness guarantees. The proof of it can be found in *Theorem 1* of [23].

Recall that RRT-SOS lacks the processes ChooseParent() and Rewire() which are essential for optimizing the path cost in RRT* [46]. Similarly, our algorithm can only guarantee to quickly find near-optimal paths. It is feasible to guarantee asymptotic optimality for us. RRT-SOS can inherit asymptotic optimality from the internal machinery of RRT* by adding ChooseParent() and Rewire(). Then, we conduct our informed sampling for a fixed iteration and allow the optimal version of RRT-SOS to perform afterward. The tree constructed by this combined sampling manner will reach the whole configure space $\mathcal{X}$ and be updated by the incremental rewiring approach. The optimality proofs will be similar to [46] since we only improve the sampling and collision-checking modules. However, modifying RRT-SOS is out of scope and contribute to a low input-output ratio since efficiently finding near-optimal solutions in uncertain nonconvex environments is more practical than pursuing asymptotic optimality.

Algorithm 2: 3D Neural Risk-Aware RRT

Input: $\boldsymbol{x}_{\text{init}}$, $\boldsymbol{x}_{\text{goal}}$, PM, Δ, $\widehat{\mathcal{C}}_R^\Delta$
Output: $\boldsymbol{\pi}$

1 $Z \leftarrow \text{PCnet}(PM)$;
2 $\mathcal{V}_f \leftarrow \boldsymbol{x}_{\text{init}}$, $\mathcal{V}_b \leftarrow \boldsymbol{x}_{\text{goal}}$, $\boldsymbol{\pi} \leftarrow \varnothing$;
3 **for** $i = 0, \ldots, N$ **do**
4 $\quad \boldsymbol{B}_{\text{sample}} \leftarrow \text{NNS}\left(\boldsymbol{B}_f^{top}, \boldsymbol{B}_b^{top}, \boldsymbol{B}_Z, \boldsymbol{B}_\Delta\right)$;
5 $\quad \boldsymbol{x}_{t+1} \leftarrow \text{argmin}\,\text{NCE}(\boldsymbol{B}_Z, \boldsymbol{B}_{\text{sample}}, \boldsymbol{B}_{\text{goal}}, \boldsymbol{B}_\Delta)$;
6 $\quad \mathcal{V}_f \leftarrow \mathcal{V}_f \cup \{\boldsymbol{x}_{t+1}\}$;
7 $\quad$ **if** SOScondition$(\mathcal{V}_f^{top}, \mathcal{V}_b^{top})$ **then**
8 $\quad\quad \mathcal{V} \leftarrow \text{concatenate}(\mathcal{V}_f, \mathcal{V}_b)$;
9 $\quad\quad \boldsymbol{\pi} = \mathcal{V}$;
10 $\quad\quad$ Break;
11 $\quad$ SWAP$(\mathcal{V}_f, \mathcal{V}_b)$;
12 $\quad$ **return** $\boldsymbol{\pi} = \varnothing$ **if** $i = N$;
13 $\boldsymbol{\pi} \leftarrow \text{LSC}(\boldsymbol{\pi})$;
14 $succ$, F_{idx}, $\boldsymbol{\pi} \leftarrow \text{RA}(\boldsymbol{\pi}, \widehat{\mathcal{C}}_R^\Delta)$;
15 **if** $succ$ **then**
16 $\quad$ **return** $\boldsymbol{\pi}$;
17 **else**
18 $\quad$ **for** $j = 0, \ldots, N_j$ **do**
19 $\quad\quad \boldsymbol{\pi} \leftarrow \text{Replan}(\boldsymbol{\pi}, Z, F_{\text{idx}}, \Delta)$;
20 $\quad\quad \boldsymbol{\pi} \leftarrow \text{LSC}(\boldsymbol{\pi})$;
21 $\quad\quad succ$, F_{idx}, $\boldsymbol{\pi} \leftarrow \text{RA}(\boldsymbol{\pi}, \widehat{\mathcal{C}}_R^\Delta)$;
22 $\quad\quad$ **return** $\boldsymbol{\pi}$ **if** $succ$;
23 $\quad$ **for** k *in* F_{idx} **do**
24 $\quad\quad \boldsymbol{\pi}_{\text{local}} \leftarrow \text{RRT-SOS}(\boldsymbol{\pi}_k, \boldsymbol{\pi}_{k+1}, \widehat{\mathcal{C}}_R^\Delta)$;
25 $\quad\quad \boldsymbol{\pi} \leftarrow \text{insert}(\boldsymbol{\pi}_{\text{local}})$;
26 $\quad$ **return** $\boldsymbol{\pi}$;

V. Simulation Experiments

The implementation and training details of our DNNs are described in Section V-A. We evaluate our algorithm by comparing with recent state-of-the-art work in uncertain environments with different acceptable risk tolerances in Section V-B. Then, the performance comparison in seen and unseen cluttered uncertain environments is conducted in Section V-C.

