2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Neural Trajectory Model: Implicit Neural Trajectory Representation for Trajectories Generation

Zihan Yu and Yuqing Tang*

Abstract— **The multi-agent trajectory planning problem is a difficult problem in robotics due to its computational complexity and real-world environment complexity with uncertainty, non-linearity, and real-time requirements. Many existing solutions are either search-based or optimization-based approaches with simplified assumptions of environment, limited planning speed, and limited scalability in the number of agents. In this work, we first attempt to reformulate single-agent and multi-agent trajectory planning problems as query problems over an implicit neural representation of trajectories. We formulate such implicit representations as Neural Trajectory Models (NTM) which can be queried to generate nearly optimal trajectory in complex environments. We conduct experiments in simulation environments and demonstrate that NTM achieve (1) sub-millisecond planning time using GPUs, (2) almost avoiding all collisions, and (3) generating almost shortest paths. We also demonstrate that the same NTM framework can also be used for refining low-quality and conflicting multi-agent trajectories into nearly optimal solutions efficiently. (Open source code is available at `https://github.com/laser2099/neural-trajectory-model`)**

I. INTRODUCTION

Trajectory planning research is usually conducted from two perspectives: single-agent and multi-agent trajectory planning. Single-agent trajectory planning entails finding trajectories between two positions in space along with the corresponding relative timestamps and is a fundamental, well-researched issue in artificial intelligence. Trajectory planning are typically achieved using search algorithms such as the A* series algorithms [1] and the RRT series algorithms [2]. Multi-agent trajectory planning (MATP) is an extension of single-agent trajectory planning that involves multiple agents. The objective is to determine paths along with relative timestamps for all agents from their respective starting points to their goals, ensuring that agents do not collide with the environment and other agents during their movements [3]. To solve the MATP problems, many state-of-the-art approaches require the use of searching-based algorithms to generate reference paths for further generation of multi-agent trajectories.

Although search-based algorithms perform well in some scenarios, they are often very time-consuming for single and multi-agent trajectory planning. In this paper, we introduce a novel method for single-agent and multi-agent trajectory planning by formulating these problems as queries over an implicit neural representation of trajectories. Using recent advances in neural representations, our approach efficiently encodes valid and nearly optimal trajectories and generalizes to new scenarios while maintaining constraints and optimality. Experiments in three different environments show that our method is at least 10 times faster than existing baseline approaches while achieving comparable planning performance. Our main contributions are summarized as follows:

- We present a novel reformulation of single-agent and multi-agent trajectory planning problems as query problems with implicit neural representation of valid and optimal trajectories in a given environment.
- Our method demonstrates fast planning speed and superior performance for both single-agent and multi-agent trajectory planning.
- We also demonstrate that the same neural formulation can be employed for multi-agent coordination and trajectories de-conflict.

Zihan Yu is with the Systems Hub, The Hong Kong University of Science and Technology(Guangzhou). This work was conducted while the first author was doing internship at IDEA.

Yuqing Tang is with the International Digital Economy Academy.

*Corresponding author is Yuqing Tang (e-mail: tangyuqing@idea.edu.cn).

II. RELATED WORK

A. Search-based single agent trajectory planning

Search-based path planning algorithms are extensively used in path and trajectory planning due to their ease of implementation. Algorithms such as A* and RRT were developed to tackle various planning challenges. Over the years, variious improved versions have emerged, Hybrid A* [4], RRT* [5], and LPA* [6]. Despite their enhanced performance, the running time remains a significant concern because these algorithms require repeated searches. To address this issue, we utilize a neural trajectory model that leverages recent advances in neural networks, thus avoiding redundant searches and achieving significant performance improvement.

B. Multi-agent trajectory planning

Multi-agent trajectory planning (MATP) problem is an extension of the single-agent trajectory planning problems. Numerous researchers attempt to solve the MATP problems by leveraging single-agent planning algorithms. The Conflict-based search (CBS) algorithm [7] conducts searches on a constraint tree (CT). EDG-TEAM [3] employs ECBS to generate initial paths and then implements a group-based optimization to further ensure the algorithm's performance. Game theoretic approaches have also been proposed to model the interactions between multi-agent [8], [9]. Search-based methods rely on simplifying assumptions about the environment. Game-theoretic methods, while addressing interactions between agents, involve complex calculations that

limit planning speed not meeting real-time requirements. In this work, we circumvent extensive searching by employing a neural trajectory models.

