Learn to Path: Using neural networks to predict Dubins path characteristics for aerial vehicles in wind

Trevor Phillips*, Maximilian Stölzle*, Erick Turricelli*,
Florian Achermann, Nicholas Lawrance, Roland Siegwart and Jen Jen Chung

***Abstract*— For asymptotically optimal sampling-based path planners such as RRT*, path quality improves as the number of samples added to the motion tree increases. However, each additional sample requires a nearest-neighbor search. Calculating state transition costs can be particularly difficult in cases with complex dynamics such as aerial vehicles in non-isotropic cost fields like wind. Computationally costly nearest neighbor searches increase the time required to add new samples to the search tree, thereby reducing the likelihood of finding low-cost paths in a given computational time. In this paper, we propose the use of a lightweight neural network to approximate nearest neighbor cost calculations. The network approach uses a low-dimensional encoding of the cost space along with a start and goal query pair and returns an estimate of the path cost that can be used for nearest neighbor and path validity estimation. We demonstrate our method for a Dubins airplane model in a 3D wind field and show that the network method achieves equivalent path lengths as an existing iterative solver $32\,\%$ faster and, when given the same search time, up to $10.8\,\%$ shorter.**

I. INTRODUCTION

Sampling-based planners such as rapidly-exploring random trees (RRTs) offer appealing properties for solving planning problems in high-dimensional search spaces with dynamic constraints. Some RRT variants, such as RRT*, provide formal guarantees including probabilistic completeness and asymptotic optimality [1]. These algorithms can effectively run as anytime solvers - providing monotonically lower cost solutions as computation time is increased. This property is particularly useful for planning with autonomous vehicles which have dynamics constraints and operate in complex cost spaces [2]. However, a limitation of these methods is that when adding a new state to the search tree they require calculation of the nearest (lowest cost) existing neighbor in the tree. RRT* also requires an additional nearest neighbor search during the rewiring stage in case of a non-symmetric cost space. This search requires an optimal cost metric that provides the true lowest cost between a pair of states. Calculating the optimal cost between any pair of states is not always straightforward and often computationally expensive [3], especially in the case of complex cost fields. Non-isotropic vector cost fields, such as wind for aerial vehicles and currents for underwater vehicles, often have no closed-form solution, requiring iterative solvers and significant computation for every start/goal-pair cost query.

*Authors contributed equally.

This research was funded in part by the Microsoft Swiss Joint Research Center. The authors are with the Autonomous Systems Lab, ETH Zürich, Zürich 8092, Switzerland. {tphillips; mstoelzl; eturricelli; acfloria; lawrancn; rsiegwart; chungj}@ethz.ch

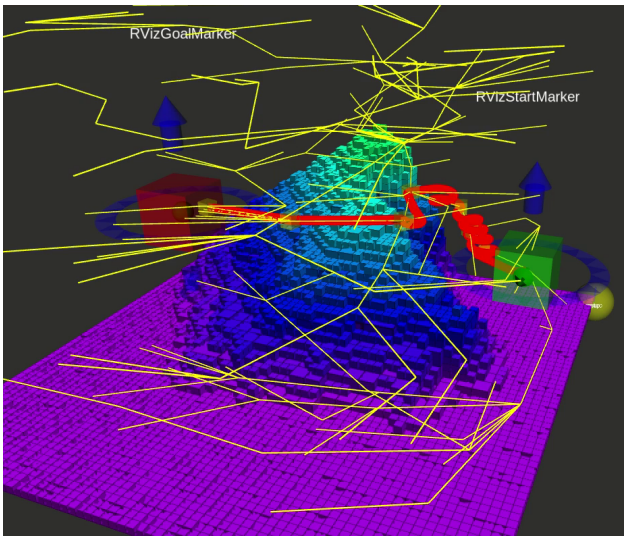


Fig. 1: Visualization of aerial vehicle path planning. The vehicle must navigate from the green to the red region, through a complex wind field and avoiding terrain. The RRT* motion tree is shown in yellow (Dubins airplane path curves are not shown). The final path, with Dubins airplane paths, is shown in red.

