NFOMP: Neural Field for Optimal Motion Planner of Differential Drive Robots With Nonholonomic Constraints

Mikhail Kurenkov, Andrei Potapov, Alena Savinykh, Evgeny Yudin, Evgeny Kruzhkov, Pavel Karpyshev, and Dzmitry Tsetserukou, *Member, IEEE*

***Abstract*—Optimal motion planning is one of the most critical problems in mobile robotics. On the one hand, classical sampling-based methods propose asymptotically optimal solutions to this problem. However, these planners cannot achieve smooth and short trajectories in reasonable calculation time. On the other hand, optimization-based methods are able to generate smooth and plain trajectories in a variety of scenarios, including a dense human crowd. However, modern optimization-based methods use the precomputed signed distance function for collision loss estimation, and it limits the application of these methods for general configuration spaces, including a differential drive non-circular robot with non-holonomic constraints. Moreover, optimization-based methods lack the ability to handle U-shaped or thin obstacles accurately. We propose to improve the optimization methods in two aspects. Firstly, we developed an obstacle neural field model to estimate collision loss; training this model together with trajectory optimization allows improving collision loss continuously, while achieving more feasible and smoother trajectories. Secondly, we forced the trajectory to consider non-holonomic constraints by adding Lagrange multipliers to the trajectory loss function. We applied our method for solving the optimal motion planning problem for differential drive robots with non-holonomic constraints, benchmarked our solution, and proved that the novel planner generates smooth, short, and plain trajectories perfectly suitable for a robot to follow, and outperforms the state-of-the-art approaches by 25% on normalized curvature and by 75% on the number of cusps in the MovingAI environment.**

***Index Terms*—Deep learning methods, motion and path planning, nonholonomic motion planning.**

I. Introduction

Recently, the area of mobile robotics has witnessed significant progress, that is leading to widespread use of robots in our daily lives. Autonomous robots have already been implemented as a replacement for human labor in goods delivery [1] and people transportation industries. Moreover, mobile robots have become an integral part of logistics in warehouses and stores, solving tasks of stock-taking [2], [3], disinfection [4], and plant inspection [5]. However, robot dimensions and maneuverability remain a challenge that engineers should consider when developing such robots. Inability to move in cluttered environments limits robot autonomy in real-world scenarios that are hardly structured. Thus, robots can get stuck in obstacles, go off a track, and perform unnecessary maneuvers and jerks during movement. Thereby, mobile robots require human operators to control their behavior under the challenging scenarios they can't cope with autonomously. This, in turn, increases the costs of robotic solution deployment. Thus, a motion planning module is an inseparable part of any robot's software.

Motion planning is responsible for building a sequence of collision-free robot positions that connect the start and goal points along a path that the robot can traverse. The task of motion planning is complicated due to several reasons. The first problem is the shape of a robot, typically a square or a complex non-circular form. It leads to the necessity of calculating the robot position as an element of SE(2) space instead of a more simple two-dimensional Euclidean space. The second issue is that motion patterns of widely used differential drive robots must satisfy non-holonomic constraints, since such robots cannot move along wheel axes. Moreover, the motion planning algorithm must be computationally efficient to be able to replan trajectories during the robot operation and, at the same time, to maintain the critical path properties, such as smoothness, length, and optimality, based on objectives that vary from case to case. Thus, the goal of this work is to develop an algorithm for solving the optimal motion planning problem in a cluttered environment for differential-drive mobile robots with non-holonomic constraints.

We propose to enhance CHOMP-like [6] algorithms by using neural field (NF) [7] representation for collision loss function, which is shown in Fig. 1. The neural field that we use is a function that returns the collision probability based on an input robot position. This collision neural field can be learned online during planning as a simple classifier. Online learning allows the neural field to be more precise near an optimized trajectory. Moreover, use of the neural field increases collision

Manuscript received 24 February 2022; accepted 10 July 2022. Date of publication 5 August 2022; date of current version 24 August 2022. This letter was recommended for publication by Associate Editor A. Kuntz and Editor H. Kurniawati upon evaluation of the reviewers' comments. This work was supported by RFBR and CNRS according to the Research Project under Grant 21-58-15006. *(Corresponding author: Mikhail Kurenkov.)*

The authors are with the Skolkovo Institute of Science and Technology, 121205 Moscow, Russia (e-mail: Mikhail.Kurenkov@skoltech.ru; Andrei.Potapov@skoltech.ru; Alena.Savinykh@skoltech.ru; evgeny.yudin@skoltech.ru; Evgeny.Kruzhkov@skoltech.ru; pavel.karpyshev@skoltech.ru; D.Tsetserukou@skoltech.ru).

Digital Object Identifier 10.1109/LRA.2022.3196886

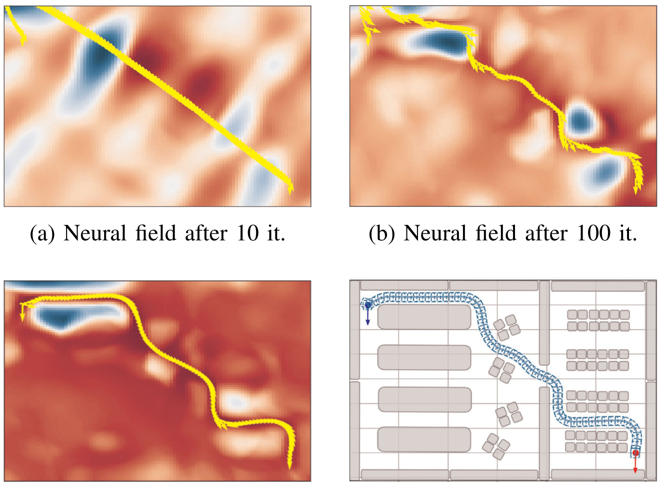

(a) Neural field after 10 it. (b) Neural field after 100 it. (c) Neural field after 200 it. (d) Resulting trajectory.

