This article has been accepted for publication in IEEE Systems, Man, and Cybernetics Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LSMC.2026.3731283

Learning Convex Probabilistic Obstacle Spaces for Efficient Uncertainty-Aware Path Planning

Jun Xiang, Jun Chen, *Senior Member, IEEE*

***Abstract*—Identifying the obstacle space is crucial for path planning. However, generating an accurate obstacle space remains a significant challenge due to various sources of uncertainty, including motion, behavior, and perception limitations. Even though an autonomous system can operate with an inaccurate obstacle space by being over-conservative and using redundant sensors, a more accurate obstacle space generator can reduce both path planning costs and hardware costs. Existing generation methods that generate high-quality output are all computationally expensive. Traditional methods, such as filtering, sensor fusion and data-driven estimators, face significant computational challenges or require large amounts of data, which limits their applicability in realistic scenarios. In this paper, we propose leveraging neural networks, commonly used in imitation learning, to mimic expert methods for modeling uncertainty and generating confidence regions for obstacle positions, which we refer to as the probabilistic obstacle space. The network is trained using a multi-label, supervised learning approach. We adopt a fine-tuned convex approximation method as the expert to construct training datasets. After training, given only a small number of samples, the neural network can accurately replicate the probabilistic obstacle space while achieving substantially faster generation speed. Moreover, the resulting obstacle space is convex, making it more convenient for subsequent path planning.**

***Index Terms*—Path planning, obstacle space, uncertainty, neural networks**

I. Introduction

AUTONOMOUS vehicles and robots operations face various uncertainties and hazards [1], [2]. One of the most important tasks of path planning is to move without collisions, in other words, to remain within a safe space at all times [3]–[6]. Safe space is also called safe region [7], obstacle-free space [8], free space [5] in other works. The first step to find a safe space is usually to find the obstacle space. While many methods exist for generating a safe space under the assumption of a known obstacle space [9], this assumption is often unrealistic in practice, and only a few approaches [10]–[12] extend to uncertain obstacles. Meanwhile, many previous path planning algorithms assume the obstacle space is polygonal [13], [14] due to the simplicity of collision checking and the compactness of the representation. The area of the obstacle space should be minimized, as a larger safe space typically results in smoother and shorter trajectories while also reducing the overall planning cost [8], [15]. Building on these insights, this paper proposes a novel method to construct a polygonal *probabilistic obstacle space* that is both compact and capable of representing uncertain obstacles.

The authors are with the Department of Aerospace Engineering, San Diego State University, San Diego, CA 92182 USA. (emails: jxiang9143@sdsu.edu; jun.chen@sdsu.edu).

In practice, safety is ensured by declaring a collision whenever the distance between an agent and an obstacle falls below a threshold, even without physical contact [16]. In this work, we focus on obstacles with relatively small volumes, such as flights or UAVs in airspace [17], where the obstacle space is determined primarily by position rather than precise geometry, a common simplification known as the point mass model. An obstacle is certain if a deterministic value represents its position, and uncertain if a probability distribution describes its position. The probabilistic obstacle space is then defined as the confidence region of this distribution.

Obstacle uncertainty can arise from perception, behavior, and motion. Perception uncertainty is common because position-locating sensors, including GPS [18], cameras [19], LiDAR [20], and radar [21], are subject to measurement noise, making it necessary to construct probabilistic obstacle spaces from noisy measurements. Sensor Uncertainty Mitigation (SUM) methods [22] can bound these errors, but they have high computational costs and require diverse, representative data. Moreover, over-conservative approaches substantially reduce safe spaces, making it difficult for planning algorithms to find feasible solutions. Behavioral uncertainty can also make an obstacle uncertain because different intents may lead to entirely different trajectories [23]. Traditional data-driven methods [24], [25] can estimate intent distributions when sufficient historical data are available; however, these distributions are often highly complex and require many parameters, making them impractical to translate directly into obstacle spaces. Motion is another common source of uncertainty, as even obstacles following a fixed path can exhibit positional variability [4], [10], [26]. The M-estimator of Tukey [27] has been used to estimate obstacle motion from single-camera images, but it is computationally expensive and requires careful parameter tuning, while Gaussian mixture models [4], [28] are also widely used to model uncertain motion.

In this paper, we propose a supervised learning approach that employs multi-label training to train a transformer-based neural network for generating polygonal probabilistic obstacle spaces under uncertainty. Numerical experiments demonstrate that the network can reliably reproduce probabilistic obstacle spaces when the input samples are drawn from obstacle types it has learned. In addition, the proposed method achieves substantially faster generation times compared to baseline approaches. The method offers four key advantages:

- *Advantage 1*: The approach does not rely on dynamic models of uncertain obstacles. Instead, it can construct obstacle spaces from only a few noisy measurements or partial observations, reducing the sensing and modeling requirements.

