Neural NMPC through Signed Distance Field Encoding for Collision Avoidance

Journal Title
XX(X):1–24

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Martin Jacquet, Marvin Harms and Kostas Alexis

Abstract

This paper introduces a neural Nonlinear Model Predictive Control (NMPC) framework for mapless, collision-free navigation in unknown environments with Aerial Robots, using onboard range sensing. We leverage deep neural networks to encode a single range image, capturing all the available information about the environment, into a Signed Distance Function (SDF). The proposed neural architecture consists of two cascaded networks: a convolutional encoder that compresses the input image into a low-dimensional latent vector, and a Multi-Layer Perceptron that approximates the corresponding spatial SDF. This latter network parametrizes an explicit position constraint used for collision avoidance, which is embedded in a velocity-tracking NMPC that outputs thrust and attitude commands to the robot. First, a theoretical analysis of the contributed NMPC is conducted, verifying recursive feasibility and stability properties under fixed observations. Subsequently, we evaluate the open-loop performance of the learning-based components as well as the closed-loop performance of the controller in simulations and experiments. The simulation study includes an ablation study, comparisons with two state-of-the-art local navigation methods, and an assessment of the resilience to drifting odometry. The real-world experiments are conducted in forest environments, demonstrating that the neural NMPC effectively performs collision avoidance in cluttered settings against an adversarial reference velocity input and drifting position estimates.

Keywords

Autonomous Navigation, Aerial Robotics, Neural Networks, Nonlinear MPC, Signed Distance Function

1 Introduction

Recent advances in autonomous navigation and aerial robotics have enabled the large-scale deployment of robots in several challenging applications, such as subterranean or forest exploration and ship inspection Fang et al. (2017); Tian et al. (2020); Tranzatto et al. (2022). However, navigation in unknown and unstructured environments while relying only on onboard sensing remains a strenuous challenge. This is particularly true in perceptually-degraded environments and conditions, where mapping systems are failure-prone Ebadi et al. (2024).

Conventional state-of-the-art navigation stacks often involve the construction of a reliable volumetric map of the scene, followed by a planning step Sucan et al. (2012); Tranzatto et al. (2022). Although this approach has proven effective in many cases, errors at the mapping level (such as a sudden localization drift) propagate downstream, adversely affecting planning and control. Furthermore, the sequential update of dense maps before motion planning induces latency, which limits the responsiveness to unforeseen events. Enhanced resilience w.r.t. collisions can be achieved by integrating map-based methodologies with local reactive strategies that rely on instantaneous exteroceptive observations.

Accordingly, the robotics community has investigated a set of approaches for the so-called mapless navigation problem, i.e., the problem of achieving collision-free navigation without relying on explicit mapping. Beyond early heuristic approaches to reactive collision avoidance Siegwart et al. (2011), more advanced strategies have emerged in recent years. While initial efforts focused on model-based methods Florence et al. (2018); Gao et al. (2019); Florence et al. (2020); Yadav and Tanner (2020); Zhou et al. (2021a), deep learning-based techniques have taken the field by storm, primarily due to their scalability to high-dimensional exteroceptive inputs such as depth images. Relevant learning strategies include Imitation Learning (IL) Loquercio et al. (2021); Lu et al. (2023), and Reinforcement Learning (RL) Hoeller et al. (2021); Ugurlu et al. (2022); Kulkarni and Alexis (2024). In addition to these end-to-end methods, recent research has investigated the use of deep learning for implicit environment representations, combined with principled control strategies Harms et al. (2024); Jacquet and Alexis (2024).

This work falls into this latter category, and its main contribution is threefold. First, we introduce a neural architecture for encoding a single range observation as an Signed Distance Function (SDF) Park et al. (2019); Sitzmann et al. (2020). Unlike prior work on (neural or non-neural) scene-scale implicit SDF encoding – which primarily targets mapping and planning

All authors are affiliated with the Autonomous Robots Lab, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Corresponding author:
Martin Jacquet, Autonomous Robots Lab, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Høgskoleringen 1, 7034 Trondheim, Norway.
Email: martin.jacquet@ntnu.no

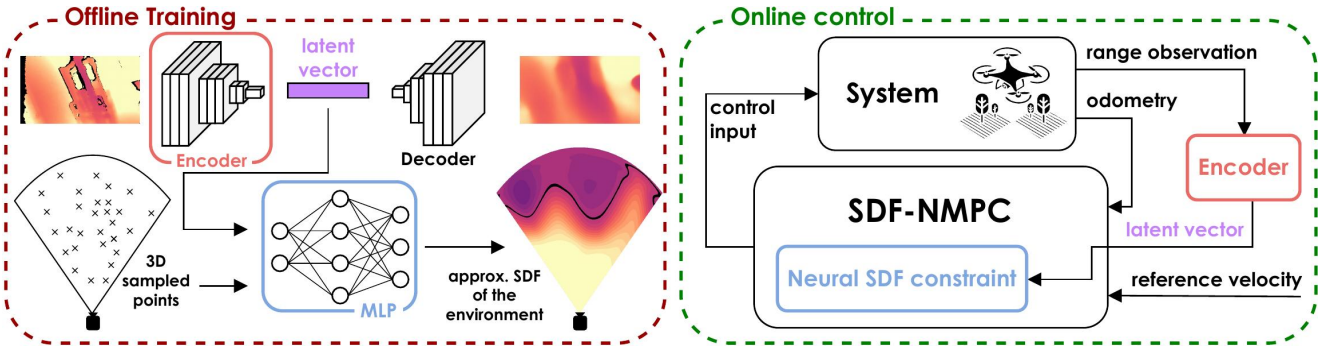

Figure 1. An overview of the proposed SDF-NMPC method. The left side depicts the neural architecture used to approximate the mapping between the input depth images and the corresponding SDF, through sampling-based training in the 3D sensor frustum. The Convolutional encoder-decoder and MLP networks are trained sequentially. The right side presents the proposed control scheme, highlighting the contributed neural-constrained NMPC for velocity tracking, and the color-coded learning-based components.

applications Ortiz et al. (2022); Wu et al. (2023) – we explicitly avoid aggregating multiple observations. Second, we integrate this representation into a neural Nonlinear Model Predictive Control (NMPC) for achieving collision-free, mapless navigation in unknown environments. We verify that the NMPC satisfies recursive feasibility (under fixed observations) and stability conditions. An overview block diagram of the contributed method, referred to as SDF-NMPC, is shown in Figure 1. Thirdly, we conduct a detailed, component-wise evaluation, including a quantitative assessment of the neural environment encoding, ablation studies, comparisons with existing methods, and real-world experiments involving drifting odometry and adversarial reference inputs.

The proposed NMPC relies solely on instantaneous range measurements from an onboard sensor and on odometry, which may suffer from position drift. The neural SDF encoding of the current range observation parametrizes a position constraint for the controller. To address the limited sensor Field of View (FoV), the predicted motion is also constrained to remain within the visible frustum. The controller generates thrust and attitude commands that satisfy both collision and FoV constraints, and is integrated in a velocity-tracking framework for real-time control of an Aerial Robot (AR). The implementation of the contributed NMPC and the associated neural networks is released open-source*.

Accordingly, this work departs from several limitations of our previous contribution in Jacquet and Alexis (2024). First, the proposed control scheme generalizes to the nonlinear dynamics of the multirotor, moving beyond the previously considered unicycle-like motions. Second, we make use of a richer 3D representation. Indeed, the SDF is well-suited for collision avoidance, being a continuous, almost everywhere differentiable 3D field that encodes both the distance and direction to the nearest obstacle. Defined purely at the position level, it can be constructed directly from sensor measurements, whereas defining safe sets for derivative states (e.g., velocity or acceleration) is often challenging for nonlinear, higher-order systems Harms et al. (2024). Moreover, the neural encoding provides a closed-form differentiable expression, enabling straightforward integration with gradient-based NMPC. The SDF formalism also facilitates the definition of a forward-invariant safe set for the receding-horizon problem, allowing us to establish recursive feasibility of the control law without introducing excessively restrictive terminal constraints. Lastly, the evaluation is substantially more comprehensive, including the explicit evaluation of the resilience of the 0-memory approach against drifting odometry.

The remainder of this article is structured as follows. Section 2 provides an overview of the related literature, followed by the problem statement in Section 3. The proposed method is presented in Section 4 and Section 5, respectively covering the neural environment encoding and the collision-avoidance NMPC framework. Finally, Section 6 presents a comprehensive evaluation of the method, before discussions and final remarks in Section 7.

2 Related Work

This section presents related work, and especially a) recent advances in neural NMPC, b) methodologies for mapless navigation, and c) neural distance fields and their applications in robotics.

2.1 Neural MPC

Nonlinear Model Predictive Control (MPC) has has become a widely adopted control strategy for constrained systems, showing strong performance for ARs Sun et al. (2022).

With advances in onboard computing and machine learning, learning-based MPC emerged as a successful alternative to robust or stochastic MPC, offering online adaptability to model mismatches and external disturbances Hewing et al. (2020). Gaussian Processes, for instance, have been used to model residual dynamics in Kabzan et al. (2019); Torrente et al. (2021). Neural Networks (NNs), capable of capturing complex data patterns, have also been used to model complex dynamics, such as aerodynamic effects for quadrotor flight Bauersfeld et al. (2021). Following a similar approach, neural NMPC has been proposed, embedding small-scale NNs to learn unmodeled dynamics Williams et al. (2017); Chee et al. (2022); Gao et al. (2024). Going further, Syntakas and Vlachos (2024) introduces an ensemble approach

*https://github.com/ntnu-arl/sdf-nmpc

based on the Monte-Carlo dropout technique, addressing the epistemic uncertainty in the neural models. RL has also been used to optimize the NMPC model, cost, and constraints to enhance closed-loop performanceGros and Zanon (2020), further extended by including a neural model for unknown dynamics Adhau et al. (2024).

An open-source toolbox for integrating large NNs into NMPC is proposed in Salzmann et al. (2023), enabling tasks like navigation in turbulent flow Salzmann et al. (2024). This paved the way for exploiting complex neural representations in NMPC. In Alhaddad et al. (2024), a transformer-based NN have been used to represent obstacle fields as repulsive cost, while Jacquet and Alexis (2024) embeds depth camera observations into a differentiable occupancy map, explicit used as a position constraint. This work builds upon this architecture.

Other recent architectures reframe NMPC as a parameterized controller learned via policy search for agile flight Song and Scaramuzza (2022), or integrate differentiable MPC into RL agents for tighter coupling between short- and long-term control Romero et al. (2024). RL-based warm-starting strategies have been proposed to initialize the NMPC with actor-critic rollouts Reiter et al. (2024), or approximate terminal costs using neural networks Alsmeier et al. (2024). Additionally, Celestini et al. (2024) leverages a Transformer to generate trajectory candidates for warm-starting the solver.

A set of novel architectures for neural MPC has been explored. In Song and Scaramuzza (2022), the NMPC is formulated as a parametrized controller learned via policy search for agile flight. Another avenue relies on NN to approximate the infinite horizon cost. In Romero et al. (2024), a differentiable MPC implementation East et al. (2020) is integrated in the RL agent, such that the MPC drives the short-term actions while the critic network manages the long-term ones. RL-based warm-starting strategies have been proposed in Reiter et al. (2024) to initialize the NMPC via policy roll-out. In Alsmeier et al. (2024), the long-horizon cost of the MPC is approximated by the NN, and used as the terminal cost of a short-horizon problem, for computational efficiency. Additionally, in Celestini et al. (2024), a Transformer is used to generate candidate trajectories for warm-starting MPC solver.

2.2 Mapless Navigation

Mapless navigation has gained attention in recent years. A first class of approaches constructs local volumetric representations from recent observations using structures like k-d trees Florence et al. (2018); Gao et al. (2019), enabling fast collision checking. Ray-casting Yadav and Tanner (2020), or depth map back-projection Matthies et al. (2014) can also be used for fast collision checks. Leveraging such efficient checks, motion primitives have been used for planning Lopez and How (2017); Bucki et al. (2020). In Zhou et al. (2021a), a spline-based planner using differentiable repulsive forces from visible obstacles is introduced. Contrary to these works, which rely on a single depth measurement to perform checks, Florence et al. (2018) introduces a data representation for querying points in a sliding window of consecutive observations, explicitly accounting for the transform uncertainty induced by the uncertain state estimation. In Zhang et al. (2025), a NMPC is proposed leveraging a k-d tree to query the N closest obstacles for each shooting node along a pre-sampled path, that are in turn used as 1D position constraints for this shooting node. The approach struggles in cluttered settings where the path must significantly diverge from the straight-line reference.