Fig. 1. Evaluation of NFOMP path planner. Red is the low obstacle neural field value. Blue is the high obstacle neural field value.

loss in places where trajectory optimization pushes a path to a shortcut through obstacles. Furthermore, the neural field creates smoother gradients that produce smoother resulted trajectories and allow avoiding small U-shaped obstacles and thin walls.

In order to enforce non-holonomic constraints, we propose to use Lagrange multipliers. These non-holonomic constraints, together with the optimization of squared differences between sequential trajectory heading angles, increase the smoothness of generated trajectories. We demonstrate that the final smoothness of resulted trajectories is similar to that of CHOMP. Moreover, we can avoid U-shaped obstacles by initializing from a sampling-based solution, even if this solution is not feasible. For example, we can initialize the path with the A* algorithm, which has the efficient implementation in the form of JPS [8].

Thus, the main contributions of this work are:

- a neural field representation of collisions, that is learned online during path planning;
- a trajectory optimization method for differential drive robots with non-holonomic constraints with the use of Lagrange multipliers;
- an optimization-based algorithm for optimal path planning of planar non-circular differential-drive robots.

II. Related Work

An exact combinatorial solution for path and motion planning problems is impossible because it is proven to be PSPACE-hard [9] for a wide class of problems. Therefore, the only way of solving the motion planning problem is finding inexact solutions. Prior works in this area can be classified according to the approach they use to either find an initial path or optimize the prior trajectory.

A. Lattice-Based Planning

The first class to consider is lattice-based planning methods [10], [11] that, in general, split the working space into discrete states, generate a graph, and then solve the motion planning problem with discrete planners. The A* algorithm is one of the most popular discrete planners [12], and has many modifications [10], [11]. Other typical representatives of discrete planners are SBPL lattice planners, such as MHA* [11] and ANA* [13]. The main drawback of this type of planners is their discretization that leads to high computational cost and omission of important environment details. Moreover, these algorithms result in edgy trajectories.

B. Random Sampling-Based Planning

The random sampling-based class of planners is represented by such algorithms as RRT [14], RRT*[15], EST [16], SBL [17], Informed RRT*[18], etc. In general, they explore the environment by states sampled randomly and validate the path feasibility, allowing them to plan in high-dimensional spaces. The probabilistic nature of sampling-based approaches affects the established convergence rate and, in practice, such algorithms often fail to find a feasible path. These methods also result in edgy trajectories, requiring additional post-processing for smooth path construction. Similarly to the class of discrete planning algorithms, random sampling-based methods have many modifications. For example, [19] proposes to use a neural network to improve the sampling distribution.

C. Optimization-Based Planning

The third class of planners is optimization-based approaches [6], [20] that transform the initial trajectory by minimizing a combination of distance and collision loss functions. Optimization-based planners are more flexible, since they are able to continuously replan in dynamic environments and to work with unfeasible trajectories during planning. As an example, the CHOMP planner [6] can initiate with any non-feasible trajectory.

The most efficient and widespread optimization approaches are the gradient descent and second-order optimization methods, such as the Gauss-Newton method and the Levenberg-Marquardt method. CHOMP is the first approach that suggests the use of the gradient descent optimization for trajectory updates. This planner can represent a robot as a set of spheres, and thus can handle planning in SE(2) and other high-dimensional spaces. However, it is not able to handle non-holonomic constraints in its default implementation. Moreover, CHOMP uses a signed distance function (SDF) for collision loss that has low capabilities to handle long walls and small U-shaped objects because it produces local minima for trajectory optimization.

CHOMP has several modifications. For example, STOMP [21] proposes the derivative-free stochastic optimization method. TrajOpt [22] is similar to CHOMP, except for the use of a different collision detection approach and a sequential quadratic programming for trajectory optimization. GPMP [20] parametrizes the trajectory with a few states and then applies a Gaussian process interpolation during optimization. GPMP2 [23] also uses a Gaussian process interpolation, but formulates the problem as nonlinear least squares. dGPMP [24]

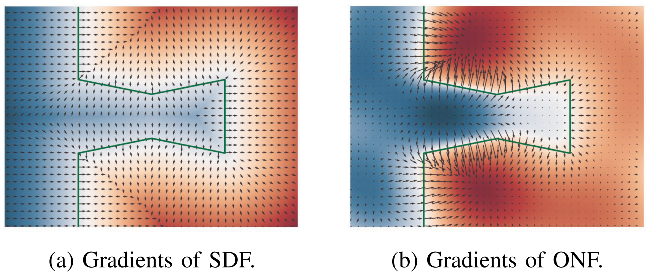

(a) Gradients of SDF. (b) Gradients of ONF.

Fig. 2. SDF and ONF gradient comparison during planning. Blue, red, and green colors represent high-loss, low-loss values, and obstacle contour (the obstacle is to the left of the contour), respectively. Arrows show the directions of the gradients.

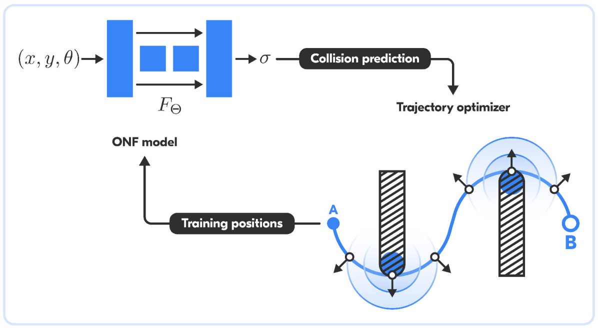


Fig. 3. Structure of NFOMP. A is the starting point, B is the goal point. Black hatched rectangles are obstacles.

is an extension of GPMP2, that automatically adapts its hyperparameters.