This additional cost penalty can limit the effectiveness of the planning algorithm, especially in applications with limited computational resources and time. This is particularly important in applications such as small UAVs operating in wind fields. Because small UAVs are lightweight and have a low moments of inertia, they can be easily disturbed by wind [4]. Furthermore, small UAVs often fly close to the ground, so wind disturbances may lead to crashes. To prevent such events, an online path planner may be used on UAVs to avoid (or even take advantage of) wind disturbances, and RRT-based methods are particularly appealing due to the anytime solution capability and elegant handling of dynamic constraints. In the context of UAVs flying in windy conditions, the optimal path is often considered as the minimum-time path, such that the local cost metric is the minimum time required to fly between two poses. Minimum time is also equivalent to minimum path length assuming a fixed air-relative speed. Using Euclidean distance between two states is not sufficient for estimating the cost, as it leads to high errors of more than $140\,\%$ of the true cost and thus results in sub-optimal paths during planning. Instead, fixed-wing UAV models often use the time-optimal Dubins airplane path [5], an approximation for fixed-wing UAV motion based on piece-wise constant curvature and straight-line path segments.

While the optimal Dubins airplane cost has a partial analytic solution for zero or uniform wind [6], the current computational bottleneck for RRT-based path planning lies in finding the optimal Dubins path between two poses in the presence of non-uniform wind [7]. Currently, this computation uses an expensive iterative solver and reduces the number of explored states in the RRT* motion tree to $0.5\,\%$ compared to the number of nodes added using a Euclidean distance approximation. Over $95\,\%$ of the time for the nearest neighbor search is spent in the iterative algorithm, which leads to fewer path planning iterations and a sub-optimal final path.

In this work, we develop a learned approach to replace the expensive iterative search for estimating path cost and validity in a non-isotropic cost field (wind) for a 3D Dubins airplane model. Our approach uses a convolutional neural network to encode the 3D wind field into a low-dimensional latent space. This code is appended to start/goal-pair location queries, and a multi-layer fully connected network estimates the path cost and validity. This provides fixed-time, high-speed inference for path cost estimation, allowing the search tree to explore more of the state space and find lower-cost paths. Our work is integrated into an RRT* planning pipeline implemented in C++ with the Open Motion Planning Library (OMPL) [8], and shown to consistently reduce the computational bottleneck for UAV path planning, thereby moving towards online, in-flight planning. A visualization of our approach is shown in Fig. 1.

II. RELATED WORK

Aerial vehicle planning in 3D wind represents a complex challenge due to a high-dimensional state space, dynamic constraints, and non-isotropic (direction dependent) costs. Mathematically appealing methods such as wavefront expansion for non-isotropic costs [9], [10] are often limited to lower dimensional spaces or have limited capabilities for dynamic constraints. Sampling-based approaches work well for higher-dimensional planning, but previous methods assume constant wind over each planning edge [11], or use costly forward-propagation methods [12].

Sampling-based planning methods such as PRM- and RRT-based approaches require a cost metric that calculates the optimal path cost between a pair of sample configurations in collision-free space. In cases with complex dynamic constraints, this metric can be hard to calculate, since it requires solving the local planning problem between two states. Previous work [13] shows that weighted variants of the Euclidean metric do not always perform well. An approach presented in [14] estimates the optimum cost-to-go metric offline using a higher-order state space projection before graph construction. Other approaches show that improvements can be made by approximating the local planner, for example by linearizing the system dynamics [15], but this method struggles with highly non-linear systems.

Recent research has explored approaches for estimating cost metrics through machine learning to train a fast online predictor from expert demonstrations. Various publications demonstrate considerable improvement over Euclidean estimates through learning to approximate the two-point boundary problem for non-holonomic wheeled robots [16], [17], and other approaches use machine learning to estimate cost for kinodynamic RRT* planning [18], [19]. Direct learning of the reachable set in the neighborhood of a quadrotor is proposed by Allen and Pavone [20], using a support vector machine (SVM). Neural RRT* [21] is a CNN which can predict the probability distribution of the optimal path on the map and guides the sampling process in a 2D environment. Reinforcement learning (RL) approaches for approximating local planners are explored to improve prediction times in PRM and RRT planners [22], [23]. However, these approaches focus on system dynamics, and do not consider asymmetric or non-isotropic environmental costs.

Our work approaches this problem by providing information about the environmental costs to the local planner cost estimator. In our application, we generate a lower-dimensional latent-space representation of the full wind field using an encoder network, inspired by previous work for wind estimation near terrain [4]. Convolutions are particularly useful for planning because they are able to learn spatial correlations. Alternative and potentially complementary approaches have also demonstrated improved planning performance by learning network-encoded environment representations to generate biased sampling distributions [24], [25], generating cost maps for inverse RL [26], or even the full planning pipeline [27]. We take a similar approach for encoding the wind field, but use the coded representation as an input for the local path cost estimator in an RRT framework. We do not use an end-to-end learning approach to avoid issues related to generalization and data scarcity.