A. Implementation and Training Details

Without loss of generality, we consider a three-dimensional configuration space $\mathcal{X} \subset \mathbb{R}^3$. An environment $\mathcal{E}$ contains

TABLE I: Comparisons of the Total Mean and Median Computation Time and Path Qualities with Standard Deviations (S.D.) of the Initial (In.) and Near-optimal (No.) Solutions in Uncertain Single 3D Environments With Ten Risk Tolerances

	Convex Environment				Nonconvex Environment			
	Time(s)		Length		Time(s)		Length	
	Mean±S.D.	Median	Mean±S.D.	Median	Mean±S.D.	Median	Mean±S.D.	Median
Ours	1.777 ± 1.503	1.528	9.606 ± 2.056	8.647	$\mathbf{1.515 \pm 1.065}$	**1.114**	2.495 ± 0.408	2.520
RRT-SOS (No.) [15]	6.838 ± 1.032	4.057	9.921 ± 2.180	9.897	12.125 ± 32.746	4.434	$\mathbf{2.393 \pm 0.555}$	**2.361**
RRT-SOS (In.) [15]	1.308 ± 1.249	1.020	11.040 ± 2.754	10.883	4.778 ± 8.576	2.173	2.500 ± 0.683	2.424
Time-Varying SOS [15]	29.975 ± 221.257	12.131	$\mathbf{9.218 \pm 2.045}$	**9.272**	-	-	-	-
Standard SOS [15]	$\mathbf{0.504 \pm 0.156}$	**0.503**	9.425 ± 1.536	9.332	-	-	-	-

one or multiple uncertain obstacle(s). Given different risk tolerances, we construct corresponding point cloud maps PMs for every environment. 10000 demonstration paths for training and 2000 for testing are collected by RRT-SOS under each risk tolerance. We store the critical 3D coordinates and the $\mathcal{L}_2$ norm costs to reach their next nodes and goals of every demonstration path. The key parameters in RRT-SOS include α and β. The former denotes the size of the sampling neighborhood of the connection between $(\boldsymbol{x}_{\text{init}}, \boldsymbol{x}_{\text{goal}})$. The latter indicates the number of samples per round. α increases after every β samples are drawn. The smaller α is, the closer samples are drawn to the line. Such a manner improves the quality of the initial path at the cost of more computation time to identify the eligible nodes. The fairness of comparison experiments with the different parameters can be maintained because we emphasize both the planning time and path quality of the found initial and resultant paths. The maximum number of bidirectional connection N is 100, and replan attempts N_j is 10. There are three nodes in each batch $\boldsymbol{B}_t$.

We train PCnet end-to-end with Inet for NNS and with Cost-Inet for NCE, respectively. The weights of PCnet are not frozen. NNS and NCE are trained with the PyTorch on a NVIDIA RTX 2080Ti. The simulated experiments are conducted on Ubuntu 18.04 with an Intel Core i9-9900KF CPU. The learning rate is $\eta = 0.001$. The batch sizes are 64 and 32 in Section V-B and V-C, respectively. The Adam optimizer [47] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ is used.

B. Comparison Experiments Under Different Risk Tolerances

We aim to verify that risk-bounded near-optimal paths in uncertain convex and nonconvex environments with different risk tolerances Δ can be quickly found by our algorithm. Some examples of planning solutions from the SBP algorithms are presented in Fig. 4. The images in the first row are the results facing an uncertain spherical obstacle. Its polynomial with $\omega \sim \mathcal{U}(1, 2)$ has been given in Section III-A, and $(x, y, z) \in [-5, 5] \times [-5, 5] \times [-5, 5]$. The second row represents the results in a nonconvex environment with uncertain parameter $\omega \sim \mathcal{N}(0.1, 0.001)$, where $(x, y, z) \in [-1, 1] \times [-1, 1] \times [-1, 1]$. Ten risk tolerances $\Delta = \{10\%, 20\%, \ldots, 100\%\}$ are used in each environment.