Deep learning offers an alternative to collision checking, and exhibits good results for mapless navigation. Nguyen et al. (2022, 2024) proposes a neural architecture to correlate depth images to near-future collisions for a given action. This method was extended with a Variational Auto-Encoder (VAE) to allow more control over the latent representation, enabling semantic augmentation for task-specific navigation Kulkarni et al. (2023). A similar approach is proposed in Lu et al. (2023) where a Q-value-inspired model predicts action values using expert policy rollouts. In Liang et al. (2024), a generative NN is used to generate candidate trajectories that satisfy traversability and coverage learned from A*-generated data.

Imitation Learning (IL) has also been employed to learn how to generate sensor-based trajectories by imitating a map-informed export policy, such as an NMPC Tolani et al. (2021), a sampling-based expert Loquercio et al. (2021), or a privileged RL agent Song et al. (2023). However, IL often lacks generalization of the learned policy. Thus, numerous works tackled this problem with RL Kahn et al. (2021a,b). For instance, Ugurlu et al. (2022) trains a policy from depth data but neglects AR dynamics during training, while Kulkarni and Alexis (2024) incorporates dynamics and achieves real-world deployment. Some RL-based approaches also learn implicit representations of time-consistent environments and robot state Hoeller et al. (2021).

A hybrid strategy leverages NNs to encode a compressed, interpretable environment representation. Several works use neural networks to learn Control Barrier Functions (CBFs) for a given system, which describes safety w.r.t. to the obstacles as described in range measurements. A relevant instance is provided by Dawson et al. (2022), where an observation-based CBF is learned for a first-order system. The method relies on approximating the LiDAR observation dynamics, which is difficult for higher-order systems and error-prone in complex environments, because the true observation dynamics are, in general, discontinuous. More accurate prediction of the next observation can be achieved with a NeRF Tong et al. (2023) and used in the CBF, but is still limited to linear (first or second order) dynamics and struggles to extend to nonlinear motions.

Instead of approximating the observation dynamics, Harms et al. (2024) proposes treating observations as parameters of safe sets in a switching approach, relying on the principle that a state safe under one observation remains safe over time, and provides a constructive approach of the LiDAR-based CBF (for an input-constrained second-order linear system). Other works approximate a CBF from obstacle-centered SDFs Long et al. (2021) or via a GP Keyumarsi et al. (2024), though both are restricted to first-order dynamics.

In general, these techniques struggle to faithfully encode observation-based safe sets for systems with complex (i.e., nonlinear or high-order) dynamics. Our previous work Jacquet and Alexis (2024) proposes an alternative that

consists of using the NN to encode a simpler, position-only representation in the form of an occupancy map, while the dynamics are handled by an NMPC. However, the occupancy map gradient is nearly zero everywhere except at obstacle boundaries, limiting expressivity Our proposed method overcomes this by encoding observations as an SDF.

2.3 Implicit Distance Fields

In computer vision, NNs are commonly used for implicit representation of surfaces or volumes, i.e., as the 0-levelset of a scalar field. Indeed, Coordinate-based networks Tancik et al. (2020) take spatial coordinates as input and output the corresponding Boolean occupancy Chen and Zhang (2019); Mescheder et al. (2019) or SDF values Park et al. (2019), offering continuous, compact representations encoded in network weights or latent spaces. These methods have been applied to shape rendering Wang et al. (2021); Azinović et al. (2022); Wang et al. (2022) and completion Dai et al. (2017); Stutz and Geiger (2018).

In robotics, SDFs are often organized as Euclidean Signed Distance Functions (ESDFs), assigning voxel-wise distances to the nearest obstacle. However, obtaining an SDF from sensor data, online, can be a challenging task. Classical approaches Newcombe et al. (2011); Oleynikova et al. (2017) often construct these incrementally from Truncated Signed Distance Functions (TSDFs), which store truncated distances along sensor rays. In Han et al. (2019) an efficient alternative using occupancy grids with fast indexing is proposed.

Consequently, neural SDFs have gained traction for mapping and localization. Neural SLAM frameworks have been introduced in Sucar et al. (2021); Zhu et al. (2022); Wang et al. (2023), where an SDF is encoded in an Multi-Layer Perceptron (MLP) and learned online from RGBD images. Other approaches build SDFs online using TSDFs-inspired schemes Yan et al. (2021); Ortiz et al. (2022), with extensions for object structure Jang et al. (2024) and hierarchical representation Vasilopoulos et al. (2024). A comprehensive mapping framework with global consistency is proposed in Pan et al. (2024). Separately, a relevant, non-neural framework uses GP-based SDF for non-neural mapping, odometry, and planning Wu et al. (2023).

Aside from mapping and navigation, implicit SDFs are also employed for manipulation, e.g., as observations to an RL agent for grasping in Mosbach and Behnke (2022). Differently, in Lee and Liu (2024) a network is trained to predict SDF of the swept volume of a manipulator arm, to ensure operator safety. A non-neural example is provided in Marić et al. (2024), where the SDF is approximated using piecewise polynomial SDFs with enforced $\mathcal{C}^1$ continuity. While relying on few parameters compared to NNs, the scalability to large, complex scenes is questionable, and the construction time remains prohibitive for real-time applications.

3 Problem Statement

We consider the problem of collision-free navigation in an unknown, static environment by exploiting the immediate high-resolution exteroceptive range observations. We assume a reference velocity to be available, which is followed when possible while remaining collision-free. The AR is assumed to be a rigid body, whose state vector is denoted by $\mathbf{x} \in \mathcal{X}$ and the input vector is written as $\mathbf{u} \in \mathcal{U}$. We assume that all collision constraints can be expressed as constraints on the position of the robot alone, and that the system state $\mathbf{x}$ contains position information, denoted $\mathbf{p} \in \mathbb{R}^3$.

The system state evolves according to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

where $\mathbf{f}$ denotes the nonlinear AR dynamics (detailed, e.g., in Lee et al. (2010)), which is recalled in Section 5.2. The AR also integrates a constrained-FoV range sensor that periodically produces observations $\mathbf{o} \in \mathcal{O}$ of a body-centric 3D volume defined by the sensor frustum, denoted $\mathbb{F} \subset \mathbb{R}^3$. The observation $\mathbf{o}$ is a function of the (potentially uncertain) state vector $\mathbf{x}$ and the unknown environment. In turn, this implies that the mapping function ξ between the state and the observation $\mathbf{o} = \xi(\mathbf{x})$ is unknown.

We specifically tailor the scope of this work to 0-memory navigation, i.e., at a given time t, the set $\mathcal{X}_{\text{free}}(t)$ of states which are not in collision is a function of $\mathbf{o}$. Because of the constrained FoV, such a set can strictly be defined only within the visible volume $\mathbb{F}(t)$. Accordingly, we define the mapping $h : \mathbb{R}^3 \times \mathcal{O} \to \mathbb{R}$ as

$$\mathcal{X}_{\text{free}}(t) = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{p} \in \mathbb{F}(t),\ h(\mathbf{p}, \mathbf{o}) \geq 0\}. \tag{2}$$

This mapping h defines the position constraint that must be satisfied to ensure collision avoidance. However, the time derivative of h cannot be obtained, as ξ is unknown and in general discontinuous. Instead, we resort to a switching update approach, as in Harms et al. (2024), that avoids an explicit approximation of the observation dynamics. At each step k, the discrete $\mathbf{o}_k$ defines a parametric mapping $h_{\mathbf{o}_k} : \mathbb{R}^3 \to \mathbb{R}$, which in turn defines (together with the corresponding frustum $\mathbb{F}_k$) the set $\mathcal{X}_{\text{free},k}$ of reachable positions w.r.t. the observation frame.

Our proposed method for tackling this problem is presented hereafter. In Section 4, we present an approach for approximating $h_{\mathbf{o}}$, while in Section 5 we present the actual system model and derive the subsequent definition of the optimal control problem.

4 Learning the Environment Representation

Given the most recent observation $\mathbf{o}$, we want to explicitly define the mapping $h_{\mathbf{o}}$ that defines the set of free positions within the sensor frustum $\mathbb{F}$. However, the very high dimensionality of the observations (typically, $\approx 1\text{e}5$ pixels or 3D points) prevents naive implementations that would directly transpose each pixel to individual constraints. It is therefore necessary to use an intermediate volumetric representation of the visible environment. However, classical approaches for computing said representations (e.g., raycasting or k-d tree search) suffer limitations such as a) slow computation, b) algorithmic computing, or c) discontinuous representations, which in turn prevents their embedding in gradient-based optimal controllers.

The core of the proposed approach is to leverage deep learning to encode the sensor observation into a single volumetric constraint, using a neural implicit SDF constructed from a single depth image.

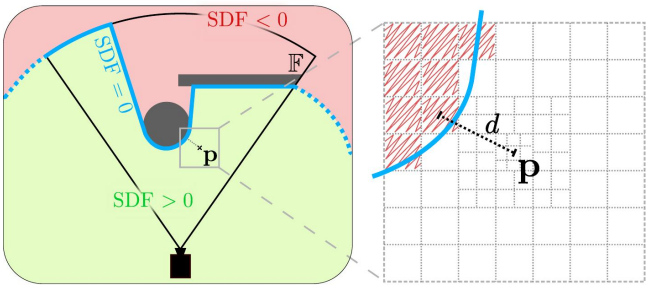

Figure 2. A 2D visualization of the distance transform (left) with two visible obstacles (gray). The blue line marks the 0-level set of the SDF, and its dashed part illustrates its heuristic extension beyond the FoV $\mathbb{F}$ for training purposes. The zoom-in (right) illustrates the grid-based approximation of the distance transform for $\mathbf{p}$ by the distance d between the central cell and the closest cell with a different occupancy (in red).

4.1 SDF from observations

We define the distance field as the spatial distance function to the surface of visible obstacles. Specifically, it must have negative values for non-visible space, and positive values for visible space, as shown in Figure 2. Such a function is called a distance transform Maurer et al. (2003). As commonly done for neural SDFs Park et al. (2019), with inherently limited representation capabilities, the SDF is truncated beyond some user-defined distance, denoted T_{SDF}. For precise surface encoding, T_{SDF} is typically a few centimeters. For navigation, however, the obstacles are represented at a larger scale and we set $T_{\text{SDF}} = 1\,\text{m}$.

However, this distance transform is defined on $\mathbb{F} \subset \mathbb{R}^3$. To ensure proper boundary conditioning during training, its definition is extended to $\mathbb{R}^3$ by assigning SDF values based on the spherical projection onto $\mathbb{F}$ (see Figure 2). This entails that an additional constraint must enforce the queried positions to be within $\mathbb{F}$, where the SDF truly relates to the observation.

Although there exist efficient parallelized algorithms for computing such a transform Criminisi et al. (2008); Cao et al. (2010), those are tailored for computing the full distance transform over a given grid. This is typically not well suited for sampling-based training. Instead, we utilize an approximation algorithm to generate the target values $\text{SDF}(\mathbf{p})$, for each sampled 3D position $\mathbf{p}$, given the observation. This algorithm, illustrated in Figure 2, consists of fitting a 3D grid of adaptive resolution on the sampled point, locating the closest cell where the occupancy changes. The grid is dense around the center (thus enforcing higher precision close to surfaces) but gets increasingly sparser for achieving efficient online computation. The algorithm seamlessly provides the direction of the spatial gradient, whose norm satisfies an eikonal equation

$$\left\| \frac{\partial \text{SDF}(\mathbf{p})}{\partial \mathbf{p}} \right\| = \begin{cases} 1 & \text{if } \|\text{SDF}(\mathbf{p})\| < T_{\text{SDF}} \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

We note that the precision of the approximation algorithm is therefore lower bounded by the thinnest resolution of the grid. This algorithm can be tensorized for online training data generation. It relies on a routine for evaluating the occupancy of arbitrary points given the input image, which can be efficiently implemented on GPU kernels via backprojection onto the pixel grid, e.g., using Warp Macklin (2022).

4.2 NN architecture

We use a two-step process to handle observations. First, a Convolutional Neural Network (CNN) encodes the observation into a compact representation. This latent vector is then passed through a MLP for further processing., exploiting the spatial correlation properties of convolutions.

The two-step approach allows for a) avoid redundant computation when evaluating the SDF at multiple points given the same observation, b) leverage spatial patterns in the input via the CNN, while keeping the MLP (which parametrizes the collision constraint) lightweight, and finally c) improve robustness to the noise and systematic errors in the observations Kulkarni et al. (2023) (e.g., stereo shadow and invalid LiDAR rays).

