2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Dynamic Neural Potential Field: Online Trajectory Optimization in the Presence of Moving Obstacles

Aleksei Staroverov[1,2,3], Muhammad Alhaddad[3], Aditya Narendra[3]
Konstantin Mironov[4], and Aleksandr Panov[1,3]

Abstract— **Generalist robot policies must operate safely and reliably in everyday human environments such as homes, offices, and warehouses, where people and objects move unpredictably. We present Dynamic Neural Potential Field (NPField-GPT), a learning-enhanced model predictive control (MPC) framework that couples classical optimization with a Transformer-based predictor of footprint-aware repulsive potentials. Given an occupancy sub-map, robot footprint, and optional dynamic-obstacle cues, our NPField-GPT model forecasts a horizon of differentiable potentials that are injected into a sequential quadratic MPC program via L4CasADi, yielding real-time, constraint-aware trajectory optimization. We additionally study two baselines: NPField-StaticMLP, where a dynamic scene is treated as a sequence of static maps; and NPField-DynamicMLP, which predicts the future potential sequence in parallel with an MLP.**

In dynamic indoor scenarios from BenchMR and on a Husky UGV in office corridors, NPField-GPT produces more efficient and safer trajectories under motion changes, while StaticMLP/DynamicMLP offer lower latency. We also compare with the CIAO* and MPPI baselines. Across methods, the Transformer+MPC synergy preserves the transparency and stability of model-based planning while learning only the part that benefits from data: spatiotemporal collision risk. Code and trained models are available at `https://github.com/CognitiveAISystems/Dynamic-Neural-Potential-Field`.

I. Introduction

Mobile robots that act autonomously within human-oriented environments can become significant assistants for humans. They can be applied in offices, shops, homes, and medical facilities. Autonomous operation requires methods for planning motion within the environment. Trajectory planning is often executed in two stages: first, a rough global path is generated via search-based [1, 2] or sampling-based [3, 4] methods; second, the global path is turned into a local trajectory under kinodynamic constraints and obstacle avoidance. The second stage is often executed online with a receding horizon strategy. Solutions for receding-horizon trajectory planning may be obtained with model predictive path integral (MPPI) [5, 6] or numerical model predictive control (MPC) [7–14]. MPPI can work with arbitrary obstacle maps, but it can produce invalid or chattering solutions. Numerical MPC provides fast and stable solutions; however, it requires an analytical representation of collision danger either as a set of constraints [9] or as a cost term, the repulsive potential. The term "repulsive potential" was introduced for the artificial potential field global planner [15]. In MPC, the trajectory is biased toward safer regions according to the repulsive potential. In this work, we consider local planning with numerical MPC in the presence of dynamic obstacles (Fig. 1).

Avoiding collisions with dynamic obstacles (e.g., walking human beings in an office environment) is especially challenging. It means that the obstacle map for each frame within the prediction horizon will be different. The prediction of obstacle flow is a distinct task [16–18], which is out of our scope. It can be solved with various methods. Here, we consider dynamic obstacles to be predictable. The dynamic appearance of the obstacle model enlarges the computational complexity of the MPC problem due to a higher number of obstacle parameters.

We follow a hybrid design principle: learn only spatiotemporal collision risk and keep optimization model-based. Specifically, we extend NPField to dynamic scenes with three neural variants that produce differentiable, footprint-aware obstacle potentials for MPC.

Our framework comprises three architectures to balance accuracy and latency. The simplest variant, StaticMLP, processes dynamic environments as sequential static frames. To improve temporal reasoning, DynamicMLP infers future potentials concurrently using parallel heads. Finally, NPField-GPT employs a non-autoregressive Transformer to predict the entire potential horizon holistically, significantly enhancing the representation of spatial and temporal obstacle dynamics. In the GPT model, coordinate information is emphasized through explicit dynamic-obstacle channels, relative-coordinate fusion, and near-obstacle weighted training.

Our main contributions are:

- A real-time MPC-compatible dynamic neural potential framework with three variants (StaticMLP, DynamicMLP, GPT) covering accuracy-latency trade-offs.
- A novel NPField-GPT architecture that predicts horizon potentials in parallel and strengthens obstacle-relative geometry conditioning.
- Extensive comparisons with CIAO* and MPPI in BenchMR scenarios and real Husky UGV office-corridor experiments.
- Open-source implementation with L4CasADi integration for differentiable neural costs inside MPC.