C. Neural Models

Neural environment representations employ neural networks to represent the geometry, and occasionally the color and texture, of intricate 3D scenes. Typically, these representations is trained on a labeled dataset to learn a function in the form of $f_\theta(p) = \sigma$, where f denotes a neural network parameterized by the weights θ, p signifies a low-dimensional query such as a 3D coordinate (x, y, z), and σ represents relevant a single scalar or a vector of scalars of measurement. Existing neural implicit representations can be classified into generalizable paradgims [10], [11] and overfit paradigms [12], [13]. The latter approach focuses on accurately representing a specific given environment by deliberately overfitting a neural network to the data collected from the environment. In this work, we adopt the implicit representation paradigm of neural models. Specifically, we extend the neural fields formulation to accommodate trajectories — overfitting a multi-trajectory function for a given environment so that valid and nearly optimal trajectories of the environment can be queried with their start-end positions as input.

III. Neural Trajectories

A. Problem Formulation

In this section, we present the problem formulation for generating desired trajectories in a multi-agent system of N agents. The goal for the ith agent is to move from a starting position s^i to a goal position g^i. The environment is represented by a Signed Distance Field (SDF) [14]. $SDF(x)$ determines the distance to the closest surface of any 3D coordinate x in the environment where (1) $SDF(x) < 0$ indicates that x is inside an object; (2) $SDF(x) = 0$ indicates that x is on the surface of an object; (3) $SDF(x) > 0$ indicates that x is outside any objects. Such an SDF can be derived from a 3D mesh model, occupancy grids, CSG (constructive solid graph) and other 3D environment representations.

A trajectory $\mathbb{W}^i$ for agent i is a sequence of time-stamped waypoints: $\mathbb{W}^i := \{(t_i^0, p_i^0), (t_i^1, p_i^1), ..., (t_i^T, p_i^T)\}$ where (1) the 4D waypoint $w_i^j = (t_i^j, p_i^j)$ is the jth waypoint ($j \in \{0, ..., T\}$) with t_i^j and p_i^j being the jth timestamp and jth coordinate; (2) $s_i = p_i^0$ and $g_i = p_i^T$ are the starting and the goal positions; (3) T is the time horizon for the trajectory.

A trajectory $\mathbb{W}^i$ is considered environmental collision-free if all its waypoints are in free space with a safety threshold S_{thresh}: i.e. $SDF(p_i^j) > S_{thresh}$ for all $p_i^j \in \mathbb{W}^i$. Two trajectories $\mathbb{W}^i$ and $\mathbb{W}^k$ are considered inter-collision free if any pair of waypoints $(t_j^i, p_j^i) \in \mathbb{W}^i$ and $(t_{j'}^k, p_{j'}^k) \in \mathbb{W}^j$ are conflict-free: (1) time separated with a threshold t_{thresh}: $|t_j^i - t_{j'}^k| > t_{thresh}$; or (2) distance separated (Euclidean distance) with a safety threshold dis_{thresh}: $\|p_j^i - p_{j'}^k\| > dis_{thresh}$.

A multi-agent trajectory generation problem is to generate a set of N trajectories $\mathbb{W} = \{\mathbb{W}^1, \mathbb{W}^2, ..., \mathbb{W}^N\}$ given the starting and goal positions $\{(s^1, g^1), (s^2, g^2), ..., (s^N, g^N)\}$ of N agents. The requirement is that all trajectories in $\mathbb{W}$ are both environmental collision-free and inter-collision-free. We seek to learn a neural function f_Θ parameterized by Θ to generate such a trajectory set:

$$\mathbb{W} = f_\Theta(\{(s^1, g^1), (s^2, g^2), ..., (s^N, g^N)\}). \quad (1)$$

