ODESolvers are also Wayfinders: Neural ODEs for Multi-Agent Pathplanning

Dwip Dalal*
Indian Institute of Technology, Gandhinagar
dwip.dalal@iitgn.ac.in

Progyan Das*
Indian Institute of Technology, Gandhinagar
progyan.das@iitgn.ac.in

Anirban Dasgupta
Indian Institute of Technology, Gandhinagar
anirbandg@iitgn.ac.in

Abstract

Multi-agent path planning is a central challenge in areas such as robotics, autonomous vehicles, and swarm intelligence. Traditional discrete methods often struggle with real-time adaptability and computational efficiency, emphasizing the need for continuous, optimizable solutions. This paper introduces a novel approach that harnesses Neural Ordinary Differential Equations (Neural ODEs) for multi-agent path planning in a continuous-time framework. By parameterizing agent dynamics using neural networks within these ODEs, we enable end-to-end trajectory optimization. The inherent dynamics of ODEs facilitate collision avoidance. We demonstrate our method's effectiveness across both 2D and 3D scenarios, navigating multiple agents amidst obstacles, underscoring the potential of Neural ODEs to transform path planning.

1 Introduction

Path-planning refers to the problem of finding a valid, low-cost curve connecting two points on an environmental map. Multi-agent path-planning, therefore, refers to the case where multiple agents are involved, and the paths are non-colliding. Multi-agent path planning is a well-established NP-hard problem, and recent techniques have often resorted to deep learning. One of the major challenges of multi-agent path planning is to avoid the collision of two agents - especially where there is no possibility of communication between them.

We model the path-planning environment using ordinary differential equation ϕ represented by a vector field $\mathcal{F}$. Since each point r in is associated with a unique vector $\mathcal{F}(r)$, the intersection of field lines would result in a contradiction. Hence, field lines never intersect. We exploit this property of vector fields to ensure that points that travel based on ϕ never collide with each other. The Neural ODE, represented by $\frac{dx}{dt} = \mathcal{F}(x)$ (where $\mathcal{F}(x)$ is a neural network), allows us to train the vector field $\mathcal{F}$. The ϕ(x,t), takes as argument the starting position(s) x_0 and a time period t, and returns $\phi(x_0, t)$, the position(s) of x_0 once it has flowed through ϕ for time t. We can then reduce the loss between $\phi(x_0, t)$ and the set of target destinations, y_0.

Our framework is highly adaptable and capable of incorporating various objectives and constraints without requiring a complete algorithmic redesign. Bridging the gap between the rich theoretical underpinnings of differential equations and the practical needs of multi-agent systems opens new avenues for research in robotics, optimization, and machine learning. To the best of our knowledge,

*Equal Contribution

DLDE-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

this is the first work that has addressed the challenge of path planning with Ordinary Differential Equations.

2 Related Works

The landscape of Multi-Agent Path Planning (MAPP) has evolved significantly with the integration of data-driven models, outperforming traditional algorithms like Dijkstra's and A*. [7] presents a well-structured optimization framework albeit with scalability limitations. On the other hand, the work in [4] introduces Symplectic networks to solve high-dimensional optimal control problems efficiently, especially notable in scenarios requiring energy conservation. Reinforcement learning, as utilized in [1] and [3], demonstrated scalable and dynamic path planning solutions, with the actor-critic architecture facilitating effective exploration and exploitation. The incorporation of LSTM units for spatial encoding in MAPP, as presented in [9], allows for capturing temporal dependencies and scalability across numerous agents and dimensions. The roadmap-based approach in [8] and the large neighborhood search techniques in [2] highlight cooperative behaviors and significant speed improvements respectively. Furthermore, the exploration of overlooked constraints in MAPP by employing world-line concepts from Special Relativity in [5] unveils new possibilities in modeling agents' trajectories. Lastly, the 'Flow-Planning' approach utilizing Neural Ordinary Differential Equations (NODEs) aims to bridge machine learning efficiency with optimization theory rigor, inspired by recent advancements in NODEs as seen in [6] which proposes a learning-based modular motion planning pipeline using NODEs for ensuring stability and safety during task execution.

3 Methodology

In this section, we describe the end-to-end methodology for training the pathfinding flow over a particular environment, beginning with constructing the SDF for a given environment, establishing a flow and registering start and target positions for each agent, and then training the Neural ODE to reach the target position without entering forbidden regions of the map.

Problem Statement We address the problem of learning optimal paths in a multi-dimensional space filled with obstacles and targets. Given an initial point A and a target point A', the objective is to find a trajectory that not only minimizes the distance between the end point of the trajectory and A', but also avoids obstacles and maximizes the curvature of the path.

We employ Neural ODEs to model the continuous-time dynamics of the system. A Neural ODE is an ODE where the derivative function f is parameterized by a neural network. Formally, given an initial point A, the trajectory τ_A (where we use τ to denote trajectory) is obtained by solving the following ODE:

$$\frac{d\tau_A}{dt} = f(\tau_A, t; \theta) \tag{1}$$

where θ represents the parameters of the neural network. The optimization objective consists of multiple terms, each serving a specific purpose described below.