Our method is based on the concept of a neural field [7] that implicitly represents the environment as a fully connected neural network. The paper most closely related to our method is the work by Adamkiewicz et al. [25]. This work proposes to use a differentiable density function as a collision loss. This density function is a neural field, which is obtained from NERF [26] that is used to render novel realistic views. This neural field is learned prior to trajectory optimization. By contrast, in our method, we use a neural field that is learned simultaneously with trajectory optimization. The recent work [27] states that implicit scene representation [26] can be used to improve their planning algorithm. Moreover, Rakita et al. [28] use a multi-layer perceptron with six hidden layers as a differentiable cost to approximate distances from collision states to compute smooth paths.

D. Path Smoothing

The smoothing techniques are applied directly after planning algorithms to simplify a path while keeping its feasibility and loss values. The most commonly used post-smoothing techniques are: SimplifyMax, that attempts to remove redundant path positions from a trajectory; Shortcut, that is an iterative process of path "short-cutting" while maintaining its validity; and B-Spline smoothing, that is an iterative process of interpolation utilizing B-splines [29]. A recently proposed gradient-informed path smoothing approach (GRIPS) [30] optimizes initially generated trajectories considering system constraints. GRIPS removes redundant positions from the trajectory and deforms its shape.

III. Methodology

This section describes the proposed neural field optimal motion planner (NFOMP). A general scheme of the method is shown in Fig. 3. First, we describe the details of the CHOMP algorithm, on which the proposed approach is based. Subsequently, we cover the proposed novel obstacle neural field model (ONF model) which predicts obstacles based on the input robot position. Then, we discuss the added Lagrange multipliers and loss functions for trajectory optimization, and the ONF model. Finally, we present the overall proposed algorithm and details of its optimization and implementation.

A. CHOMP Algorithm Description

The proposed algorithm is based on the CHOMP planner, which uses the gradient descent to find collision-free trajectories. The main part of its loss function consists of squared finite differences between neighbor positions, that is similar to Laplacian regularization. The authors of CHOMP proposed to use second order differences, thus minimizing an acceleration term. Our algorithm, in turn, utilizes the first order differences between sequential positions. In addition to the classic quadratic regularization term for the robot position, we use squared differences for robot heading angles. Optimization of quadratic differences for robot heading angles, together with enforced non-holonomic constraints for a differential drive robot, is equal to minimization of a curvature, which is similar to minimization of a squared acceleration, utilized in the original paper. We will demonstrate this property by comparing our method with CHOMP in the experiment section.

The distance loss $L_{distance}$ is represented as follows:

$$L_{distance} = \sum_{i=0}^{N-1} \left((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \right. \\ \left. + w_{direction}(\theta_{i+1} - \theta_i)^2\right). \tag{1}$$

Here, we represent SE(2) trajectory positions with x and y coordinates in the absolute frame, and θ heading angle. (x_0, y_0, θ_0) is the fixed start position, and (x_N, y_N, θ_N) is the fixed goal position. N is the number of positions for trajectory parametrization and equals to 100.

B. Neural Field Obstacle Model

The main problems with CHOMP-type optimizers, which use a signed distance function, are the trajectory shortcutting, and problems with long thin walls and small U-shaped obstacles. In various scenarios, this can lead to building an impossible path and getting stuck at local extremum points.

The problems of trajectory short-cutting and small U-shaped obstacles are directly related to the loss factor, which is responsible for both smoothness of a trajectory, and avoidance of collisions. If the length of a resulting path is significant, the trajectory can be optimized through obstacles to improve overall smoothness, despite collision penalties, and, if the length of the resulting path is short, the resulting trajectory may deviate too far from the obstacles, which will increase its length. This problem is solved by dGPMP2 [24], that utilizes a neural network for adjusting the global collision weight in the loss function. By contrast, in our work, we adjust the collision weight locally.

For U-shaped and uneven obstacles, directions of gradients of a collision loss function are perpendicular to obstacles, and they can even head in the opposite direction from feasible paths, which can cause a resulting trajectory to get stuck at the local extremum, as shown in Fig. 2. By contrast, the proposed method produces gradients, which push the optimized trajectory along walls. Both of these problems are avoided by using online neural field training during trajectory optimization, which predicts the collision probability for a given position. Moreover, online learning allows local impact on high-collision locations to prevent trajectory shortcutting. It also leads to a rough representation of the environment early in training and creates gradients with large optimization steps to overcome local minima such as U-shaped obstacles. To overcome the problem with thin walls, we utilize trajectory reparametrization, Lagrange multipliers for collision loss, sampling random positions on trajectory for trajectory loss calculation, and sampling more positions in collision areas for ONF training.

Generally, the ONF model is based on a simple perceptron consisting of two hidden layers with 100 neurons with a ReLU activation function and skipped connections for faster initial convergence. The Gaussian positional embedding proposed in Fourier Feature Networks [31] is applied to map input states into the higher dimensional space before transferring them to the ONF model. The embedding is represented in the form $sin(B \cdot p)$, where B is the matrix with elements sampled from a normal distribution with the variance σ, that varies for different environments, and $p = [x, y]$. To optimize the collision model, we sample positions randomly all over the environment, along the trajectory, and in the areas with high collision loss. In our implementation, we used 100 positions along the trajectory and 220 sampled points for collision model learning: 100 near the trajectory, 100 in the areas with a high neural field collision function, and 20 randomly in the environment.