[1]Cognitive AI Systems Lab
[2]National University of Science and Technology MISIS
[3]Moscow Independent Research Institute of Artificial Intelligence
[4]Ufa University of Science and Technology

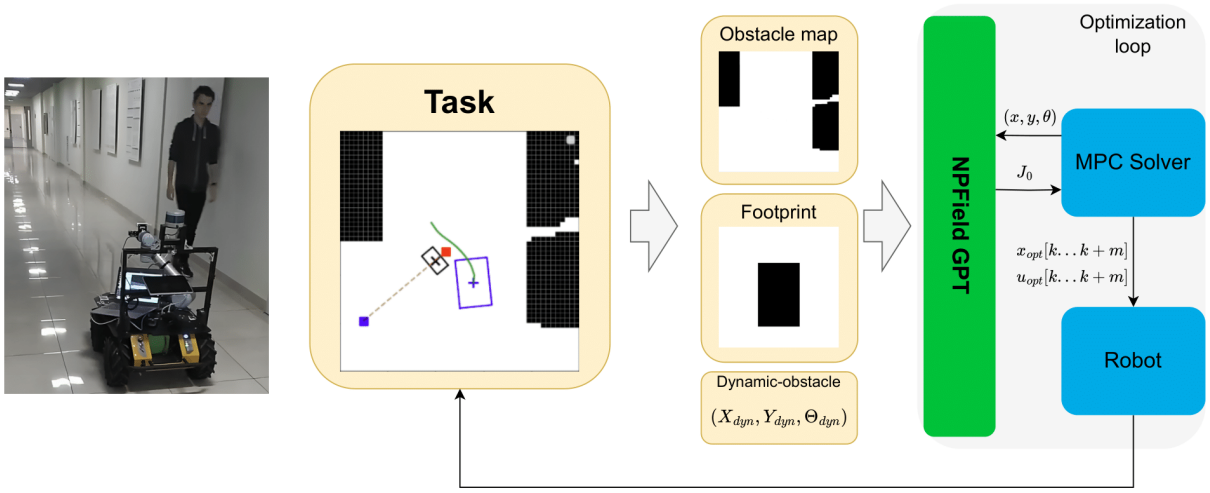

Fig. 1. Overview of NPField-GPT integrated with MPC for dynamic obstacle avoidance. Inputs include occupancy map, robot footprint, robot pose, and dynamic-obstacle state $(x_{dyn}, y_{dyn}, \theta_{dyn})$. The neural model predicts a footprint-aware repulsive potential horizon, which is consumed by MPC to optimize $\mathbf{x}_{opt}$ and $\mathbf{u}_{opt}$ in real time.

II. Related works

This section discusses existing approaches to motion planning, especially local planning in the presence of moving obstacles and collision avoidance using neural networks. The general planning task consists of finding a trajectory from a given start to a given goal. Well-known global planners such as A* [1], Theta* [2], PRM [3], or RRT [4] generate a reference of intermediate positions of the robot (a global geometric path). Following this reference leads to approaching the destination point. However, in the common case, the geometric path is unaware of how to provide smooth and collision-free motion between intermediate positions. Basic global planners have special extensions, which provide planning with respect to kinodynamic constraints. A* search may be executed on a lattice of kinodynamically feasible motion primitives [19], which results in an executable trajectory instead of a rough geometric path. Motion primitives allow the planner to check that there are no collisions between the intermediate state, including collisions with dynamic obstacles [20–23]. Sampling-based planners may also be extended to satisfy kinodynamic constraints [24] and replan while avoiding dynamic obstacles [25].

The aforementioned methods aim to add dynamic collision avoidance into the global planning procedure, which might be computationally excessive in case of long global plans and short local horizons. Prediction of dynamic obstacles is often obtained from actual sensor data and requires replanning within a relatively short prediction horizon. Therefore, we further narrow down to receding horizon approaches for local planning.

A. Receding horizon planning and dynamic obstacles

The task of local planning is to turn a fixed part of the rough global plan into a smooth trajectory segment under consideration of obstacles and kinodynamic constraints. Model Predictive Path Integral (MPPI) [5] achieves this via sampling random trajectory segments and generating a good solution based on these samples. Collision check for sample trajectories may be systematically extended to the case of dynamic obstacles [26, 27]. Computational heaviness and nondeterministic nature are disadvantages of MPPI; therefore, it may be used in cases when the process model is too complicated for MPC. Numerical MPC solves the local planning task as an optimal control problem. Obstacle avoidance is formalized as a set of constraints (e.g. [9]) or as an additional cost term of the optimization problem. This formalization leads to additional problem parameters describing obstacle properties. The number of parameters should be small to preserve MPC performance. Many approaches approximate the obstacle map with a low-dimensional model. Either obstacles or free space may be approximated with a set of simple geometric figures: points [8], circles [9, 13], rectangles [10], polylines [28], or polygons [7, 14, 29]. Some of these works explicitly consider dynamic environments [28, 29]; others can be adapted straightforwardly (e.g., CIAO* [9, 10] uses independent approximations of the free space at each timestep; these approximations can reflect dynamic obstacles). The limitations of geometric approximations are computational challenges and loss of fidelity. Many works (e.g. [7, 14]) do not consider how to obtain a geometric approximation from an arbitrary obstacle map. In practice, maps are often provided as occupancy grids (a matrix projected onto the map; zero values correspond to free-space cells, while ones correspond to occupied cells). High-resolution grids have too many parameters to be passed to the MPC solver in raw form. In the next subsection, we consider approaches where obstacle data are modeled with neural networks.