This article has been accepted for publication in IEEE Systems, Man, and Cybernetics Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LSMC.2026.3731283

- *Advantage 2*: All generated obstacle spaces are convex and linear, which simplifies their integration into path planning algorithms and ensures computational tractability.
- *Advantage 3*: Training requires only a small amount of labeled data, which lowers the burden of data collection and annotation and makes the approach practical for large-scale deployment.
- *Advantage 4*: The generation process is extremely fast, enabling real-time use in online planning and decision-making systems.

II. Problem Statement

The main goal of this paper is to generate the *probabilistic obstacle space* of an uncertain obstacle from a *limited data sample* using a *neural network*:

$$o^{\alpha} = NN(w, \mathbf{s}), \tag{1}$$

where o^{α} is the probabilistic obstacle space at confidence level $\alpha \in (0, 1)$, w denotes the neural network parameters, and $\mathbf{s} = {s^i}_{i=1}^{n}$ is the set of observed n samples.

Definition (Probabilistic Obstacle Space). Let o^{true} be the true occupied region of an uncertain obstacle o, and let $\mathbb{P}$ denote the underlying probability measure that characterizes uncertainty in the obstacle's position. A set o^{α} is called a probabilistic obstacle space at confidence level α if it satisfies

$$\mathbb{P}[a \in o^{true} \mid a \notin o^{\alpha}] < 1 - \alpha. \tag{2}$$

That is, if the agent a avoids o^{α}, then the probability of entering the true obstacle region o^{true} is bounded above by $1 - \alpha$.

Since o^{true} is generally unknown due to imperfect perception and prediction, o^{α} provides a tractable confidence-region approximation. Our definition generalizes several prior notions, including the ϵ-shadow of obstacle spaces [10], [11], Gaussian-distributed obstacles [29], adaptive bounding boxes [30], and α-probabilistic occupied sets [31]. In the experiments, sample coverage is used as an empirical approximation of the conditional collision-risk objective in Eq. (2). It measures the probability that the true obstacle position lies inside the generated region under the available observations.

The neural network in Eq. (1) is thus trained to approximate the mapping

$$f : \mathcal{S}^n \to \mathcal{O}, \quad f(\mathbf{s}) \mapsto o^{\alpha}, \tag{3}$$

where $\mathcal{S}^n$ is the sample space of n measurements and $\mathcal{O}$ is the family of convex polygonal sets in $\mathbb{R}^d$.

We illustrate three representative examples of probabilistic obstacle spaces and their associated data samples.

Motion uncertainty: Consider the scenario of a moving object following a fixed planned trajectory. However, many factors, such as environmental disturbances, control uncertainties, and human factors, can introduce uncertainty in the actual trajectory. Therefore, a single planned trajectory may correspond to infinitely many possible realizations of motion, all of which the ego agent must avoid. To capture this variability, we construct a probabilistic obstacle space that encompasses the possible deviations. Formally, we define the sample set $\mathbf{s} = {s^i}_{i=1}^{n}$ as the trajectories of n flights following the same planned path.

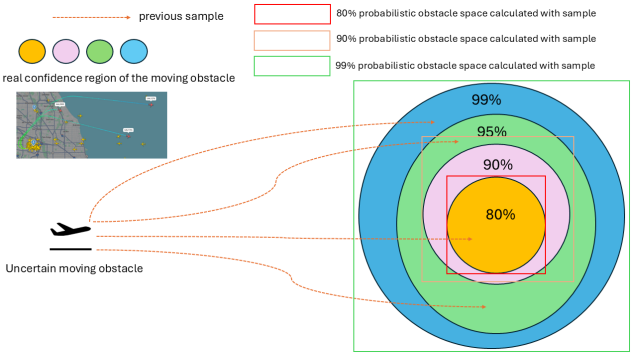


Fig. 1. Behavior uncertainty: flights taking off from the same runway with a different destination will have a different route

Behavior uncertainty: Another important source of uncertainty arises from the intent of other agents. As shown in Fig. 1, multiple aircraft may depart from the same runway within a narrow time window, and each departure may be cleared for a different standard instrument departure depending on destination, weather, and traffic flow. From the UAV's perspective, the resulting obstacle space is therefore governed by a probability distribution over several candidate routes rather than a single deterministic path.

Sensor uncertainty: We use noisy sensor data of two adjacent circular obstacles. Because the measurements are corrupted by noise, the exact obstacle positions cannot be determined without calibration or advanced signal processing techniques. Furthermore, a single measurement outcome may be consistent with multiple possible true obstacle positions. To account for this, we define $\mathbf{s}$ as the set of n noisy sensor measurements, from which a probabilistic obstacle space is constructed.

Since the obstacle space is inherently probabilistic, its exact boundary may be irregular or complex. For path planning, however, a tractable geometric representation is preferred. In particular, polygonal approximations are widely used because they provide a convex, linear structure that can be efficiently integrated into optimization-based planners. To this end, instead of directly generating the full probabilistic obstacle space, we represent it using a convex polygon defined by its vertices. Specifically, the neural network outputs four vertices that characterize the polygonal approximation of the probabilistic obstacle space:

$$\tilde{\mathcal{V}} = NN(w, \mathbf{s}), \tag{4}$$

where w denotes the trained parameters of the neural network and $\tilde{\mathcal{V}}$ is the predicted set of polygon vertices.