Finally, we calculate the collision loss for trajectory optimization as following:

$$L_{collision} = \sum_{i=0}^{N-1} (softplus(F_\Theta(\widetilde{x_i}, \widetilde{y_i}, \widetilde{\theta_i})). \tag{2}$$

Here, F_Θ is the neural field function with Θ parametrization, and $(\widetilde{x_i}, \widetilde{y_i}, \widetilde{\theta_i})$ is the interpolated position randomly chosen between (x_i, y_i, θ_i) and $(x_{i+1}, y_{i+1}, \theta_{i+1})$.

C. Lagrange Multipliers

To enforce non-holonomic constraints, we use Lagrange multipliers that are multiplied by constraint deltas that should be zero, and maximized during optimization. We used non-holonomic constraint deltas δ_i from the equation:

$$\begin{aligned}\delta_i = (x_{i+1} - x_i)sin((\theta_{i+1} + \theta_i)/2) \\ - (y_{i+1} - y_i)cos((\theta_{i+1} + \theta_i)/2).\end{aligned} \tag{3}$$

Thus, our constraint loss function L_{constr} is given by:

$$L_{constr} = \sum_{i=0}^{N} (\delta_i^2 + \lambda_i \delta_i), \tag{4}$$

where λ_i are the Lagrange multipliers.

D. Final Loss for Trajectory Optimization

Thus, the final loss function consists of three parts: the collision loss function, which pushes a trajectory apart from collisions; Laplacian regularization, which minimizes a path length and smoothness of a trajectory; and constraint loss, which forces positions to non-holonomic constraints. The constraint loss includes two terms: quadratic, and linear with Lagrange multipliers. For collision loss, we use the softplus function to the output of the neural network. Thus, the resulting trajectory loss can be calculated from:

$$L_{traj} = L_{dist} + w_{col} L_{col} + w_{const} L_{const}, \tag{5}$$

where w_{col} and w_{constr} are equal to 100 each. For collision model optimization, we use the output of the neural field with the sigmoid activation function:

$$L_{onf} = \sum_{j=0}^{N} \sigma(F_\Theta(x_j, y_j, \theta_j)). \tag{6}$$

Here, $F_\Theta(x_j, y_j, z_j)$ is our neural network parametrized with Θ parameters. This neural network consists of two hidden layers with 100 neurons.

E. Details of Optimization

The optimization is based on the Adam optimizer [32] for both the obstacle neural field model and the trajectory loss, with the learning rate of 0.02 for the collision model and 0.05 for the trajectory model, and the betas of (0.9, 0.9). Learning rates should be conformed to each other for stable planning. The Lagrange's multipliers are optimized with the learning rate of 0.1. One iteration of collision model optimization is performed after each iteration of trajectory optimization. The σ parameter of the Gaussian encoding is the most influential on the method's performance. Its value depends on a scale of environment map. It equals 3 for parking dataset and 10 for other datasets.

Gradients of the loss function are preconditioned based on an inverse of pseudo Hessian, as in the article [33], that is the sum of the quadratic form matrix and the identity matrix with some coefficients. For gradient preconditioning, we use the following equation:

$$g_i = \eta(\alpha H + I)^{-1} \nabla(L_{traj}), \tag{7}$$

where η is the learning rate, α is the hyperparameter that equals 0.5, and H is the Hessian of $L_{distance}$. $\eta(\alpha H + I)^{-1}$ can be

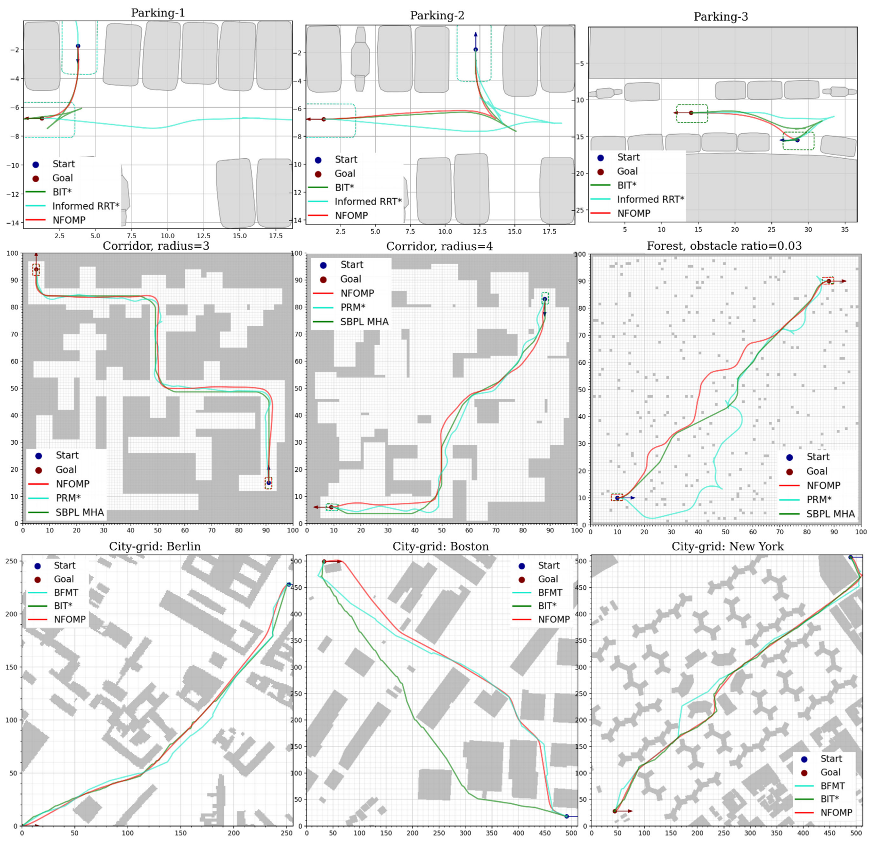


Fig. 4. The robot trajectories generated by NFOMP (red line), PRM* (cyan), and SBPL MHA (green) in different scenarios.

calculated once at the start of optimization, due to the Hessian H being independent of an optimized trajectory.