B. Neural Trajectory Model

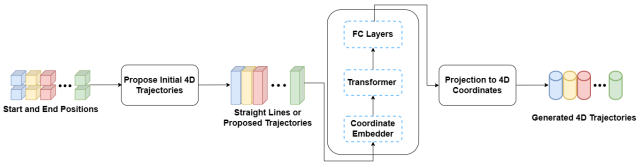

Fig. 1: Neural Trajectory Model

As shown in Figure 1, our neural trajectory model accepts a set of N start-end positions as input and output N corresponding 4D trajectories. The 4D trajectories proposing step is first employed to propose an initial set of 4D trajectories for N start-end positions. Without the loss of generality, we employ a simple line interpolation for initial trajectories proposal. T discrete 3D points are sampled with uniform intervals along a straight line between the start and end position (s, g) as follows: $l_{s\to g} = \left\{p_j = s + (g - s)/T \times j\right\}_{j=0}^{T}$ where $l_{s\to g} \in \mathbb{R}^{(T+1)\times 3}$ serves as the initial path proposal. Equally separated time-stamps $\{t_0, ..., t_T\}$ are sampled between 0 and T and prepend to the path proposal $l_{s\to g}$ to form the trajectory proposal:

$$L_{s\to g} = \{(t_j, p_j)\}_{j=0}^{T} \quad (2)$$

Other initial trajectories proposal methods can also be used. For example, in Section IV-F we demonstrate to feed trajectories created by other methods or a single-agent neural trajectory model into the transformer as initial trajectories to generate desired high-quality trajectories. We adapt the positional embedding approach in the original transformer paper [15] to 4D positional embeddings.

$$E_l = CoordinateEmbedder(L_{s\to g}), \quad (3)$$

Coordinate embeddings are then fed into a transformer [15], conducting attention mechanism to capture the relationship among coordinates of the proposed trajectories. At the end, a fully-connection layer is applied to project the transformer's vector sequence outputs to 4D coordinates sequences of time horizon T for the N agents satisfying desired properties:

$$f_\Theta = FC(transformer(E_l)). \quad (4)$$

Although this work focuses on a static environment trajectory generation, it can be easily extended to accommodate a dynamic environment with sensory input.

C. Model Training

As shown in Figure 2, the neural trajectory model is trained with ground truth trajectories data along with additional regularization of collision-free and performance metrics (.e.g. travel distance, time, etc.) requirements. The training data D is of the from $D = \{(SG_j, \mathbb{W}_j)\}$ where each SG_j

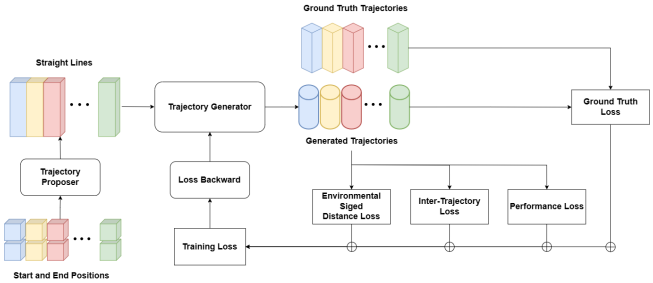

Fig. 2: Neural Trajectory Model Training

is N start-goal positions and each $\mathbb{W}_j$ is the corresponding ground-truth multi-agent trajectories which satisfy collision-free and the desired performance requirement. The goal of this training process is to enable the model to generalize the trajectory generation beyond the seen start-goal positions to unseen start-goal positions in the same environment. We will introduce the training loss functions first and describe the ground truth data curation processes in Section III-F.

Ground Truth Supervision: We compute the point-wise L1 losses between the model generated trajectories and ground-truth trajectories to enable fast convergence to plausible trajectories.

$$l_{gt} = \frac{1}{T \times N} \sum_{i=1}^{N} \sum_{j=1}^{T} (\|\hat{w}_j^i - w_j^i\|_1) \tag{5}$$

where $\hat{w}_j^i$ is the jth model generated trajectory waypoint and w_j^i is the corresponding ground truth for the ith agent.

Environmental Safety Loss: We point-wisely compute signed distance values of trajectories towards the environment obstacles and ensure a safety distance margin threshold from every waypoints to the obstacles. We adapt the differential SDF implementation in the Kaolin library [16] to enable loss back-propagation of the environmental safety loss. Neural SDFs [10] can also be used here.

$$l_{sdist} = \frac{1}{T \times N} \sum_{i=1}^{N} \sum_{j=1}^{T} max\{(S_{thresh} - SDF(p_j^i), 0\}. \tag{6}$$

Inter-Trajectory Conflict Loss: We ensure distances between any two waypoints of two trajectories are separated by at least $w_{thresh} = \langle t_{thresh}, dist_{thresh} \rangle$ by the following inter-collision loss:

$$l_{inter} = \frac{1}{T \times N} \sum_{i,k=1}^{N} \sum_{j,j'=1}^{T} max(0, w_{thresh} - \|w_j^i - w_{j'}^k). \tag{7}$$

Performance Loss: We ensure the generated trajectories to be at least as good as the ground-truth length $\hat{d}^i$ as an example performance metric loss:

$$l_{dist} = \frac{1}{N} \sum_{i=1}^{N} max(0, \hat{d}^i - d^i) \tag{8}$$

where $\hat{d}^i$ and d^i are the ground truth and the predicted travel distance of agent i respectively.