We then construct quadrilaterals with the four vertices in a consistent non-self-intersecting order and prevent bow-tie. Although collinear predictions are not formally excluded, none were observed in millions of generated quadrilaterals.

This article has been accepted for publication in IEEE Systems, Man, and Cybernetics Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LSMC.2026.3731283

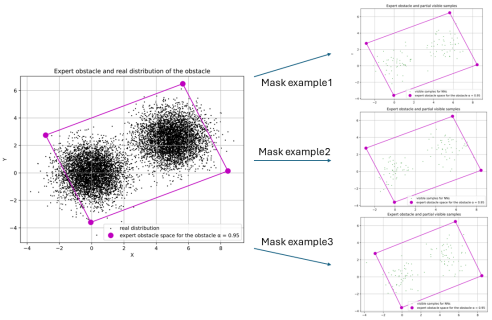


Fig. 2. Supervised learning: we iteratively pick a few samples from o^{true} as inputs to the neural network and task the network with generating the corresponding probabilistic obstacle space.

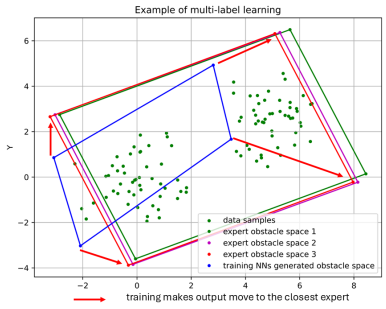


Fig. 3. Multi-label learning: during training, the network is guided to produce outputs that align with the closest expert solution.

III. Method

A. Supervised learning and expert method

Previous research [32] has proven that the transformer can process the visible (unmasked) parts of the data, reconstructing the entire input from the encoded, incomplete data (masked data). Therefore, if we only have a few samples from a distribution, the transformer may be able to reconstruct the entire distribution. To train the neural network, we first require expert solutions generated by an expert method to serve as supervision during the training process.

In this paper, we employ the expert method, as proposed in our previous work [33]. The expert method can generate the probabilistic obstacle space of an uncertain object. This data-driven method builds on a kernel density estimator (KDE) that can generate the probabilistic obstacle space, accelerated by a fast Fourier transform (FFT). These KDE values are then utilized within an Integer Linear Programming (ILP) framework to find a minimal parallelogram box that approximates the probabilistic obstacle space. This ILP solution effectively encapsulates the area where the state is expected to stay with high confidence.

For each obstacle distribution, multiple expert labels are generated from different sampled obstacle configurations. Each sampled configuration can yield a different feasible quadrilateral, and every expert-generated quadrilateral is constructed by the procedure in our previous work [33] to satisfy the prescribed target coverage.

After obtaining the expert polygon vertices $\tilde{\mathcal{V}}$, we prepare the input data for network training. Fig. 2 illustrates how different subsets of samples are presented to the neural network during training. At each training step, a small random subset of samples is selected, and the transformer is trained to generate the corresponding polygon vertices. Through this supervised learning process, the network is repeatedly exposed to multiple partial views of the same obstacle, learning to associate incomplete sample sets with the full probabilistic obstacle space.

B. Multi-label learning and loss function

Recall the non-uniqueness of confidence regions [34]: for any given distribution, there exist infinitely many subsets that capture the same probability mass. Consequently, for each uncertain obstacle, there are infinitely many valid probabilistic obstacle spaces. This motivates the use of multi-label learning rather than single-label learning. Multi-label learning not only reduces computational overhead but also allows the model to share learned features across multiple labels, potentially leading to better generalization. Fig. 3 illustrates this process. Given the current network parameters, the neural network produces an output polygon. The loss is then computed between this output and each available expert probabilistic obstacle space. The smallest loss among them is used to update the network weights. In practice, multiple expert probabilistic obstacle spaces are provided for each obstacle during training. Similar to many regression-based approaches, we employ the mean squared error (MSE) loss, as shown in Eq. 5, which minimizes the average discrepancy between the network output and the expert labels.

The minimum-loss strategy is used because these labels are alternative valid solutions for the same underlying obstacle distribution. Averaging the losses over all labels could penalize a prediction that closely matches one valid quadrilateral but differs from the other valid quadrilaterals. Selecting the minimum loss instead accommodates the inherent multimodality of the solution space.

$$\text{Loss} = \min_{j\in[1,10]} \frac{1}{8}\sum_{i=1}^{8}(\hat{y}_{ij} - y_i)^2 \quad \text{where} \quad \hat{y}_{ij} \in \tilde{\mathcal{V}} \tag{5}$$

where:

- we have 8 (four 2d vertices) elements in the output vector,
- y_i is is the generated value for the i-th element,
- $\hat{y}_{ij}$ is the j-th expert vertices for the i-th element.