For the validation of our method, PyTorch framework is used for automatic differentiation. The code of our method is freely available on GitHub[1]. The resulting trajectories of our approach are shown in Fig. 4.

IV. Experiments and Results

A. Implementation Details

To validate the performance of the proposed NFOMP planner, we used novel open-source planning benchmark Bench-MR [34]. Bench-MR provides environments, planners realizations, path quality estimation utilities, and the ability to evaluate the proposed approach in comparison to built-in planners. In this paper, we obtained path metrics for various built-in planners on diverse scenarios including randomly generated corridors, forest environments, and predefined parking and city grids from the Moving AI path-finding benchmark [35]. NFOMP was tested along with built-in feasible sampling-based planners (RRT [14], PRM [36], EST [16], KPIECE), asymptotically (near) optimal planners (BFMT [37], RRT*[15], PRM*[15], Informed RRT*[18], BIT*[38]) and lattice-based planners (MHA*[11]). This set of planners was chosen based on the wide statistics provided in [34]. The results for the GPMP2 method in Moving AI, corridor and random forest are presented as well. During the tests, the following metrics of planners were obtained: **success statistics** (aggregate) is the ratio of found, collision-free, and exact solutions; **computation time** is the time in seconds (s) required to complete path planning; **path length** is the length of the generated path in meters (m); **cusps** is the number of stops, turns and abrupt changes of robot direction; **the maximum and normalized curvature** (lower value corresponds to smoother maneuvers) and **angle-over-length** (AOL) is the smoothness of the generated path. The experiments were conducted for the following cases:

- Parking scenarios, i.e., pulling forward into a parking space, backing into a parking space, and parallel parking. On each scene, 5 experiments with different start and goal points were launched. Results are shown in Fig. 5.
- Random forest dataset map is the 100×100 polygon with one-cell obstacles of 3% density. For testing, 10 experiments were launched at different randomly generated positions of obstacles, and different start and end points.
- 10 randomly generated corridors are presented by 100×100 cells with corridors of three cells wide.
- The Cities Dataset from the Moving AI Lab Pathfinding Benchmarks [35], that contains five occupancy grid scenes of cities. For our experiments, we selected 50 of the most complex scenarios of the *Berlin_0_256* scene, where the start and end points are not in collision.

[1]https://github.com/MisterMap/pytorch-motion-planner

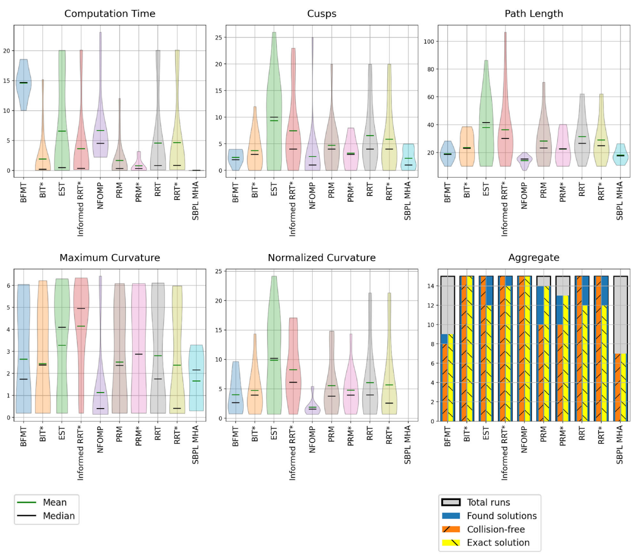


Fig. 5. Consolidated graphs of NFOMP path planner metrics on parking scenarios.

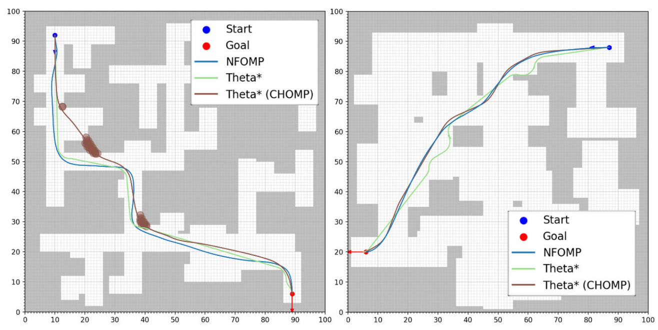
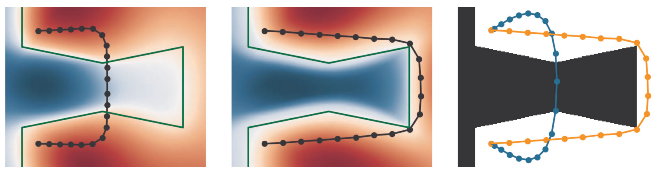


Fig. 6. Comparison of CHOMP and NFOMP performance on the corridor dataset.

