Safe and Efficient Path Planning under Uncertainty via Deep Collision Probability Fields

Felix Herrmann[†,1], Sebastian Zach[†,1], Jacopo Banfi, Jan Peters[1,3,4], Georgia Chalvatzaki[‡,1,3] and Davide Tateo[‡,1]

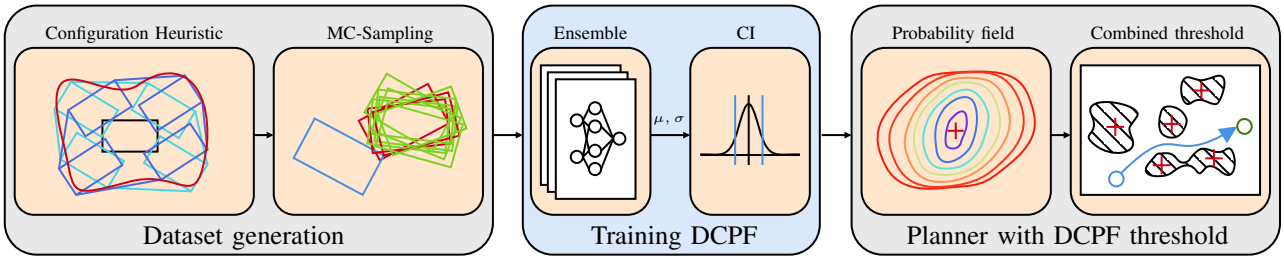

Fig. 1. Proposed pipeline for learning and planning with Deep Collision Probability Fields (DCPF): (**Left**) we start by generating a dataset with Monte Carlo sampling, balanced by selecting the configurations with a heuristic based on Minkowski Sums; (**Centre**) an ensemble of DCPF networks is trained on this dataset. The upper bound of the Confidence Interval (CI) of this ensemble is then used to approximate collision probability fields for an obstacle. (**Right**) In a scenario with multiple obstacles, we can threshold the combined collision probability estimate to exclude unsafe states when planning.

***Abstract*— Estimating collision probabilities between robots and environmental obstacles or other moving agents is crucial to ensure safety during path planning. This is an important building block of modern planning algorithms in many application scenarios such as autonomous driving, where noisy sensors perceive obstacles. While many approaches exist, they either provide too conservative estimates of the collision probabilities or are computationally intensive due to their sampling-based nature. To deal with these issues, we introduce Deep Collision Probability Fields, a neural-based approach for computing collision probabilities of arbitrary objects with arbitrary unimodal uncertainty distributions. Our approach relegates the computationally intensive estimation of collision probabilities via sampling at the training step, allowing for fast neural network inference of the constraints during planning. In extensive experiments, we show that Deep Collision Probability Fields can produce reasonably accurate collision probabilities (up to 10^{-3}) for planning and that our approach can be easily plugged into standard path planning approaches to plan safe paths on 2-D maps containing uncertain static and dynamic obstacles. Additional material, code, and videos are available at `https://sites.google.com/view/ral-dcpf`.**

I. Introduction

Safety is one of the most critical issues that needs to be solved to deploy autonomous agents in the real world. This is particularly important for autonomous robots and cars, as the perception of the world is subject to different sources of uncertainty such as noisy sensor measurements, approximate target tracking models, sensor malfunctioning, or adversarial attacks [1]. To deal with uncertainty, path planning and control approaches could incorporate some form of probabilistic constraints [2]. Using such constraints, we can force the Collision Probability (CP) to take very small values while preventing the planner from generating overly conservative paths, leveraging the accurate uncertainty estimate coming from modern perception systems.

However, even for 2D cases, with robots and obstacles of simple shapes (cf. Fig. 1), whose pose is described by a Gaussian distribution, it is impossible to calculate the CP in closed form. To incorporate CP constraints in path planning, researchers explored two main directions: approximate methods, such as [3], [4], [5], [6], and sampling-based methods [7], [8], [1]. Both approaches have some key issues. To ensure the satisfaction of the CP constraints, approximate methods are prone to be over-conservative, preventing the robot from maneuvering in tight gaps. Conversely, sampling-based methods are often computationally intensive and do not scale well when reducing the CP constraint. This issue is critical in autonomous driving since it requires real-time decisions and, at the same time, very low CP values.

To tackle these issues, we introduce Deep Collision Probability Fields (DCPF), a neural-based approach for computing collision probabilities. The key idea of DCPF is to relegate a time-consuming Monte-Carlo estimate of the CP to the dataset generation phase while allowing for fast neural network evaluation of the constraints during planning. Our network design is inspired by Signed Distance Function (SDF), a technique extensively used to quickly compute distances between objects of arbitrary shapes. We reformulate the SDF approach in the setting of collision probabilities and, based on the ideas presented in [9], we introduce a novel inductive bias that enforces reasonable predictions even where the training data is sparse or non-existent, by gradually reducing the CP to zero as the distance between the two objects increases and forcing the CP to approach one if the distance is small. DCPF has three significant advantages. First, we can learn the CP from data under arbitrary probability densities. Second, we can create smooth and differentiable distance fields, a particularly desirable property for trajectory optimization. Finally, using deep neural networks allows for

† Equal contribution, ‡ Equal supervision
[1] Computer Science department, TU Darmstadt
[3] Hessian.AI
[4] German Research Center for AI (DFKI), Research Department: Systems AI for Robot Learning

fast and parallel querying of data points, combining the strengths and avoiding the weaknesses of the approaches presented in the literature.

In extensive experiments, we show that Deep Collision Probability Fields can produce reasonably accurate collision probabilities (up to 10^{-3}) for planning, and we demonstrate the effectiveness of our approach in many different simulated path planning tasks under uncertainty, in terms of computational time and CP accuracy. We conclude the paper with validation in the real world in a navigation scenario with two TIAGo mobile robots.

A. Related Work

Prior works that studied CP constraints in the context of path planning can be classified along several orthogonal dimensions. The most important are (1) the method used to compute the CP, namely, sampling-based or approximated, (2) the type of uncertainty (Gaussian or more complex), (3) where uncertainty is assumed to be present (robot, obstacles, or both), and (4) whether the CP constraint is enforced on a single trajectory step, or along the full trajectory. Due to abundant literature on the subject, only a few approaches are discussed below. We refer to [6] for a more comprehensive overview.

Lambert et al. [7] assume Gaussian uncertainty on both the robot's and obstacles' pose, and use Monte Carlo sampling to compute step-wise collision probabilities which, in turn, allow to compute a safe speed profile. More recently, Schmerling and Pavone [8] considered uncertainty in the robot's execution of its nominal trajectory, and use Monte Carlo with importance sampling to compute the trajectory CP. Banfi et al. [1] consider complex, potentially multi-modal obstacles' uncertainties that might arise in adversarial contexts, and propose a method based on the Sequential Ratio Probability Test (SPRT) to compute CPs for partial trajectories obtained during planning. A common advantage of these sampling-based methods is that they naturally come with theoretical guarantees, as it is easy to identify precise standard errors, confidence intervals, etc., of the computed CPs. However, a common downside of these methods is that they are computationally intensive, and might require thousands of samples to give reasonably precise estimates of small CPs. While in this paper we leave the investigations of theoretical guarantees for future work, we remark that our approach could already be used to identify promising regions of the planning space on which such guarantees could be derived via sampling.

Other approaches for CP estimation consider different types of approximations to reduce the computation time. Bry and Roy [3] assume the uncertainty on the robot's pose to be Gaussian and use the approximation of checking the ellipse defined by the covariance matrix and a desired chance bound for enforcing step-wise CP constraints. A similar approach is used by Kamel et al. in [5]. Hardy and Campbell [2] consider Gaussian uncertainty on the obstacles' positions and use rectangular bounding boxes for both the robot and the obstacles to obtain a CP upper bound. Thomas et al. [6] consider Gaussian uncertainty on both the robot's pose and obstacles and approximate their shapes as ellipsoids. This allows us to formulate the collision condition as a distance between ellipsoids. These methods are very fast, but their conservative nature might prevent the robot from finding feasible paths that are both safe and low-cost.