III. METHOD

We propose a neural network approach to estimate the wind-aware, time-optimal cost for a Dubins airplane path between a given start and goal pose in a full 3D wind field. The estimator is then used online for nearest-neighbour calculations of an RRT* planner.

A. Inputs and Outputs

Variational Autoencoders (VAEs) have shown promising results predicting local wind fields from given boundary wind speeds and a terrain map [28]. This network encodes the wind at each location in a spatially uniform three-dimensional grid. Each cell in the grid is four dimensional, with one dimension for each of the three wind directions, and a fourth dimension indicating if the cell is terrain or free space. As such a network is capable of running at a constant rate during flight to predict the local wind field around the UAV, its wind encoding can be directly used as a low-dimensional latent space representation of the local wind field. The reuse of this wind encoding for the cost prediction neural network prevents any additional computational overhead from re-encoding the wind field to a lower dimension and only needs to be done once per planning run. This design choice is motivated by the fact that the wind

encoder has an average inference time of $1.8\,\mathrm{s}$ while the cost estimation (see Section III-C) exhibits an average inference time of $9.5 \times 10^{-6}\,\mathrm{s}$.

The input $\mathbf{x}$ for the downstream neural network used to estimate cost consists of the wind encoding $\mathbf{y}_{\text{enc}} \in \mathbb{R}^{1024}$, the wind scale $U_{\text{norm}} \in \mathbb{R}^1$ which represents the strength of the wind field, the length of each wind grid cell in all Cartesian dimensions $l_{\text{grid}} \in \mathbb{R}^3$, as well as the start $\mathcal{X}_{\text{start}} \in \mathbb{R}^3 \times \mathbb{S}^1$ and goal $\mathcal{X}_{\text{goal}} \in \mathbb{R}^3 \times \mathbb{S}^1$ poses (see Equation 1). Each pose consists of 3D Cartesian coordinates respective to the wind grid and a planar heading ψ.

The wind grid is encoded to a vector of length 1024 based on the results of a hyperparameter selection study. A lower wind encoding size resulted in poorer performance of the network on the test set with an increase in the test loss by $330\,\%$ and $350\,\%$ for wind encoding sizes of 256 and 64 respectively. An increase of the wind encoding size to 4096 led to generalization challenges between training and validation set as the number of trainable parameters increased and thus the test loss increased by $330\,\%$.

The output $\mathbf{y}$ of the model consists of the cost $C \in \mathbb{R}^+$ and the path validity prediction $p_{\text{valid}} \in \{0, 1\}$ (2). The ground truth cost is equivalent to the length of the wind-aware Dubins path. The ground truth validity is set to zero if the iterative algorithm fails to converge, otherwise to one. In the case of an invalid path, the cost is set to zero.

$$\mathbf{x}_n = \begin{bmatrix} \mathbf{y}_{\text{enc}} & U_{\text{norm}} & l_{\text{grid}} & \mathcal{X}_{\text{start}} & \mathcal{X}_{\text{goal}} \end{bmatrix} \tag{1}$$

$$\mathbf{y}_n = \begin{bmatrix} C & p_{\text{valid}} \end{bmatrix} \tag{2}$$

Inputs are normalized to the same order of magnitude with the goal of speeding up the training and preventing large gradients and saturation of the non-linear activation layers (for example ReLUs). During the training of the VAE on the wind fields, the wind speeds are normalized with a wind speed scalar U_{norm}, which signifies the strength of the wind. The coordinates of the start and goal poses are normalized by the dimension of the wind field in the respective axis. The heading angle of the UAV at the start and goal poses is normalized by 2π.

B. Loss Function

We use a relative ℓ_1 loss for the cost prediction C and a binary cross entropy loss with logits for the path validity prediction loss $\mathcal{L}_{p_{\text{valid}}}$. If a path is invalid, all the other losses except the path validity loss are set to zero irrespective of the prediction as seen in Equation 3. We normalize the cost loss with the ground-truth path length to prevent over-fitting to long paths with larger absolute cost losses.