C. Neural networks architecture

The overall architecture of the neural networks is designed based on our previous work [35]. The input to the neural network consists of sampled points and the desired confidence level. Before sending these inputs to the neural network, we normalize them first. The confidence level and the sample points are then embedded in a higher dimension using a linear layer. The embedded items are concatenated together. These concatenated embeddings are then sent to the transformer layer. The sample points are treated as a set rather than a sequence. No position-dependent label is assigned to the

sample-point channels; therefore, permuting the input points only permutes their intermediate point channels, while the confidence-level channel aggregates the same set of point features and produces the same polygon output.

D. Input normalization, embedding, and concatenate

Because sample points are drawn from different tasks, they have very different means and variances, which can cause challenges for the training process of neural networks. Normalizing those points before feeding them into the neural network is very crucial for several reasons. First, if some inputs have larger values and higher variance than others, they may dominate the training process. For example, if one target (or feature) has a very large value compared to others, its gradient will be disproportionately large. Second, normalization can accelerate the training process. When the scales of the inputs are not consistent, the model might spend more time adjusting weights to accommodate the varying scales of input data rather than learning meaningful patterns. Third, normalization can prevent overfitting by reducing bias toward dominant features and indirect regularization effects.

We use the Min-Max scaling [36] to normalize the sample points. The Min-Max scaling equation is defined by:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}. \tag{6}$$

Min-max scaling scales sample points to the range [0, 1] without changing the relative position between points. At the same time, since all the points are sampled from a reasonable distribution, so no outlier can heavily influence the normalization.

After normalizing, we use a single linear layer to embed the inputs. Compared to vision or natural language inputs, the inputs in our task are much more straightforward to understand. We primarily aim to determine the relationship between points rather than to interpret their meanings. Additionally, the transformer model already incorporates non-linearities. Therefore, we choose to embed the inputs into relatively low dimensions and use only a single linear layer to decrease the computing complexity.

After embedding, the point tensor has dimensions (N, C), where N is the dimension of the series of points and C is commonly referred to as the channel in the field of machine learning. We concatenated the confidence level tenor and the point tensor together along the N dimension. So, the concatenated tensor has dimensions ($N + 1$, C), while the first channel is the channel of the confidence level.

E. Transformer and Output Layer

We use only the transformer encoder [37] because the task predicts all polygon vertices simultaneously rather than sequentially. The encoder consists of six identical layers, each containing multi-head self-attention, a feedforward network, residual connections, and layer normalization. Compared with recurrent or convolutional layers, self-attention is more suitable for this point-set input because it enables parallel computation and allows each point to directly attend to all other points, capturing long-range dependencies without relying on local patches or sequential processing.

The Transformer is selected here to test whether a learning-based method can reproduce or improve upon the expert and baseline methods while retaining efficient parallel inference. We do not claim that it is universally superior to MLP, PointNet, or DeepSets. PointNet and DeepSets may achieve better performance, and a comparative study of alternative learning architectures is left to future work.

Unlike masked autoencoder reconstruction tasks, our model does not use mask tokens or a decoder. The visible samples do not indicate which part of the obstacle space they come from, and their order is irrelevant: the point sets $(1, 1), (2, 2)$ and $(2, 2), (1, 1)$ represent the same distribution. Since the goal is not to reconstruct the full distribution but to generate a convex bounding region, a simple output module is sufficient.

Following ViT [38], we use a multilayer output layer instead of a single linear layer. This module further refines the encoder features and predicts the final convex bound using only the transformed confidence-level channel, which has already attended to all embedded point channels through the transformer. Finally, the predicted output is denormalized to match the original coordinate scale.

F. Parameters

For the transformer, the embedding dimension N_{emd} is 864. We use 6 attention heads, so each head processes 144 embedding dimensions of each channel. The number of attention layer is also 6. Optimization is performed using the Adam optimizer with a dynamic learning rate to ensure gradual convergence. Additionally, a dropout rate of 0.2 is applied to prevent overfitting by randomly omitting certain neurons during training.

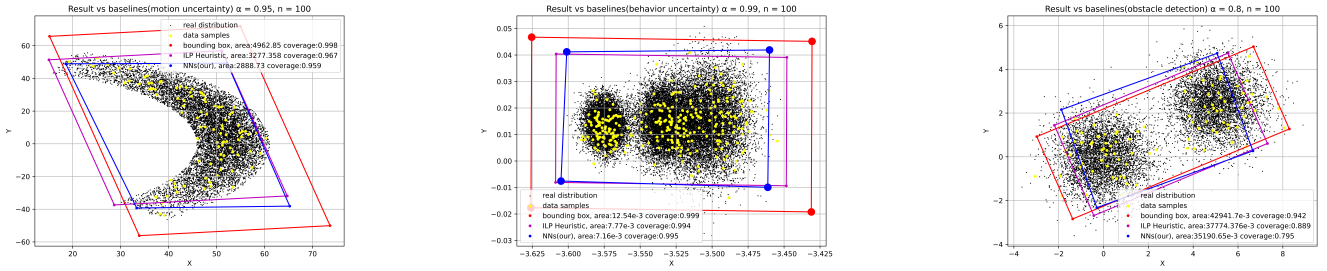
The reported model is trained for five epochs on a single NVIDIA RTX 3090 GPU. The training set contains 320,000 input sets of 100 samples, with 100 expert vertex sets prepared for each confidence level; ten corresponding expert labels are randomly assigned to each selected input during a training iteration.