B. Trajectory optimization with neural collision model

Learning collision model for numeric trajectory optimization was considered in several works [30–34]. It is challenging to balance the precision and computational complexity of the collision model; therefore, existing works are constrained in terms of computation time and complexity of the maps. [30–33] use neural models inspired by Neural Radiance Field [35]. These models learn the structure of a single obstacle map (in 2D or 3D) and may be used for navigation within the learned map. [30, 31] exploit network inference for trajectory optimization, while [32, 33] optimize trajectory within the learning procedure. Instead of learning a single obstacle map, the models from [34, 36] take an obstacle representation as input. Both architectures include two submodels: the first reduces the dimensionality of the obstacle representation, while the second calculates the collision score. Therefore, the first submodel provides a compact vector of parameters for a real-time MPC solver, while the second is integrated into this solver using the L4CasADi [31] framework. The difference is that NPField [36] utilizes occupancy grids for footprint-aware collision avoidance of a wheeled mobile robot, while [34] utilizes depth images for 3D collision avoidance of an aerial robot. To our knowledge, there are no neural obstacle models that provide MPC avoidance from dynamic obstacles in real time. [33] exploits a NeRF learning procedure for trajectory optimization in the presence of moving obstacles; however, the computational procedure takes tens of seconds, which is non-real-time. This work aims to provide a model that allows online MPC collision avoidance with a neural model of static and dynamic obstacles.

C. Positioning and novelty.

Unlike classical geometric MPC [9, 10] or sampling-based methods [5], our approach naturally encodes footprint-aware costs from raw occupancy grids. Furthermore, in contrast to prior neural models that learn single scenes offline [30, 32, 33] or use non-map inputs [34], our framework provides real-time, differentiable repulsive potentials for dynamic obstacle avoidance directly within the MPC loop.

III. Background

Following NPField, we define the statement and notations for model predictive local planning. We consider a non-holonomic wheeled robot with a differential drive. System state vector $\mathbf{x} = \{x, y, \theta, v\}$ includes 2D robot coordinates x and y, its orientation θ and linear velocity v (directed according to θ). Control vector $\mathbf{u} = \{a, \omega\}$ includes robot acceleration a (directed according to θ) and angular velocity ω. We consider the model with continuous time dynamics and discrete time control (i.e., u is considered to be constant within the single timestep). A formal statement for the trajectory optimization problem is the following:

$$\{\mathbf{x}_{opt}[i], \mathbf{u}_{opt}[i]\}_{i=k}^{k+m} = \arg\min \sum_{i=k}^{k+m} (\|\mathbf{x}[i] - \mathbf{x}_r[i]\|_{w_x} + \|\mathbf{u}[i]\|_{w_u} + J_o(\mathbf{x}[i], \mathbf{p}_o[i])), \tag{1a}$$

s.t.

$$\frac{dx}{dt} = v\cos\theta, \quad \frac{dy}{dt} = v\sin\theta, \quad \frac{dv}{dt} = a, \quad \frac{d\theta}{dt} = \omega. \tag{1b}$$

Here $\mathbf{x}_r$ is a reference path, $\mathbf{p}_o$ is a vector of obstacle parameters. The total cost function consists of three terms: path following term ($\|\mathbf{x}[i] - \mathbf{x}_r[i]\|_{w_x}$ is a weighted distance between actual trajectory and a reference path), control minimization term ($\|\mathbf{u}[i]\|_{w_u}$ is a weighted norm of the control input), and repulsive potential J_o. The aforementioned statement may be adapted to systems with other dynamic models (e.g., holonomic or car-like robots): this requires the change of $\mathbf{x}$, $\mathbf{u}$, and (1b), while the obstacle model will keep the same. Neural potential function is a neural network, which is trained to predict J_o based on $\mathbf{x}$ and $\mathbf{p}_o$. In NPField $\mathbf{p}_o$ is an embedding of the obstacle map obtained from the neural encoder. Reference potential for each point is calculated based on the signed distance function (SDF) to the obstacle border. The reference potential for the whole robot is chosen as the maximum potential of the points within its footprint. The reference potential is non-differentiable (as is the SDF), and the maximum over footprint cells is computed algorithmically, so it cannot be used directly in the MPC loop. Instead, reference potential is used to generate a dataset for training the neural potential field. The difference between static and dynamic environments is that in the first case $\mathbf{p}_o$ is constant for the whole trajectory, while in the second case it depends on time.

a) Why potentials instead of explicit trajectory prediction?: Explicitly predicting obstacle trajectories can provide precise future states but introduces (i) an additional module to train and maintain, (ii) error compounding over the horizon, and (iii) coupling between prediction errors and MPC feasibility. Our approach estimates repulsive potentials that directly shape the MPC cost. This has two benefits: (a) optimization remains robust to small prediction errors because potential encodes spatial risk rather than a single hypothesized obstacle pose, and (b) potentials are differentiable and integrate naturally with L4CasADi for efficient gradient computation. When high-quality motion predictors are available, StaticMLP can consume predicted trajectories in the form of future obstacle maps, while DynamicMLP/NPField-GPT can be conditioned on predictor states $I_{dyn} = (x, y, \theta)$ and infer horizon risk directly from the current map context. Thus, our framework complements modern predictors by converting their outputs into optimization-friendly costs.