B. Problem Formulation

We consider a mobile robot operating in a n-dimensional environment specified by a bounded region $X \subset \mathcal{R}^n$. We assume X is composed of a static untraversable region $X_{\text{obs}} \subset X$ —known without uncertainty— and free space $X_{\text{free}} \subseteq X$, with $X_{\text{free}} \cap X_{\text{obs}} = \emptyset$. Let $O = \{o^k\}$ be a set of k independent obstacles. Each obstacle is completely described by the tuple $o = \langle \mathcal{C}_o, p_o(\boldsymbol{q}_o), \mathcal{V}_o(\boldsymbol{q}_o) \rangle$ where $\mathcal{C}_o \subset \mathbb{R}^n$ is the space of all possible object configurations —such as position, orientation, and lengths— describing the properties of the obstacle, $p_o(\boldsymbol{q}_o)$, is a probability distribution of each obstacle configuration $\boldsymbol{q}_o \in \mathcal{C}_o$, representing the uncertainty of the perceived obstacle, and $\mathcal{V}_o(\boldsymbol{q}_o)$ is the set of points occupied by the obstacle assuming that the true obstacle configuration is $\boldsymbol{q}_o$. We assume that the robot's position is known without uncertainty. Therefore, the robot can be defined as the tuple $r = \langle \mathcal{C}_r, \mathcal{V}_r(\boldsymbol{q}_r) \rangle$, where $\mathcal{C}_r$ denotes the configuration space of the robot and $\mathcal{V}_r(\boldsymbol{q}_r)$ denotes the set of points occupied by the robot at configuration $\boldsymbol{q}_r \in \mathcal{C}_r$. In this paper, we focus on the special case where the robots and the obstacles are 2D rectangles with sides l_1, and l_2, with isotropic Gaussian uncertainty. Under these assumptions, the obstacles can be described by the following parameter vector $\boldsymbol{q}_o = [x, y, \phi, l_1, l_2]^T$ with $\boldsymbol{q}_o \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

The collision probability $p_{\text{coll}}(r, o)$ between robot r and obstacle o is defined as

$$p_{\text{coll}}(r, o) = \int_{\mathcal{D}} p_o(\boldsymbol{q}_o) d\boldsymbol{q}_o, \tag{1}$$

where $\mathcal{D} = \{\boldsymbol{q}_o \,|\, \boldsymbol{q}_o \in \mathcal{C}_o, \mathcal{V}_r(\boldsymbol{q}_r) \cap \mathcal{V}_o(\boldsymbol{q}_o) \neq \emptyset\}$. Our goal is to compute an approximated version of $p_{\text{coll}}(r, o)$ with a neural network, which we denote $\hat{p}_{\text{coll}}(r, o)$, and show that it can be used in several planning settings. To this aim, we define the following family of planning problems.

Let $\boldsymbol{q}_r(0)$ be the initial robot configuration and $\mathcal{Q}_G \subset X_{\text{free}}$ is the set of admissible goals states. Let $\mathcal{P}$ be the space of all possible paths. We assume that every path $\pi = [a_0, \ldots, a_T] \in \mathcal{P}$ is composed by a set of T motion primitives $a(t)$. Each motion primitive moves the robot from a configuration $\boldsymbol{q}_r(t)$ to a new configuration $\boldsymbol{q}_r(t + \Delta t)$. Finally, let $r(t)$ and $o(t)$ denote the tuples describing robot and obstacle at step t.

We consider planning problems of the following form

$$\begin{aligned} \arg\min_{\pi} \quad & c(\pi) = \sum_{a_t \in \pi} c(a_t) \\ \text{s.t.} \quad & 1 - \prod_i \left(1 - p_{\text{coll}}(r(t), o_i(t))\right) \leq p_{\max} \quad \forall t, \\ & \boldsymbol{q}_r(t) \in X_{\text{free}} \, \forall t, \quad \boldsymbol{q}_r(T) \in \mathcal{Q}_G, \end{aligned}$$

where $p_{\max} \in [0, 1]$ is the maximum CP allowed for the planned trajectory and $c(\pi)$ is a cost function, like path

length, computed by summing the cost of each motion primitive composing the path.

Notice that in this formulation we assume that the robot will be able to perfectly track the computed trajectory, but we do not make any assumption regarding the number of obstacles, their nature— static or dynamic —and the magnitude of their future uncertainty. For dynamic obstacles tracked with a Kalman filter, this could simply be obtained by applying the filter prediction step [2].

II. Deep Collision Probability Fields

Our goal is to approximate the collision probability between a robot and an arbitrary object efficiently by means of a neural network, hence, we introduce DCPF that exploit neural approximations to obtain rapidly accurate CP estimates. We make two modeling assumptions: first, we consider the coordinate frame defined in the obstacle's center of mass, instead of the world coordinate frame. Second, we assume a parametric form for the obstacle's probability distribution, $p_o(\boldsymbol{q}_o; \boldsymbol{\omega}_o)$ where $\boldsymbol{\omega}_o$ represents the distribution's parameters. For example, assuming Gaussian uncertainty, the distribution parameters are the covariance matrix entries, $\boldsymbol{\omega}_o = \boldsymbol{\Sigma}_o$. Notice that, given that the coordinate system is obstacle-centric, the obstacle distribution mean for position and rotation is zero. We assume that the distribution family is known and that the distribution parameters are estimated from a perception system. Thanks to these assumptions we can easily impose an inductive bias in the network's structure, forcing the probability of collision to smoothly decrease to zero, when the distance between the object and the robot increases, while making it close to one when the two objects overlap. This inductive bias, inspired by the one presented in [9], exploits the fact that sufficiently far from the object, both the shape and the object uncertainty are irrelevant, as the collision probability will drop to zero. This bias allows us to efficiently learn the probability field for any configuration while keeping the probability approximation smooth.

A. Network structure

Using the above-mentioned assumptions and implementing the distance-based inductive bias, we obtain the following network structure

$$\begin{aligned}\hat{p}_{\text{coll}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o) =& (1 - \sigma^1_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o)) \left((1 - \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o)) f_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o) \right) \\ &+ \sigma^1_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o), \end{aligned} \tag{2}$$

with the current robot configuration $\boldsymbol{q}_r$, the parameters of the obstacle density $\boldsymbol{\omega}_o$, and the vector of learnable parameters of the neural network $\boldsymbol{\theta}$. Furthermore, for $i \in \{1, 2\}$, we define two functions as

$$\sigma^i_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o) = \text{sigmoid}\left(s^i \alpha^i_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o) \cdot \left(\rho^i_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o) - ||\boldsymbol{x}||_2 \right) \right), \tag{3}$$

where $\rho^i_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o)$ defines a soft threshold for switching from a local approximated collision probability to the one predicted by the inductive bias (zero or one), $\alpha^i_{\boldsymbol{\theta}}(\boldsymbol{q}_r, \boldsymbol{\omega}_o)$ regulates the sharpness of the change between the two modes, and s^i is a sign multiplier, with $s^1 = 1$ and $s^2 = -1$. The first regularizer $\sigma^1_{\boldsymbol{\theta}}$ biases the output towards one when the query point is close to the obstacle. The second regularizer $\sigma^2_{\boldsymbol{\theta}}$ moves the value of p_{coll} closer to 0 when the robot is close to the obstacle.

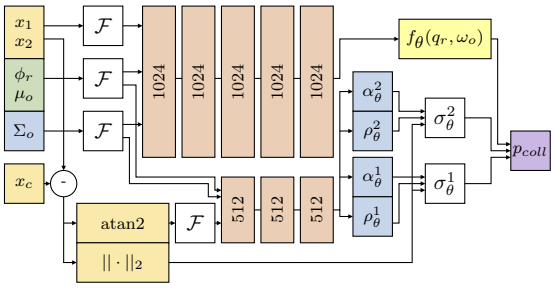

Fig. 2. The network structure of DCPF: input data is processed using Fourier features. The processed features are fed to a deep neural network with 5 fully connected 1024-dimensional hidden layers. An additional set of 3 fully connected 512-dimensional hidden layers compute the input for the shaping functions α and ρ, which influence the mode switching of the two regularizers $\sigma^1_{\boldsymbol{\theta}}$ and $\sigma^2_{\boldsymbol{\theta}}$, that implement the euclidean distance bias.