B. Comparison With CHOMP and GPMP2

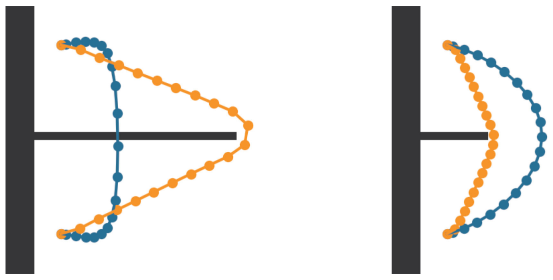
To validate our hypothesis, we compared NFOMP and classical optimization-based methods, such as CHOMP and GPMP2, on different hand-crafted scenarios. Firstly, the comparison was conducted on the random corridor Bench-MR dataset. An initial trajectory for our approach was calculated using the A* algorithm, while CHOMP was initialized with Theta* method because Bench-MR does not provide A* planner implementation. Fig. 6 demonstrates the difference between NFOMP and CHOMP. This scenario demonstrates that CHOMP cuts angles at turns, whereas NFOMP successfully found a short, smooth, and feasible path without collisions. Another environment is presented in the right picture of Fig. 6. In such simple conditions, both CHOMP and NFOMP found close geometrical trajectories with a similar quality.

The performance of NFOMP was also compared with GPMP2. Fig. 7 presents the results of the GPMP2 planner and NFOMP planner on a U-shaped obstacle, and Fig. 8 shows the result on a thin wall. All these trajectories are obtained with the same parameters for each algorithm. For GPMP2, we



(a) NFOMP path after 30 iterations. (b) NFOMP path after 100 iterations. (c) GPMP2 (blue), NFOMP (orange).

Fig. 7. GPMP2 and NFOMP trajectory comparison in U-shape scenario.



(a) Long thin wall scenario: GPMP2 (blue), NFOMP (orange). (b) Short thin wall scenario: GPMP2 (blue), NFOMP (orange).

Fig. 8. GPMP2 and NFOMP trajectory comparison in scenarios with long and short thin walls.

used the collision weight that equals 1. The results show that GPMP2 produces long paths for a small wall, and a shortcut path when a wall is long. Moreover, GPMP2 gets stuck in a local minimum for a small U-shape obstacle. By contrast, NFOMP successfully optimizes paths for all these scenes. Fig. 7(c) depicts how the neural field collision function pushes the optimized path toward a free space.

C. Result on Bench-MR Dataset

This section presents the results of experiments on scenes *Berlin_0_256* from the Moving AI dataset (51 scenarios), parking (15 scenarios), corridor (10 scenarios), and random forest (10 scenarios). The BenchMR realization for each planner was used. For GPMP2, we use the trajectory initialization from the A* algorithm, and different collision cost weights for different scenes (0.01 for MovingAI, 0.1 for random corridor and forest). Also, we use a sphere parametrization of the robot shape for GPMP2. The *"Solution"* column comprises the number of collision-free paths out of the found paths (e.g., for SBPL MHA* in *Berlin_0_256* scenario the path is found in 33 out of 51 scenes, and all 33 paths are collision-free; the same situation is with smoothing heuristics implemented for RRT: they generate not collision-free paths). The results of the Moving AI dataset are presented in Table I. The *"Cusps"* column demonstrates the cumulative values through all scenes in each scenario. The remaining columns show the averaged value of the metrics [see Section IV-A] for each of the planners. Experiments were performed using the Reeds-Shepp steer function for interpolation. For each scene, there is a time limit for path planning in one scenario, and both RRT* and PRM* were launched with a fixed