End Point Loss The end point loss $loss_A$ is computed as the Mean Squared Error (MSE) between the last point of the trajectory $\tau_A[-1]$ (where τ denotes the trajectory) and the target point A'.

$$\mathcal{L}_{MSE} = MSE(\alpha \times \tau_A[-1], \alpha \times A') \tag{2}$$

Obstacle Avoidance (SDF-based Loss) The SDF-based loss $loss_B$ ensures that the trajectory avoids obstacles. For each point p in τ_A, we compute its Signed Distance Function (SDF) value. We add a sharp sigmoid drop-off with a constant β to both heavily penalize trajectories that fall even marginally inside the obstacle, and to ignore trajectories that fall even marginally outside.

$$\mathcal{L}_{SDF} = sigmoid(-\beta \times SDF(p)) \tag{3}$$

Curvature Regularization To encourage the trajectory to curve, we introduce the curvature κ at each point in τ_A, computed as:

$$\kappa = \frac{dx \times d^2y - dy \times d^2x}{(dx^2 + dy^2)^{1.5}} \tag{4}$$

The curvature regularization is then the negative sum of the absolute curvature values:

$$\mathcal{L}_\kappa = -\sum |\kappa| \tag{5}$$

The total multi-objective loss is given by:

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda \mathcal{L}_{SDF} + \gamma \mathcal{L}_\kappa \tag{6}$$

3.1 Fast Marching Method for fast SDF calculation

The Signed Distance Function (SDF) provides a way to represent the geometry of a binary image. It assigns each pixel a value that corresponds to the shortest distance from that pixel to the boundary of the object. Positive values are assigned to pixels inside the object, and negative values are assigned to pixels outside the object.

1. For each pixel $p = (x, y)$ inside the object (black pixel), find the minimum Euclidean distance $d(x, y)$ to a background pixel (white pixel). Set $\phi(x, y) = d(x, y)$.
2. For each pixel $p = (x, y)$ outside the object (white pixel), find the minimum Euclidean distance $d(x, y)$ to an object pixel (black pixel). Set $\phi(x, y) = -d(x, y)$.

The time complexity of this method is $O(N^2)$ for an image with N pixels, which makes it computationally expensive for large images. We can alleviate this problem with a technique called the Fast Marching Method (FMM), which is as follows –

1. Initialize SDF with large positive and negative values for object and background pixels, respectively.
2. Start with an initial set of points (usually the boundary of the object) and assign them an SDF value of zero. Add these points to a priority queue.
3. Use the priority queue to propagate the SDF values to the neighboring pixels. Update the SDF values according to the Eikonal equation

$$\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2 = 1$$

4. Continue the process until the priority queue is empty.

The time complexity of the Fast Marching Method is approximately $O(N \log N)$ for an image with N pixels, making it more efficient than the brute-force method. We present here a sample environment and its SDF.

3.2 Establishing a path-planning flow.

In our approach, we leverage Neural Ordinary Differential Equations (Neural ODEs) to generate flows in n-dimensional spaces. A flow is formally defined as a continuous, time-parameterized mapping from a state space to itself. Explicitly, a flow on a set X is a mapping, ϕ, such that,

$$\phi : X \times \mathrm{R} \to X$$

such that, for all $x \in X$ and some $s, t \in \mathrm{R}$, we have $\phi(x, 0) = x$ and $\phi(\phi(x, t), s) = \phi(x, t + s)$. The second statement, essentially, would be the composition of two flows. A Neural ODE allows us to parameterize and train over these flows, by modeling the dynamics of a point, or multiple points, over the flow.

Given an initial point $\mathbf{x}_0$ in $\mathbb{R}^n$, our Neural ODE model describes the evolution of this point through the state space by solving the ordinary differential equation $\frac{d\mathbf{x}}{dt} = f_\theta(\mathbf{x}, t)$. Here, f_θ is a neural network parameterized by θ, and t serves as the continuous time variable. This formulation inherently ensures the smoothness and continuity of the generated flow. The neural network f_θ is trained to minimize the `MSELoss` between the end of the trajectories $\phi(X, t)$ of the agents, (X being the starting positions of each agent) and the corresponding target destinations $\mathcal{T}$.

However, that only ensures that the agents will move from one point to another – we also need to constrain them to the non-forbidden regions of the environment. To make sure the agents don't flow over the forbidden regions of the environment, we sample points $\mathcal{P}$ from the trajectory of the agents and set the loss function to $\Sigma_{i=1}^{n} SDF(P_i)$, where $SDF(kx)$ is the signed distance function corresponding to the environment, where x is the coordinate of the agent and k is a sharpness constant, that determines how sharply the distance function falls off after encountering an obstacle.