The architecture of DCPF, depicted in Fig. 2, is composed of two neural networks. The larger network consists of 5 fully connected hidden layers, each 1024-dimensional. This network takes the robot's position and configuration, and the obstacle's configuration and variance, each individually encoded with random Fourier features to compute f_θ. In addition, the smaller network uses 3 fully connected 512-dimensional hidden layers to compute α^i_θ and ρ^i_θ, the shaping parameters for the two regularizers. As input of this network, we exchange the positional information of the robot with just the angle from the position in polar coordinates, as the distance to the center $||\boldsymbol{x}||_2$ is directly used in the regularizer.

We use GeLU activations for all layers except for the output layers of the small network, which have their specific activation functions. In particular, we limit the range of $\alpha^1_{\boldsymbol{\theta}}$ and $\alpha^2_{\boldsymbol{\theta}}$ to the interval $[1, 21]$ by applying a sigmoid function and a bias on the output unit. $\rho^1_{\boldsymbol{\theta}}$ is bounded to the interval $[0, 12]$ and $\rho^{(2)}_{\boldsymbol{\theta}}$ is constrained to be positive, as the last layer is a softplus. The ranges discussed above are selected to give reasonable output values and ensure the stability of the training process in the autonomous driving setting. We also constrain the collision probability output $f_{\boldsymbol{\theta}}$ in the interval $[0, 1]$, using a sigmoid function.

Conservative CP estimates: Approximating a function with a neural network may lead to approximation errors, especially in areas far from the original dataset distribution. To reduce the chances of significant approximation errors, we learn an ensemble of DCPFs. Using this approach, we are able not only to provide a more accurate value of the collision probability but also to obtain a measure of the model prediction uncertainty. In the rest of our work, on top of the vanilla approach using a single neural network, we will exploit the ensemble technique in three different ways. The first option is to take the *mean*, where we average the prediction of multiple ensembles, reducing the error due to overfitting. Alternatively, we can take the *maximum* collision probability value, ensuring that the estimate we select is always conservative. This is particularly useful in case the

collision probability constraint bound is critical. Finally, we can exploit the measure of the prediction uncertainty by computing the 95% confidence interval of our prediction to build an upper confidence bound around the mean.

B. Data generation

We resort to a sampling-based method to generate an accurate dataset that does not leverage approximations to compute CPs. Specifically, since we are not bound by time constraints at this stage, we resort to Simple Monte Carlo sampling. Another key observation is that the desired accuracy of the CP estimate directly depends on the CP itself. For example, suppose that after having drawn 10k samples, our current CP estimate is .5; in this case, we would accept an error of .01, especially if our planning constraint $p_{\max}$ is .01 or .1. Following from this observation, we define the following probability intervals: $[0, .01)$, $[.01, .1)$, $[.1, 1]$. Each interval is then associated with a different desired accuracy for a Central Limit Theorem (CLT) based 95% confidence interval estimate[1]: $\pm.01$, $\pm.001$, $\pm 10^{-4}$. Therefore, we can stop drawing samples as soon as the current CLT-based 95% confidence interval falls below the accuracy associated with the current CP estimate. To speed up this procedure, we only recompute the CP estimate and associated confidence interval after having drawn a batch of samples. We use batches of $4{\cdot}10^4$ samples, and $4{\cdot}10^6$ maximum total samples, computed as the worst-case number of samples needed for the smallest probability interval.

Another issue to consider during the training is to sample properly the robot configurations to be evaluated. Indeed, if the dataset size is a concern, sampling uniformly may require a prohibitively large amount of samples to properly cover the areas where the estimation of collision probability is particularly sensible. We propose a sampling method based on the Minkowski Sum (MS) between the robot and the obstacle. The MS is the shape obtained by sliding the robot shape around the obstacle [11]. Our key idea is to compute a shape that resembles the isolines of the CP landscape, particularly close to the constraint budget level, and subsequently sample points from it. In a scenario without uncertainty, this shape can be trivially obtained by computing the MS between the robot and the obstacle. To take the uncertainties $\boldsymbol{\Sigma}_o = (\sigma_x, \sigma_y, \sigma_\phi, \sigma_{l_1}, \sigma_{l_2})$ into account we apply various strategies. For positional and shape variance we construct an inflated shape s_{inflated} by computing a MS between the shape of the obstacle s_o and an ellipse $s_e \sim N(\sigma_x + \sigma_{l_1}, \sigma_y + \sigma_{l_2} | (\sigma_x + \sigma_{l_1}, \sigma_y + \sigma_{l_2}))$. The effects of rotational variance are approximated by constructing the shape $s_{\text{rotational}}$, by computing the union of multiple s_{inflated} for rotated obstacles s_o. On top of that, we smooth the obtained shape by growing and shrinking $s_{\text{rotational}}$. The rotations are a discretization of the interval $[-\phi_m, \phi_m]$ with $\phi_m \sim \mathcal{U}(\sigma_\theta, 3.1 \cdot \sigma_\theta)$. In our experiment, we found that three rotations are sufficient, for this heuristic to result in sample configurations for a well-balanced dataset.

[1]See [10], Eq. (2.20), and the following paragraph for the special case of estimated CP = 0 or 1.

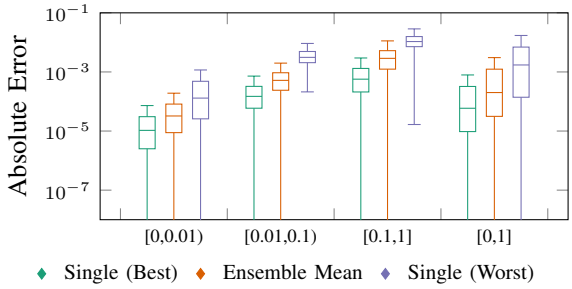

Fig. 3. Box plot of absolute error evaluated on test dataset

C. Training

We train our model by optimizing the following loss

$$\mathcal{L}(\mathcal{D}) = \sum_{\boldsymbol{q}_r, \boldsymbol{\omega}_o, \bar{p} \in \mathcal{D}} \mathcal{H}\left(\hat{p}_{\text{coll}}\left(\boldsymbol{q}_r, \boldsymbol{\omega}_o\right), \bar{p}\right) + \gamma \cdot \mathcal{R}\left(\boldsymbol{q}_r, \boldsymbol{\omega}_o\right), \quad (4)$$

where $\mathcal{H}$ is the binary cross-entropy between the CP prediction of the network, $\bar{p}$ the target value computed in the dataset, and $\mathcal{R}$ is a regularization term, weighted by a γ coefficient, composed by the sum of the following two terms

$$\mathcal{R}\left(\boldsymbol{q}_r, \boldsymbol{\omega}_o\right) = |\Delta\rho_\theta| + \sum_{i=\{1,2\}} \text{sigmoid}\left(s^i \alpha_\theta^i \cdot \Delta\rho_\theta / 2\right), \quad (5)$$

with $\Delta\rho_\theta = \rho_{\boldsymbol{\theta}}^2 - \rho_{\boldsymbol{\theta}}^1$ the difference of the mode switching parameters. The first term of the regularizer forces the network to switch between the regimes of data-driven collision probabilities to the inductive bias of zero or one probabilities as soon as possible, by bringing the thresholds $\rho_{\boldsymbol{\theta}}^i$ closer together. The second term enforces the thresholds to be in the right order by minimizing the influence of the bias in their center. The regularization term is particularly important in areas with low sample density.

III. Evaluation

We empirically evaluate DCPF in three sets of experiments. The first experiment evaluates the CP predictions obtained by DCPF on a dataset inspired by an autonomous driving scenario, examining the impact of number and size of network layers. In the second experiment, we use the best network obtained in the first experiment to tackle two simulated planning scenarios: one with static obstacles and one with dynamic obstacles. In the third experiment, we validate DCPF in the real world with two TIAGo robots (Fig. 9), where one acts as the agent and the other acts as a dynamic obstacle.