We use a β-VAE Higgins et al. (2017) architecture with a ResNet-10 network as the encoder. It incorporates ReLU activations, batch normalizations, and dropout regularization for improved generalizability. The final convolution volume is passed through an average pooling, before flattening and processing through an Fully Connected (FC) layer for computing the mean ($\boldsymbol{\mu}$) and standard deviation (std) ($\boldsymbol{\sigma}$) parametrization of the Gaussian space. Accordingly, we use a deconvolution network based on ResNet-10 as the decoder.

The latent encoding $\mathbf{z}$ is processed using a coordinate-based MLP Tancik et al. (2020) that approximates the distance transform for the corresponding observation.

Similar to Ortiz et al. (2022), positional embedding is first applied to the 3D point $\mathbf{p}$, mapping it to a high-dimensional space using periodic activation functions. This sinusoidal mapping of input coordinates allows MLPs to better represent higher frequency content Tancik et al. (2020). We use the off-axis positional embedding $\boldsymbol{\gamma}$ Barron et al. (2022), defined as

$$\boldsymbol{\gamma}(\mathbf{p}) = \begin{bmatrix} \mathbf{p} \\ \sin(2^0 \mathbf{A}\mathbf{p}) \\ \cos(2^0 \mathbf{A}\mathbf{p}) \\ \vdots \\ \sin(2^{L-1} \mathbf{A}\mathbf{p}) \\ \cos(2^{L-1} \mathbf{A}\mathbf{p}) \end{bmatrix} \in \mathbb{R}^{2L+3}, \tag{4}$$

where L is the number of frequency bands, and the rows of $\mathbf{A} \in \mathbb{R}^{12\times 3}$ are vertices of a unit-norm icosahedron.

This embedding is concatenated to $\mathbf{z}$, and processed via an MLP with 4 hidden layers and using sine activations Sitzmann et al. (2020). The input to the MLP is also feed-forwarded to the third hidden layer. The MLP also uses dropout regularization. Various sizes for the hidden layers are evaluated in Sections 6.2 and 6.3.

We note that the VAE operates directly on pixels, and is therefore agnostic of the sensor FoV. However, because of the convolution operations, the network is trained for a specific input image ratio, which tends to differ between depth cameras (e.g., $16:9$) and LiDAR range images (e.g., $4:1$).

The SDF MLP operates on spatial data and implicitly encodes the intrinsic projection matrix, and is therefore trained for a specific sensor.

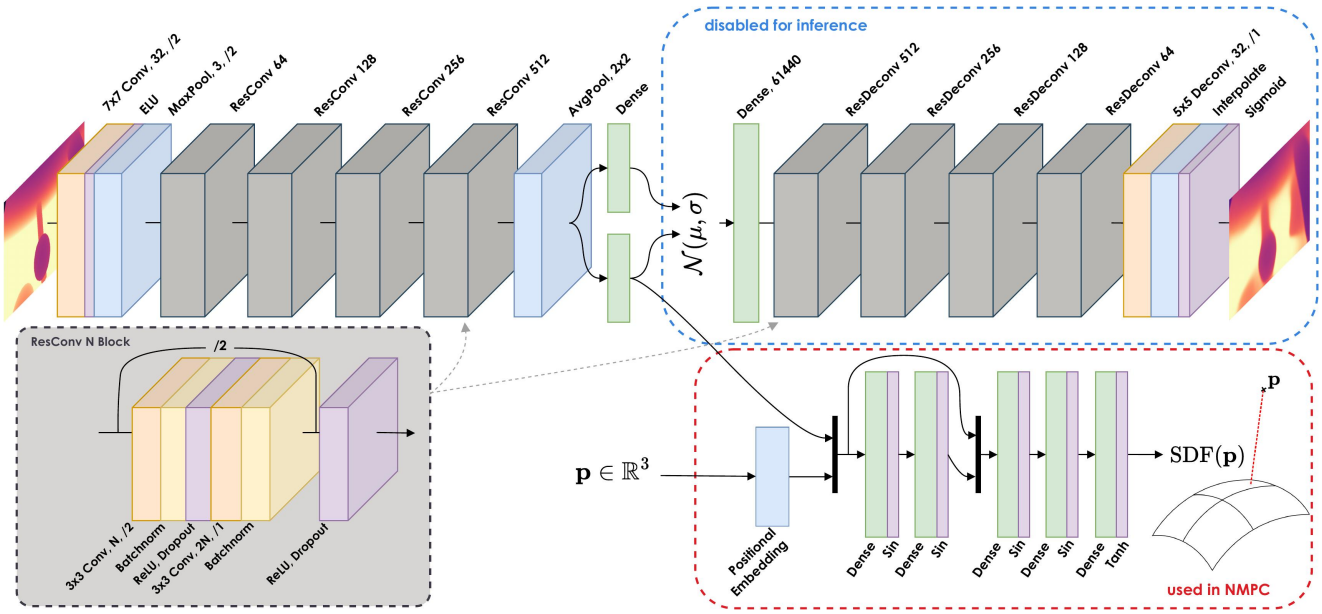

Figure 3. Architecture of the neural networks. Convolution and deconvolution layers are pictured in orange, linear layers in green, activations in purple, and pooling layers in blue. The encoder computes a Gaussian latent representation of the image, parametrized by its mean and std. The latent is sampled and decoded to reconstruct the input image. The decoder (blue rectangle) is used for training the VAE but is disabled for inference. The Residual block for a given size N is pictured in the gray rectangle and instantiated in the network structure. Residual Deconvolution blocks follow the exact same structure.

	Depth camera		**LiDAR**	
	train	test	train	test
Real	15165	1685	18682	2076
Gazebo	48039	5338	23707	2634
Aerial Gym	91697	10189	89673	9964

Table 1. Number of images used in the training and testing datasets. The sensors for each modality are an Intel D455 depth camera and an Ouster OS1 LiDAR.

4.3 Training procedure

The VAE and MLP are trained sequentially.

4.3.1 VAE Training First, the VAE is trained using a dataset including images from real sensors, as well as simulated images collected from Gazebo, and from Aerial Gym Kulkarni et al. (2025) based on Isaac Sim and using Warp rendering. The first two sets consist of indoor and outdoor scenes, while the latter consists of geometric shapes of arbitrary positions and orientations. The number of images for each set is reported in Table 1. The training set is randomly divided into training and validation subsets, using an 85%-15% ratio. The input images are cropped up to a maximum range, and normalized to $[0,\ 1]$. The maximum range $d_{\max}$ describes the range of environmental information encoded by the network. In practice, we set $d_{\max} = 5\,\mathrm{m}$, which provides a sufficient horizon for effectively managing local collision avoidance.

We follow the training procedure in Kulkarni et al. (2023), i.e., no reconstruction loss is computed for the invalid pixels caused by stereo shadow or obstructions.

Because the encoder is used to prevent collisions, all obstacles must be reconstructed despite the high compression rate, in particular those close to the robot. We therefore propose to bias the VAE to reconstruct close obstacles with higher accuracy. This is achieved by scaling the loss function according to the distance value of the target pixel. The scaling factor is computed such that it equals 1 for 0-valued pixels, and $w \in (0,\ 1)$ for 1-valued pixels, where w is a tunable parameter. The intermediate values are obtained using a quadratic interpolation with a horizontal tangent at 0.

The pixel-wise MSE reconstruction loss function of the VAE for an observation $\mathbf{o}$ is hence defined as

$$\mathcal{L}_{\text{MSE}}(\mathbf{o}) = \sum_{\mathbf{o}_{ij} \neq 0} (\mathbf{o}_{ij}^2(w-1)+1)(\text{VAE}(\mathbf{o})_{ij} - \mathbf{o}_{ij})^2, \quad (5)$$

where the subscript $\bullet_{ij}$ refers to the pixel on the i-*th* line and j-*th* column.

Additionally, a β-scaled Kullback-Leibler Divergence (KLD) metric $\mathcal{L}_{\text{KL}}$ is used to enforce that the latent vector $\mathbf{z}$ follows a proper normal distribution fitting the true posterior, as commonly done for variational encoders. It is defined as

$$\mathcal{L}_{\text{KL}} = -\frac{\beta_{\text{norm}}}{2}(1 + \log(\boldsymbol{\sigma}^2) - \boldsymbol{\mu}^2 - \boldsymbol{\sigma}^2), \quad (6)$$

where the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the predicted mean and std of the latent distribution, and β_{norm} is a scaling factor computed from the latent and image sizes, according to Higgins et al. (2017).

4.3.2 SDF Training Then, the weights of the encoder are frozen, and the MLP is trained to approximate the SDF. The training is also fully supervised, using regression of the target SDF value and gradient at sampled 3D points. Target values for sampled points are computed following the procedure described in Section 4.1. The training is performed using the same dataset as the VAE. However, because the SDF ground truth approximation relies on backprojection onto the image plane, the image must contain only valid pixels. The images from the real sensors are therefore not used for

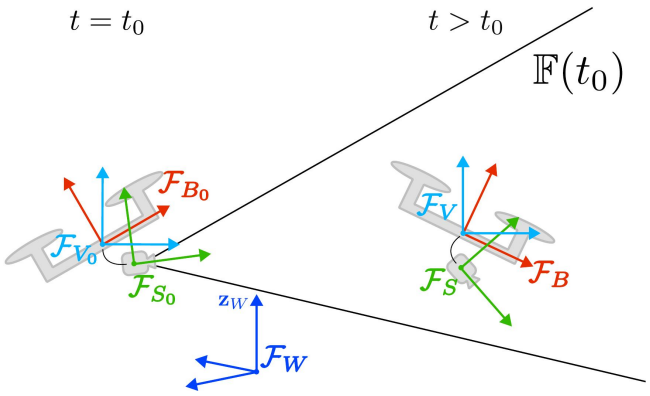

Figure 4. Planar representation of the relevant 3D frames used in the paper, including the inertial frames at time t_0 where the sensor observation is captured (left), and the sensor frustum $\mathbb{F}$.

training the MLP. The sampling process is such that points are sampled a) within $\mathbb{F}$, b) in a ball centered on O_S (to enforce accuracy close to the origin), c) close to obstacles (for increased surface accuracy), and d) in a ball larger that $\mathbb{F}$ for boundary conditioning. We use the sampling ratios 0.4, 0.35, 0.2 and 0.05, respectively.

The loss function is defined as the weighted sum of two MSE losses, for both the target SDF value and its gradient.

We note that the MSE loss on the unit norm gradient includes the satisfaction of the eikonal property (3). We report, in Appendix A, the training losses and metrics for both the VAE and SDF networks, which are used in the remainder of the paper.

5 Neural NMPC

5.1 Mathematical Notations and Coordinate Frames

We denote a coordinate frame $\bullet$ as $\mathcal{F}_\bullet$, its' origin as $O_\bullet$ and canonical base $(\mathbf{x}_\bullet, \mathbf{y}_\bullet, \mathbf{z}_\bullet)$. We denote a generic world inertial frame with $\mathcal{F}_W$. The position and orientation of the AR are represented using the body-aligned frame $\mathcal{F}_B$ attached to the geometric center of the robot O_B. Exteroceptive measurements made by the body-mounted range sensor are described in a sensor frame $\mathcal{F}_S$. Further, we define the vehicle frame $\mathcal{F}_V$, such that $O_V = O_B$, $\mathbf{z}_V = \mathbf{z}_W$, and which is yaw-aligned with $\mathcal{F}_B$.

Lastly, we consider the inertial frames $\mathcal{F}_{B_0}$, $\mathcal{F}_{V_0}$, $\mathcal{F}_{S_0}$, which at time $t = t_0$ coincide with $\mathcal{F}_B$, $\mathcal{F}_V$ and $\mathcal{F}_S$, respectively. A visual summary of the relevant frames is provided in Figure 4.

The relative position of $\mathcal{F}_a$ w.r.t. $\mathcal{F}_b$, expressed in $\mathcal{F}_b$, is denoted by ${}^b\mathbf{p}_a$. Similarly, the orientation of $\mathcal{F}_a$ w.r.t. $\mathcal{F}_b$ expressed in $\mathcal{F}_b$ is denoted by ${}^b\mathbf{R}_a$. The unit quaternion representation of the rotation ${}^b\mathbf{R}_a$ is denoted ${}^b\mathbf{q}_a$, and its Euler angle representation is ${}^b\boldsymbol{\eta}_a = \begin{bmatrix}\phi\ \theta\ \psi\end{bmatrix}$, following the ZYX Tait–Bryan convention.