We report the statistical results, i.e., mean values, standard deviations (S.D.), and medians, of computation time and path qualities in Table I. We give the results of both the initial (In.) and near-optimal (No.) solutions from RRT-SOS [15] to get rid of unfairness affected by its different parameters. RRT-SOS has to sample average of 11.761 ± 10.754 and 4.214 ± 8.565 nodes for finding different paths in the two environments, respectively. By contrast, our method needs 2.612 ± 2.013 and 2.487 ± 1.963 nodes to decently solve the problems, thanks to the high success rates of informed sampling 95.558% and 96.285% in the environments. The fewer nodes that need to be sampled, the fewer iterations need to compute, resulting in less planning time. Our method can take less time to find better initial paths in the nonconvex environment. The initial paths are exactly our resultant near-optimal solutions, whereas RRT-SOS has to take more time to refine them. Although the standard SOS optimization [15] finds the comparable results with less planning time, its success rate is only 60.15% in the convex environment with different Δ. Both the time-varying SOS [15] and standard SOS optimization methods exhibit 0% success rates in the nonconvex environment although we adjust the numbers of linear pieces and iteration of their heuristic algorithms. The high polynomial order of obstacle risk contours leads the runtime of these methods to increase exponentially, exhausting the time budget eventually. The optimization-based methods often encounter the unstable numerical computation and do not guarantee completeness.

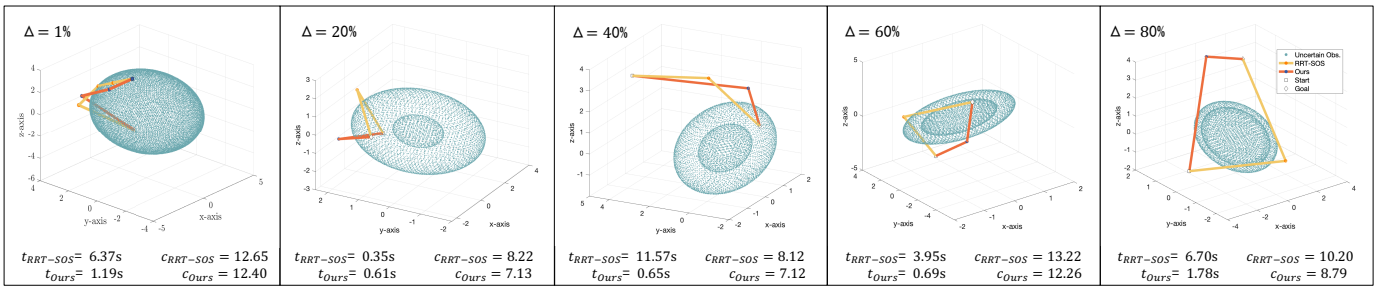
To simulate the real situation, we separately specify a quite low risk tolerance $\Delta = 1\%$. In the convex environment, the computation time is 1.882 ± 1.225s and 7.352 ± 1.118s, and 2.851 ± 2.168 and 13.589 ± 9.658 samples are drawn for ours and RRT-SOS (No.), respectively. The corresponding path lengths by them are 11.677 ± 2.168 and 11.885 ± 2.159. In the nonconvex environment, ours takes 1.684 ± 1.148 seconds and 2.856 ± 1.978 nodes for the path quality of 3.538 ± 0.658 on average, while RRT-SOS (No.) takes 10.892 ± 21.549 seconds and 5.625 ± 5.584 nodes for the solution length of 3.795 ± 0.845 on average. Some representative example trajectories are depicted in Fig. 4 as well. In short, our 3D NR-RRT can efficiently find near-optimal paths while keeping the collision risk bounded, regardless of the specified risk tolerance, start and goal states in a seen uncertain environment.