TABLE I
Percentage of predictions within confidence interval table for different Network Sizes

Network	$[0.0, 10^{-2})$	$[10^{-2}, 10^{-1})$	$[10^{-1}, 1]$	$[0.0, 1.0]$
3x128	0.666045	0.602677	0.906839	0.723093
3x512	0.407508	0.569119	0.888136	0.594182
4x512	0.757707	0.701193	0.939518	0.798590
4x1024	0.761130	0.667497	0.934856	0.790180
5x1024	0.770757	0.722915	0.946965	0.812132
6x1024	**0.779680**	**0.747732**	**0.951831**	**0.823841**
7x1024	0.757662	0.698077	0.947665	0.800255

A. DCPF Estimation of CP

We build a dataset of one billion samples, with an 80%-10%-10% training-validation-test split, ensuring an equal balancing across the different probability intervals (see Section II-B). We assume the robot dimensions to be fixed to a width of 4.07 and height of 1.74, while the $\boldsymbol{q}_o$ is obtained by drawing from a Gaussian distribution with $\boldsymbol{\Sigma}_o \in [0, \sqrt{2}]^5$. The dataset generation phase took 80 hours on a computer equipped with an AMD Ryzen 9 5950X 16-Core and an RTX 3080Ti (12 GB). We use the Adam optimizer for training with $2.4 \cdot 10^{-4}$ learning rate. We evaluate the network varying the number of hidden layers in 3 to 7 and the number of hidden layers neurons in 128, 512, and 1024, while the network for the regularized was kept fixed to 3 layers of 512 neurons. For every setting, we use an ensemble of 10 networks and we evaluate the average prediction. We consider two metrics, the Mean Absolute Error (MAE) and the Percentage of Accurate Predictions (PAP). The MAE measures the absolute average error of the network prediction over the full dataset. Since our dataset samples are approximations we also use PAP to measure the percentage of accurate predictions, where we consider a prediction as accurate if it is within the confidence interval of the collision probability estimate.

In Table I, we report the ablation study on the network size using the PAP as metric. We obtain accurate predictions for each network size and the network precision improves increasing the number of neurons. However, increasing the number of layers does not have a consistent positive impact. Due to the results of our ablation, we select a network with 6 hidden layers of 1024 neurons for the subsequent experiments.

In Figure 3, we evaluate the performance of our selected network by reporting the boxplots of absolute errors for different CP buckets. The results clearly show that our prediction error for the best network is very accurate, falling most of the time in the desired bucket accuracy. By looking at the worst prediction of each ensemble, we notice a significant drop in the accuracy. However, the presence of these outliers in the prediction is strongly mitigated by using the ensemble mean, which brings the distribution close to the best network, indicating that these errors are not very frequent in the dataset. This proves that, by using an ensemble technique, we provide highly accurate CP computations, ensuring the safety of the planning algorithm. It is possible to enforce this safety guarantee with high probability by using a confidence interval to provide a conservative estimate of the network.

TABLE II
INFERENCE COMPUTATION TIME PER SAMPLE

Method	Mean Time [ms]	Std Time [ms]	Max Time [ms]
SPRT $p_{\max} = 0.1$	0.558	0.589	19.123
SPRT $p_{\max} = 0.01$	0.900	0.743	12.392
SPRT $p_{\max} = 0.001$	9.378	7.227	114.030
z-Test $p_{\max} = 0.1$	0.918	25.520	1761.018
z-Test $p_{\max} = 0.01$	3.634	49.207	2409.020
z-Test $p_{\max} = 0.001$	68.661	598.436	8554.228
Network CPU $b = 1$	14.770	3.477	122.731
Network GPU $b = 1$	2.636	13.067	415.570
Network CPU $b = 16$	1.523	0.299	10.774
Network GPU $b = 16$	0.181	0.908	28.870
Network CPU $b = 1024$	0.718	0.007	0.878
Network GPU $b = 1024$	0.011	0.013	0.412

To evaluate the performance in terms of computation time, we compare our approach with the Sequential Probability Ratio Test (SPRT) and the Z-Test methods [1] to ensure the CP is below a set of fixed constraints. We use $p_{\max} \in \{10^{-1}\, 10^{-2}, 10^{-3}\}$ and a sample function based on the separating axis theorem. The CP distribution is approximately reciprocal. For every approach, we test the performance evaluation on the CPU using a single configuration, while for DCPF we also evaluate the performance with batch sizes 1, 16, and 1024, in GPU and CPU. The computation time reported is the computation time per configuration, i.e., the total computation time divided by the batch size. Both Z-Test and SPRT sample a maximum of $4 \cdot 10^6$ times for each constraint respectively. Our results, presented in Table II, show that SPRT is a competitive method only when using a high CP collision probability, and together with the Z-Test provides a highly variant behavior in terms of computation time. Our approach has much more consistent computation times. It is worth noting that the mean computation time can be heavily affected by the outliers, particularly for lower CP budgets. Our method shines particularly with high batch sizes, making it suitable when checking collision probabilities of long trajectories or in combination with parallelized planners, e.g. [12], [13], [14].

B. Planning with DCPF

a) Static Obstacles: For this work we consider two settings. A a narrow passage scenario with two obstacles and a random obstacle planning setting.

In the first planning scenario, we consider a square workspace containing two static obstacles forming a narrow passage, as shown Fig. 4, with uncertainty described by

$$\boldsymbol{\Sigma}_1 = \mathrm{diag}(0.05, 0.2, 0.03, 0.0001, 0.0001)$$
$$\boldsymbol{\Sigma}_2 = \mathrm{diag}(0.15, 0.4, 0.13, 0.01, 0.015),$$

where Σ_i represents the uncertainty distribution of the i-th obstacle ${\boldsymbol{q}_o}^i = \left[x^i, y^i, \phi^i, l_1^i, l_2^i\right]^T$. The goal of the robot is to find a path leading to the goal region to the left of the workspace while accounting for a given CP constraint $p_{\max} \in \{10^{-1}, 10^{-2}, 10^{-3}\}$.

The planner assumes a bicycle model for the robot and plans using Hybrid-A* using a set of 10 motion primitives. To check the satisfaction of the CP constraint by evaluating if the sate is below the lower bound of the 95% confidence interval around the mean of 10 DCPF Networks. We consider two baselines to compare against our method. The first is a common Simple Monte Carlo baseline [7], also used in [8], [1]. Specifically, we compute the CP estimate via Simple Monte Carlo and use a one sided z-test to check if it can be concluded, with 95% confidence, that the CP constraint is not violated. The second baseline is the SPRT approach introduced in [1]. In both cases, we run experiments for $p_{\max} \in \{10^{-1}, 10^{-2}, 10^{-3}\}$, and examine the impact

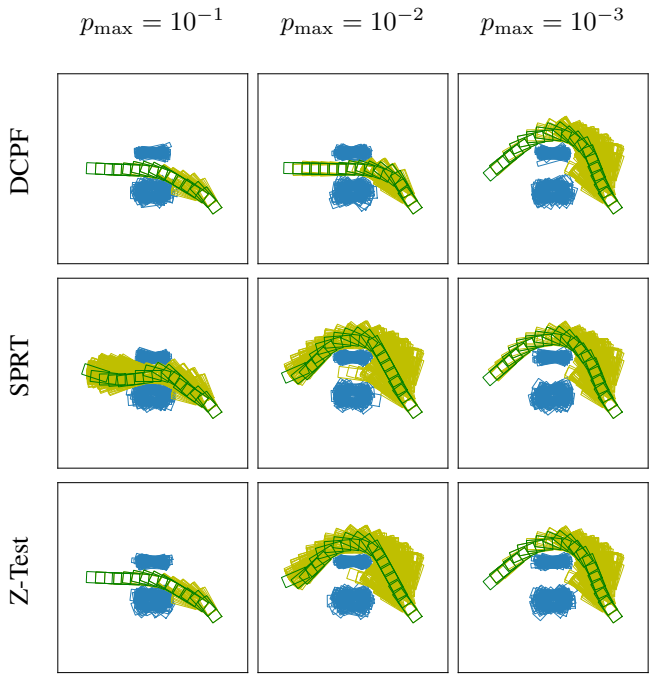

Fig. 4. Comparison of paths generated using A* using different sampling algorithms. Blue rectangles represent obstacles sampled from the obstacle distribution, while the other rectangles represent a configuration of the robot checked by the planner. The configurations chosen as part of the solution are marked in dark green. SPRT samples a maximum of 4×10^6 and the Z-Test uses 10^5 samples at most.