As our two losses have different orders of magnitude and represent different quantities, we need to balance their contribution to the total loss to ensure convergence of all training tasks. The total loss is computed as follows:

$$\mathcal{L}\left(\mathbf{y}_n, f(\mathbf{x}_n, \theta)\right) = \begin{cases} w_C \cdot \mathcal{L}_C + w_{p_{\text{valid}}} \cdot \mathcal{L}_{p_{\text{valid}}} & \text{if valid path} \\ w_{p_{\text{valid}}} \cdot \mathcal{L}_{p_{\text{valid}}} & \text{otherwise} \end{cases} \tag{3}$$

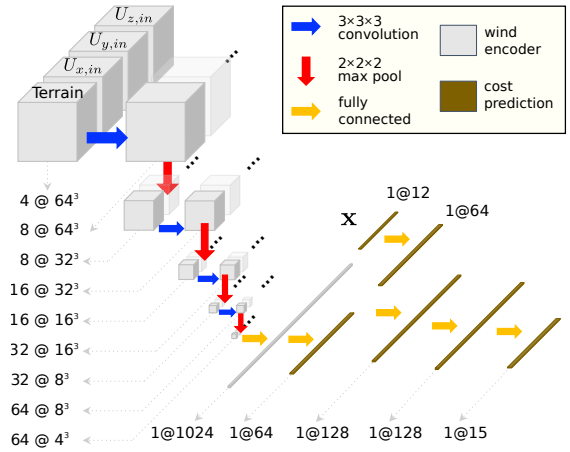


Fig. 2: The Learn to Path network. The wind field encoding (left) is concatenated with the start/goal pair locations (center) to predict path cost and path parameters.

where w corresponds to the individual weights of the loss components which need to be tuned manually. As a result of a hyperparameter selection study, we selected a cost weight $w_C = 1$ and a path validity weight of $w_{p_{\text{valid}}} = 0.5$.

C. Network Architecture

A main challenge in our problem is the different orders of magnitude of the inputs: the wind velocities of the wind grid have a much higher dimensionality than the start and goal poses. The first step to alleviate this is to split the network into two parts and train a wind encoder using a VAE to infer an intermediate wind encoding of size 1024. In a second step, downsampling is used on the wind encoding and upsampling on the poses and parameters to bring both parts to the same dimensionality. Subsequently, the downsampled wind encoding is concatenated with the upsampled parameters to ensure convergence and an equal weighting of inputs.

The second challenge is that the network needs a large enough number of trainable parameters to be able to solve the cost estimation problem with sufficient accuracy. If the prediction quality is not good enough, the RRT* planner will misjudge the cost and sample sub-optimal nodes and paths, which leads to a higher final path length. However, if the number of parameters is too high, the inference time increases resulting in fewer completed path planning iterations over a fixed time during deployment, which can also lead to longer paths. Therefore, we found an optimal trade-off for both the wind encoding size and the network architecture to accomplish a good cost estimation performance with a lightweight model architecture.

We evaluated various neural network architectures but present only the best-performing model for the sake of brevity. The network consists of about 9.97×10^4 learnable parameters. As shown in Fig. 2, the wind encoding is downsampled from the size 1024 to 64 using a fully-connected layer. At the same time, the start and goal poses (each of size 4), the wind scale and the wind grid size (size of 3) are upsampled together using a fully-connected layer to a size of 64. Then the downsampled wind encoding is concatenated with the upsampled parameters. Two additional

fully-connected layers with a size of 128 follow before the dimension is reduced to the output size. After each fully-connected layer, a Leaky ReLU [29] rectifier with a slope of 0.01 is used as the non-linearity.

D. Dataset Generation

To train our model, we created datasets that include wind fields and parameterized wind-aware Dubins airplane paths.

Wind field: Our wind data consists of static 3D wind fields with a size of approximately $1.1 \times 1.1 \times 0.8\,\mathrm{km}^3$ [4]. The wind fields are regular grids containing the terrain as a distance field, and the wind speeds as a 3D vector at each cell. Separate wind fields are used in the training, validation, and test split to ensure our model is robust to different environmental conditions. We use 5373 local wind fields in the training set, 6708 in the validation set and 4764 in the test set.

When building the dataset, random 64^3 regions are sampled from each wind field, and Dubins airplane paths are formed within the sampled region.

Sampling strategy: To build a Dubins airplane path, we first sample a start and goal pose in the 64^3 region. Poses are sampled to mimic the distribution of start-goal cost queries seen by nearest-neighbor search in the iterative algorithm's motion tree. More specifically, a start pose $\mathbf{x}_s$ is sampled uniformly at random from the field. The goal position $\mathbf{x}_g$ is generated by sampling an offset planar x-y distance d_p from a (positive-only) truncated Gaussian distribution $\mathcal{N}\left(2152, 1069^2\right)$, a bearing direction $\psi_p \sim \mathcal{U}\left(0, 2\pi\right)$, and an altitude difference $z_p \sim \mathcal{N}\left(0, 1058^2\right)$. Sampled goal positions are restricted to lie inside the wind grid. The sample distribution parameters were determined from an analysis of the cost queries performed when new states are added to the RRT* motion tree.