C. Comparison Experiments in Seen and Unseen Cluttered Uncertain Environments

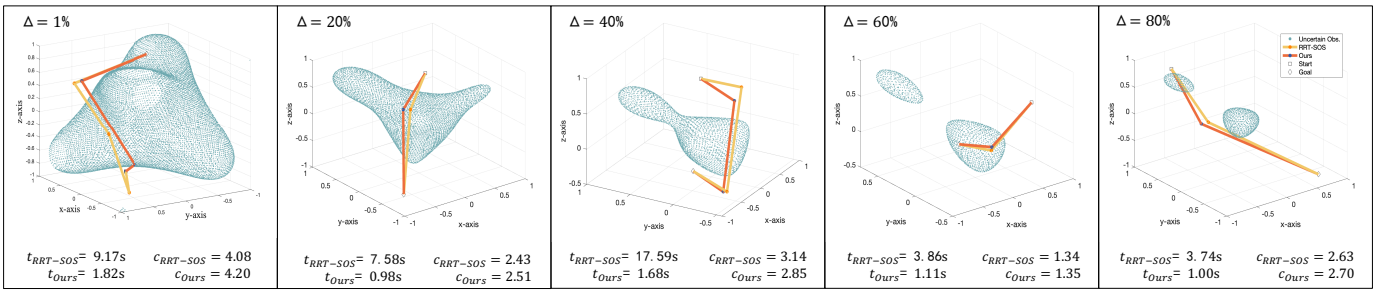
We create ten seen and two unseen cluttered environments with $[x, y, z] \in [-5, 5] \times [-5, 5] \times [-5, 5]$. The unseen maps are not utilized in the training. Seven uncertain 3D obstacles with $\Delta = 20\%$ are randomly placed in each environment. Four of them are spherical with $\omega \sim \mathcal{U}(0.5, 1)$, two are peanut-shell-shaped with $\omega \sim \mathcal{N}(0.1, 0.1)$, and one is fox-head-shaped with

TABLE II: Comparisons of the Total Mean and Median Planning Time and Path Qualities with Standard Deviations (S.D.) of the Initial (In.) and Near-optimal (No.) Solutions in Ten Seen and Two Unseen Uncertain Cluttered Nonconvex 3D Environments

	Seen Cluttered Environments					Unseen Cluttered Environments				
	Time(s)		Length		Nodes	Time(s)		Length		Nodes
	Mean±SD	Median	Mean±SD	Median	Total	Mean±SD	Median	Mean±SD	Median	Total
Ours w/ NCE	$\mathbf{5.792 \pm 9.214}$	**4.553**	$\mathbf{7.997 \pm 2.251}$	**8.059**	**22999**	$\mathbf{4.541 \pm 3.615}$	**4.017**	$\mathbf{7.974 \pm 2.224}$	**7.986**	**4330**
Ours w/o NCE	8.679 ± 11.603	8.092	8.095 ± 2.353	8.145	29781	8.499 ± 7.976	7.389	8.101 ± 2.282	8.113	4998
RRT-SOS (No.) [15]	45.740 ± 90.023	18.176	9.114 ± 2.281	9.127	121483	48.859 ± 139.745	16.118	9.137 ± 2.266	9.136	28961
RRT-SOS (In.) [15]	11.377 ± 13.981	7.172	9.640 ± 2.672	9.530	121483	10.803 ± 19.156	6.191	9.686 ± 2.689	9.518	28961

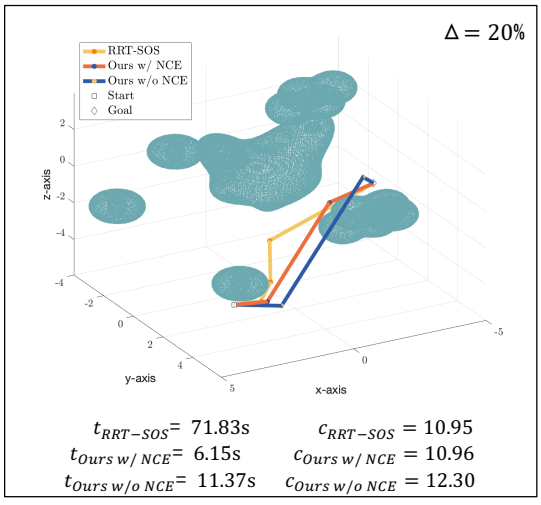


(a) Comparison Examples Facing An Uncertain Convex 3D Obstacle with Different Risk Tolerances

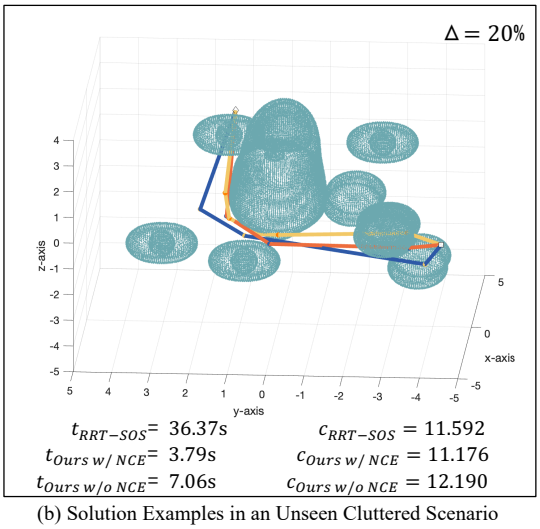


(b) Comparison Examples Facing An Uncertain Nonconvex 3D Obstacle with Different Risk Tolerances