IV. Result

A. Experiment setting

To evaluate the effectiveness of the proposed method, we conduct experiments on three types of uncertain obstacles: flying obstacles subject to motion uncertainty, obstacles with uncertain destinations, and obstacles whose positions are estimated using noisy sensor measurements. We feed the neural network with samples drawn from the true obstacle distribution and desired confidence level α, then evaluate whether the generated probabilistic obstacle space achieves the desired coverage, i.e., whether it bounds the specified proportion of the distribution. Obstacles of the same type share the same underlying distribution but may exhibit different geometric shapes.

This article has been accepted for publication in IEEE Systems, Man, and Cybernetics Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LSMC.2026.3731283



(a) motion uncertainty (b) behavior uncertainty (c) sensor uncertainty

Fig. 4. Black points are sampled from the real obstacle distribution. Colorful polygons are example outputs when the three methods take the same yellow visible points. Yellow points are enlarged for visibility.

1) Training: We trained a neural network that can work with varying values of α, where α can be 80%, 85%, 90%, 95%, and 99%. To accelerate the training process, we prepared 100 sets of expert vertices for each desired confidence level and prepared 320,000 sets of 100 samples before training began. In each training iteration, we randomly select one set of samples and one of α as the neural network's input, then randomly assign ten sets of corresponding expert vertices as the target. We can prepare the training dataset for a known uncertain obstacle within a few hours. It supports the *Advantage 3* of the proposed method, the training dataset is easy to make.

2) Evaluation and Baselines: To evaluate the generated probabilistic obstacle space o^{α}, we estimate its coverage using 100,000 Monte Carlo samples. The error between the measured coverage α_i and the desired confidence level $\hat{\alpha}_i$ is measured by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\alpha_i - \hat{\alpha}_i)^2, \tag{7}$$

where $n = 1000$. We also report the correctness rate,

$$\text{Correctness Rate} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}[\alpha_i > \hat{\alpha}_i]\,. \tag{8}$$

We compare our neural-network method with two KDE-based baselines: bounding box (BB) [39], [40] and ILP Heuristic (ILP). Since these baselines are not learning-based, we evaluate them using both the same number of samples and 20× more samples. The MINLP optimal algorithm [41] is included as a reference for the minimum-area convex region satisfying the coverage requirement. We tested the trained neural networks by presenting them with 5,000 random tasks (1000 for each α) from each type of uncertain obstacle. Each task contains 100 samples randomly generated from the real distribution of the uncertain obstacle and has never been seen by the neural network. The neural network generates the corresponding o^{α} for those samples.

We use 100 input samples because substantially smaller sets, particularly five samples, are generally insufficient to estimate the underlying distribution reliably. Thus, 100 samples are treated as a limited but practically meaningful setting and remain substantially fewer than those commonly required by sampling- or calibration-based uncertainty estimation methods.

Table I shows that NN achieves consistently low MSE and stable coverage across all uncertainty types and confidence levels. BB usually achieves high correctness, but its obstacle-space areas are much larger, indicating conservative behavior. ILP can generate smaller regions, but its correctness is less stable, especially with limited samples. Compared with both baselines, NN provides a better balance between coverage accuracy, correctness, and obstacle-space compactness, and its results are generally closer to the optimal-area reference.

B. Overall Discussion

Across motion, behavior, and sensor uncertainty, the proposed NN consistently outperforms traditional BB and ILP methods. Three key observations can be drawn:

- **Accuracy with Few Samples:** NN achieves high accuracy and low MSE even with very limited data, whereas baselines require 20× more samples to approach similar performance. Its empirical coverage ranges also remain closely aligned with the target confidence levels, indicating more accurate confidence calibration. In contrast, the baseline methods frequently produce coverage values near 100%, suggesting overly conservative uncertainty regions.
- **Reliability:** NN maintains correctness rates above 93% in all scenarios, ensuring that the generated probabilistic obstacle spaces meet or exceed the desired confidence levels.
- **Space Efficiency:** NN produces compact obstacle spaces that are significantly smaller than BB while avoiding the instability of ILP. This efficiency is particularly critical in path planning applications where over-approximation reduces safe space.
- **Training cost:** Because the proposed method is learning-based, it requires a training stage before inference. However, once trained, the model can be applied repeatedly to scenarios represented in the training dataset without requiring retraining. The current results were obtained after only five training epochs, requiring less than one hour on a single NVIDIA RTX 3090 GPU without using any training-acceleration techniques.