TABLE I
PLANNING STATISTICS FOR DIFFERENT BENCHMARKS

Planner	**Solutions**	**Time [s]**	**Path Length**	**Cusps**	**Max. Curv.**	**Norm. Curv.**	**AOL**
Scenario: Berlin_0_256 **Steering:** Reeds-Shepp **Time limit:** 0:45							
BFMT	48/51	2.83	363.10	42	0.89	9.90	0.04
BIT*	44/51	46.25	352.99	35	0.99	12.38	0.05
EST	46/51	4.48	494.89	469	1.86	22.22	0.09
Informed RRT*	33/51	45.06	345.92	17	0.76	5.84	0.03
KPIECE	44/51	**2.17**	954.29	1204	1.18	43.67	0.09
PRM	25/51	45.08	358.31	164	1.74	17.40	0.08
PRM*	28/51	45.08	350.75	48	0.95	12.20	0.05
RRT	36/51	8.69	418.42	326	2.10	16.09	0.07
RRT*	36/51	45.06	**344.00**	15	0.66	4.16	0.02
SBPL MHA*	33/33	12.07	363.57	44	1.04	-	0.03
RRT+GRIPS	50/51	45.08	388.42	52	0.89	3.60	0.02
RRT+B-Spline	51/51	45.06	390.50	226	2.41	14.00	0.07
RRT+Shortcut	51/51	45.06	383.05	190	1.97	10.54	0.05
RRT+SimplifyMax	44/51	45.06	359.50	81	1.37	4.89	0.02
A* + GPMP2	29/51	6.25	585.72	**4**	1.52	7.75	0.02
NFOMP(ours)	50/51	17.05	355.98	6	**0.45**	**2.67**	**0.01**
Scenario: Parking **Steering:** Reeds-Shepp **Time limit:** 0:20							
BFMT	8/9	14.61	19.13	22	2.65	4.01	0.48
BIT*	15/15	1.88	23.46	56	2.44	4.74	0.52
EST	15/15	6.55	38.00	140	3.28	9.84	0.73
Informed RRT*	15/15	3.62	36.41	112	4.15	8.26	0.63
PRM	10/14	1.68	28.35	66	2.52	5.54	0.51
PRM*	10/13	0.76	22.55	42	2.88	4.76	0.49
RRT	15/15	4.57	31.43	98	2.81	6.05	0.65
RRT*	15/15	4.64	28.97	88	2.38	5.68	0.60
SBPL MHA*	7/7	**0.01**	18.23	**16**	1.66	4.42	0.46
RRT+GRIPS	15/15	4.53	23.06	31	1.81	3.65	**0.33**
RRT+B-Spline	1/15	4.53	330.48	2803	1.36	12.84	1.76
RRT+Shortcut	12/15	4.53	35.01	137	3.36	7.16	0.81
RRT+SimplifyMax	15/15	4.53	23.20	34	1.66	3.32	0.35
NFOMP(ours)	15/15	6.66	**14.16**	39	**1.13**	**1.88**	1.07
Scenario: Corridor **Steering:** Reeds-Shepp **Time limit:** 1:00							
BFMT	10/10	29.73	160.56	27	**0.37**	7.86	0.12
BIT*	10/10	60.08	166.50	49	1.31	10.67	0.17
EST	10/10	56.62	298.40	500	2.79	49.17	0.65
Informed RRT*	10/10	60.09	153.36	19	1.48	7.81	0.10
KPIECE	0/10	28.72	611.24	1524	1.80	114.92	0.85
PRM	8/10	60.14	183.92	157	2.41	17.19	0.32
PRM*	8/10	60.16	174.58	106	1.52	12.80	0.28
RRT	10/10	40.38	247.28	219	2.62	19.25	0.35
RRT*	10/10	60.07	153.67	21	0.95	8.42	0.11
SBPL MHA*	10/10	**0.62**	**152.46**	10	1.03	-	0.08
RRT+GRIPS	6/10	40.67	201.85	98	1.99	12.93	0.22
RRT+B-Spline	0/10	40.67	655.44	3171	2.12	56.68	1.52
RRT+Shortcut	6/10	40.67	225.98	204	2.50	17.31	0.35
RRT+SimplifyMax	10/10	40.67	167.95	43	1.90	8.53	0.13
A* + GPMP2	7/10	0.75	167.75	18	1.72	14.77	0.37
NFOMP(ours)	10/10	25.21	155.80	**8**	1.06	**6.81**	**0.07**
Scenario: Random forest **Steering:** Reeds-Shepp **Time limit:** 1:00							
BFMT	10/10	24.75	130.28	27	0.92	8.99	0.14
EST	9/10	44.00	318.53	463	2.52	49.58	0.59
Informed RRT*	10/10	60.09	126.15	14	0.54	7.14	0.10
KPIECE	0/10	24.65	600.85	1077	1.57	111.31	0.66
PRM	8/10	60.16	179.34	139	3.16	16.02	0.34
PRM*	8/10	60.22	158.86	89	2.22	13.59	0.27
RRT	10/10	15.66	210.20	130	2.66	20.35	0.28
RRT*	10/10	60.11	124.76	19	**0.68**	6.67	0.11
SBPL MHA*	10/10	**5.72**	**118.52**	**9**	0.70	**3.27**	**0.07**
RRT+GRIPS	5/10	15.90	170.01	61	2.45	11.44	0.19
RRT+B-Spline	3/10	15.90	553.92	2981	2.10	65.02	1.73
RRT+Shortcut	7/10	15.90	180.37	112	3.15	13.33	0.27
RRT+SimplifyMax	10/10	15.90	154.32	62	1.93	9.54	0.18
A* + GPMP2	1/10	0.65	201.76	6	4.2	10.69	0.11
NFOMP(ours)	10/10	19.21	122.39	88	2.25	8.53	0.29

time budget and returned the best solution found by them. All the best results are outlined in **bold**, the second-best results are underlined.

On the Moving AI dataset, our method shows the best performance in terms of curvature (both normalized and maximum), AOL, and the number of cusps. Moreover, NFOMP has the largest number of collision-free solutions, the average length of which is comparable to the top results at almost a third of the time limit. The best result in terms of path length was obtained by the RRT* planner, though the planner couldn't find fully collision-free trajectories. The SPBL MHA* planner found only 33 collision-free solutions out of 51 launches. The normalized curvature for the method is not calculated because the obtained path shows multiple in-place turns, thus pushing the metric to infinity. GPMP2 found only 29 collision-free solutions out of 51. However, it has the smallest value of the number of cusps. Our planner outperforms GPMP2 in the path length metric, the normalized and maximal curvature.

On parking scenarios, the PRM* and SBPL MHA* planners have the shortest path planning time; however, at the same time, on some scenarios, they build a path with collisions or do not build it at all. The PRM* and RRT* algorithms have the computational time lower than the time budget because several scenarios have possibility to connect the goal and start positions without collision in the one movement. The average path length built by our planner is the shortest; additionally, the constructed paths have the smallest average curvature with a relatively low number of cusps (small total number of cusps for PRM* and BFMT is due to the lack of results on a part of the scenarios).

For corridor scenarios, all planners successfully construct paths from the start point to the end point. The average path length planned by NFOMP is slightly larger than the best results, while the total number of cusps is the smallest. GPMP2 has the problem described in the section, that leads to unfeasible solutions and increased normalized and maximum curvature.

On the random forest dataset, such metrics of our planner, as path length and solution time, are slightly worse than the best results. However, it outperforms most planners in path smoothness terms, but loses to SBPL MHA* and RRT* by the number of cusps, and normalized and maximal curvature.

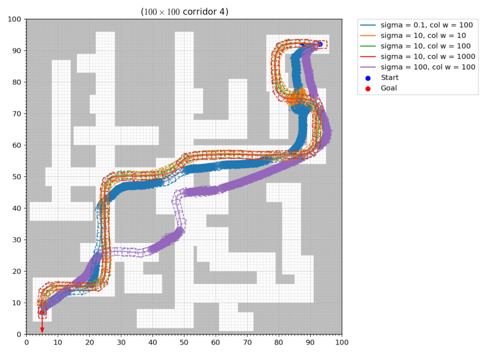


Fig. 9. Sensitivity analysis of different parameters.