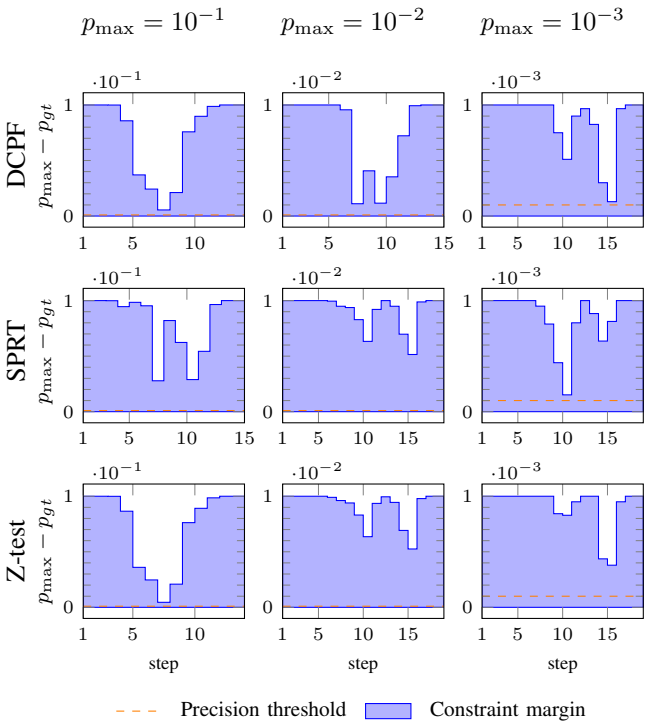

Fig. 5. The distance of the CP constraint $p_{\max}$ to the CP p_{gt}, computed with the CLT-based method outlined in II-B, during the planned trajectories seen in figure 4. More conservative methods will have a higher minimal distance. For values below the precision threshold it cannot be verified whether the constraint is respected, due to the uncertainty of p_{gt}

of using a limited number of samples for testing. In the first scenario we use a maximum of $4 \cdot 10^6$ samples for the SPRT baseline and tested different sample maxima of $\{10^2, 10^3, 10^4, 10^5, 10^6, 4 \cdot 10^6\}$ for the Z-Test.

From Figure 4 we can see that the trajectory computed by the planner changes the homotopy class for collision probabilities smaller than 10^{-3}. Instead, both Z-test and SPRT change the homotopy class for lower constraints. This happens as both methods will mark a state as unsafe if they cannot decide with the given sample budget if the CP constraint is respected. This behavior is highlighted in Figure 5, showing that our method gets closer to the constraint, while not violating it, whereas the other methods stay further away, not allowing the car to pass through the narrow gap.

We further analyze the behavior of our algorithm by computing the difference $\hat{p}_{\text{coll}} - \bar{p}$, where $\bar{p}$ is the ground truth value, computed using the z-test with an unlimited amount of samples. The results, presented in Figure 6 show that our approach in general computes accurate collision probabilities while finding a less conservative path than Z-test with limited computation. Indeed, the prediction mostly falls into the confidence interval of the ground truth, meaning

TABLE III
RANDOM OBSTACLES IN MAP EXPERIMENT RESULTS

Metric	DCPF	z-Test $n = 1e6$	SPRT $n = 4e6$
No solution	23	47	30
Path duration [s]	26.133 ±6.651	26.133 ±6.651	26.101 ±6.680
Planning Time[s]	335.220 ±359.047	6764.291 ±5182.787	1223.666 ±869.418

we cannot distinguish the error from noise in the data generation process. In the experiments, our approach always outputs conservative CP estimates thanks to the confidence interval upper bound, reflected in a positive error in the CP estimate when the error is outside the ground truth confidence interval. This result is particularly important to provide the required level of safety.

We also test our approach in a realistic scenario and present the results in Table III. In this second experiment, we consider 10 different cleaned-up occupancy maps of real-world environments [15], [16], [17] populated with 120 random obstacles per free $100\,\text{m}^2$. Width and height are between $0.1\,\text{m}$ and $3\,\text{m}$. The uncertainties are drawn uniformly between 0.001 and 0.1. The start and goal position are chosen, so that the distance without considering the map

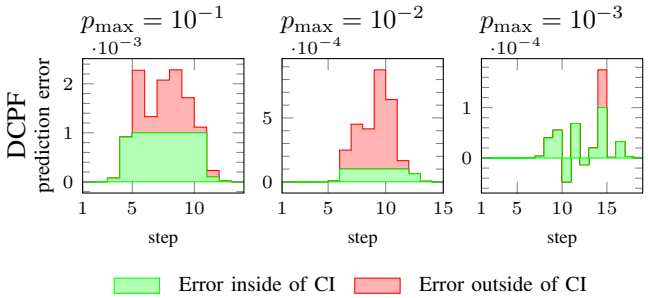

Fig. 6. Error of CP estimate during the planned trajectory for DCPF in figure 4 The green bar represents an estimation error inside the confidence interval of the ground truth calculation, while the red bar represents an error outside the confidence interval.

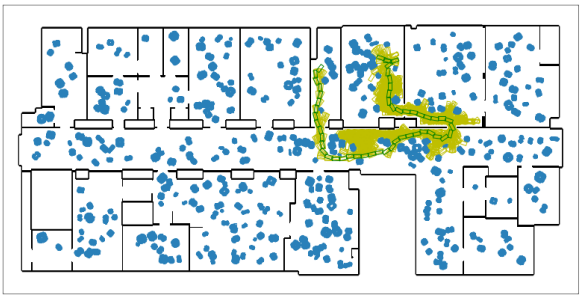

Fig. 7. Path found in between random obstacles in an occupancy map with hybrid A* using DCPF to ensure satisfaction of the CP constraint $p_{\max}$

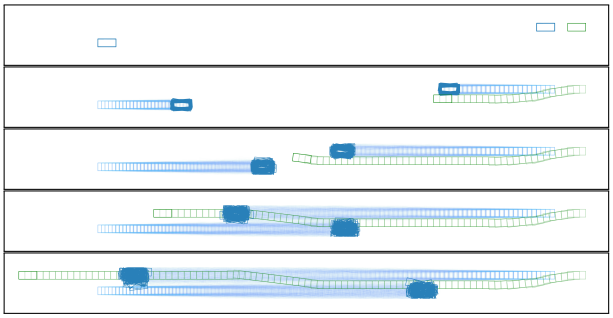

Fig. 8. Example of the paths computed during the dynamic overtake experiments

is between $35\,\mathrm{m}$ and $40\,\mathrm{m}$. We set the CP constraint $p_{\max}$ to 10^{-3} and compare our method against s-Test and SPRT baselines in 10 random configurations per map. The sample maxima are set to 10^6 for the z-Test and 4×10^6 for the SPRT baseline. Due to the set computing time limitations the z-Test and SPRT baselines find fewer solutions. The mean and standard deviation of the path length and planning time are only computed for the paths, where all of the methods found a solution. Also here the DCPF approach outperforms the baselines in all metrics.

b) Dynamic Obstacles: To test the performance with dynamic obstacles we implemented a time-dependent version of Hybrid-A*, and we tested our approach in a dynamic overtake setting, depicted in Fig 8. The objective of the task is to reach the end of the lane as soon as possible, by overtaking a car running in front of the controlled vehicle, while another car, driving in the opposite lane, is approaching. The position of the non-controlled car is known with uncertainty. The uncertainty grows over time as it would happen if a Kalman filter computes the prediction.