These results validate the effectiveness of the proposed learning-based framework in generating accurate and reliable

TABLE I
RESULT COMPARISON UNDER MOTION, BEHAVIOR, AND SENSOR UNCERTAINTY

Type	CL	NN MSE(range)	BB small MSE(range)	BB large MSE(range)	ILP small MSE(range)	ILP large MSE(range)	CR: NN/BBs/BBI/ILPs/ILPI/Opt	Area: NN/BBs/BBI/ILPs/ILPI/Opt
Motion	80%	0.32%(78.95–80.44)	11.98%(80.28–99.80)	12.73%(88.66–96.29)	2.83%(75.04–91.87)	2.01%(75.36–84.46)	98.7/100/100/57.7/79.8/100	1415.07/2342.52/2309.52/1783.52/1514.04/1172.71
	85%	0.25%(84.44–86.21)	9.88%(85.39–99.91)	10.98%(91.93–98.82)	2.95%(75.24–94.01)	2.31%(81.43–90.18)	99.8/100/100/52.4/64.4/100	1762.30/2790.63/2815.55/2138.71/1902.27/1579.52
	90%	0.23%(89.40–90.91)	7.36%(89.60–99.99)	8.63%(95.93–99.75)	2.69%(75.63–98.43)	1.90%(87.40–94.16)	99.7/99.8/100/52.8/82.3/100	2229.01/3367.96/3470.37/2652.60/2365.27/2107.79
	95%	0.49%(95.22–96.02)	4.32%(92.60–100)	4.81%(98.59–100)	2.20%(85.67–99.88)	2.72%(94.35–99.06)	100/99.4/100/74.8/98.9/100	3530.87/4525.46/4576.97/3382.33/3374.61/2742.49
	99%	0.98%(99.96–99.99)	1.33%(98.03–100)	0.99%(99.96–100)	1.12%(97.86–100)	0.98%(99.19–100)	100/61/100/59.7/100/100	4833.31/5749.98/6127.58/4967.29/4820.67/3711.22
Behavior	80%	1.79%(79.54–84.27)	11.31%(84.04–96.41)	10.47%(85.60–93.33)	2.74%(75.00–88.70)	2.40%(75.00–81.90)	93.1/100/100/10.8/3.9/100	2.75E-3/3.64E-3/3.46E-3/2.59E-3/2.39E-3/2.15E-3
	85%	1.76%(82.34–88.25)	9.56%(88.09–98.02)	9.45%(90.69–96.35)	2.62%(75.31–90.94)	3.01%(85.60–93.33)	95.2/100/100/27.9/16.2/100	3.44E-3/4.41E-3/4.35E-3/2.99E-3/2.89E-3/2.39E-3
	90%	1.13%(88.31–91.80)	7.37%(93.11–99.40)	7.44%(95.89–98.67)	1.76%(81.48–96.56)	1.37%(85.62–94.55)	99.8/100/100/69.2/55.8/100	4.41E-3/7.37E-3/5.81E-3/3.83E-3/3.67E-3/2.99E-3
	95%	0.85%(94.14–97.83)	4.09%(95.82–99.95)	4.22%(97.98–99.67)	1.11%(90.61–98.80)	0.74%(92.35–97.50)	95.6/100/100/68.2/65.4/100	5.69E-3/7.89E-3/7.98E-3/5.03E-3/4.87E-3/3.95E-3
	99%	0.58%(99.18–99.94)	0.80%(98.42–100)	0.91%(99.72–99.97)	0.40%(97.03–99.95)	0.25%(98.28–99.61)	100/99.7/100/56.5/75.8/100	9.21E-3/11.19E-3/11.16E-3/8.01E-3/7.63E-3/6.46E-3
Sensor	80%	0.42%(79.12–82.05)	11.04%(87.56–99.42)	16.26%(93.24–98.43)	3.66%(72.00–93.54)	4.12%(72.01–86.43)	99.7/100/100/31.4/14.7/100	35.22/45.65/46.14/36.06/33.95/33.01
	85%	0.35%(84.45–86.63)	10.18%(91.98–99.80)	12.10%(95.65–98.49)	3.70%(72.17–95.76)	3.15%(74.27–93.60)	99.8/100/100/67.3/57.9/100	41.58/52.16/50.12/42.42/40.69/34.23
	90%	0.24%(89.51–91.01)	8.20%(93.26–99.94)	8.12%(96.52–99.63)	4.98%(86.16–98.78)	4.37%(89.70–98.32)	99.8/100/100/97.2/99.8/100	51.38/63.18/59.10/53.86/52.35/42.79
	95%	0.62%(94.08–96.29)	4.20%(94.84–99.99)	4.28%(98.64–99.86)	3.25%(92.45–99.73)	3.22%(95.48–99.53)	98.9/99.9/100/99.1/100/100	64.53/75.85/71.90/67.46/66.08/55.01
	99%	0.87%(97.66–99.94)	1.87%(97.09–99.99)	0.93%(99.73–99.99)	0.80%(98.97–99.99)	0.86%(99.49–99.98)	98.7/95.2/100/99.9/100/100	99.03/101.49/106.67/95.28/96.10/73.96

probabilistic obstacle spaces under diverse forms of uncertainty.