The overall training objective is as follows:

$$loss = \lambda_1 l_{gt} + \lambda_2 l_{sdist} + \lambda_3 l_{inter} + \lambda_4 l_{dist} \tag{9}$$

where λ_1, λ_2, λ_3 and λ_4 are the corresponding combination weights.

D. Neural Trajectory Inference

The inference process of NTM is shown in Algorithm 1. It consists of two main steps and an optional optimization step. The first two steps are the trajectory proposal step and the neural generation step. An optional optimization step (section III-E.) can be employed to further optimize the output.

Algorithm 1: Neural Trajectory Inference

Input: Start-Goal positions:
$SG_{pairs} = \{(s^1, g^1), (s^2, g^2), ..., (s^N, g^N)\}$
Output: Trajectories: $\mathbb{W} = \{\mathbb{W}^1, \mathbb{W}^2, ..., \mathbb{W}^N\}$

$InitTrajs \leftarrow GetLines(SG_{pairs})$;
$\mathbb{W} \leftarrow NeuralTrajectory(InitTrajs)$;
if *further_optimize* ***is*** *True* **then**
  /* Optional trajectory optimization step */
  $\mathbb{W} \leftarrow TrajectoriesOptimizer(\mathbb{W})$;
end
return $\mathbb{W}$;

Multi-trajectories Coordination and De-Conflicting: The neural trajectory model can not only support trajectory planning but also trajectory de-conflicting. With coordination tasks, the inference procedure in Algorithm 1 will take trajectories with conflicts or non-optimal trajectories as input and output better conflict-resolved and better optimized trajectories.

E. A Trajectory Optimizer

Algorithm 2: Trajectories Optimizer

Input: Initial Trajectories: $\mathbb{W}^0 = \{\tau^1, \tau^2, ..., \tau^N\}$
Output: Optimized Trajectories: $\mathbb{W}^*$

$\mathbb{W} \leftarrow \mathbb{W}^0$;
repeat
  /* w_j^is are 4D waypoints in $\mathbb{W}$ */
  $\nabla_{w_i^j} \leftarrow \frac{\partial}{\partial w_i^j} J(\mathbb{W})$;
  $\mathbb{W} \leftarrow \mathbb{W} - \gamma \nabla_{w_i^j}$;
until *desired trajectories are obtained or time-out*;
return $\mathbb{W}$;

The Trajectory Optimizer as specified in Algorithm 2 makes use of the following loss functions to optimize proposal trajectories:

$$J = \lambda_1 l_{sdist} + \lambda_2 l_{inter} + \lambda_3 l_{dist} \tag{10}$$

where λ_1, λ_2 and λ_3 are the corresponding combination weights. l_{sdist} (Equation 6), l_{inter} (Equation 7), and l_{dist}

(Equation 8) are the environmental loss, the inter-trajectory collision loss and the performance loss defined in Section III-C. In Algorithm 2 line 3 takes gradients with respect to trajectories waypoint coordinates instead of taking gradients with respect to neural model parameters as in Section III-C. γ in Line 4 is the gradient descent update learning rate.

F. Training Data Curation

To support the training process described in Section III-C, we devise the data curation process as in Algorithm 3. It requires an environment model *ENV*. We sample from the environment M start-goal positions. Then we apply a trajectory proposer (Line 3) to generate initial trajectories for optimization. We use multi-agent A-Star algorithms [1] as our trajectories proposer. In particular, we use the implementation available in [17]. Then the trajectory optimizer described in Algorithm 2 is applied to optimize the trajectories to obtain nearly optimal trajectories with no environmental and inter-agent collisions.