Lower and upper bounds on any variable are respectively denoted with $\underline{\bullet}$ and $\overline{\bullet}$, and $\otimes$ denotes the Hamilton product of two quaternions.

5.2 System Modeling

The AR considered hereafter is a standard co-planar multirotor. It is modeled as a rigid body of mass m centered in O_B, and actuated by 4 or more co-planar propellers. We assume that the robot is fully enclosed in a sphere of radius r.

We denote t_0 as the time at which the most recent observation is acquired. We chose to represent the system position and orientation w.r.t. the inertial, switching frame $\mathcal{F}_{V_0}$, that is, the vehicle frame at t_0. The heading with respect to $\mathcal{F}_{V_0}$ is defined in quaternion form, denoted ${}^{V_0}\mathbf{q}_V$. It has 2 non-zero components (the scalar and z ones), respectively denoted ${}^{V_0}q_w$ and ${}^{V_0}q_z$.

Therefore, the system state is described, dropping the subscript $\bullet_B$ for legibility, by the vector

$$\mathbf{x} = \left[{}^{V_0}\mathbf{p}^\top,\ {}^{V_0}q_w,\ {}^{V_0}q_z,\ {}^{V_0}\mathbf{v}^\top\right]^\top \in \mathbb{R}^8, \tag{7}$$

where ${}^{V_0}\mathbf{v}$ is the velocity of O_B, expressed in $\mathcal{F}_{V_0}$.

Accordingly, we select the system input variables in order to control the 3D thrust (magnitude and orientation through roll and pitch) and yaw rate, as typically done for co-planar multirotors Furrer et al. (2016). The system input vector is therefore

$$\mathbf{u} = \left[T,\ {}^V\phi,\ {}^V\theta,\ {}^B\omega_z\right]^\top \in \mathbb{R}^4, \tag{8}$$

where T is the collective thrust of the actuators along $\mathbf{z}_B$, ${}^V\phi$ and ${}^V\theta$ are the roll and pitch angles of ${}^V\mathbf{R}_B$, and ${}^B\omega_z$ is the angular rate around $\mathbf{z}_B$, expressed in $\mathcal{F}_B$. It is assumed that lower and upper bounds on $\mathbf{u}$ are known, or obtained through an identification campaign.

The system dynamics (1) considered in the problem formulation are therefore instantiated as

$${}^{V_0}\dot{\mathbf{p}} = {}^{V_0}\mathbf{R}_B\,{}^{V_0}\mathbf{v}, \tag{9a}$$

$$\begin{bmatrix}{}^{V_0}\dot{q}_w\\ 0\\ 0\\ {}^{V_0}\dot{q}_z\end{bmatrix} = \frac{1}{2}\begin{bmatrix}{}^{V_0}q_w\\ 0\\ 0\\ {}^{V_0}q_z\end{bmatrix} \otimes \begin{bmatrix}0\\ 0\\ 0\\ {}^B\omega_z\end{bmatrix}, \tag{9b}$$

$${}^{V_0}\dot{\mathbf{v}} = \frac{T}{m}\,{}^{V_0}\mathbf{R}_B\mathbf{z}_B - g\mathbf{z}_V, \tag{9c}$$

where ${}^{V_0}\mathbf{R}_B = {}^{V_0}\mathbf{R}_V\,{}^V\mathbf{R}_B$ is the composition of the rotations described respectively by ${}^{V_0}\mathbf{q}_V$ and $({}^V\phi, {}^V\theta)$, and g is the magnitude of the gravity vector.

We further motivate this choice of dynamics representation in Appendix B.

The onboard range sensor, whose principal axis is $\mathbf{x}_S$, is rigidly attached such that ${}^B\mathbf{p}_S$ and ${}^B\mathbf{R}_S$ are constant and known. Its FoV $\mathbb{F}$ is described by the halved vertical and horizontal angular apertures, respectively α_V and α_H, as well as a maximum distance $d_{\max}$. In practice, we set $d_{\max} = 5\,\mathrm{m}$ and match the maximum range handled by the VAE and SDF networks.

5.3 Local Navigation Neural MPC

5.3.1 Obstacle Avoidance Constraint We assume that the observation $\mathbf{o}$ is encoded into a latent representation $\mathbf{z}$ through the VAE (which is a convenience wording shortcut

referring to taking the mean of the latent distribution through the encoder part of the VAE). The neural approximation of the SDF defined in Section 4.1 defines the function

$$\begin{aligned} \mathbb{R}^3 &\to \mathbb{R} \\ \mathbf{p} &\mapsto \mathrm{SDF}_{\boldsymbol{\theta},\mathbf{z}}(\mathbf{p}) \,, \end{aligned} \tag{10}$$

which is parametrized by $\mathbf{z}$ and the MLP weights, denoted as $\boldsymbol{\theta}$. Ensuring that the motion of the AR remains collision-free is achieved by constraining the SDF value of positions of the open-loop trajectory w.r.t. the observation $\mathbf{o}$. Recalling that $\mathbf{o}$, acquired at $t = t_0$, describes a position constraint w.r.t. $\mathcal{F}_{S_0}$, we write the position of O_B in $\mathcal{F}_{S_0}$ as a closed-form function of the NMPC state and parametrized by the roll and pitch angles at time t_0, as

$${}^{S_0}\mathbf{p}_B = {}^{B}\mathbf{R}_S^\top \left({}^{V_0}\mathbf{R}_{B_0}^\top \, {}^{V_0}\mathbf{p}_B - {}^{B}\mathbf{p}_S \right). \tag{11}$$

Then, we impose on the NMPC the constraint

$$\mathrm{SDF}_{\boldsymbol{\theta},\mathbf{z}}({}^{S_0}\mathbf{p}_B) \geq r + \epsilon, \tag{12}$$

where $\epsilon > 0$ is a user-defined safety margin added to the robot-enclosing radius $r > 0$.

5.3.2 Field of View Constraints Because the collision-free set is defined in the sensor frustum $\mathbb{F}(t_0)$, the predicted motion must evolve within this volume. This results in a set of two additional constraints, respectively relative to the horizontal and vertical translations of the sensor O_S w.r.t. to $\mathcal{F}_{S_0}$.

We denote the transformations from Euclidean coordinates to the azimuth and elevation angles as $\mathcal{S}_{\text{azimuth}} \colon \mathbb{R}^3 \to \mathbb{R}$ and $\mathcal{S}_{\text{elevation}} \colon \mathbb{R}^3 \to \mathbb{R}$, respectively, which are computed as

$$\begin{aligned} \mathcal{S}_{\text{azimuth}}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) &= \operatorname{atan2}(y, x), \\ \mathcal{S}_{\text{elevation}}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) &= \operatorname{atan2}(z, \sqrt{x^2 + y^2}). \end{aligned} \tag{13}$$

The resulting constraints are then written as

$$\begin{aligned} -\alpha_H \leq \mathcal{S}_{\text{azimuth}}({}^{S_0}\mathbf{p}_S) \leq \alpha_H, \\ -\alpha_V \leq \mathcal{S}_{\text{elevation}}({}^{S_0}\mathbf{p}_S) \leq \alpha_V. \end{aligned} \tag{14}$$

We note that an additional maximum range constraint would be redundant, as it is already encoded in (12) that every position past $d_{\max}$ is unsafe.

5.3.3 Objective Function for Velocity Tracking We assume that a reference velocity and a reference heading, both expressed w.r.t. $\mathcal{F}_{V_0}$, are provided by any high-level planner, respectively denoted ${}^{V_0}\mathbf{v}_{\text{ref}}$ and ${}^{V_0}\psi_{\text{ref}}$.

Following Brescianini and D'Andrea (2020), we express the heading errors in quaternions. Let ${}^{V_0}\mathbf{q}_{\text{ref}}$ be the quaternion representation of ${}^{V_0}\psi_{\text{ref}}$. The orientation error $\mathbf{q}_e$ is given by

$$\mathbf{q}_e = \begin{bmatrix} q_{e,w} \\ 0 \\ 0 \\ q_{e,z} \end{bmatrix} = {}^{V_0}\mathbf{q}_{\text{ref}} \otimes \begin{bmatrix} {}^{V_0}q_w \\ 0 \\ 0 \\ {}^{V_0}q_z \end{bmatrix}^{-1}, \tag{15}$$

and minimizing the heading error is therefore equivalent to minimizing $q_{e,z}$.

Additionally, a control input objective penalizes the magnitude of the roll, pitch, and yaw rate, as well as the deviation from mg of the vertical projection of the thrust $T_z = T\cos({}^{V}\phi)\cos({}^{V}\theta)$.

The stage cost $\ell(\mathbf{x}, \mathbf{u})$ is thus written as

$$\ell(\mathbf{x}, \mathbf{u}) = \left\| \begin{bmatrix} q_{e,z} \\ {}^{V_0}\mathbf{v} - {}^{V_0}\mathbf{v}_{\text{ref}} \end{bmatrix} \right\|_{\mathbf{Q}}^2 + \left\| \begin{bmatrix} T_z - mg \\ {}^{V}\phi \\ {}^{V}\theta \\ {}^{B}\omega_z \end{bmatrix} \right\|_{\mathbf{R}}^2, \tag{16}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are tunable positive semidefinite and positive definite weight matrices, respectively, and $\|\bullet\|_{\mathbf{M}}^2$ is the weighted squared norm operator defined as

$$\|\mathbf{a}\|_{\mathbf{M}}^2 = \mathbf{a}^\top \mathbf{M} \mathbf{a}, \tag{17}$$

for an arbitrary vector $\mathbf{a}$ and diagonal weight matrix $\mathbf{M} \geq 0$.

5.4 Theoretical Analysis

In this study, we construct a terminal constraint to ensure recursive feasibility of the control law. Specifically, we enforce the terminal state to be such that there exists a sub-optimal terminal control policy that is recursively feasible under all constraints.

We then derive sufficient conditions on an appropriate terminal cost to ensure that the optimal cost becomes a non-increasing Lyapunov function under the sub-optimal terminal control policy. This step is then used to quantify a region of local stability that scales with the choice of the free parameters in the terminal cost.

For the remainder of the analysis, we consider the evolution of the system dynamics under a fixed observation. This means that we effectively neglect inaccuracies of the SDF approximation and the observation-dependent nature of $\mathcal{X}_{\text{free}}$ by assuming that $\mathcal{F}_{S_0}$ and $\mathrm{SDF}_{\boldsymbol{\theta},\mathbf{z}}$ are not time-varying. This partial approach does not directly prove the feasibility of the overall control problem under switching and noisy observations. However, it ensures that the control law never enters unsafe states under the current observation. Since the trajectory remains within the sensor frustum, this approach ensures that the (extended) open-loop trajectory is collision-free at all times.

The yaw dynamics are disregarded since they are decoupled from position dynamics for the considered AR.

5.4.1 Recursive Feasibility For all terminal states $\mathbf{x}_N$, we define a corresponding set $\mathcal{X}_N(\mathbf{x}_N)$ as the set of all states reached by recursively applying a "maximum braking to standstill" policy. This set is trivially forward invariant for such a policy and is bounded (both in velocity and position). To make the braking policy recursively feasible, it remains to ensure that it satisfies the input constraints, and that all states in $\mathcal{X}_N(\mathbf{x}_N)$ satisfy the collision avoidance constraints, i.e., $\forall \mathbf{x} \in \mathcal{X}_N(\mathbf{x}_N)$,

$$\mathrm{SDF}_{\boldsymbol{\theta},\mathbf{z}}({}^{S_0}\mathbf{p}) \geq r + \epsilon, \tag{18a}$$

$${}^{S_0}\mathbf{p} \in \mathbb{F}(t_0). \tag{18b}$$

The maximum braking policy, denoted $\pi_{\text{b}}(\mathbf{x})$, is defined as the solution to the constrained optimization problem

$$\{\pi_{\text{b}}(\mathbf{x}), \lambda_b\} = \underset{T,\phi,\theta,\lambda}{\operatorname{argmax}} \|\mathbf{a}_{\text{b}}(\mathbf{v})\| \tag{19a}$$

$$s.t. \quad \mathbf{a}_{\text{b}}(\mathbf{v}) = \frac{T}{m}{}^{V_0}\mathbf{R}_B\mathbf{z}_B - g\mathbf{z}_V \tag{19b}$$

$$\mathbf{a}_{\text{b}}(\mathbf{v}) = -\lambda\mathbf{v} \tag{19c}$$

$$\lambda \geq 0 \tag{19d}$$

$$0 \leq T \leq \overline{T} \tag{19e}$$

$$\underline{\phi} \leq \phi \leq \overline{\phi} \tag{19f}$$

$$\underline{\theta} \leq \theta \leq \overline{\theta} \tag{19g}$$

where $\mathbf{a}_{\text{b}}(\mathbf{v})$ is the deceleration opposite to the velocity vector, and Equations (19e), (19f) and (19g) are the input constraints imposed on the system. The policy ensures a deceleration along a straight line via Eq. (19c), and the system position thus evolves along a straight line toward the hovering equilibrium. Therefore, it holds that

$$\forall \mathbf{x} \in \mathcal{X}_N, \|\mathbf{p} - \mathbf{p}_N\| \leq d_{\text{b}}, \tag{20}$$

where d_{b} is the braking distance traveled while applying π_{b} given by

$$d_{\text{b}}(\mathbf{v}_N) = \frac{\mathbf{v}_N^2}{2\ \|\mathbf{a}_{\text{b}}(\mathbf{v}_N)\|}, \tag{21}$$

since the system is subject to a constant deceleration $\mathbf{a}_{\text{b}}(\mathbf{v}_N)$.