Therefore, the resultant loss function $\mathcal{L}$ is,

$$\mathcal{L} = MSE_{Loss}(\phi(X,t), \mathcal{T}) + \alpha \Sigma_{i=1}^{n} SDF(P_i),$$

where α is a constant that helps determine the right weights for the losses. In the case of coincident target destinations, i.e., in the case where we want $\phi(x_0, t) = \phi(x_1, t)$ in the solution, an approximate answer can be achieved by randomly perturbing the ending points so they do not end up at the exactly same location.

The optimization problem becomes somewhat more complex as the number of agents increases, but it allows us to plan differentiable, non-grid-bound routes for agents from starting location to target without any prior training on a dataset over any number of dimensions, given an environment whose SDF can be reliably calculated.

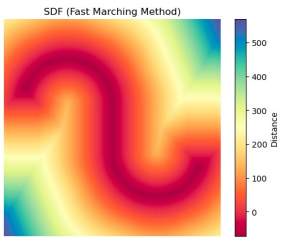

Figure 1: SDF of 2D maze primitive maze shape obstacle. The bodies are confined in the space.

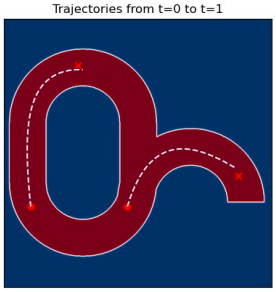

Figure 2: Multi-body SDF solving challenge. The bodies have to go from the start point to the end point without moving out of the SDF.

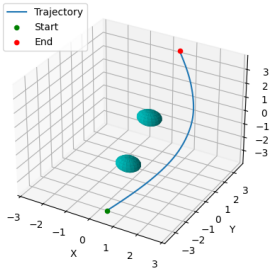

Figure 3: Obstacle Avoidance in 3D using Neural ODE.

4 Results

As we can see, our method designs non-intersecting paths from the starting points to the ending points for each planning problem. We can extend the problem to the basic case of a circular obstacle. The paths planned clearly avoid the obstacle, with non-intersecting paths. On generating a number of random `(starting point, ending point)` tuples, we see that our method generates consistent paths for each case.

We shift to the dataset of mazes and record the performance. Since our method is dataless, we can reduce each pathfinding problem to an optimization problem. We will soon be quoting our results.

5 Conclusion

Multi-agent path planning stands as a significant challenge in various domains, from robotics to autonomous vehicles. Traditional approaches, while having their merits, often grapple with issues such as computational efficiency, optimality, and safety. This paper introduced , a groundbreaking methodology that harnesses the power of Neural Ordinary Differential Equations (Neural ODEs) for multi-agent path planning in a continuous-time framework. By modeling the path-planning environment as a differentiable flow and employing a neural network to parameterize the dynamics, this method brings forth the strengths of continuous-time modeling, including memory efficiency and adaptability to irregular time steps.

In essence, the paradigm represents a significant step forward in the domain of multi-agent path planning. By bridging the gap between differential equations and practical multi-agent systems, this approach paves the way for novel research opportunities in robotics, optimization, and machine learning. Future endeavors in this direction may explore further optimization techniques, integration with more complex environments, and expanding the application domains of this approach.

References

[1] Huifeng Guan et al. "AB-Mapper: Attention and BicNet Based Multi-agent Path Finding for Dynamic Crowded Environment". In: *arXiv preprint arXiv:2110.00760* (2021).

[2] Jiaoyang Li et al. "MAPF-LNS2: fast repairing for multi-agent path finding via large neighborhood search". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 9. 2022, pp. 10256–10265.

[3] Zuxin Liu et al. "MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 11748–11754.

[4] Tingwei Meng et al. "SympOCnet: Solving Optimal Control Problems with Applications to High-Dimensional Multiagent Path Planning Problems". In: *SIAM Journal on Scientific Computing* 44.6 (2022), B1341–B1368.

[5] Vismay Modi et al. "Multi-Agent Path Planning with Asymmetric Interactions In Tight Spaces". In: *arXiv preprint arXiv:2204.00567* (2022).

[6] Farhad Nawaz. "Learning Safe and Stable Motion Plans with Neural Ordinary Differential Equations". In: *Arxiv* (). URL: https://ar5iv.org/abs/2308.00186.

[7] Takuma Okubo and Masaki Takahashi. "Simultaneous optimization of task allocation and path planning using mixed-integer programming for time and capacity constrained multi-agent pickup and delivery". In: *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*. IEEE. 2022, pp. 1088–1093.

[8] Keisuke Okumura et al. "CTRMs: Learning to Construct Cooperative Timed Roadmaps for Multi-agent Path Planning in Continuous Spaces". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2022, pp. 972–981.

[9] Marc Schlichting. "Long Short-Term Memory for Spatial Encoding in Multi-Agent Path Planning". In: *Journal of Guidance, Control, and Dynamics* (2022).