An analysis of our neural network's performance indicated that the time-optimal cost of short paths and low-altitude paths are predicted with less accuracy. For this reason, we introduced a bias during generation of the training data such that, with probability $p_p = 0.25$, the planar distance is sampled from a uniform distribution $\mathcal{U}\left(0, 500\,\mathrm{m}\right)$. Independently, the altitude difference is sampled from $\mathcal{U}\left(-100\,\mathrm{m}, 100\,\mathrm{m}\right)$ with probability $p_a = 0.25$.

We compute 100 Dubins paths for every wind field to make up our dataset. This results in 537300 paths in the training set, 670800 paths in the validation set and 476400 paths in the test set.

E. Wind-aware Dubins Airplane Path

The Dubins airplane path is the shortest path between a start and goal pose with a bounded curvature [5]. It consists of segments with a minimum allowable turn radius r_{min} and straight line segments. The climbing angle is constant throughout the whole path.

We extend the original idea, which does not consider wind, to the *Wind-aware Dubins airplane path*: the addition of wind to the Dubins path computation means that in order for the UAV to finish at the desired goal pose, our algorithm must account for shifts in sections of the path caused by wind. The resulting paths can differ from their original versions with respect to total path length and shape.

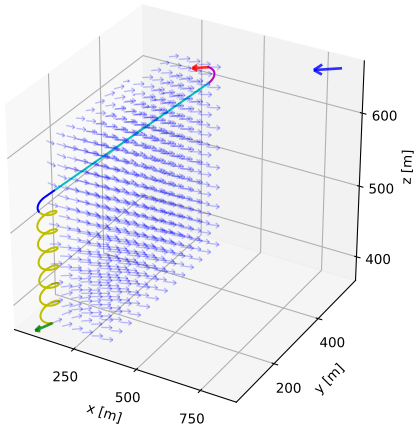


Fig. 3: A wind-corrected Dubins path: Arrows indicate the start pose (green) original goal in inertial space (blue) and wind-aware adjusted goal in wind-relative space (red). Wind pushes the UAV towards the original goal, hence the wind-aware path is shorter. Each segment of the path (helix spiral, right turn, straight, left turn) is colored differently (yellow, dark blue, light blue, purple).

An iterative algorithm is used to compute the path in the presence of wind [7]. An example is shown in Fig. 3. In brief, the iterative algorithm consists of three parts: first, the path *without* wind is computed. Second, the wind information is added by simulating the vehicle flying the path and drifting away due to the wind resulting resulting in a shifted goal pose *with* the respective wind drift. Third, a new virtual goal state is calculated by subtracting calculated wind drift from the original goal pose. The three steps are then repeated iteratively with the new virtual goal pose until the ground-relative path end point and the original goal state have converged within a certain threshold.

In some cases, the path between two states may be classified as *invalid*. This indicates that the end pose of the path has not converged below a specified distance threshold within a number of allowed iterations.[1] For example, too strong wind can make the path infeasible.

If at any point the Dubins path drifts outside the bounds of the 64^3 wind cube, we use the wind from the closest point inside the cube during the iterative algorithm.

IV. RESULTS

Performance of path planning: Integration of our neural network into a full path-planning framework reduces the runtime of nearest-neighbor search within the RRT* algorithm [1]. Specifically, the $Nearest(\cdot,\cdot)$, $Near(\cdot,\cdot,\cdot)$ and $c(\cdot)$ functions. The $Nearest(\cdot,\cdot)$ function is called when adding a new sampled state to the tree, and finds the single nearest neighbor from the existing motion tree to the new state. The $Near(\cdot,\cdot,\cdot)$ function used during the rewiring step finds a set of the k-nearest neighbors from the motion tree to a new state. The $c(\cdot,\cdot)$ function returns the cost between two states (or infinity if the path is invalid).