Fig. 4: Comparison examples from our algorithm (orange) and RRT-SOS (yellow). The 3D images in 4(a) and 4(b) present the resultant paths from the two methods handling the uncertain convex and nonconvex obstacles with different Δ, respectively. The corresponding computation time (t) and costs (c) of the example paths are at the bottom.



(a) Solution Examples in a Seen Cluttered Scenario



(b) Solution Examples in an Unseen Cluttered Scenario

Fig. 5: Comparison examples in seen and unseen uncertain cluttered 3D environments from our algorithm with the neural cost estimator (orange, ours w/ NCE), our algorithm without the neural cost estimator (blue, ours w/o NCE), and RRT-SOS (yellow). The computation time (t) and the costs (c) of the example paths are at the bottom correspondingly.

$\omega \sim \mathcal{N}(1,1)$. 10000 valid start and goal pairs for training and 2000 pairs for testing are randomly generated in every seen scenario. We also randomly generate 2000 new pairs only for testing in each unseen scenarios.

Given the 20000 testing start and goal pairs in the ten seen and 4000 in the two unseen environments, we execute the algorithms to collect the statistical results in Table II. We use three methods to solve the problems, namely our algorithm with neural cost estimator (ours w/ NCE), ours without neural cost estimator (ours w/o NCE), and RRT-SOS. Ours w/o NCE indicates that our neural network sampler generates only one node at every iteration. The node is added to the set directly without cost estimation by NCE. We can see from Table II that ours w/ NCE achieves the best performance in solving the problems in terms of the planning time, computation resources, and path quality. Because our algorithm expands nodes with a nonuniform strategy learned from the excellent demonstration paths, the likelihood of extreme results is reduced. The much smaller variance of our 3D NR-RRT indicates that it has good robustness. The number of nodes from our methods is roughly one-sixth of those from RRT-SOS, which means our learning-based heuristic approaches save computing resources and reduce the waste of time sampling in the unfeasible space. Our algorithm uses the informative heuristic to let the search tree directly explore the promising space, saving the computational cost for finding near-optimal solutions. The results presented in the first and last rows of Table II indicate that 3D NR-RRT also finds better initial solutions but takes less time than RRT-SOS, which overcomes the natural drawback of SBP methods. As the results of an ablation study, the first two rows show that NCE improves planning efficiency and quality by selecting the appropriate nodes. It eliminates the nodes which might need replanning or become redundant during performing LSC. Therefore, our algorithm requires less computation time and fewer resources to find near-optimal paths. We select some planned paths in a seen and an unseen environment and their corresponding statistic results for illustration in Fig. 5.

VI. Conclusions and Discussions

In this article, we solve a general class of problems about how to quickly find relatively safe and short paths under environmental uncertainty. An efficient and general risk-bounded sampling-based path planning framework in uncertain nonconvex environments is proposed, named 3D NR-RRT. Aided by simple yet effective DNN models, it requires a little computation time and few resources to find risk-bounded near-optimal solutions avoiding diverse probabilistic obstacles. The neural bidirectional informed sampling and cost estimation contribute to fewer planning iterations and better solution quality in the challenging environments. Our numerical experiments show that 3D NR-RRT consistently achieves great superiority compared with the existing risk-bounded path planners in the complex uncertain environments.

In the face of the workspaces involving uncertain nonconvex 2/3D obstacles, our method can be used as a risk-bounded geometric path planner in the C-spaces that include positions, such as $\mathbb{R}^3$, the special Euclidean group SE(2), and SE(3). However, it is not applicable to the C-spaces not containing positions, such as the joint space of the manipulator. In the future, we will consider kinodynamic constraints and motion uncertainties, and deploy our method in realistic applications.

References

[1] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation," *arXiv preprint arXiv:2103.02828*, 2021.

[2] S. Daftry, N. Abcouwer, T. Del Sesto, S. Venkatraman, J. Song, L. Igel, A. Byon, U. Rosolia, Y. Yue, and M. Ono, "Mlnav: Learning to safely navigate on martian terrains," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5461–5468, 2022.