IV. Neural Network-Based Potential Field Generation

We study three variants for producing footprint-aware repulsive potentials in dynamic scenes: NPField-StaticMLP, NPField-DynamicMLP, and NPField-GPT. All variants share a modular pipeline with three parts: (i) map/footprint encoding (yellow block), (ii) point-wise potential prediction for MPC query states (green block), and (iii) an auxiliary map-reconstruction decoder used only in training (red block). The auxiliary decoder regularizes the spatial embedding and is removed at inference time.

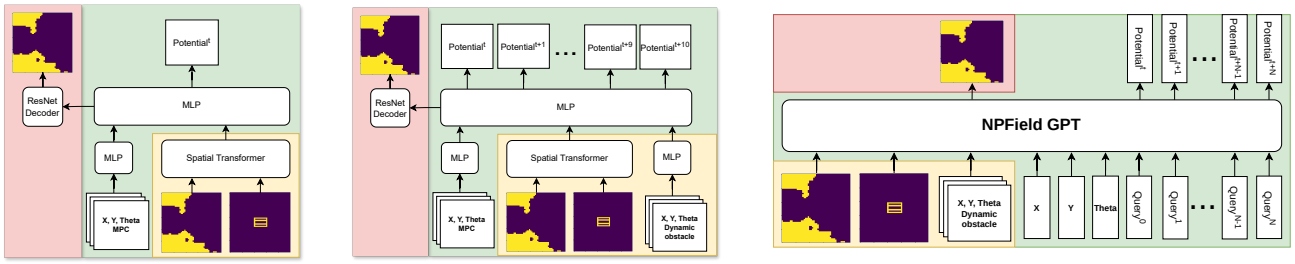

Fig. 2. StaticMLP (left), DynamicMLP (middle), and NPField-GPT (right). Yellow: map/footprint encoder; green: potential predictor for sampled states; red: auxiliary decoder used only during training. NPField-GPT predicts horizon potentials in one Transformer pass using learnable query tokens and coordinate-enhanced conditioning.

NPField-StaticMLP models dynamic navigation as a sequence of static maps and evaluates one-step potentials for each frame. This keeps the predictor simple but requires explicit future-map construction from obstacle forecasts.

NPField-DynamicMLP predicts all horizon steps in parallel with separate MLP heads conditioned on dynamic obstacle state $I_{dyn} = (x, y, \theta)$. This is efficient but temporal coupling between steps is weak because each step-specific head is independent.

NPField-GPT uses a Transformer backbone with non-autoregressive horizon prediction. Instead of generating step $t+1$ from step t, NPField-GPT appends learnable query tokens (10 for the horizon) and predicts the full potential sequence in a single forward pass.

a) Token sequence.: The input to the Transformer is a single sequence of tokens. The *context* is built from 11 tokens in a fixed order: (1) map embedding (CNN plus spatial self-attention over the occupancy grid, projected to dimension d); (2) robot footprint embedding; (3)–(4) dynamic-obstacle position (x_{dyn}, y_{dyn}) each encoded by a linear layer; (5)–(6) dynamic-obstacle orientation θ_{dyn} as $\sin\theta_{dyn}$ and $\cos\theta_{dyn}$; (7)–(8) query robot position (x, y); (9)–(10) query robot orientation θ as $\sin\theta$ and $\cos\theta$; (11) a learned *relative-coordinate fusion* token that combines robot and obstacle coordinates via an MLP (context plus delta terms) and trainable gains to emphasize obstacle-relative geometry. The map and footprint encodings are computed once from the occupancy sub-map and footprint image; the coordinate tokens depend on the query pose (x, y, θ) and dynamic state $I_{dyn} = (x_{dyn}, y_{dyn}, \theta_{dyn})$. After the context, $H = 10$ learnable query vectors are appended (the vectors themselves are fixed parameters, not a function of (x, y, θ)). Learned positional embeddings are added to the full sequence; the stack is passed through the causal Transformer and layer norm. Because each query token attends over all preceding context tokens—including the robot pose (x, y, θ) and dynamic-obstacle state—the output at the query positions is conditioned on the query pose. The hidden states at the query positions are projected by a linear head and passed through sigmoid to obtain the predicted potential at each horizon step. Thus the potential horizon is predicted in one forward pass, with dependence on the query pose arising through attention to the context.

b) Training loss.: Let $\hat{y}_{b,h}$ and $y_{b,h}$ denote the predicted and target potential for batch index b and horizon step h. Let d_b be the Euclidean distance from the query pose (x_b, y_b) to the dynamic obstacle $(x_{dyn,b}, y_{dyn,b})$. We use a distance-dependent weight $w(d) = 1 + \alpha \exp\left(-(d/\sigma)^2/2\right)$ with $\alpha = 2$, $\sigma = 0.5$ to upweight errors near the obstacle. The potential loss is the weighted mean squared error

$$\mathscr{L}_{\text{pot}} = \frac{1}{BH} \sum_{b,h} w(d_b) \, (\hat{y}_{b,h} - y_{b,h})^2. \tag{2}$$

An auxiliary occupancy reconstruction loss $\mathscr{L}_{\text{map}}$ is computed by decoding the hidden state at the last context token (the relative-coordinate fusion token) through a small decoder and applying cross-entropy to the ground-truth occupancy grid; this stabilizes the spatial embedding. The total training loss is

$$\mathscr{L} = \mathscr{L}_{\text{pot}} + \lambda_{\text{map}} \mathscr{L}_{\text{map}}, \tag{3}$$

with $\lambda_{\text{map}} \geq 0$ (default 1). The auxiliary decoder and $\mathscr{L}_{\text{map}}$ are used only during training and removed at inference.