[1]We allow up to 12 iterations and maximum error of $\sqrt{3}$ m^2

			Number of nodes in motion tree (larger is better)			Seconds required to fly the path (smaller is better)		
Experiment	**Mean wind speed [m/s]**	**Max wind speed [m/s]**	**Iterative**	**NN**	**NN k-nearest**	**Iterative**	**NN**	**NN k-nearest**
1	0.0023	0.023	235 ± 6	**402 ± 6**	396 ± 6	**169.4 ± 10.2**	171.9 ± 14.6	169.9 ± 9.3
2	0.0029	0.063	247 ± 5	415 ± 8	**418 ± 7**	209.3 ± 38.2	**200.2 ± 41.4**	203.8 ± 42.8
3	3.1	4.2	162 ± 5	**326 ± 5**	300 ± 5	137.2 ± 27.1	**118.8 ± 18.7**	129.4 ± 19.0
4	3.2	5.2	127 ± 5	**263 ± 5**	229 ± 4	169.4 ± 33.2	**158.2 ± 27.4**	166.6 ± 32.0
5	7.7	11.6	132 ± 3	**254 ± 5**	225 ± 4	**87.7 ± 9.9**	90.2 ± 10.1	89.6 ± 8.5
Median			160	**326**	300	159.6	**148.1**	156.5

TABLE I: Mean and standard deviation of motion tree size and flight time for iterative and neural network path planning, in various wind conditions. The UAV airspeed is $15\,\mathrm{m\,s^{-1}}$. The iterative algorithm estimates the cost required for RRT* path planning using an iterative search for the wind-adjusted Dubins path. NN uses a neural network to approximate the cost and path validity for both $Near(\cdot,\cdot)$ and $Nearest(\cdot,\cdot)$ functions, or for only the $Near(\cdot,\cdot)$ function (k-nearest).

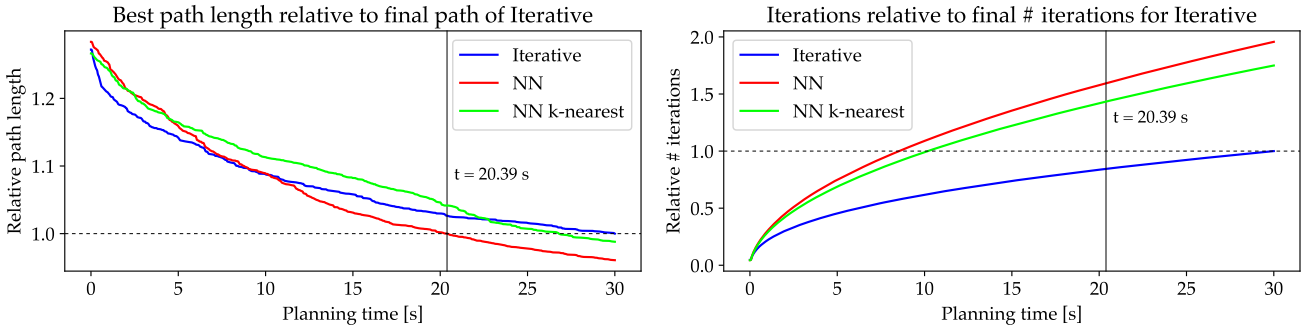


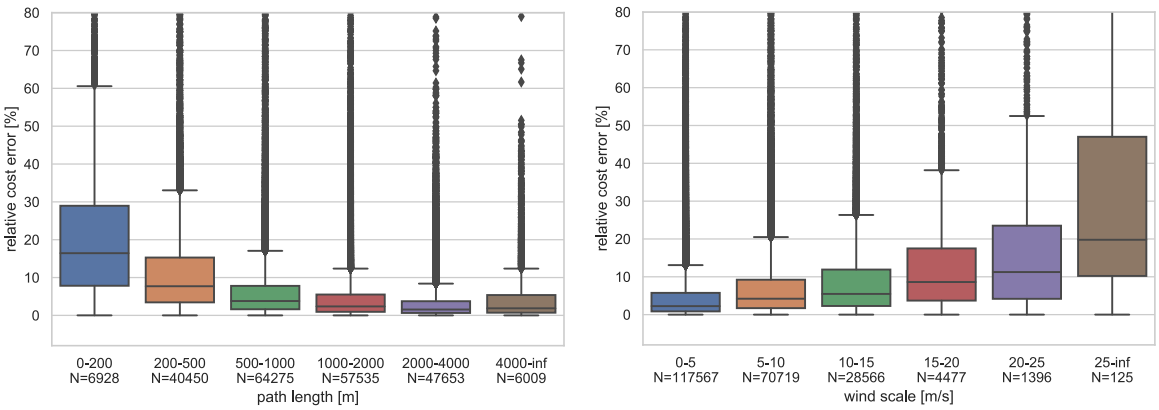
Fig. 4: Here we show the best path length from start to goal in the motion tree and the total number of RRT* iterations, for a fixed planning time of 30 s. Data consists of 5 experiments and 100 trials per experiment. Path length and number of iterations are normalized by the mean final path length (respectively, final number of iterations) of the iterative method *per experiment*. The plots show the mean across all experiments and all trials. Our method is able to run approximately twice as fast as the iterative method and achieves an equivalent path length after approximately 20 s.