The condition (18a) is thus enforced by the terminal constraint

$$\text{SDF}_{\boldsymbol{\theta},\mathbf{z}}({}^{S_0}\mathbf{p}_{B,N}) - d_{\text{b}} \geq r + \epsilon. \tag{22}$$

Because $\mathbb{F}(t_0)$ is convex, the second condition (18b) is satisfied if

$${}^{S_0}\mathbf{p}_{S,N} \in \mathbb{F}(t_0), \quad {}^{S_0}\mathbf{p}_{S,E} \in \mathbb{F}(t_0), \tag{23}$$

where $\mathbf{p}_{S,E}$ is the position of O_S in the hovering equilibrium, obtained by translation of $d_{\text{b}}(\mathbf{v}_N)$ along the direction of $\mathbf{v}_N$. This constraint is enforced as in Eq. (14).

To include these two terminal constraints in the Optimal Control Problem, we compute a closed-form approximation of the braking distance d_{b}. We employ an i-*th* degree 3-variate polynomial, whose coefficients are obtained through a least-square fitting on the target values obtained through Equations (19) and (21). Table 2 reports the approximation errors for a given set of actuation constraints, evaluating polynomials of various degrees to provide insights on the expected accuracy. We note that directly approximating d_{b} is easier than approximating $\mathbf{a}_{\text{b}}$ since the latter is discontinuous in $\mathbf{v} = 0$, while Equation (21) can be trivially extended by continuity. Hereafter, we assume that an approximation of d_{b} of arbitrary precision is available.

5.4.2 Stability The current problem setting does not allow the establishment of asymptotic stability under arbitrary reference velocities due to the conflicting objectives in the stage cost (reference velocity tracking) and terminal cost and policy (coming to a stop). Further, the collision constraint may prevent the AR from converging to the desired reference velocity. An intuitive example of this is when a forward reference velocity is given while a wall is blocking the way; in this case, the constraint brings the system to a full stop. As a result, asymptotic stability (to a zero-cost equilibrium) cannot be established in general.

Degree of the fitted polynomial	3	4	5	6	7
# of params.	20	35	56	84	120
RMSE [cm]	4.0	2.6	2.4	1.7	1.6

Table 2. Fitting error, in centimeters, of various polynomial approximations of $d_{\text{b}}(\mathbf{v})$ as computed through Equations (19) and (21). The results are evaluated on the ball $\|\mathbf{v}\| \leq 3$, with a resolution of 0.05 (i.e., $\approx 0.9\,\text{M}$ points).

We instead derive sufficient conditions such that the optimal cost of the controller remains non-increasing. This is achieved by selecting an appropriate terminal cost $V(\mathbf{x}_N)$ ensuring that the optimal cost, denoted J^* and introduced in section 5.5, remains non-increasing under the sub-optimal terminal braking policy π_{b}. This enforces the existence of a feasible solution at the next time step that bounds J^*.

We emphasize that while stability is established under certain assumptions on the reference velocity $\mathbf{v}_{\text{ref}}$, the region in which the cost function is guaranteed to be non-increasing can be made arbitrarily large by the choice of the terminal cost parameter p. This is sufficient for the intended use case, as hard constraints are acting on the system state to ensure feasibility.

The Lyapunov stability condition on J^* is written

$$J^*(\mathbf{x}_{t+1}) - J^*(\mathbf{x}_t) \leq 0. \tag{24}$$

Expanding the two terms, this condition can be equivalently written as a condition on the terminal cost $V(\mathbf{x})$

$$V(\mathbf{x}_{N+1}) - V(\mathbf{x}_N) \leq -\ell(\mathbf{x}_N, \pi_{\text{b}}(\mathbf{x}_N)), \tag{25}$$

where $\ell(\mathbf{x}_N, \pi_{\text{b}}(\mathbf{x}_N))$ is the stage cost, evaluated in $\mathbf{x}_N$ while applying the braking policy (19).

We define the terminal cost as

$$V(\mathbf{x}) = p\ \mathbf{v}^\top\mathbf{v}, \tag{26}$$

where $p > 0$. Specific conditions on p ensuring that Eq. (24) holds are provided in Appendix C, including a narrow case when the system terminal velocity vanishes to zero (as well as a strategy to overcome this limitation by using a reference governor).

5.5 NonLinear Programming

The discrete-time NonLinear Programming (NLP) over the receding horizon T, sampled in N shooting points, at a given instant t, given a range image captured at $t_0 \leq t$ and compressed into a latent vector $\mathbf{z}$ via a network parametrized by $\boldsymbol{\theta}$, is expressed as

$$J^* \quad = \min_{\substack{\mathbf{x}_0 \dots \mathbf{x}_N \\ \mathbf{u}_0 \dots \mathbf{u}_{N-1}}} \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_N) \tag{27a}$$

$$
\begin{aligned}
s.t.\ \ & \mathbf{x}_0 = \mathbf{x}(t) && \text{(27b)}\\
& \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), && k\in\{0,N-1\} \quad \text{(27c)}\\
& \underline{\mathbf{v}} \le \mathbf{v}_k \le \overline{\mathbf{v}}, && k\in\{0,N\} \quad \text{(27d)}\\
& \underline{\mathbf{u}} \le \mathbf{u}_k \le \overline{\mathbf{u}}, && k\in\{0,N-1\} \quad \text{(27e)}\\
& r + \epsilon \le \mathrm{SDF}_{\boldsymbol{\theta},\mathbf{z}}({}^{S_0}\mathbf{p}_{B,k}), && k\in\{0,N-1\} \quad \text{(27f)}\\
& -\alpha_H \le \mathcal{S}_{\text{azimuth}}({}^{S_0}\mathbf{p}_{S,k}) \le \alpha_H, && k\in\{0,N\} \quad \text{(27g)}\\
& -\alpha_V \le \mathcal{S}_{\text{elevation}}({}^{S_0}\mathbf{p}_{S,k}) \le \alpha_V, && k\in\{0,N\} \quad \text{(27h)}\\
& r \le \mathrm{SDF}_{\boldsymbol{\theta},\mathbf{z}}({}^{S_0}\mathbf{p}_{B,N}) - d_{\mathrm{b}}(\mathbf{v}_N), && \text{(27i)}\\
& -\alpha_H \le \mathcal{S}_{\text{azimuth}}({}^{S_0}\mathbf{p}_{E}) \le \alpha_H, && \text{(27j)}\\
& -\alpha_V \le \mathcal{S}_{\text{elevation}}({}^{S_0}\mathbf{p}_{E}) \le \alpha_V, && \text{(27k)}
\end{aligned}
$$

where $\mathbf{x}(t)$ is the state estimate at t, and $\mathbf{f}$ synthetically denotes the discretized dynamics defined in Equation (9). We note that the NLP could be solved without the terminal constraint and cost related to recursive feasibility and stability.

5.6 Implementation Details

The above NLP is implemented in Python using Acados Verschueren et al. (2021) and Casadi Andersson et al. (2019). The neural network is implemented with PyTorch, and it is interfaced with the NMPC using L4Casadi Salzmann et al. (2023). The NLP is transformed into an Sequential QP solved with Interior Point Method Frison and Diehl (2020) with a Real-Time Iteration (RTI) scheme.

We use Levenberg–Marquardt (LM) regularization to improve the stability of computing sensitivities. We remark that this has a strong impact on the stability of the SQP solution under switching observations, and thus switching constraints. This is a consequence of both using a neural constraint parametrized with a large number of neurons, and of using the RTI suboptimal solving strategy, which relies on a reliable warm start from the previous solution. We empirically set the LM regularization factor to 10.

The FoV and neural constraints (27f), (27g), and (27h) are implemented as slackened constraints. This allows both to retain feasibility w.r.t. noisy feedback on the initial state $\mathbf{x}_0$, and w.r.t. the switching observations and the corresponding SDF approximation errors. We impose a $L1$-norm penalty with a weight of 20 on both slack variables for the FoV constraints, and both $L1$ and $L2$ penalties on the avoidance constraint, with respective weights 200 and 20. The large penalty on the neural constraint aims to compensate for the potentially large magnitude of the velocity tracking error when the environment imposes a full stop on the robot (e.g., in front of the wall with a large forward velocity reference).

The proposed method relies on only a few tunable hyperparameters. Those are, namely, the MPC weight (including those for the slack variables) which are tuned as for any other controller; the length T and sampling N of the horizon; the LM regularization factor; the safety margin ϵ; and the size of the SDF network, which is discussed in the next subsection. We note that the safety margin ϵ is chosen to be at least greater than the neural SDF RMSE, evaluated in Section 6.2. Table 3 presents the values for these parameters for each experimental subsection hereafter. The LM regularization factor and slack variables weights detailed in this section are not repeated, as they are constant in all the simulations and experiments.

Section	diag($\mathbf{Q}$)	diag($\mathbf{R}$)	T [s]	N	ϵ [m]
Section 6.3	$\begin{bmatrix}20\\5\\5\\5\end{bmatrix}$	$\begin{bmatrix}0.04\\50\\50\\5\end{bmatrix}$	1.5	20	0.1
Section 6.4	$\begin{bmatrix}20\\5\\5\\2\end{bmatrix}$	$\begin{bmatrix}0.04\\25\\25\\5\end{bmatrix}$	1.5	20	0.2
Section 6.5	$\begin{bmatrix}20\\5\\5\\2\end{bmatrix}$	$\begin{bmatrix}0.04\\25\\25\\5\end{bmatrix}$	1.5	20	0.2
Section 6.6	$\begin{bmatrix}20\\5\\5\\5\end{bmatrix}$	$\begin{bmatrix}0.04\\50\\50\\5\end{bmatrix}$	1.5	10	0.3

Table 3. SDF-NMPC parameters used in the experimental sections.

6 Validation

In this section, we evaluate the components of the proposed method. First, the encoding capabilities of the VAE and SDF networks are evaluated. Then, the neural controller is evaluated through randomized simulations to assess its sensitivity to the various parameters. The proposed method is then compared against two state-of-the-art mapless navigation methods. We perform an ablation study to quantify the resilience to odometry drift. Finally, the controller is integrated into a real system and validated in hardware experiments.

In order to properly study the properties of the proposed NMPC method, we avoid relying on a map-informed high-level planner generating the velocity reference provided to the NMPC. Instead, we consider a worst-case setting and implement a naive, obstacle-agnostic goal-seeking planner. It provides a reference ${}^{V_0}\mathbf{v}_{\text{ref}}$ along the straight line to reach the given goal position. The magnitude v_{ref} of ${}^{V_0}\mathbf{v}_{\text{ref}}$ is a user-defined parameter. In the laterally-constrained FoV case, the planner also provides a reference heading ${}^{V_0}\psi_{\text{ref}}$ such that the sensor aligns with the direction of motion.

6.1 VAE Evaluation

This subsection presents a qualitative and quantitative evaluation of the VAE reconstruction. We chose a latent space dimension of 128, which offers a good trade-off between quality and compression for the same input image size (480×270 pixels) Kulkarni et al. (2023). We compare the proposed distance-weighted (biased) VAE, introduced in Section 4.3, with a standard (unbiased) VAE. The tuning parameter is fixed at $w = 0.01$. We also include a baseline compression method based on the Fast Fourier Transform (FFT), which retains the 64 complex frequencies (i.e., 128 real values) with the largest magnitudes.