V. Dataset Preparation and Generation Details

The dataset collection procedure includes the choice of static maps, robot footprints, and dynamic obstacles. Static maps were chosen based on the MovingAI [37] city map dataset and an occupancy grid for an office building of the Institute. The Husky UGV footprint is used in this work. In addition to the robot footprints, one footprint is used for the dynamic obstacle: a 0.7×0.5 m rectangle that moves in linear motion with a random orientation. Dataset collection includes the following steps:

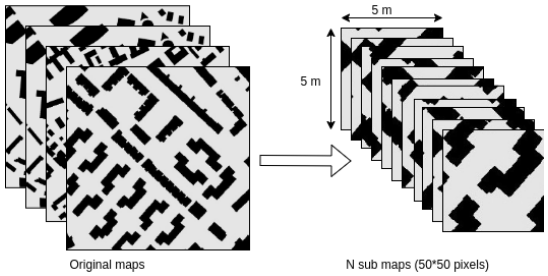

Fig. 3. Cropping original maps into sub-maps

1) The maps are cropped into sub-maps and represented as images with 50×50 resolution (Fig. 3), where each pixel corresponds to 0.1 m (10×10 cm) in the real environment (i.e., each sub-map covers 5×5 m). This yields 1000 unique sub-maps for data generation.
2) For each sub-map, one agent position with three orientations is sampled. Assuming 0.3 m/s constant speed, 10 forward positions per orientation are generated (31 variants per sub-map: 1 static-only; 3×10 with the agent).
3) Every sub-map copy is transformed into a costmap where each cell is filled with the reference potential.
   a) The signed distance function (SDF) is computed for each cell (distance to the nearest obstacle border; positive in free space, negative in obstacles).
   b) Repulsive potential per cell is $J_o = w_1(\pi/2 + \arctan(w_2 - w_2\,\mathrm{SDF}))$. This sigmoid is low far from obstacles, approaches w_1 asymptotically inside obstacles, and has maximum derivative on the obstacle border.
4) Two robot footprints are used, both corresponding to a real Husky UGV mobile manipulator (folded and outstretched arm).
5) For each footprint, the potential at each query pose is obtained by placing the footprint at 10 random orientations per pixel for each copy of every sub-map; the robot footprint is projected onto the map and the maximum potential over the covered cells is taken as the repulsive potential.

The resulting dataset is tabular with fields: sub-map ID, query robot pose, dynamic obstacle pose/direction, footprint ID, and target potential.

Data splits and validation. We split the dataset into training/validation/test sets by sub-map (70/15/15%). All figures that reference a *validation* configuration use sub-maps and agent placements not present in training. We also include direction-reversal stress tests by flipping I_{dyn} on held-out validation sub-maps to evaluate robustness.

VI. Implementation and Computational Profile

We solve the nonlinear MPC problem via Sequential Quadratic Programming (SQP) using Acados [38] with CasADi [39] for differentiation. To integrate our PyTorch [40] potential model into the Acados solver, we wrap it using L4CasADi [31], which provides an exact symbolic description suitable for accurate gradients (unlike local approximations such as ML-CasADi [41]). This allows us to directly evaluate the predicted horizon potential in the Acados model, optimizing a nonlinear least-squares objective that combines tracking, control effort, and the neural potential term.

a) End-to-end runtime breakdown.: On an AMD Ryzen 5 3500 CPU and NVIDIA GeForce RTX 2080 (CUDA), typical per-replan times are:

- map/footprint preprocessing and h_{map} encoding (GPU): 6–12 ms;
- per-sample point embedding and NPField forward over the horizon (GPU): NPField-StaticMLP: 30–40 ms, NPField-DynamicMLP: 55–80 ms, NPField-GPT: 120–160 ms (Transformer with parallel query-token horizon prediction);
- Acados SQP solve with L4CasADi cost (CPU): 250–550 ms depending on variant and scenario complexity;
- ROS I/O and sensor updates: 10–20 ms.

This yields end-to-end medians consistent with Table I. The bottleneck remains the MPC solve, while NPField-GPT adds moderate overhead due to the larger Transformer backbone and enhanced coordinate conditioning.

b) Scalability.: Let N be the horizon length and M the number of dynamic obstacles. For NPField-StaticMLP, complexity scales as $\mathcal{O}(N)$ forward calls of a single-step potential; for NPField-DynamicMLP, a constant number of heads gives $\mathcal{O}(1)$ forward depth per time step (batched over N); for NPField-GPT, horizon prediction is parallel in one Transformer pass, with complexity dominated by attention over context and query tokens. MPC cost is dominated by the nonlinear solve and scales roughly quadratically in N; we use warm starts and structured costs. With multiple obstacles, I_{dyn} and token counts grow linearly in M; we cap M per local window and prioritize nearest obstacles.

c) Training.: For NPField-GPT we use typical Transformer settings: 4 layers, 4 heads, embedding size 576; 10 learnable horizon query vectors; map encoder output projected to the model embedding width before Transformer fusion. Optimization uses an Adam-family optimizer with gradient clipping and optional mixed precision.