We conducted five experiments, each using a different wind field selected specifically from the test dataset to analyze performance in various wind speeds and terrain features. Table I gives a comparison of how quickly the motion tree grows using an iterative solver versus a neural network in a fixed amount of time, as well as the time required for the UAV to fly the path. Statistics for each experiment and planner combination are calculated over 100 trials. In each trial, the planner is allowed to run for 30 s and the shortest path is extracted. The start and goal poses are fixed for each experiment, so variability across trials in an experiment is due to the random nature of RRT*. The neural network was evaluated on a CPU, because the model is small enough that overhead from transferring data to/from a GPU actually slows down inference. We also evaluated the path planner using the neural network *only* for the k-nearest neighbor search. Our hypothesis was that the nearest neighbor search might be sensitive to returning the wrong (not nearest) state resulting in inferior planning performance.

Over all experiments the neural network based approach resulted in 4 % shorter paths after 30 s of planning time. It required 32 % less time to find a solution compared to the baseline method with the full planning time, shown in Fig. 4. While the effects vary over the different wind fields we observed that the neural network based approach performance was comparable to or better than the baseline iterative method. The neural network performance is sufficient for the nearest neighbor search, thus using it for all neighborhood searches performs best.

Cost estimation and path validity prediction results: Here we examine the accuracy of our neural network. The relative cost error is defined as the ℓ_1 cost error divided by the ground-truth path length ($\frac{\|\hat{C}-C\|}{C}$). The network predicts the cost with a mean relative error of 3.52 %, which is sufficient for use in our planning framework. There exists a strong correlation between the relative cost error and the ground-truth length of the path as seen in Fig. 5a. Even a bias in the dataset towards short paths does not reduce the relative error on short paths below 8 %. Figure 5b shows a correlation between the average wind speed of the wind field and the relative cost error. The relative cost error is significantly higher for stronger winds. This may be due to a bias of the training dataset towards wind fields with weaker wind magnitudes. Furthermore the proposed model is able to predict the validity of a path with a mean accuracy of 92.95 %. Paths in very weak wind are usually valid and those in strong wind more often invalid, making validity easier to predict and reducing path validity loss compared to paths with medium wind speed (8–13 $\mathrm{m\,s^{-1}}$) which could be either valid or invalid.

Spatial analysis of relative cost error: As the start and goal poses are randomly sampled based on predetermined distributions, we investigated in detail the correlation between relative cost error and position of the start and goal pose in the wind grid. We conducted this analysis for a chosen subset of wind fields with different mean wind



(a) Relative cost loss distribution over path length

(b) Relative cost loss distribution over wind scale

Fig. 5: Distribution of relative cost error in bins of the ground-truth path length and wind scale. The relative cost error is defined as the ℓ_1 cost error divided by the ground-truth path length. The white dot marks the median, the black bar signifies the 25th and 75th percentile.

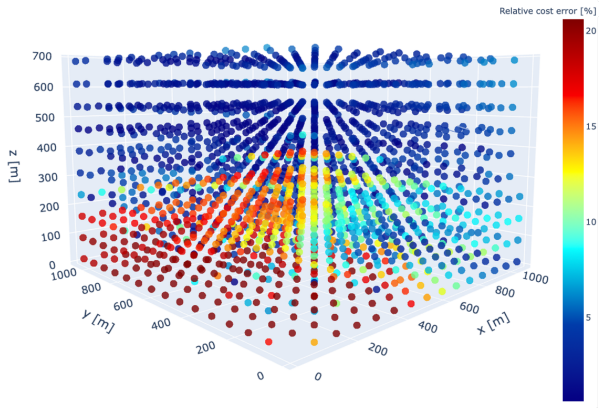


Fig. 6: Spatial distribution of relative cost error depending on position of goal pose in wind field with an average wind speed of $6.94\,\mathrm{m/s}$ (medium wind strength). The start pose it always set to $(0, 0, 0)$. Both start and goal pose have a heading of $0°$.

speeds. We always placed the start pose at grid coordinates $(0, 0, 0)$ with a heading of $0°$ and subsequently placed the goal pose at every position in the discretized wind field again with a heading of $0°$. This allowed us to analyse the correlation between relative cost error and the planar angle, planar distance and vertical distance between the start and goal poses.