Quantitative results on the test dataset are summarized in Table 4. These results show that the biased VAE shifts reconstruction error toward the background, yielding better reconstruction in the foreground i.e., for the obstacles of interest for navigation. Overall, it outperforms both the unbiased VAE and the FFT-based reconstruction. Reconstruction quality is generally lower for LiDAR images compared to depth images, since the broader FoV perceives more objects and the resulting image has a higher spatial

Sensor	Pixels considered	FFT	Vanilla VAE	Biased VAE
Depth camera	full image	0.358	**0.182**	0.313
	non-background	0.562	0.489	**0.342**
LiDAR	full image	0.640	**0.375**	0.681
	non-background	0.850	0.745	**0.551**

Table 4. Pixel-wise RMSE [m] using a FFT, a standard VAE, and the proposed biased VAE, on the LiDAR and Depth Camera datasets, both on the full image and on the non-background pixels.

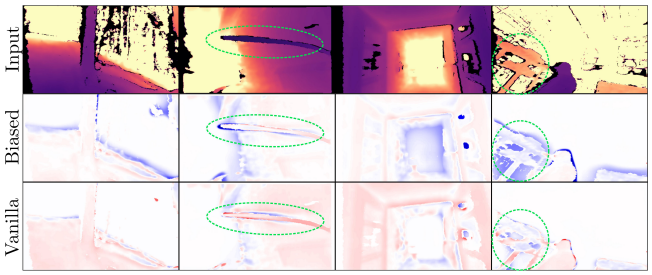

Figure 5. Reconstruction error using the proposed biased VAE and a vanilla VAE. The blue pixels correspond to reconstructions "closer" than the actual pixel value. The green circles highlight instances of thin obstacles whose reconstructions are improved.

frequency. Thus, more information is passed through the same latent bandwidth.

The biased VAE tends to reconstruct pixels closer to the camera. This is illustrated in selected instances in Figure 5. This results in a conservative approximation of the obstacles, in particular w.r.t. thin or small obstacles. This (even partial) reconstruction of an obstacle indicates that it is encoded in the latent representation, and therefore can be captured by the SDF network.

To assess generalization, we compute the reconstruction error on depth images from the TartanAir dataset Wang et al. (2020). A total of $30,000$ randomly sampled images are evaluated. These are cropped at $d_{\max} = 5\,\mathrm{m}$, and images with little to no nearby content (within $d_{\max}$) are excluded to ensure a representative sample.

The resulting pixel-wise RMSE is $0.225\,\mathrm{m}$, which is consistent with the error magnitudes reported in Table 4 for our testing depth images. Specifically, this value is slightly lower as the TartanAir environments typically feature lower spatial frequency than the randomized training scenes.

The average inference time of the VAE encoder, respectively on CPU (Intel Core i7-12800H) and GPU (NVIDIA GeForce RTX 3080 Ti Laptop), is $84.03\,\mathrm{ms}$ and $1.27\,\mathrm{ms}$. This highlights the importance of GPU acceleration for real-time deployment.

6.2 SDF Reconstruction

We now evaluate the SDF-approximating MLP.

6.2.1 Neural Network Size We assess how the reconstruction errors vary with the number of neurons. Network size is indeed a critical parameter, as it directly affects both the solving time of the NMPC, and the quality of the obstacle avoidance (through the accuracy of the environment encoding).

Abbrev.	SDF_{64}	SDF_{128}	$\mathbf{SDF_{256-64}}$	SDF_{256}	$\mathrm{SDF}_{512-256}$	$\mathbf{SDF_{512}}$
Layer sizes	[64 64 64 64]	[128 128 128 128]	[256 256 128 64]	[256 256 256 256]	[512 512 256 256]	[512 512 512 512]
# of params	41217	115201	246401	361473	869889	1247233
Query time	0.53	0.66	0.69	0.73	0.89	1.06

Table 5. Details of the 6 evaluated network sizes, with their naming abbreviations. The average query time [ms] for the SDF value and the corresponding gradient on a single query point is evaluated on an Intel Core i7-12800H CPU. The two networks highlighted in bold are further evaluated for closed-loop performances in Section 6.3.

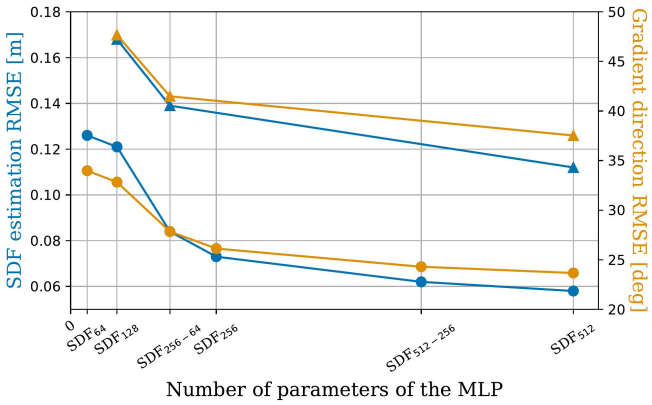

Figure 6. RMSE of the SDF estimation (blue) and of its gradient orientation (orange), for the depth (circles) and LiDAR (triangles) images, as a function of the neural network size.

Specifically, we evaluate 6 sets of layer sizes, and report the corresponding SDF reconstruction errors (and gradient direction errors) in Figure 6. The evaluated network sizes, detailed in Table 5, are named after the following convention: SDF_N denotes a network where all four layers have N neurons, and SDF_{N-M} denotes a network with decreasing layer sizes from N in the first layer to M in the last. We chose as the smallest architecture SDF_{64} the one used in Jacquet and Alexis (2024), which has been shown successful in encoding an occupancy map, i.e., a different (simpler) spatial representation. The network size SDF_{256} is similar to the one used in Ortiz et al. (2022) for online learning of a weight-encoded SDF. Metrics are computed on a 3D grid with a resolution of $10\,\mathrm{cm}$ within the sensor frustum. The evaluation is performed solely on simulated data, as the ground truth cannot be obtained for images that contain invalid pixels. It can be observed that the RMSE is higher in the LiDAR case, which is consistent with the VAE results in Table 4.

In order to provide visual insights on the neural 3D field, some instances of 2D SDF reconstruction are depicted in Figure 7 for the slices $z_B = 0$. It illustrates the convergence of the network-predicted SDF $(r + \epsilon)$-level set (blue) toward the ground truth (cyan) as network size increases. Notably, the SDF_{128} architecture exhibits significant penetration into obstacle, rendering it unsuitable for navigation tasks. Indeed,

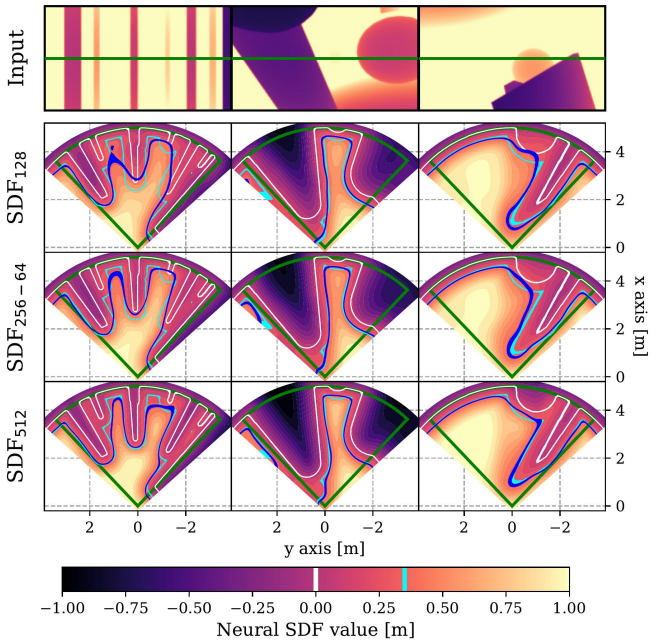

Figure 7. 2D slice of the estimated SDF at $z = 0$ (shown as the green line). The green cones depict the FoV. The color gradient shows the neural SDF, and the white curve marks the visible obstacle surface. The blue and cyan curves are the neural $(r + \epsilon)$-level sets, used as constraint, and its ground truth, respectively.

Input size	k-d tree		GP		neural SDF	
	constr. (CPU)	query (CPU)	constr. (GPU)	query (CPU)	VAE (GPU)	query (CPU)
129600	41.0	0.5	-	-	1.3	0.7
14400	4.1	0.2	587.3	2.4	-	-
1296	0.3	0.1	38.7	2.3	-	-
190	0.1	0.1	5.5	1.9	-	-

Table 6. Computation times [ms] for the construction and query (of the value and gradient) for different SDF approximation methods applied to depth images, with varying input dimensions.

Figure 6 shows a strong error decrease between SDF_{128} and SDF_{256-64}. We select SDF_{256-64} as the default architecture for the remainder of the paper, as it provides a practical trade-off between computational efficiency and reconstruction accuracy. For comparison, we also evaluate SDF_{512} to explore how further accuracy improvements translate to closed-loop navigation performance.

6.2.2 Comparison with Other SDF Approximations To contextualize the performance of our method relative to existing SDF encoding approaches, we compare computation times in Table 6. Specifically, we benchmark against a GP-based method Wu et al. (2023), implemented using the GPU-accelerated GPyTorch library Gardner et al. (2018). As an additional baseline, we include a k-d tree implementation. Although it does not provide an analytical SDF and thus cannot be used in our framework, it serves as a useful point of reference for raw query performance.

While these methods compute a different SDF representation, preventing a direct one-to-one comparison, we include the reconstruction errors reported in Wu et al. (2023) and Ortiz et al. (2022) for reference alongside the values

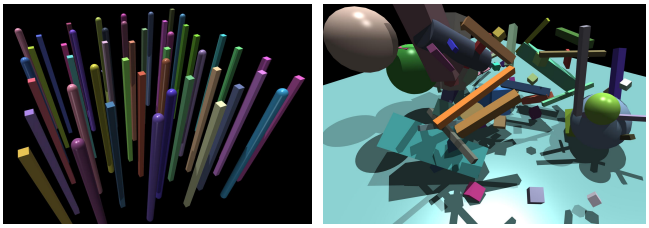

Figure 8. Instances of the two classes of environments used in the ablation study.

reported in Figure 7. The former reports an instance of scene RMSE of 7.7 cm, and the latter reports errors between 3 and 7 cm across different scenes. These values are comparable in scale to the RMSE obtained with our neural SDF (albeit generally lower). The gradient error reported in Ortiz et al. (2022) is also comparable, lying between 25° and 30°.

6.2.3 Generalizability We further evaluate the SDF reconstruction error on depth images from the TartanAir dataset Wang et al. (2020). Using the same 30,000 images as in Section 6.1, we compute the reconstruction error with the $\text{SDF}_{256\text{-}64}$ network, on a 3D grid with a resolution of 10 cm. The recorded RMSE is 9.1 cm (+8%) for the SDF estimation, and the gradient direction errors is 31° (+10%), both remaining within the same range as those reported in Figure 6 and in related works.

6.3 Ablation Study

In this section, we evaluate the sensitivity of performances w.r.t. the various hyperparameters. Although NMPC is typically paired with a low-level feedback controller for added robustness against model mismatches, we isolate our analysis by modifying the NMPC to output body wrench commands directly. We perform the simulations in the Aerial Gym simulator Kulkarni et al. (2025), which supports customizable physics stepping frequencies, randomized obstacle generation, and parallel rollouts. Control is executed at 50 Hz, while the physics engine runs at 500 Hz.

The simulated AR weights 1.25 kg and has a radius $r = 25$ cm. We fix the safety margin $\epsilon = 10$ cm. Each rollout samples a new obstacle configuration within a $10 \times 10 \times 5$ m environment, as well as the start and goal locations on opposite sides. Rollouts terminate colliding with an obstacle, or timing out after 30 s, if the goal is not reached, indicating that the robot is stuck in a dead-end. This behavior is expected in highly cluttered scenarios, as the planner only has access to local information.

Performance metrics include success, failure, and timeout rates, as well as other metrics indicating the navigation performance: average speed, average minimum neural SDF evaluation during the rollout, the true SDF values based on range observations. The robot consistently reaches its target speed for a significant portion of time; specifically, we verify that the average 90-*th* percentile speed remains above $0.95 v_{\text{ref}}$.