Our local planner works together with Theta* [2] global planner, which generates global plans as polylines. Note that Theta* uses a simplified version of the robot footprint (a circle with a diameter equal to the robot width) as it fails to provide a safe path with a complete footprint model. This simplified model does not guarantee the safety of the global plan; therefore, the safety of the trajectory is provided by our local planner.

VII. Experiments

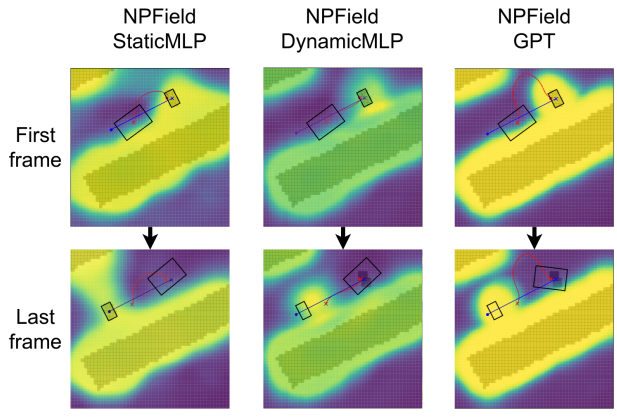

Fig. 4. First and last frames of a representative validation scenario for each method. The heatmap encodes the predicted repulsive potential. The larger rectangle denotes the robot footprint; the smaller rectangle denotes the dynamic obstacle.

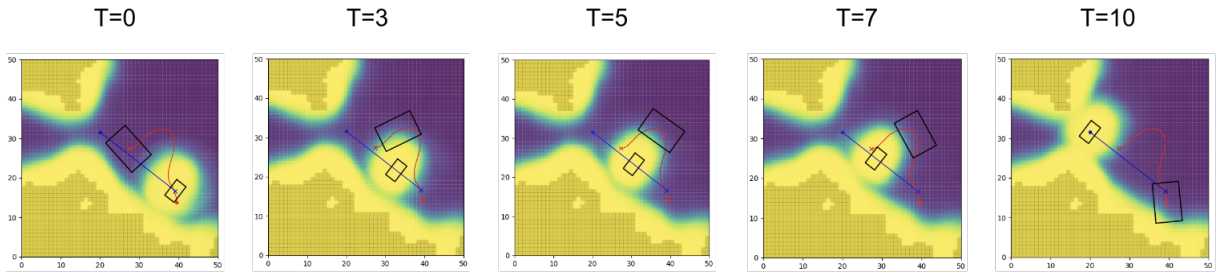

Fig. 5. Five sequential time steps of MPC trajectory optimization. The sequence illustrates how NPField-GPT adapts the full-horizon potential in real time, enabling smooth collision avoidance without explicit future-map rollout.

A. Numerical experiments

We evaluated our algorithm on 100 scenarios using the [42] framework, which includes tasks like navigating through narrow passages. Figure 5 shows a representative example: five successive MPC replan steps in which NPField-GPT steers the robot around a moving obstacle by continuously updating the predicted potential horizon. The assessment covered standard metrics such as planning time, path length, smoothness, and angle-over-length, where a lower value is preferable for all. Additionally, we introduced a custom metric, "safety distance" (the minimum value of the SDF).

The experimental results, as detailed in Table I, compared three versions of the Dynamic NPField algorithm: StaticMLP, DynamicMLP, and NPField-GPT, as well as MPPI and the CIAO* [9, 10] trajectory optimization algorithm. Figure 4 compares the predicted potential fields at the first and last frames of a validation scenario. The qualitative differences are clear: StaticMLP tends to merge the dynamic obstacle into the surrounding wall, losing its distinct boundary because the obstacle footprint is small relative to the map. DynamicMLP produces overly diffuse boundaries and does not always capture the temporal displacement of the obstacle correctly. NPField-GPT yields the sharpest and most accurate potentials, maintaining well-defined boundaries for both static walls and the moving obstacle throughout the horizon.

For each waypoint of the reference, CIAO* approximates the free space around the robot with a convex figure (circle or rectangle) and constrains the robot to be inside this figure. DynamicMLP and NPField-GPT directly use the data about the bounding box and motion direction of the dynamic obstacle, while StaticMLP and CIAO* require the projection of the predicted bounding boxes onto the obstacle map.