Fig. 6 shows much higher relative cost errors for goal positions close to the start position at the origin of the coordinate system. In practice this poor cost estimate has a small effect in RRT*, as only $5.3\,\%$ of start-goal queries have a planar distance of $500\,\mathrm{m}$ or less. Additionally, the neural network is better at estimating the cost of paths with large altitude changes than paths with a low altitude difference between the start and goal poses.

V. DISCUSSION

The results in Section IV showed much higher relative cost errors on small paths and in wind fields with stronger wind speeds. Even after adding a bias to the dataset favoring short paths, the relative cost error for short paths remained significantly higher than for long paths. It is conceivable that our network is *too* lightweight to accurately solve a task of this difficulty. Additionally, due to the UAV's minimum turning radius, short paths may require unexpected or large maneuvers to reach the goal pose, which could be difficult for our network to predict.

It proved to be very beneficial to split the networks in two parts, as the encoding of the high dimensional wind input is computationally demanding and thus would have slowed down the cost estimation too much. Still, there exists an important trade-off between quality of prediction (stemming from the chosen number of trainable parameters) and the runtime of the network which correlates to the number of path planning iterations in a fixed time online on the hardware.

VI. CONCLUSIONS

We have proposed a deep-learning based approach to predict cost and path validity of time-optimal paths for a UAV in complex wind. Our model is able to predict the cost of Dubins paths with a mean relative error of $3.52\,\%$ and path validity with an accuracy of $92.95\,\%$. In a fixed planning time, our approach doubles the size of the RRT* motion tree compared to previous iterative methods and achieves an equivalent path length $32\,\%$ faster.

This work can be extended in multiple ways:

- A separate network with more trainable parameters could be used for short paths as the cost is much harder to estimate for those paths.
- As a further extension of our work, information about other airplanes or obstacles could be encoded to affect path validity and cost. Additionally, we see applications in aquatic domains to encode information about local currents and waves for cost estimation in path planning.

REFERENCES

[1] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[2] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 3276–3282.

[3] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[4] F. Achermann, N. R. Lawrance, R. Ranftl, A. Dosovitskiy, J. J. Chung, and R. Siegwart, "Learning to predict the wind for safe aerial vehicle planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2311–2317.

[5] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2379–2384.

[6] Y. Lin and S. Saripalli, "Path planning using 3D dubins curve for unmanned aerial vehicles," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 296–304.

[7] P. Oettershagen, F. Achermann, B. Müller, D. Schneider, and R. Siegwart, "Towards fully environment-aware UAVs: Real-time path planning with online 3D wind field prediction in complex terrain," *arXiv preprint arXiv:1712.03608*, 2017.

[8] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[9] M. Soulignac, "Feasible and optimal path planning in strong current fields," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 89–98, 2011.

[10] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331–341, 2007.

[11] A. Chakrabarty and J. Langelaan, "UAV flight path planning in time varying complex wind-fields," in *2013 American Control Conference*, 2013, pp. 2568–2574.

[12] N. R. Lawrance and S. Sukkarieh, "Path planning for autonomous soaring flight in dynamic wind fields," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2499–2505.

[13] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," in *IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1998, pp. 630–637.

[14] Y. Li and K. E. Bekris, "Learning approximate cost-to-go metrics to improve sampling-based motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4196–4201.

[15] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5021–5028.

[16] L. Palmieri and K. O. Arras, "Distance metric learning for RRT-based motion planning for wheeled mobile robots," in *2014 IROS Machine Learning in Planning and Control of Robot Motion Workshop*, 2014.

[17] L. Palmieri and K. O. Arras, "Distance metric learning for RRT-based motion planning with constant-time inference," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 637–643.

[18] M. Bharatheesha, W. Caarls, W. J. Wolfslag, and M. Wisse, "Distance metric approximation for state-space RRTs using supervised learning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 252–257.

[19] W. J. Wolfslag, M. Bharatheesha, T. M. Moerland, and M. Wisse, "RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1655–1662, 2018.

[20] R. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 1374.

[21] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[22] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5113–5120.

[23] H. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "RL-RRT: Kinodynamic motion planning via learning reachability estimators from rl policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[24] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6582–6588.

[25] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7087–7094.

[26] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[27] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, 2020.

[28] B.-A. Danciu, "Incorporating on-line measurements into deep neural network predictions for estimating wind near terrain with a fixed-wing UAV," Master's thesis, ETH Zürich, Switzerland, 2020.

[29] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 30, no. 1, 2013, p. 3.