We test this challenging planning task under CP constraint $p_{\max} \in \{.1, .01, .001\}$, and we provide the results in Table IV. In this table, "overtake before" and "overtake after" refer

TABLE IV
OVERTAKE EXPERIMENT RESULTS

$p_{\max}$	10^{-1}	10^{-2}	10^{-3}
Overtake before	29	29	27
Overtake after	21	18	18
No overtake	0	3	5
Violations	0	0	0
MAE $(\cdot 10^{-3})$	0.1452 $\pm$0.0006	0.0187 $\pm$0.0001	0.0031 $\pm$0.0000

Fig. 9. TIAGo robot performing an overtake by planning under uncertainty and collision probability bounds

to an overtake happening, respectively, before or after the car riding on the opposite lane reaches the one in front of the agent. The results show a clear effect on the planned overtake for different CP constraints. Lowering the budget forces the car to wait for the other car to pass by before the overtake is complete more often, as a result of a bigger safety margin. For the same reasons, the number of episodes where no overtake happens increases when tightening the CP constraint. These results show that the proposed task is quite challenging for the CP evaluation. Despite the difficulty of the task, our method provides an accurate estimate of CP for every $p_{\max}$ level and, in the experiment setting, we have no constraint violations. This accurate estimation can also be seen by inspecting the values of the mean absolute error and the mean violation, which are also extremely low.

To improve the relevance of our evaluation, we again use 10 different occupancy maps populated with obstacles moving in rectangular paths or back and forth. For the starting and goal positions, 250 configurations are chosen, such that a significant portion of the map has to be traversed and various moving obstacles have to be avoided. We set the CP constraint $p_{\max}$ to 10^{-3} and the maximum sample counts for the z-Test and SPRT baselines to 4×10^6. Table V shows the results, which show that while the baselines seem to find shorter paths, the planning time is significantly shorter.

c) Real robot experiments: We also provide a small proof-of-concept experiment, where we deploy our approach on a real robot setup. Here we implement the overtake experiment using two TIAGo robots, as shown in Figure 9. The TIAGo robot that is running in front of the controlled one is changing the width of his arms, enlarging his footprint, making his bounding box uncertain in every instant. To allow for an easy deployment and exploit the advantages of the

TABLE V
DYNAMIC OBSTACLES IN MAP EXPERIMENT RESULTS

Metric	DCPF	z-Test $n = 4e6$	SPRT $n = 4e6$
No solution	151	192	158
Path duration [s]	173.668 $\pm$124.989	146.848 $\pm$81.375	159.187 $\pm$102.153
Planning Time[s]	1342.551 $\pm$3181.698	6245.175 $\pm$6432.084	2721.024 $\pm$4905.856

parallelization coming from our method, we implement a simple motion-primitive-based planner. These motion primitives are pre-generated spline-based overtake trajectories, followed by a simulated bicycle drive controller to estimate the timestamp at each point. We then evaluate them at different discretization levels to quickly narrow down options while ensuring that the chosen trajectory does not violate the CP constraint. The algorithm runs real-time on the system and the overtake happens without any collision happening. In future works, we expect to deploy DCPF planners in more complex robotics setups with ease.

IV. Conclusion

In this paper, we introduced DCPF, a neural approach to compute efficiently CP. Differently from sampling-based approaches, DCPF shifts the heavy part of the computation offline, allowing for a fast, time-consistent, yet very accurate, evaluation of CP during planning. Our method can easily incorporate obstacle uncertainty in many different settings, including dynamic environments and shape uncertainty. Furthermore, DCPF provides a differentiable representation of the collision probability, allowing for an easy combination with trajectory combination methods. Additionally, to increase the applicability in safety-critical settings, we present an ensemble approach to deal with model prediction uncertainty and provide conservative CP estimations.

Our experiments show that DCPF can speed up the CP evaluation, particularly in the setting of parallelizable planning algorithms. As the inference time of the neural network is consistent, we are free from the need to specify a planning time budget, which may cause the standard method to wrongly label safe configurations. Our planning experiments show that DCPF provides a better evaluation of the safety of the states than the sampling-based methods with a limited computation budget. Finally, we show that we can deploy our approach in the real world, by performing a real-world overtaking task using two Tiago robots, under perception and dynamic uncertainty. While we focus on the simple setting of rectangular obstacles (crucial for autonomous driving), DCPF can be easily extended to support arbitrary shapes, as it only requires generating a proper dataset. This includes non-convex shapes that are particularly computationally intensive for sample-based CP estimation methods.

In future works, we will investigate the applicability of our approach in more challenging settings and we will combine our method with state-of-the-art parallel planning algorithms: this would allow us to reduce massively the planning time, allowing methods of planning under uncertainty to scale to complex real-world scenarios.

ACKNOWLEDGMENT

This researcher has been supported by the BMBF collaborative project KIARA (grant no. 13N16274), and the the DFG EN Project iROSA (CH 2676/1-1). The authors acknowledge the use of the high-performance computer Lichtenberg at the NHR Centers NHR4CES at TU Darmstadt. This is funded by the Federal Ministry of Education and Research, and the state governments participating based on the resolutions of the GWK for national high-performance computing at universities.

We sincerely thank Sophie Lüth for her help and technical assistance with the real-world deployment, and Matteo Luperto for providing evaluation datasets.

References

[1] J. Banfi, Y. Zhang, G. E. Suh, A. C. Myers, and M. Campbell, "Path Planning Under Malicious Injections and Removals of Perceived Obstacles: A Probabilistic Programming Approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6884–6891, Oct. 2020.

[2] J. Hardy and M. Campbell, "Contingency Planning Over Probabilistic Obstacle Predictions for Autonomous Road Vehicles," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 913–929, Aug. 2013, conference Name: IEEE Transactions on Robotics.

[3] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 723–730.

[4] A. Lee, Y. Duan, S. Patil, J. Schulman, Z. McCarthy, J. Van Den Berg, K. Goldberg, and P. Abbeel, "Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5660–5667.

[5] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 236–243.

[6] A. Thomas, F. Mastrogiovanni, and M. Baglietto, "Exact and bounded collision probability for motion planning under gaussian uncertainty," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 167–174, 2021.

[7] A. Lambert, D. Gruyer, G. S. Pierre, and A. N. Ndjeng, "Collision Probability Assessment for Speed Control," in *2008 11th International IEEE Conference on Intelligent Transportation Systems*, Oct. 2008, pp. 1043–1048, iSSN: 2153-0017.

[8] E. Schmerling and M. Pavone, "Evaluating Trajectory Collision Probability through Adaptive Importance Sampling for Safe Motion Planning," Jun. 2017, arXiv:1609.05399 [cs].

[9] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized deep signed distance fields for reactive motion generation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6673–6680.

[10] A. B. Owen, *Monte Carlo theory, methods and examples*. unpublished, 2013, sec. 2.4.

[11] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 1192–1198, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/6386125

[12] A. Lambert, A. Fishman, D. Fox, B. Boots, and F. Ramos, "Stein variational model predictive control," *arXiv preprint arXiv:2011.07641*, 2020.

[13] A. T. Le, G. Chalvatzaki, A. Biess, and J. Peters, "Accelerating motion planning via optimal transport," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[14] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1916–1923.

[15] E. Whiting, J. Battat, and S. Teller, "Topology of urban environments," in *Computer-Aided Architectural Design Futures (CAAD-Futures) 2007: Proceedings of the 12th International CAADFutures Conference*. Springer, 2007, pp. 114–128.

[16] A. Aydemir, P. Jensfelt, and J. Folkesson, "What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4675–4682.

[17] F. Amigoni, V. Castelli, and M. Luperto, "Improving repeatability of experiments by automatic evaluation of slam algorithms," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7237–7243.

APPENDIX

A. Max samples computation

In Section II-B we set our worst-case maximum total samples to $4 \cdot 10^6$. We compute this number as the infimum of the Gaussian test

$$n \geq 1.96^2 \cdot \frac{p \cdot (1-p)}{\epsilon^2}$$

with

$$\epsilon = \begin{cases} 0.0001 & \text{for} \quad 0 \geq p > 0.01 \\ 0.001 & \text{for} \quad 0.01 \geq p > 0.1 \\ 0.01 & \text{for} \quad 0.1 \geq p \geq 1 \end{cases}$$

where ϵ refers to the accuracy associated with the probability bin. Using these values, the highest lower bound that needs to be fulfilled is at $n \geq 3.803.183$ for the accuracy ϵ of 0.0001.