We consider two classes of environments, pictured in Figure 8. First, the obstacles are vertical pillars of circular or square sections, with diameters (or diagonals) ranging from 0.2 to 0.4m. These are sampled using Poisson discs, with a tunable minimum inter-obstacle distance d_{min}. This setup creates an effectively 2D navigation problem, enabling easier

	Altered parameters	Success Rate	Timeout Rate	Failure Rate	Avg. Velocity	Avg. min. neural SDF	Avg. min. SDF
Pillars	*Baseline*	1.0	0	0	1.75	0.339	0.446
	01) sensor frequency 2Hz	1.0	0	0	1.74	0.359	0.482
	02) higher state noise ($\sigma = 0.05$)	0.985	0	0.015	1.70	0.378	0.473
	03) higher state noise ($\sigma = 0.07$)	0.96	0	0.04	1.59	0.508	0.441
	04) $v_{\text{ref}} = 3\text{m/s}$	0.93	0	0.07	2.34	0.368	0.464
	05) $v_{\text{ref}} = 3\text{m/s}$, SDF_{512}	0.97	0	0.03	2.34	0.368	0.464
	06) $\alpha_H = 65^\circ$	0.995	0	0.005	1.63	0.236	0.376
	07) $\alpha_H = 65^\circ$, $v_{\text{ref}} = 3\text{m/s}$	0.95	0	0.05	2.1	0.288	0.413
	08) $\alpha_H = 65^\circ$, $v_{\text{ref}} = 3\text{m/s}$, SDF_{512}	0.99	0	0.01	2.18	0.333	0.392
	09) $\alpha_H = 180^\circ$	0.99	0	0.01	1.48	0.303	0.350
	10) $\alpha_H = 180^\circ$, $v_{\text{ref}} = 3\text{m/s}$	0.945	0	0.055	2.18	0.321	0.361
	11) $\alpha_H = 180^\circ$, $v_{\text{ref}} = 3\text{m/s}$, SDF_{512}	0.99	0	0.01	2.05	0.297	0.354
	12) $d_{\text{min}} = 0.80\text{m}$	0.92	0	0.08	1.53	0.304	0.419
	13) $d_{\text{min}} = 0.75\text{m}$	0.77	0.03	0.2	1.41	0.304	0.419
	14) $d_{\text{min}} = 0.80\text{m}$, $\alpha_H = 65^\circ$	0.97	0	0.03	1.35	0.236	0.376
	15) $d_{\text{min}} = 0.75\text{m}$, SDF_{512}	0.95	0	0.05	1.64	0.352	0.406
	16) $d_{\text{min}} = 0.75\text{m}$, $\alpha_H = 65^\circ$	0.88	0.05	0.07	1.23	0.220	0.359
	17) $d_{\text{min}} = 0.75\text{m}$, $\alpha_H = 65^\circ$, SDF_{512}	1.0	0	0	1.47	0.272	0.343
	18) $d_{\text{min}} = 0.75\text{m}$, $\alpha_H = 180^\circ$	0.215	0.78	0.05	0.83	0.258	0.305
	19) $d_{\text{min}} = 0.75\text{m}$, $\alpha_H = 180^\circ$, SDF_{512}	1.0	0	0	1.42	0.277	0.295
	20) $d_{\text{min}} = 0.75\text{m}$, $v_{\text{ref}} = 1\text{m/s}$	0.94	0	0.06	1.09	0.421	0.374
Random	*Baseline*	0.88	0.065	0.055	1.72	0.288	0.345
	21) $v_{\text{ref}} = 3\text{m/s}$	0.88	0.02	0.1	2.43	0.419	0.465
	22) SDF_{512}	0.905	0.05	0.045	1.76	0.453	0.442
	23) $\alpha_H = 65^\circ$	0.925	0.03	0.045	1.71	0.298	0.352
	24) $\alpha_H = 65^\circ$, $v_{\text{ref}} = 1\text{m/s}$	0.885	0.095	0.02	1.13	0.207	0.252
	25) $\alpha_H = 65^\circ$, SDF_{512}	0.925	0.035	0.04	1.67	0.301	0.320
	26) $\alpha_H = 180^\circ$	0.95	0.05	0	1.70	0.241	0.310
	27) $\alpha_H = 180^\circ$, SDF_{512}	0.97	0.025	0.05	1.65	0.256	0.402
	28) $+20$ obstacles	0.87	0.05	0.08	1.70	0.454	0.450
	29) $+20$ thin obstacles	0.82	0.05	0.13	1.68	0.448	0.421
	30) $+20$ thin obstacles , SDF_{512}	0.88	0.03	0.09	1.72	0.435	0.420
	31) $+20$ thin obstacles , $\alpha_H = 65^\circ$	0.88	0.03	0.09	1.69	0.258	0.296
	32) $+20$ thin obstacles , $\alpha_H = 180^\circ$	0.9	0.03	0.07	1.56	0.240	0.355
	33) $+20$ thin obstacles , $\alpha_H = 180^\circ$, SDF_{512}	0.92	0.03	0.05	1.61	0.234	0.368

Table 7. Evaluation metrics in the two classes of environments in various simulation settings. Each line corresponds to metrics gathered over 200 randomized rollouts for a single set of parameters, enumerated and described in the first column.

qualitative interpretation. Second, we use 3D randomly sampled cuboids, spheres, pillars, and rods, with smallest dimensions as low as $0.2\,\text{m}$. We also assess the impact of thinner obstacles (with smallest dimensions $< 0.05\,\text{m}$), such as small cuboids and rods.

The baseline conditions are defined after the Intel D455 depth camera FoV (i.e., $\alpha_H = 45^\circ$) at $25\,\text{Hz}$, the SDF_{256-64} network and $v_{\text{ref}} = 2\,\text{m/s}$. Gaussian noise is applied on the state feedback provided to the NMPC and on the Gaussian noise is applied to both state feedback and sensor observations (with stds $\sigma = 0.03$ and $\sigma = 0.05$, respectively), and control wrench perturbations are also introduced. Rollout visualizations for both environment classes are shown in Extension 1. Results are reported in Table 7 and discussed below.

Pillar Environments First, we observe (lines 1-3) that the framework exhibits resilience to reduced sensor frequency and moderate state noise—maintaining failure rates under 4% even with noisy feedback. The parameters that critically affect the success metrics are the reference velocity, the obstacle density, and the SDF network size. At higher velocities (lines 4-5, 7-8, and 10-11), reduced reaction times increase sensitivity to SDF approximation errors. This is mitigated by using the SDF_{512} network, which enables sharper reconstructions (see Figure 7). In the simple baseline environment (lines 1-11), the FoV only marginally impacts the performance. However, increasing the obstacle density (lines 12-20) causes a drop in success (down to 20% failure rate, line 13). This is attributed to the slackened constraint (27g) which allows slight lateral drift and results in collisions in denser layouts. A larger FoV mitigates this issue. It can be noted that the baseline SDF network struggles to reconstruct properly the narrow gaps in dense environments, especially for large FoV (line 18). This leads the neural constraint to prevents motion entirely, causing timeouts despite no actual dead-ends being present (since we ensure $d_{\text{min}} > 2(r + \epsilon) = 0.7\,\text{m}$).

3D Random Clutter Simulations in 3D environments show similar trends, though baseline failure rates are non-zero (5.5%), reflecting the known difficulty of mapless methods in cluttered settings with limited FoV. Slack FoV constraints also contribute to lateral collisions in dense areas. In such unstructured environments, increasing the FoV dramatically improves the success rate (lines 23-27), approaching 100% success rate in the 360° FoV case (lines 26-27). The framework also scales favorably with increasing obstacle density (lines 28), but performance degrades in the presence of thin obstacles (lines 29-33) (from 8% to 13% failure rate). Thin obstacles are challenging to encode and reconstruct, especially with the smaller $\text{SDF}256 - 64$

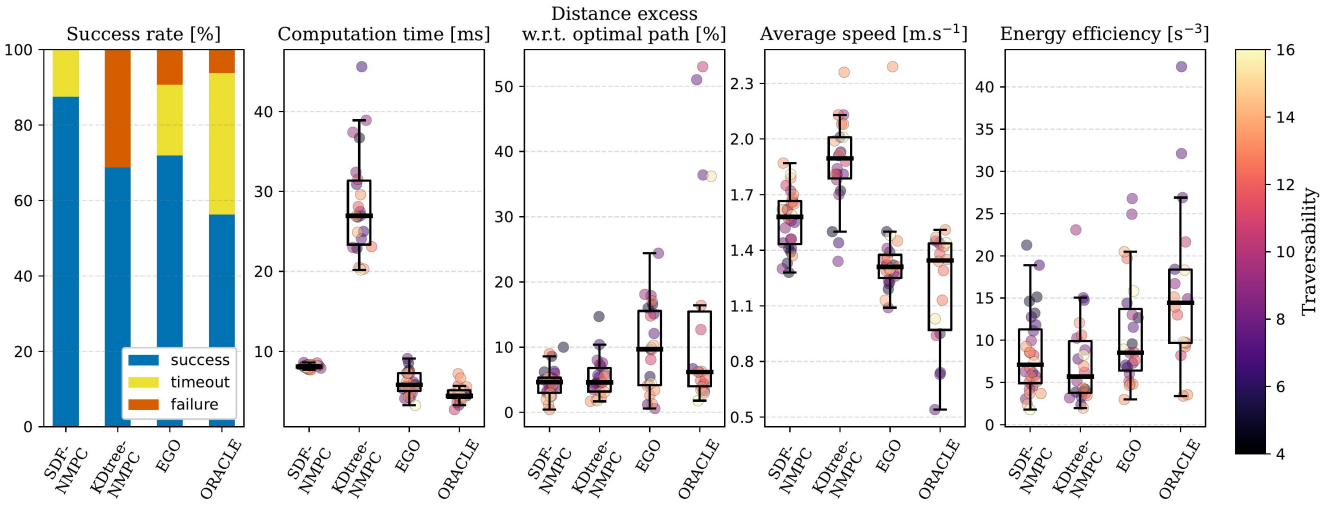

Figure 9. Comparative metrics for the 4 evaluated methods. The boxplots are overlayed with the individual samples for each of the successful rollouts, colored by traversability, as defined in Yu et al. (2023).

network. The larger SDF512 improves results (line 33), reducing the failure rate to 5%.

Across all simulations, the final columns of Table 7 indicate that the SDF network is generally conservative in its distance estimates. This suggests that most SDF constraint violations stem from discontinuities in the observations and poor estimations in the SDF network.

The average NMPC solving time on an Intel Core i7-12800H CPU is $8.51\,\mathrm{ms}$ using SDF_{256-64}, and $15.18\,\mathrm{ms}$ using SDF_{512}.

6.4 Comparison with Existing Methods

In this section, the proposed SDF-NMPC method is evaluated against three state-of-the-art collision avoidance approaches based on local depth observations, both neural and non-neural:

- the spline-based EGO-Planner from Zhou et al. (2021a) (and more specifically, the improved implementation EGO-Swarm Zhou et al. (2021b, 2022)),
- the neural, motion-primitive-based collision predictor ORACLE from Nguyen et al. (2024),
- and the k-d tree-based NMPC from Zhang et al. (2025).

A key distinction of our method is its formal consideration of safety as an explicit constraint in the optimization, rather than the heuristic collision score in ORACLE or the optimization co-objective in EGO and KDtree-NMPC.

The comparison is conducted using the Flightmare simulator Song et al. (2021) and benchmarking metrics from in Yu et al. (2023):

1. the path optimality, i.e. the ratio of excess traveled distance relative to an A* baseline,
2. the average speed, normalized by the optimal path length,
3. and the energy efficiency (i.e., the integral of the jerk).

All methods are evaluated in the same 32 randomly sampled environments, both indoor and outdoor Yu et al. (2023). The obstacles are sampled through Poisson disc distributions with radii chosen to cover a large range of traversability (≈ 4 to 16). The target navigation speed is $2\,\mathrm{m/s}$.

Baseline parameters are drawn from publicly available implementations, and fine-tuned for performance in the evaluation environments. Simulations are performed on a workstation equipped with an NVIDIA GeForce RTX 3090 GPU and an AMD Ryzen Threadripper 3970X CPU. A visualization of the simulations is included in Extension 1, and results are reported in Figure 9.

The proposed NMPC method (using SDF_{256-64}) outperforms the other methods in terms of success rate, achieving zero collisions. While being more computationally demanding than EGO and ORACLE, it maintains a control frequency above $100\,\mathrm{Hz}$ and is more efficient than the KDtree-NMPC. Notably, its computation time is more consistent around the average, whereas the baselines show increased computational load as traversability decreases.