Algorithm 3: Training Data Generation Process

Input: An environment SDF model *SDF*
Output: Trajectories Dataset: $D = \{(SG_j, \mathbb{W}_j)\}$
1 **repeat**
2 Sample $SG_j = \{(s_j^1, g_j^1), ..., (s_j^N, g_j^N) \mid SDF(s_j^i) > 0, SDF(g_j^i) > 0\}$ from the environment model *ENV* ;
3 $\mathbb{W}^0 \leftarrow TrajectorProposer(SG_j)$;
4 $\mathbb{W}_j \leftarrow TrajectoryOptimizer(\mathbb{W}^0)$;
5 $D \leftarrow D \cup \{(SG_j, \mathbb{W}_j)\}$;
6 **until** *M data instances are obtained or time-out*;
7 **return** D

IV. Experiments

In this section, we conduct experiments to validate the proposed methods. Experiment are run on a workstation equipped with an Intel(R) Xeon(R) Gold 5218 CPU and an RTX 3090 GPU.

A. Evaluation Metrics

We use the following evaluation metrics to quantitatively compare our approaches with baselines. (1) **Environmental Collision Rate (ECR)**: ECR is the ratio of the number of collision trajectories to the total number of trajectories. (2) **Inter Collision Rate (ICR)**: ICR is the ratio of the number of inter-trajectory collisions to the total number of trajectories. (3) **Travel Distance (TD)**: Travel Distance refers to the average length of all trajectories in a trajectory planning task. (4) **Calculation Time (CT)**: Calculation time, measured in seconds (s), is the time required to generate trajectories for agents.

It is also important to note that although the search-based baselines are sound, they don't guarantee completeness for collision-free paths. In our study, we treated unsolvable cases within given computational cost as collision cases.

B. Data

We employ the data curation process described in Section III-F to generate ground truth data for our experiments. Their statistics are shown in Table I.

TABLE I: Experiment Datasets

Environment	Single-agent			Multi-agent		
	train	valid	test	train	valid	test
Stonehenge	512	110	128	324	51	72
Ice-forest	/	/	/	1003	197	203
Building-forest	/	/	/	1710	317	330

C. Single Agent Trajectory Planning

The neural trajectories can be applied to perform single-agent trajectory planning. We conduct single-agent experiments in the Stonehenge environment from [17]. We compare the proposed method with Nerf-Nav [17], an advanced planning algorithm and the classical search-based algorithm A* [1] (implementation of [17]). In this scenario, the NTM is trained with 512 trajectories. We evaluate the models on 128 start and end position pairs, which were randomly sampled from the environment, excluding the training data. The evaluation results are presented in Table II.

TABLE II: Single-agent Experiments in Stonehenge Env

Method	ECR	TD	CT
Nerf-Nav [17]	0.0653	1.2567	2.0
A* [1]	0.104	**1.098**	0.1840
NTM(Ours)	**0.054**	1.1448	**0.0025**
Ground Truth	0.0	1.1011	/

From the results, we can see that NTM can generate trajectories much faster, resulting in shorter and safer routes compared to the baseline methods.

D. Multi-agent Trajectory Planning

For MATP, we experiment with an eight-agent trajectory planning problems. We compare our method with a SoTA approach EDG-TEAM [3] and a classical MATP algorithm ECBS [18]. Evaluations are conducted across three distinct environments: Stonehenge [17], Ice Forest [3], and Building Forest Environment (re-created and adapted from [19]).

For the Stonehenge environment, we train the NTM on 324 trajectories and test it on 72 distinct start-end position pairs. In the ice forest environment, the NTM is trained on 1003 trajectories and evaluated on 245 unique start and end position pairs. Lastly, in the building forest environment, the NTM is trained on 1710 trajectories and assessed on 330 different start and end position pairs. The testing start and end position pairs are randomly chosen from the environment, excluding the training data. The results are provided in Table III.