[3] H. Ma, F. Meng, C. Ye, J. Wang, and M. Q. Meng, "Bi-risk-rrt based efficient motion planning for mobile robots," *IEEE Trans. Intell. Veh*, vol. 7, no. 3, pp. 722–733, 2022.

[4] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[5] 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.

[6] H. Peng, H. Zhao, X. Wang, and Y. Li, "Robust motion trajectory optimization of overhead cranes based on polynomial chaos expansion," *ISA transactions*, vol. 110, pp. 71–85, 2021.

[7] 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.

[8] A. M. Jasour, A. Hofmann, and B. C. Williams, "Moment-sum-of-squares approach for fast risk estimation in uncertain environments," in *2018 IEEE conference on decision and control (CDC)*. IEEE, 2018, pp. 2445–2451.

[9] 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.

[10] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[12] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

[13] 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.

[14] J. Wang, X. Jia, T. Zhang, N. Ma, and M. Q.-H. Meng, "Deep neural network enhanced sampling-based path planning in 3d space," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3434–3443, 2021.

[15] A. Jasour, W. Han, and B. Williams, "Convex risk bounded continuous-time trajectory planning in uncertain nonconvex environments," in *Robotics: Science and Systems*, 2021.

[16] 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.

[17] C. Dawson, A. Jasour, A. Hofmann, and B. Williams, "Provably safe trajectory optimization in the presence of uncertain convex obstacles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6237–6244.

[18] 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.

[19] 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.

[20] 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.

[21] A. M. Jasour and B. C. Williams, "Risk contours map for risk bounded motion planning under perception uncertainties." in *Robotics: Science and Systems*, 2019.

[22] W. Han, A. Jasour, and B. Williams, "Non-gaussian risk bounded trajectory optimization for stochastic nonlinear systems in uncertain environments," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 044–11 050.

[23] F. Meng, L. Chen, H. Ma, J. Wang, and M. Q.-H. Meng, "Nr-rrt: Neural risk-aware near-optimal path planning in uncertain nonconvex environments," *IEEE Transactions on Automation Science and Engineering*, 2022, doi10.1109/TASE.2022.3215562.

[24] 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.

[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] A. Dixit, M. Ahmadi, and J. W. Burdick, "Risk-averse receding horizon motion planning," *arXiv preprint arXiv:2204.09596*, 2022.

[27] H. Peng, B. Shi, X. Wang, and C. Li, "Interval estimation and optimization for motion trajectory of overhead crane under uncertainty," *Nonlinear Dynamics*, vol. 96, pp. 1693–1715, 2019.

[28] D. Lenz, M. Rickert, and A. Knoll, "Heuristic search in belief space for motion planning under uncertainties," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2659–2665.

[29] C. Danielson, K. Berntorp, A. Weiss, and S. Di Cairano, "Robust motion planning for uncertain systems with disturbances using the invariant-set motion planner," *IEEE Transactions on Automatic Control*, vol. 65, no. 10, pp. 4456–4463, 2020.

[30] È. Pairet, J. D. Hernández, M. Carreras, Y. Petillot, and M. Lahijanian, "Online mapping and motion planning under uncertainty for safe navigation in unknown environments," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3356–3378, 2021.

[31] H. Peng, J. Bao, G. Huang, Z. Li, and X. Wang, "Chance-constrained sneaking trajectory planning for reconnaissance robots," *Applied Mathematical Modelling*, vol. 112, pp. 224–237, 2022.

[32] M. Castillo-Lopez, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "Model predictive control for aerial collision avoidance in dynamic environments," in *2018 26th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2018, pp. 1–6.

[33] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3620–3625, 2020.

[34] H. Wang, X. Ding, J. He, K. Margellos, and A. Papachristodoulou, "Safety-aware optimal control in motion planning," *arXiv preprint arXiv:2204.13380*, 2022.

[35] 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

[36] G. Flaspohler, N. A. Roy, and J. W. Fisher III, "Belief-dependent macro-action discovery in pomdps using the value of information," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 108–11 118, 2020.

[37] 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.

[38] 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.

[39] 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.

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[41] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3859–3866.

[42] 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.

[43] 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.

[44] M. M. Tobenkin, F. Permenter, and A. Megretski, "Spotless polynomial and conic optimization," *View online*, 2013.

[45] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, Version*, vol. 4, 2019.

[46] 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.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.