The neural-network output does not provide a formal guarantee that the prescribed confidence level is satisfied in every instance. Because of long-tail behavior, a newly sampled point can occasionally lie far outside the observed range even when a substantially larger region covers all available samples. Rare violations may therefore remain in an evaluation with 100,000 Monte Carlo trials. For safety-critical path planning, a safety margin or post-hoc coverage verification should be applied when a strict instance-wise guarantee is required.

V. GENERATING SPEED ANALYSIS

Generating time is crucial in deciding whether a method can be used in online applications. Therefore, we compare the time to generate one correct probabilistic obstacle space by each method to prove that our method is more useful in online applications.

Unlike the expert method, which performs KDE, FFT, and ILP optimization to construct each solution, the proposed method uses the expert only during offline label generation and directly predicts the polygon during online inference. This introduces a one-time training cost but substantially reduces repeated online computation while maintaining comparable or better coverage accuracy and compactness than the evaluated baselines. Fig. 5 illustrates the average time cost for generating effective outcomes. For reproducibility, the timing comparison reports the hardware and software environment, CPU/GPU usage, whether preprocessing is included, and the timing statistic used for each method together with the numerical values shown in Fig. 5. The results clearly demonstrate that our method significantly outperforms the baseline approaches, achieving execution times that are an order of magnitude faster. The result supports that the *Advantage 4* of the proposed method, NN can generate a useful probabilistic obstacle space very fast.

VI. CONCLUSION AND FUTURE

In this paper, we propose using supervised learning to train a transformer-based neural network to generate a confidence region of the real obstacle space distribution, which we define as the probabilistic obstacle space. Once trained, the neural network can robustly generate vertices of the polygon that cover the desired percentage of the real obstacle region, given only a few samples. The primary advantage of the proposed

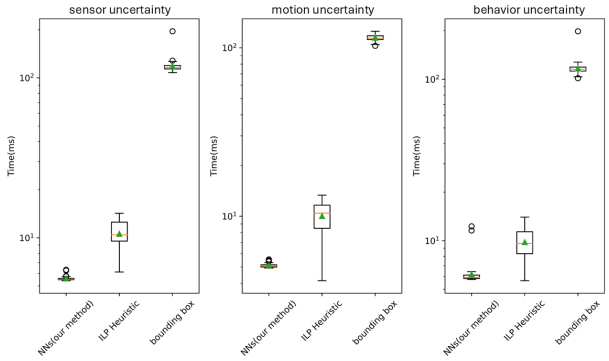


Fig. 5. Generating speed comparison

method is its extremely fast generation time while maintaining performance comparable to that of the baseline methods. Meanwhile, training is also very simple because training inputs can be generated very fast. Only a few experts are required for the labels. No dynamic information about the obstacle such as disturbance is required.

ACKNOWLEDGMENT

The authors would like to acknowledge the support from the National Science Foundation under Grants CCF-2402689 and CCF-2523829. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not reflect the views of NSF.

REFERENCES

[1] E. L. Thompson, A. G. Taye, W. Guo, P. Wei, M. Quinones, I. Ahmed, G. Biswas, J. Quattrociocchi, S. Carr, U. Topcu *et al.*, "A survey of evtol aircraft and aam operation hazards," in *AIAA AVIATION 2022 Forum*, 2022, p. 3539.

[2] J. Zhang, P. Zu, K. Liu, and M. Zhou, "A herd-foraging-based approach to adaptive coverage path planning in dual environments," *IEEE Transactions on Cybernetics*, vol. 54, no. 3, pp. 1882–1893, 2024.

[3] 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.

[4] X. Zhang, J. Ma, Z. Cheng, S. Huang, S. S. Ge, and T. H. Lee, "Trajectory generation by chance-constrained nonlinear mpc with probabilistic prediction," *IEEE Transactions on Cybernetics*, vol. 51, no. 7, pp. 3616–3629, 2021.

[5] J. Roche, V. De-Silva, and A. Kondoz, "A multimodal perception-driven self evolving autonomous ground vehicle," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9279–9289, 2022.

[6] Z. Zhang, W. Zhang, and X. Ren, "A bi-criteria obstacle avoidance scheme synthesized by time-varying penalty strategy neural network for mobile parallel manipulators," *IEEE Transactions on Cybernetics*, vol. 56, no. 3, pp. 1337–1350, 2026.

[7] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 139–146.