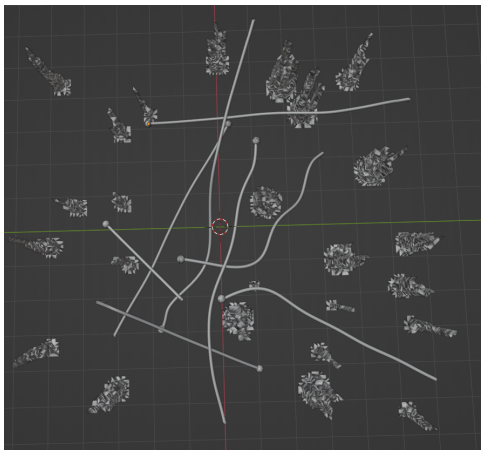

Fig. 3: Eight-agent trajectory planning in Ice Forest Env

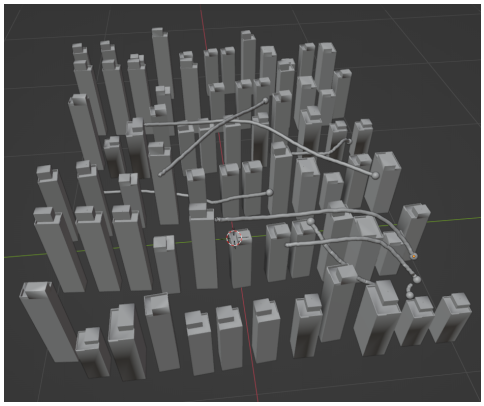

Fig. 4: 8-agent trajectory planning in Building Forest Env

TABLE III: Comparison of eight-agent planning in a Stonehenge, Ice Forest, Building Forest environment

	Method	ICR	ECR	TD	CT
Stonehenge	EDG [3]	0.048	0.312	**1.705**	0.017
	ECBS [18]	0.088	0.325	1.992	0.21
	NTM (Ours)	**0.032**	**0.027**	1.877	**0.0025**
	Ground Truth	0.0	0.0	1.598	/
Ice forest	EDG [3]	0.033	0.146	5.717	0.022
	ECBS [18]	0.0412	0.187	5.871	0.32
	NTM (Ours)	**0.022**	**0.0**	**5.179**	**0.0021**
	Ground Truth	0.004	0.004	5.109	/
Bldg forest	EDG [3]	0.0521	0.083	1.115	0.042
	ECBS [18]	0.121	0.107	1.219	0.35
	NTM (Ours)	**0.024**	**0.012**	**0.994**	**0.0025**
	Ground Truth	0.009	0.003	0.924	/

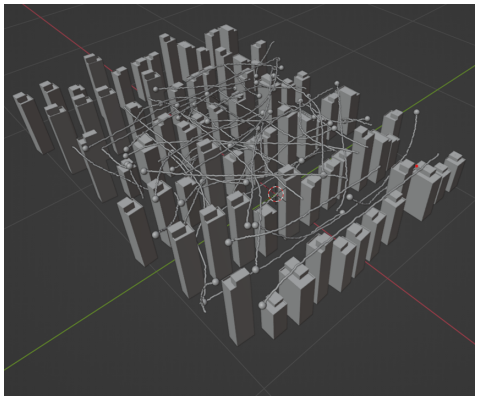

Fig. 5: Large-scale trajectory planning in Building Forest Env

Compared to the baseline methods, the NTM achieves a significant improvement in computation time, being at least 10 times faster (note that the exact number is not meaningful; only the relative magnitude is relevant as the CPU and GPU implementation are not directly comparable). Moreover, the NTM outperforms other methods in other metrics as well, demonstrating the advantages of learning from known trajectory experiences and generalizing beyond them to unseen scenarios in the same environment.

E. Towards Large Scale Trajectories Models

We also conducted preliminary experiments to assess the scalability of our approach when dealing with a substantial number of agents. In this particular experiment, we simulated scenarios representing future air traffic. Given 64 pairs of start and end positions, the NTM successfully generates collision-free trajectories. We also modify the vanilla transformer [15] to support large number coordinates attention across agents[1]. Figure 5 depicts a screenshot of these scenarios.

We further compare the computation time as the number of agents increases, as illustrated in Table IV. As we can see in the table the computation time is not sensitive to the number of agents with our specific implementation of multi-trajectory transformer attention patterns in NTM. This demonstrates the potential for solving large-scale agent trajectory planning problems. We will conduct further research in this direction to fully understand the potential of this approach.

[1]See our open source code for more details.

F. Towards Multi-agent Trajectory Coordination

Apart from trajectory planning, we conduct experiments in Building Forest to explore the potential usage of the NTM in multi-agent coordination. Given 61 batches trajectories with collisions, we employ the inference process (Algorithm 1) described in Section III-D with the trajectory proposal step replaced by given trajectory sets. The implementation can solve conflicts and collisions. The coordination result is depicted in Table V. As we can see in the results, the collision rates are reduced significantly to almost no collisions. More comprehensive experiments will be conducted to fully explore the potential of NTM in this area.