B. Ablation studies

This section presents the results of the ablation studies performed in the network. Specifically, we ablate the dataset size showing the training and validation loss in Fig. 10 and the MAE in Fig. 12. We also ablate the network size, showing in Fig. 11 the training and validation loss and the respective MAE in Fig. 13. results show that our method is sufficiently reliable even with smaller dataset sizes and that it is not too dependent on the network size, provided sufficient network capacity.

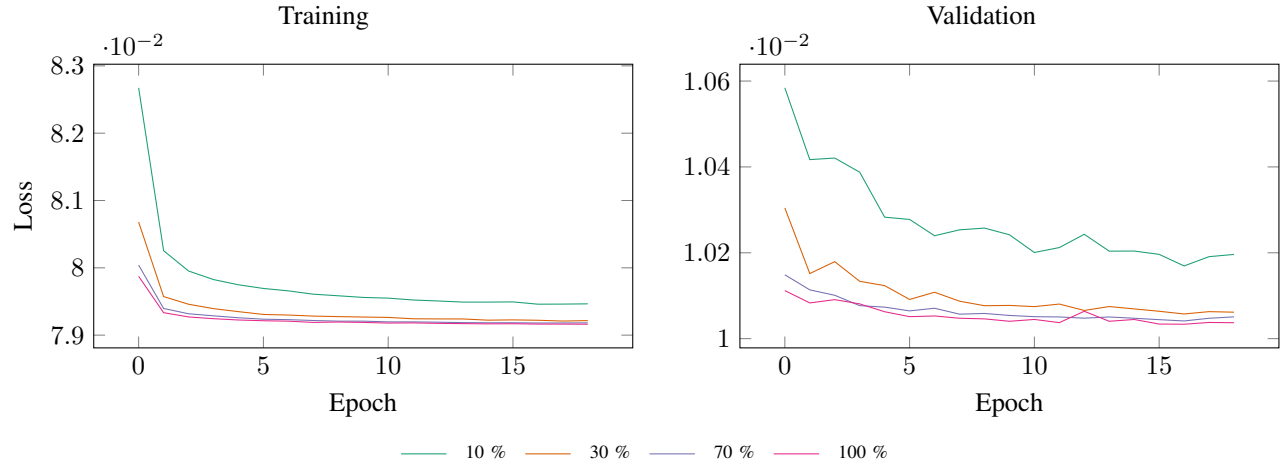

Fig. 10. Loss for different dataset sizes

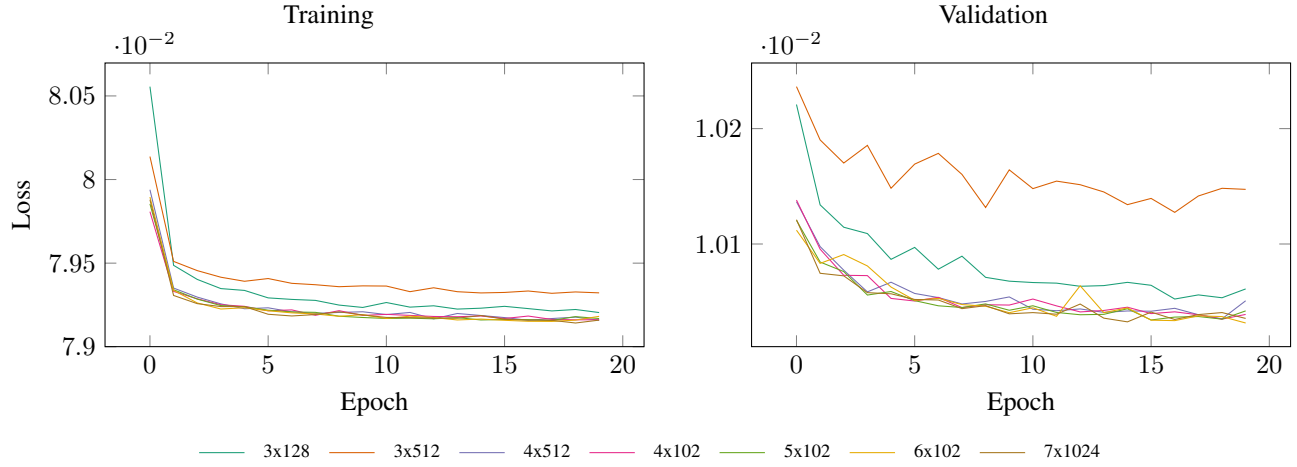

Fig. 11. Loss for different network sizes

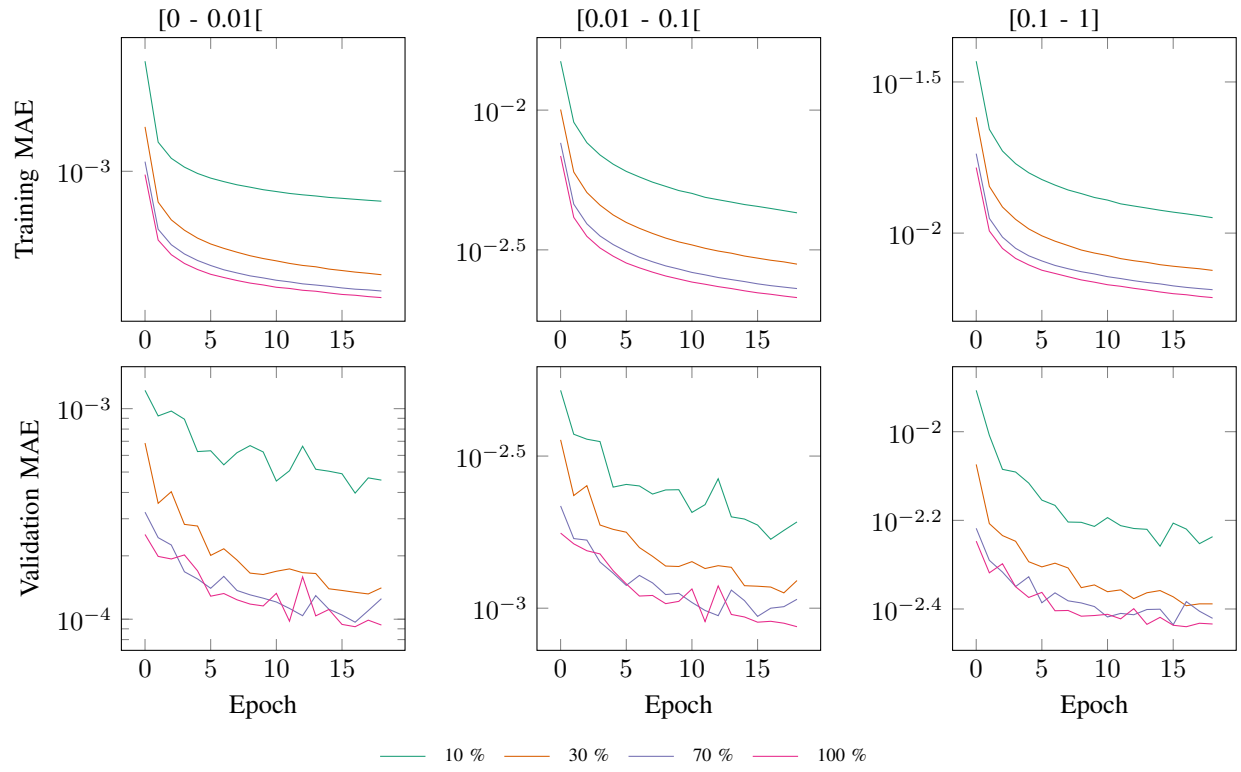

Fig. 12. MAE on train and validation dataset for different dataset sizes

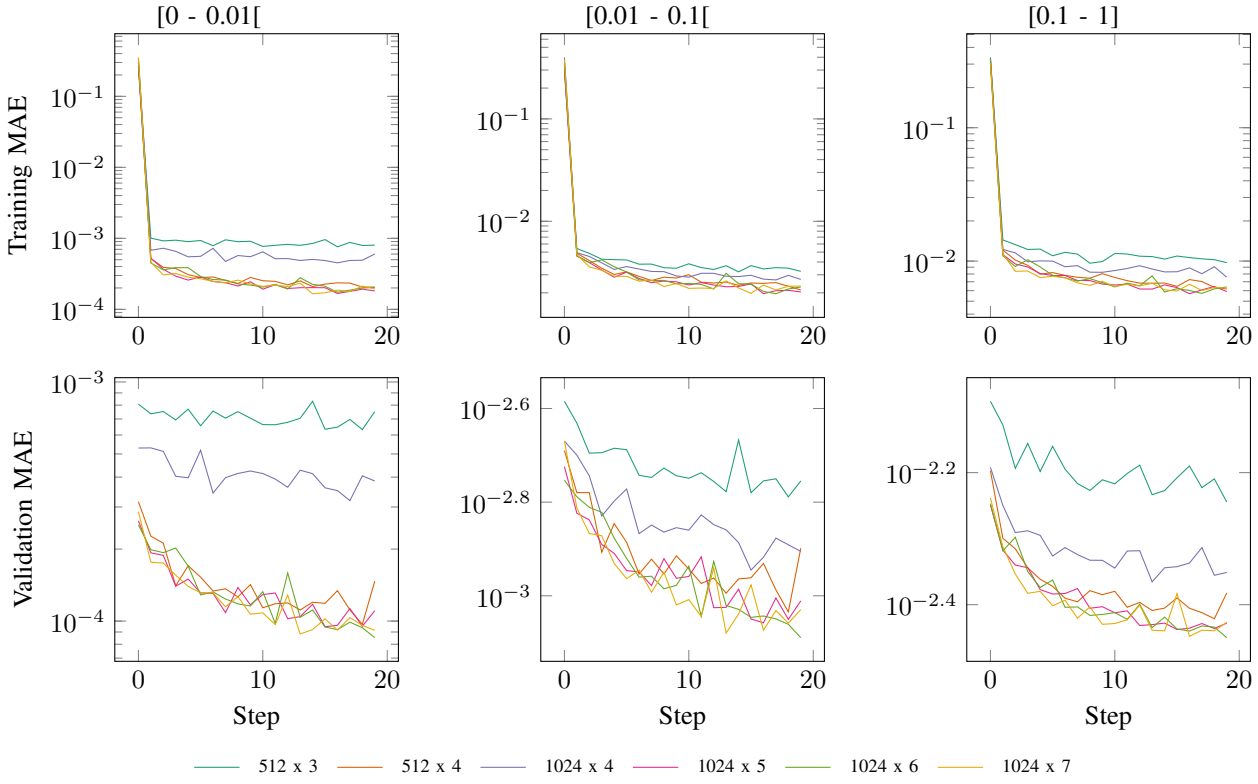

Fig. 13. MAE on train and validation dataset for different network sizes

C. Additional details from experiments

In this section, we report additional details from the experimental section presented in the main paper. Figure 14 presents the runtime experiments as a boxplot. Tables VI, VII and VIII instead report more detailed results for the random obstacle experiment for different levels of $p_{\max}$ with 50 obstacles in an otherwise empty square map of $50\,\mathrm{m}$ by $50\,\mathrm{m}$. Notice that the computational advantages of SPRT vanishes for very low CP budgets.

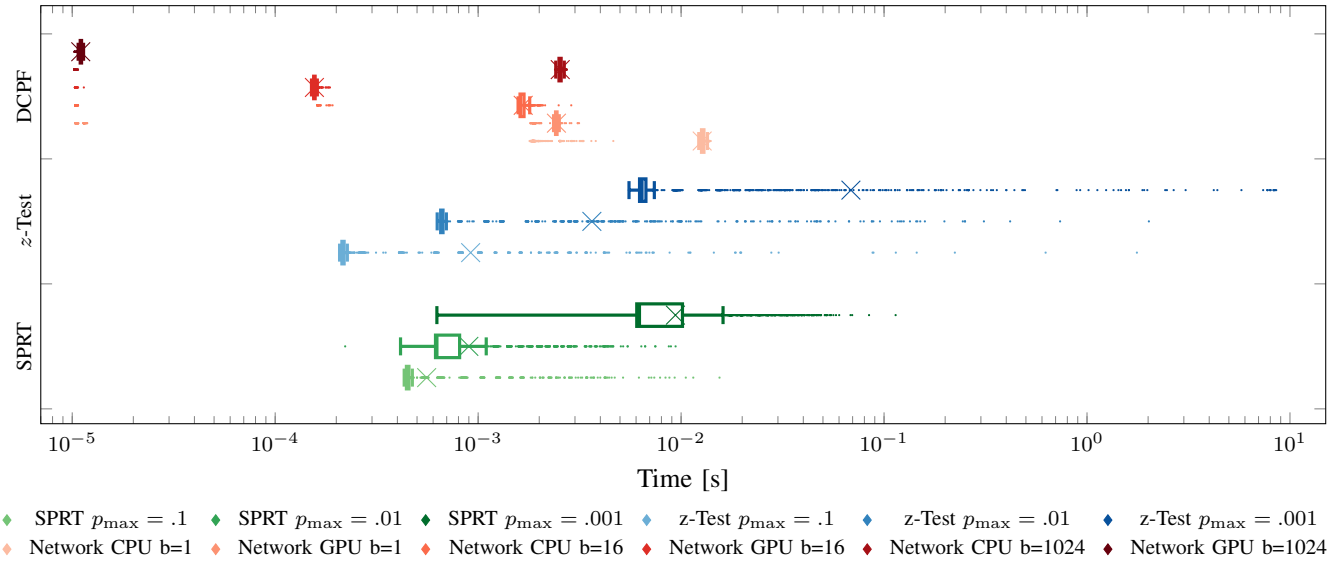

Fig. 14. Box plot of inference computation time (per sample), where b is the number of points evaluated in parallel and p_{max} is the probability constraint. If not otherwise specified b=1. The distribution of CP is approximately reciprocal. In the plot, outliers are represented by points and the mean of the distribution is represented by an x mark.