[8] Q. Wang, Z. Wang, M. Wang, J. Ji, Z. Han, T. Wu, R. Jin, Y. Gao, C. Xu, and F. Gao, "Fast iterative region inflation for computing large 2-d/3-d convex regions of obstacle-free space," *IEEE Transactions on Robotics*, 2025.

[9] B. Tian, P. Li, H. Lu, Q. Zong, and L. He, "Distributed pursuit of an evader with collision and obstacle avoidance," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 512–13 520, 2022.

[10] 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.

[11] L. Shimanuki and B. Axelrod, "Hardness of motion planning with obstacle uncertainty in two dimensions," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1151–1166, 2021.

[12] D. Zhu, H. Huang, and S. X. Yang, "Dynamic task assignment and path planning of multi-auv system based on an improved self-organizing map and velocity synthesis method in three-dimensional underwater workspace," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 504–514, 2013.

[13] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2015, pp. 109–124.

[14] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[15] Y. Wan, Y. Zhong, A. Ma, and L. Zhang, "An accurate uav 3-d path planning method for disaster emergency response based on an improved multiobjective swarm intelligence algorithm," *IEEE Transactions on Cybernetics*, vol. 53, no. 4, pp. 2658–2671, 2023.

[16] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[17] M. Mitici and H. A. P. Blom, "Mathematical models for air traffic conflict and collision probability estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1052–1068, 2019.

[18] M. G. Ferguson, "Global positioning system (gps) error source prediction," Tech. Rep., 2000.

[19] C. Zhu, M. Joerger, and M. Meurer, "Quantifying feature association error in camera-based positioning," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 2020, pp. 967–972.

[20] A. Hassani and M. Joerger, "A new point-cloud-based lidar/imu localization method with uncertainty evaluation," in *Proceedings of the 34th international technical meeting of the satellite division of the institute of navigation (ion gnss+ 2021)*, 2021, pp. 636–651.

[21] Z. Hong, Y. Petillot, K. Zhang, S. Xu, and S. Wang, "Large-scale radar localization using online public maps," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3990–3996.

[22] M. Abramson, M. Refai, M. G. Wu, and S. Lee, "Applying sensor uncertainty mitigation schemes to detect-and-avoid systems," in *AIAA Aviation 2020 Forum*, 2020, p. 3264.

[23] J. Li, X. Shi, F. Chen, J. Stroud, Z. Zhang, T. Lan, J. Mao, J. Kang, K. S. Refaat, W. Yang, E. Ie, and C. Li, "Pedestrian crossing action recognition and trajectory prediction with 3d human keypoints," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1463–1470.

[24] J. Xiang and J. Chen, "Data-driven probabilistic trajectory learning with high temporal resolution in terminal airspace," *Journal of Aerospace Information Systems*, pp. 1–11, 2025.

[25] W. Zhi, L. Ott, and F. Ramos, "Probabilistic trajectory prediction with structural constraints," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9849–9856.

[26] J. Miura and Y. Shirai, "Modeling motion uncertainty of moving obstacles for robot motion planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3. IEEE, 2000, pp. 2258–2263.

[27] C. Demonceaux, A. Potelle, and D. Kachi-Akkouche, "Obstacle detection in a road scene based on motion analysis," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 6, pp. 1649–1656, 2004.

[28] M. Kristan, V. Sulić Kenk, S. Kovačič, and J. Perš, "Fast image-based obstacle detection from unmanned surface vehicles," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 641–654, 2016.

[29] P. Wu, L. Li, J. Xie, and J. Chen, "Probabilistically guaranteed path planning for safe urban air mobility using chance constrained rrt," in *AIAA Aviation 2020 Forum*, 2020, p. 2914.

[30] A. Timans, C.-N. Straehle, K. Sakmann, and E. Nalisnick, "Adaptive bounding box uncertainties via two-step conformal prediction," in *European Conference on Computer Vision*. Springer, 2024, pp. 363–398.

[31] A. P. Vinod and M. M. K. Oishi, "Probabilistic occupancy via forward stochastic reachability for markov jump affine systems," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3068–3083, 2021.

[32] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[33] P. Wu and J. Chen, "Efficient box approximation for data-driven probabilistic geofencing," *Unmanned Systems*, pp. 1–12, 2023.

[34] J. Neyman, "Outline of a theory of statistical estimation based on the classical theory of probability," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 236, no. 767, pp. 333–380, 1937.

[35] J. Xiang and J. Chen, "Imitation learning-based convex approximations of probabilistic reachable sets," in *AIAA AVIATION FORUM AND ASCEND 2024*, 2024, p. 4356.

[36] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[39] W. Zhao and R. Wen, "The algorithm of fast collision detection based on hybrid bounding box," in *2012 International Conference on Computer Science and Electronics Engineering*, vol. 3. IEEE, 2012, pp. 547–551.

[40] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[41] P. Wu and J. Chen, "Data-driven zonotopic approximation for n-dimensional probabilistic geofencing," *Reliability Engineering & System Safety*, vol. 244, p. 109923, 2024.