D. Sensitivity Analysis

We conducted a sensitivity analysis to show an influence of two important hyperparameters on planning performance. These hyperparameters are the collision loss weight and the Gaussian encoding sigma. The result is shown in Fig. 9. The changes in the Gaussian encoding sigma can result in poor performance. However, the collision loss weight has a little influence on the final solution. This behavior is the result of the adaptive nature of a neural field, and differs from the behavior of GPMP2, that has a significant dependence on the collision loss weight, as demonstrated in the dGPMP2 article [23].

V. CONCLUSIONS AND DISCUSSION

We have presented a novel algorithm for solving the optimal motion planning problem in cluttered environments for differential-drive mobile robots with non-holonomic constraints. This algorithm, due to the transformation of the map into a neural field, makes it possible to obtain smoother gradients that form an optimized trajectory and avoid small U-shaped obstacles. Moreover, this operation allows getting a smooth trajectory much faster than the sampling-based and discrete algorithms. Additionally, in order to further improve the trajectory smoothness, we propose to use Lagrange multipliers for enforcing non-holonomic constraints.

To validate the proposed method, an extensive set of experiments was carried out on several datasets. One of the disadvantages of the proposed method is a relatively large computation time. However, it is strongly influenced by the fact that the algorithm was implemented in Python using the PyTorch framework, and the algorithms it was compared with were written in C++.

Regarding other metrics related to the quality of trajectories, such as curvature and the number of cusps, our algorithm is significantly superior to the most state-of-the-art approaches, outperforming them by 25% on normalized curvature and by 75% on the number of cusps in the MovingAI environment. The length of the estimated path is slightly inferior to the optimal planners, and is longer by less than 2% on all datasets. The fact that our algorithm is not limited to a configurable space makes it possible to extend it for solving different problems. Thus, we suggest that our planner can be effectively used to plan movements of manipulators and swarms of drones.

References

[1] S. Protasov et al., "CNN-based omnidirectional object detection for hermesbot autonomous delivery robot with preliminary frame classification," in *Proc. IEEE 20th Int. Conf. Adv. Robot.*, 2021, pp. 517–522.

[2] A. Petrovsky et al., "Customer behavior analytics using an autonomous robotics-based system," in *Proc. IEEE 16th Int. Conf. Control Automat., Robot. Vis.*, 2020, pp. 327–332.

[3] I. Kalinov et al., "Warevision: CNN barcode detection-based UAV trajectory optimization for autonomous warehouse stocktaking," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6647–6653, Oct. 2020.

[4] S. Perminov et al., "UltraBot: Autonomous mobile robot for indoor UV-C disinfection," in *Proc. IEEE 17th Int. Conf. Automat. Sci. Eng.*, 2021, pp. 2147–2152.

[5] P. Karpyshev, V. Ilin, I. Kalinov, A. Petrovsky, and D. Tsetserukou, "Autonomous mobile robot for apple plant disease detection based on CNN and multi-spectral vision system," in *Proc. IEEE/SICE Int. Symp. Syst. Integration*, 2021, pp. 157–162.

[6] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 489–494.

[7] Y. Xie et al., "Neural fields in visual computing and beyond," *Comput. Graph. Forum*, vol. 41, no. 2, pp. 641–676, 2022.

[8] D. Harabor and A. Grastien, "Improving jump point search," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2014, vol. 24, pp. 128–135.

[9] S. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[10] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proc. Int. Conf. Automated Plan. Scheduling*, vol. 5, 2005, pp. 262–271.

[11] F. Islam, V. Narayanan, and M. Likhachev, "Dynamic multi-heuristic A," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2376–2382.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[13] J. V. D. Berg, R. Shah, A. Huang, and K. Goldberg, "Anytime nonparametric A," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 105–111.

[14] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[16] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. Int. Conf. Robot. Automat.*, vol. 3, 1997, pp. 2719–2726.

[17] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics Research*. Berlin, Germany: Springer, 2003, pp. 403–417.

[18] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 2997–3004.

[19] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7087–7094.

[20] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 9–15.

[21] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 4569–4574.

[22] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Proc. Robot.: Sci. Syst.*, 2013, pp. 1–10.

[23] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Proc. Robot. Sci. Syst.*, 2016, vol. 12, no. 4.

[24] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable Gaussian process motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 10598–10604.

[25] M. Adamkiewicz et al., "Vision-only robot navigation in a neural radiance world," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4606–4613, Apr. 2022.

[26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NERF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.

[27] Z. Zhang and D. Scaramuzza, "Fisher information field: An efficient and differentiable map for perception-aware planning," 2020, *arXiv:2008.03324*.

[28] D. Rakita, B. Mutlu, and M. Gleicher, "RelaxediK: Real-time synthesis of accurate and feasible robot arm motion," in *Proc. Robot. Sci. Syst.*, Pittsburgh, PA, USA, 2018, pp. 26–30.

[29] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[30] E. Heiden, L. Palmieri, S. Koenig, K. O. Arras, and G. S. Sukhatme, "Gradient-informed path smoothing for wheeled mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1710–1717.

[31] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 7462–7473, 2020.

[32] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[33] B. Nicolet, A. Jacobson, and W. Jakob, "Large steps in inverse rendering of geometry," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–13, 2021.

[34] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, "Bench-MR: A motion planning benchmark for wheeled mobile robots," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4536–4543, Jul. 2021.

[35] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 2, pp. 144–148, Jun. 2012.

[36] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[37] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2072–2078.

[38] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3067–3074.