TABLE VI
RESULTS FROM THE RANDOM OBSTACLE EXPERIMENTS FOR $p_{\text{MAX}} = 0.1$

	DCPF	z-Test $n = 1000$	z-Test $n = 10000$
No solution found	24	24	24
Mean path length [m]	32.6711	32.5658	32.6579
Std path length [m]	4.2901	3.9714	4.1724
Mean time [s]	106.2414	20.4188	131.9221
	z-Test $n = 100000$	z-Test $n = 1000000$	SPRT $n = 4000000$
No solution found	25	25	24
Mean path length [m]	32.4400	32.4400	32.8289
Std path length [m]	3.8201	3.8201	5.0923
Mean time [s]	1277.1769	4890.3260	20.0671

TABLE VII
RESULTS FROM THE RANDOM OBSTACLE EXPERIMENTS FOR $p_{\text{MAX}} = 0.01$

	DCPF	z-Test $n = 1000$	z-Test $n = 10000$
No solution found	24	24	24
Mean path length [m]	33.1316	33.0133	33.1200
Std path length [m]	4.8050	4.9652	5.2712
Mean time [s]	121.3385	35.7087	110.7654
	z-Test $n = 100000$	z-Test $n = 1000000$	SPRT $n = 4000000$
No solution found	25	25	19
Mean path length [m]	33.1333	33.1333	33.9753
Std path length [m]	5.2823	5.2823	6.0878
Mean time [s]	907.5650	3499.4843	39.4890

TABLE VIII
RESULTS FROM THE RANDOM OBSTACLE EXPERIMENTS FOR $p_{\text{MAX}} = 0.001$

	DCPF	z-Test $n = 1000$	z-Test $n = 10000$
No solution found	21	23	23
Mean path length [m]	33.7089	33.6234	33.4286
Std path length [m]	5.3205	5.3623	5.4877
Mean time [s]	140.6172	284.8185	327.5759
	z-Test $n = 100000$	z-Test $n = 1000000$	SPRT $n = 4000000$
No solution found	21	21	22
Mean path length [m]	33.7342	33.7342	33.2692
Std path length [m]	5.9058	5.9058	4.4338
Mean time [s]	1090.1761	3434.1721	269.8011