Overall, CIAO* showed average performance, which aligns with its status as a state-of-the-art method. MPPI was faster than CIAO* but frequently failed to find optimal trajectories, resulting in generally inferior performance. StaticMLP was the fastest, registering a computation time of 366.62 ms, which is over 400 ms faster than CIAO*. In contrast, NPField-GPT achieved the safest trajectory with the shortest path, as shown by the highest safety distance, lowest path length, and lowest AOL, though at the cost of increased computation time. This behavior suggests that NPField-GPT is the most appropriate choice for real-world scenarios where computation time can be traded for enhanced safety and path efficiency. In Table I, we abbreviate planner names as NPF-SMLP, NPF-DMLP, and NPF-GPT.

TABLE I
COMPARATIVE STUDIES ON DYNAMIC OBSTACLES SCENARIOS

Planner	Time, ms	Length, m	Smoothness	AOL	Safety distance, m
MPPI	563.82	4.13	0.036	0.026	0.036
CIAO*	780.83	3.62	0.028	0.019	0.112
NPF-SMLP	**366.62**	2.76	0.013	0.033	0.093
NPF-DMLP	493.50	2.15	0.017	0.058	0.109
NPF-GPT	663.81	**2.06**	**0.011**	**0.015**	**0.115**

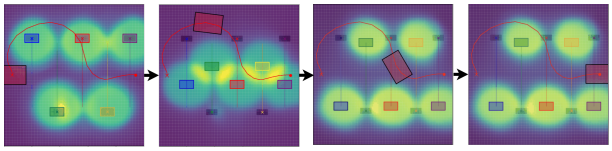

Fig. 6. Example of multi-obstacle scenarios solved by NPField-GPT. The potential field is obtained by summing per-obstacle predictions from the single-obstacle model without retraining.

B. Extension to multiple dynamic obstacles

NPField-GPT naturally extends to scenes with multiple dynamic obstacles via additive superposition of repulsive potentials, requiring no architectural changes or retraining. Given M dynamic obstacles, the combined occupancy map (with all obstacles rendered at the current timestep) is encoded once, while the per-obstacle dynamic state $I_{dyn}^{(j)} = (x_{dyn}^{(j)}, y_{dyn}^{(j)}, \theta_{dyn}^{(j)})$ is supplied individually to produce M independent embeddings. At each MPC query point, the model is evaluated M times and the resulting horizon potentials are summed: $J_o^{\text{multi}} = \sum_{j=1}^{M} J_o^{(j)}$. This superposition is integrated into the Acados solver via a single L4CasADi wrapper that loops over embeddings internally. We evaluate this scheme on scenarios with 2–5 moving obstacles (Fig. 6).

C. Real robot experiments

We tested the concept using a Husky UGV mobile manipulator as a ROS module. For local planning and control,

we utilized MPC, and for global planning, we used the Theta* planner [2]. Our testing scenario involved the robot maneuvering through a complex map. The entire navigation stack included a Cartographer [43] and RTAB-Map [44] for the global planner, with a second RTAB-Map for the local planner. All components of the control system were implemented as ROS nodes, with a central node managing communication with the Husky UGV hardware. Real-time planning and execution can be viewed in the supplementary video.

VIII. Conclusion, limitations and future work

This article presents a novel approach to MPC collision avoidance, which extends Neural Potential Field for the case of an environment with dynamic obstacles. We propose three neural architectures (NPField-StaticMLP, NPField-DynamicMLP, NPField-GPT) for predicting dynamic neural potentials and integrate them with real-time MPC.

Overall, the performance measures in Table I show that our controllers require hundreds of milliseconds to replan the trajectory. This is sufficient for an indoor mobile robot with limited velocity and a 1–2 Hz replanning rate, while faster systems such as cars or drones require a higher rate. Performance may be improved by using more powerful hardware and developing faster models and implementations.

It is worth noting that NPField is a learning-based method and therefore it has no theoretical safety guarantees. Therefore, the obtained solution has to be checked for collisions before its execution.

A significant assumption for the MPC problem is that for each dynamic obstacle, we know either a prediction of its future trajectory (NPField-StaticMLP) or a constant direction of its movement. We compensate for the inaccuracy of this assumption via replanning the trajectory on each step. Meaningful directions of future research include integrating neural potential forecasting with advanced motion prediction techniques, uncertainty-aware obstacle models, and joint multi-obstacle architectures that capture interaction effects beyond additive superposition.

Acknowledgments

The study was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, IGK 000000C313925P4B0002).

References

[1] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[2] Alex Nash et al. “Thetaˆ*: Any-angle path planning on grids”. In: *AAAI*. Vol. 7. 2007, pp. 1177–1183.

[3] L.E. Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[4] Steven M. LaValle and Jr. James J. Kuffner. “Randomized Kinodynamic Planning”. In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400.

[5] Grady Williams et al. “Aggressive driving with model predictive path integral control”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1433–1440.

[6] Grady Williams et al. “Information theoretic MPC for model-based reinforcement learning”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1714–1721.

[7] Lars Blackmore, Masahiro Ono, and Brian C Williams. “Chance-constrained optimal path planning with obstacles”. In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094.

[8] Jie Ji et al. “Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints”. In: *IEEE Transactions on Vehicular Technology* 66.2 (2016), pp. 952–964.

[9] Tobias Schoels et al. “An NMPC Approach using Convex Inner Approximations for Online Motion Planning with Guaranteed Collision Avoidance”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3574–3580.