G. Ablation Study on Inference Procedure

To further illustrate the impact of additional trajectory optimization refinement, we compare the metrics before and after performing the refinement process(Algorithm 1). The results in Building Forest Environment are displayed in Table VI. From the result, we can observe that the performances

TABLE IV: Computation time vs. number of agents

	Single-agent	Eight-agent	64-agent
CT(s)	0.00253	0.00258	0.00272

TABLE V: Multi-Trajectories Coordination

	ICR	ECR
w/ Coordination	0.8688	0.1475
w/o Coordination	0.0164	0.0328

of trajectories are improved with the trajectory optimizer, demonstrating the effectiveness of trajectory refinement and the possibilities of applying neural trajectories along with an appropriate choice multi-trajectory optimizer to balance between computation speed and solution qualities.

H. Limitations

Although NTM achieves superior performance, there also exist some limitations. Currently, NTM is designed primarily for static environments with fixed SDF, which limits its effectiveness for dynamic collision avoidance tasks. Additionally, NTM does not incorporate motion and physical dynamics, which may lead to potential issues with dynamic feasibility. Future research will focus on three main areas: (1) conducting a comprehensive study on the application of NTM to large-scale multi-agent planning and coordination problems, (2) extending NTM to handle dynamic environmental models that incorporate sensing inputs, and (3) experimenting with real-world scenarios by integrating motion and physical dynamics variables into trajectory representations to improve the handling of physical environment interactions.

V. CONCLUSION

In this study, we introduce a neural trajectory model designed to generate nearly optimal trajectories for both single-agent and multi-agent scenarios. Our approach extends the neural field formulation to represent valid and nearly optimal trajectories in complex environments through an implicit neural representation. Additionally, the model supports generating conflict-free trajectories by querying the model with start-goal positions. Through extensive simulation experiments, we demonstrate that NTM achieves (1) sub-millisecond planning times when utilizing GPUs, (2) near-complete avoidance of environmental collisions, (3) near-complete avoidance of inter-agent collisions, and (4) the generation of nearly shortest paths. Furthermore, we illustrate that the NTM framework is also effective for trajectory correction and multi-trajectory conflict resolution, transforming low-quality and conflicting multi-agent trajectories into nearly optimal solutions with high efficiency.

TABLE VI: Trajectory refinement Optimization Performance

	ICR	ECR	TD	CT
w/o Opt	0.024	0.012	0.994	0.0025
5-step Opt	0.021	0.009	0.962	0.1652
10-step Opt	0.018	0.005	0.943	0.3207

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] Z. Yu and L. Xiang, "NPQ-RRT*: An improved RRT* approach to hybrid path planning," *Complexity*, vol. 2021, pp. 1–10, 2021.

[3] J. Hou, X. Zhou, Z. Gan, and F. Gao, "Enhanced decentralized autonomous aerial robot teams with group planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9240–9247, 2022.

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[5] 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.

[6] S. Koenig and M. Likhachev, "Incremental A*. inadvances in neural information processing systems," 2002.

[7] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[8] Q. Zhang, R. Langari, H. E. Tseng, D. Filev, S. Szwabowski, and S. Coskun, "A game theoretic model predictive controller with aggressiveness estimation for mandatory lane change," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 1, pp. 75–89, 2019.

[9] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, "Deep interactive motion prediction and planning: Playing games with motion prediction models," in *Learning for Dynamics and Control Conference*, pp. 1006–1019, PMLR, 2022.

[10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

[11] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pp. 608–625, Springer, 2020.

[12] Q. Zhang, J. Hou, Y. Y. Adikusuma, W. Wang, and Y. He, "NeuroGF: A neural representation for fast geodesic distance and path queries," *arXiv preprint arXiv:2306.00658*, 2023.

[13] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural geometric level of detail: Real-time rendering with implicit 3D shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11358–11367, 2021.

[14] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, no. 2, pp. 158–175, 1995.

[15] 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.

[16] C. Fuji Tsang, M. Shugrina, J. F. Lafleche, T. Takikawa, J. Wang, C. Loop, W. Chen, K. M. Jatavallabhula, E. Smith, A. Rozantsev, O. Perel, T. Shen, J. Gao, S. Fidler, G. State, J. Gorski, T. Xiang, J. Li, M. Li, and R. Lebaredian, "Kaolin: A pytorch library for accelerating 3D deep learning research." `https://github.com/NVIDIAGameWorks/kaolin`, 2022.

[17] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[18] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 5, pp. 19–27, 2014.

[19] "Free3D." `https://free3d.com/`. Accessed: 2023-08-12.