[10] Tobias Schoels et al. “CIAO*: MPC-based Safe Motion Planning in Predictable Dynamic Environments”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 6555–6562.

[11] Zhiqiang Zuo et al. “MPC-based cooperative control strategy of path planning and trajectory tracking for intelligent vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 6.3 (2020), pp. 513–522.

[12] Damir Bojadžić et al. “Non-holonomic RRT & MPC: Path and trajectory planning for an autonomous cycle rickshaw”. In: *arXiv preprint arXiv:2103.06141* (2021).

[13] Jun Zeng, Bike Zhang, and Koushil Sreenath. “Safety-critical model predictive control with discrete-time control barrier function”. In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 3882–3889.

[14] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. “Safety-Critical Control and Planning for Obstacle Avoidance between Polytopes with Control Barrier Functions”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 286–292.

[15] O. Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505.

[16] Ajay Jain et al. “Discrete residual flow for probabilistic pedestrian behavior prediction”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 407–419.

[17] Neha Sharma, Chhavi Dhiman, and S Indu. “Pedestrian intention prediction for autonomous vehicles: A comprehensive survey”. In: *Neurocomputing* 508 (2022), pp. 120–152.

[18] Youshaa Murhij and Dmitry Yudin. "OFMPNet: Deep end-to-end model for occupancy and flow prediction in urban environment". In: *Neurocomputing* (2024), p. 127649.

[19] Jonathan Butzke et al. "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 258–265.

[20] Mike Phillips and Maxim Likhachev. "Sipp: Safe interval path planning for dynamic environments". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 5628–5635.

[21] Jiahui Lin et al. "Search-based online trajectory planning for car-like robots in highly dynamic environments". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 8151–8157.

[22] K. S. Yakovlev et al. "Safe interval path planning and flatness-based control for navigation of a mobile robot among static and dynamic obstacles". In: *Automation and Remote Control* 83.6 (2022), pp. 903–918.

[23] Zain Alabedeen Ali and Konstantin Yakovlev. "Safe interval path planning with kinodynamic constraints". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 10. 2023, pp. 12330–12337.

[24] Luigi Palmieri and Kai O Arras. "A novel RRT extend function for efficient and smooth mobile robot motion planning". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 205–211.

[25] Michael Otte and Emilio Frazzoli. "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning". In: *The International Journal of Robotics Research* 35.7 (2016), pp. 797–822.

[26] Ihab S Mohamed, Kai Yin, and Lantao Liu. "Autonomous navigation of agvs in unknown cluttered environments: log-mppi control strategy". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10240–10247.

[27] Steven Patrick and Efstathios Bakolas. "Path Integral Control with Rollout Clustering and Dynamic Obstacles". In: *arXiv preprint arXiv:2403.18066* (2024).

[28] Julius Ziegler et al. "Trajectory planning for Bertha — A local, continuous method". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 450–457.

[29] Giuseppe Franze and Walter Lucia. "A receding horizon control strategy for autonomous vehicles in dynamic environments". In: *IEEE Transactions on Control Systems Technology* 24.2 (2015), pp. 695–702.

[30] Michal Adamkiewicz et al. "Vision-Only Robot Navigation in a Neural Radiance World". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4606–4613. DOI: 10.1109/LRA.2022.3150497.

[31] Tim Salzmann et al. "Learning for CasADi: Data-driven Models in Numerical Optimization". In: (2023). arXiv: 2312.05873.

[32] Mikhail Kurenkov et al. "NFOMP: Neural Field for Optimal Motion Planner of Differential Drive Robots With Nonholonomic Constraints". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10991–10998.

[33] Maksim Katerishich et al. "DNFOMP: Dynamic Neural Field Optimal Motion Planner for Navigation of Autonomous Robots in Cluttered Environment". In: *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2023, pp. 1984–1989.

[34] Martin Jacquet and Kostas Alexis. *N-MPC for Deep Neural Network-Based Collision Avoidance exploiting Depth Images*. 2024. arXiv: 2402.13038 [cs.RO].

[35] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV].

[36] Muhammad Alhaddad et al. "Neural potential field for obstacle-aware local motion planning". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 9313–9320.

[37] N. Sturtevant. "Benchmarks for Grid-Based Pathfinding". In: *Transactions on Computational Intelligence and AI in Games* 4.2 (2012), pp. 144–148.

[38] Robin Verschueren et al. *Acados: a modular open-source framework for fast embedded optimal control*. 2020. arXiv: 1910.13753 [math.OC].

[39] Joel A E Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[40] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[41] Tim Salzmann et al. "Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms". In: *IEEE Robotics and Automation Letters* 8.4 (2023), pp. 2397–2404.

[42] Eric Heiden et al. "Bench-MR: A Motion Planning Benchmark for Wheeled Mobile Robots". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4536–4543.

[43] Wolfgang Hess et al. "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1271–1278.

[44] Mathieu Labbé and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation". In: *Journal of Field Robotics* 36.2 (Oct. 2018), pp. 416–446. ISSN: 1556-4967.