In terms of path optimality, both our NMPC and KDtree-NMPC outperform EGO and ORACLE, benefiting from objectives of minimizing deviation from a straight path. Although KDtree-NMPC achieves higher speeds, our method still surpasses EGO and ORACLE in this regard, demonstrating strong tracking performance without compromising safety.

Finally, energy efficiency is comparable across most methods, except for ORACLE, whose switching acceleration and yawing rate outputs do not account for dynamic feasibility and smoothness.

6.5 Resilience to Drifting Odometry

Because the collision avoidance is formulated in the local frame, our framework is resilient to drifting odometry, which would hinder the construction of a reliable map. In this section, we present an ablation study evaluating the method's resilience to increasing odometry drift and decreasing sensor frequency. The latter is relevant since, without new measurements, the method relies on the forward-propagated last received measurement; thus, a lower frequency implies a higher sensitivity to drift.

In this study, we perform simulations in Gazebo, and deteriorate the state feedback provided to the NMPC.

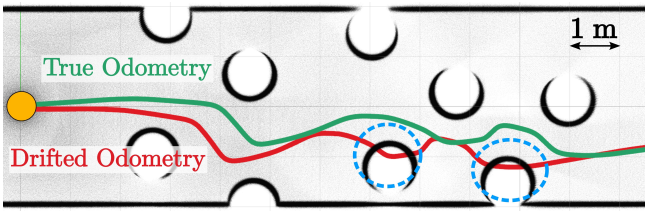

Figure 10. Top-view of a 2D trajectory where the NMPC receives drifting odometry (red). Despite the drift, the real system (green trajectory) avoids collisions, as the controller operates in a local observation frame independent of map-based positioning. The orange circle depicts the robot size.

		Sensor Frequency				
		100 Hz	30 Hz	10 Hz	1 Hz	0.5 Hz
RPE	0%	10	10	10	10	7
	∼ 5%	10	9	8	7	1
	∼ 10%	10	8	6	1	0
	∼ 15%	7	3	2	0	0

Table 8. Number of successes out of 10 rollouts with increasing RPE and decreasing sensor frequency. The reference velocity is 2 m/s.

Specifically, velocity and yaw rate are perturbed using Gaussian noise and a random-walk bias. Position and heading are then obtained through integration. The tilt angles are not altered, as those are non-drifting quantities directly observable using an IMU.

The parameters of these noises are empirically selected to achieve the desired Relative Position Error (RPE) Geiger et al. (2012) (computed for a delta of 10 m). This metric computes the position RMSE per delta of traveled distance. It offers a more meaningful measure of drift than absolute position error. For reference, the Visual-Inertial Odometry (VIO) methods ROVIO Bloesch et al. (2017) and VINS-Mono Qin et al. (2018) both report RPE of 1 to 2% for the same delta.

We perform these simulations in random, cluttered environments where spheres of 1 m radius are sampled using 3D Poisson Discs with a radius of 1.5 m. The environment is 50 m long, enclosed by walls, and the robot must reach a waypoint on the opposite end (with a reference speed of 2 m/s). This setup is intentionally challenging due to high clutter and a constrained FoV, making it a rigorous test of the method's resilience to drift. An illustrative, 2D example of such an environment is provided in Figure 10. We evaluate the success rate of the NMPC by performing rollouts in each scenario.

Table 8 reports the number of successes out of ten ten different environments in each RPE / sensor frequency setting. The first column and first row correspond to the limit cases (perfect odometry and 100 Hz sensor frequency – i.e., the control frequency). We remark that the success rate is not 10 out of 10 in the two extreme cases (top right and bottom left cells). This is due to a) the severely degraded velocity estimate in high RPE scenarios that leads to poor predictive performance (the NMPC formulation is not robust); and b) the uncertainties in the SDF in the very cluttered environments lead to collisions when the sensor rate becomes too low.

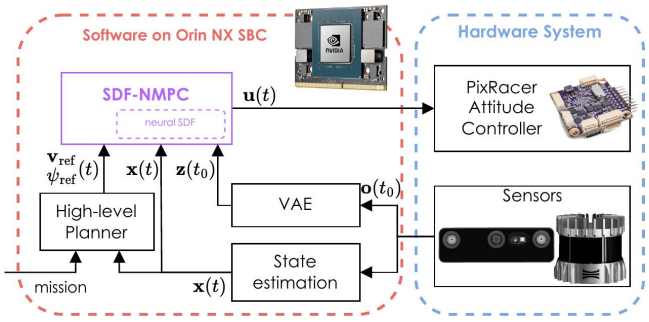

Figure 11. Block diagram on the SDF-NMPC framework. The contributed controller, which integrates the neural SDF network as a constraint, is highlighted in purple.

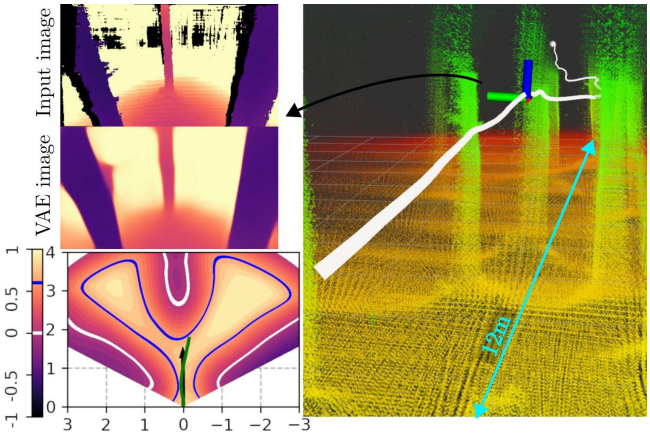

Figure 12. Third-person view of a trajectory (white path) among trees, showing the aggregated pointclouds from the depth camera. On the left is the input depth image at a given time instant, along with its VAE reconstruction, and a top-view of the $z_B = 0$ slice of the neural SDF, its 0- and $(r + \epsilon)$-levelsets (respectively white and blue), the reference velocity (black arrow), and the predicted trajectory (green line).

These results highlight the local aspect of the method, which achieves collision avoidance while relying on short-term, local consistency of the state estimate in the weakest sense without explicitly considering robust formulations (i.e., allows forward propagation in between two successive range measurements).

6.6 Real World Experiments

6.6.1 System Setup We finally present hardware experiments for both the depth camera and 360° LiDAR cases. Experiments are conducted with a custom-built RMF-class quadrotor De Petris et al. (2022), with dimensions $0.52 \times 0.52 \times 0.3$ m and mass 2.58 kg. The AR integrates PX4-based autopilot avionics for low-level control, together with an NVIDIA Orin NX Single-Board Computer (SBC) running ROS Noetic. The exteroceptive sensor used for navigation is either an Ouster OS0-64 LiDAR or a Luxonis OAK-D Pro Wide depth camera (whose halved horizontal FoV α_H is 63°). The sensor frequencies are set to 20 Hz. Further, the system employs a Texas Instruments IWR6843AOP FMCW radar sensor. Odometry is obtained by fusing a VectorNav VN-100 IMU with the LiDAR observations using Comp-SLAM Khattak et al. (2020), or with the radar measurements

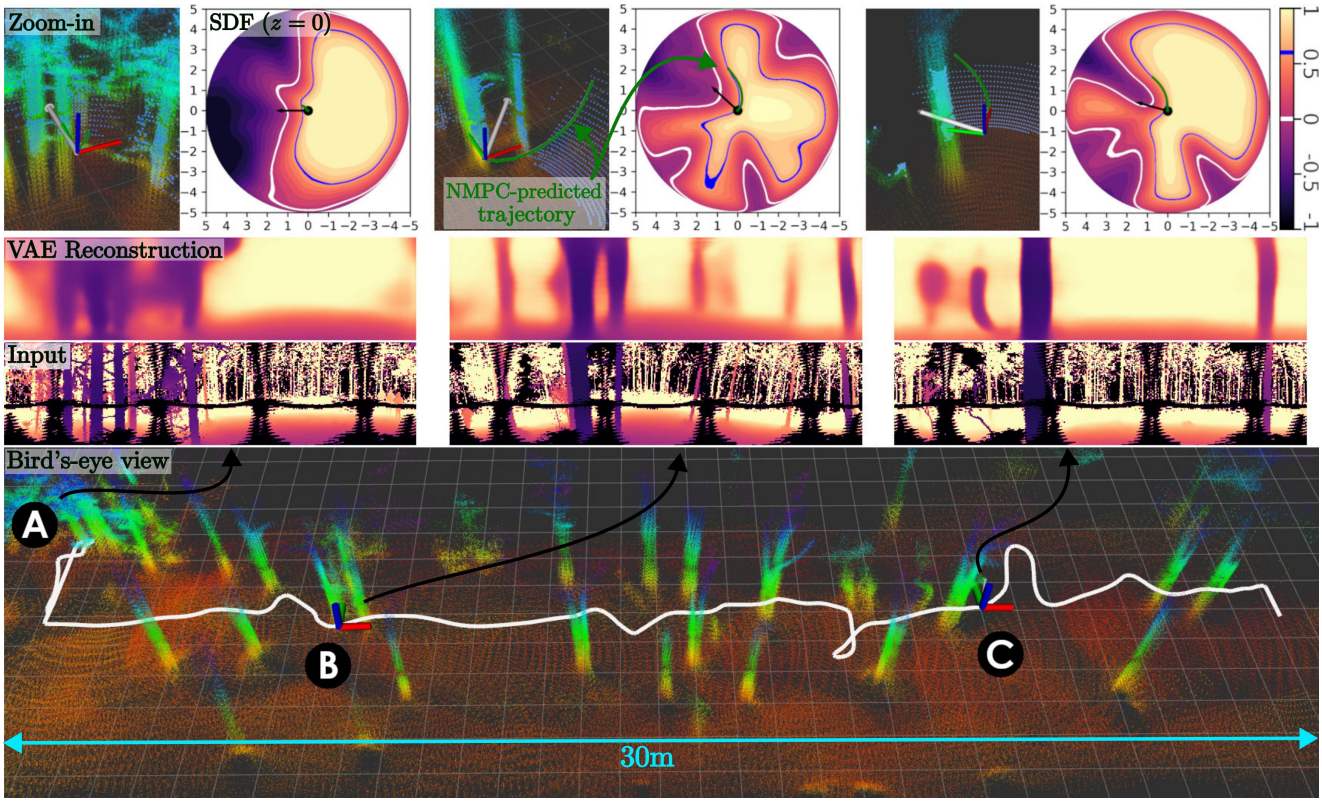

Figure 13. Bird's eye visualization of a LiDAR-based experiment among trees. The top-most row depicts three zoomed-in time instances A, B, and C, along with the $z_B = 0$ slice of the neural SDF, following the same nomenclature as Figure 12. The next two rows show the corresponding LiDAR input range image and its VAE reconstruction.

Nissov et al. (2024) in the drifting case. A block diagram of the system is depicted in Figure 11.

The VAE encoding is executed on the Orin NX GPU, with an average inference time of 12.4 ms. The neural NMPC solver (using SDF_{256-64}) runs on the CPU, with an average solving time of 15.4 ms. The resulting control frequency, including the reference velocity reference generation and the overhead induced by the Python implementation, is ≈ 40 Hz.

Recordings of each of the three following experiments are presented respectively in Extension 2, Extension 3, and Extension 4, along with relevant visualizations to appreciate the action of the NMPC.

6.6.2 Experiment with Depth Camera This first experiment is conducted using the depth camera for navigation, with LIDAR-inertial odometry. Figure 12 presents an overview of the experiment. The AR is tasked with reaching a waypoint 15 m ahead, through a 12 m-long tree-filled section. The reference speed is set to $v_{\text{ref}} = 1.5\,\text{m/s}$, and the observed 90th percentile velocity is 1.25 m/s. The resulting motion properly avoids the trees, despite the naive environment-agnostic reference velocity. The systematic noise in the input image (black pixels) is handled by the VAE. The encoded SDF shows a decrease toward the obstacles, and the constraint embedded in the NMPC (pictured as the blue line) becomes active, deflecting the trajectory to the side.

6.6.3 Experiment with LiDAR with Adversarial Reference Velocity This second experiment is conducted using the LiDAR for navigation and LiDAR-inertial odometry. Unlike the previous setup, the reference policy is not provided by the goal-seeking planner. Instead, an adversarial velocity input is provided by a human operator through a remote joystick, actively trying to collide the AR with the trees. Figure 13 pictures three time instances along the experiments which highlight that the NMPC effectively prevents collision by either deflecting the trajectory through some free space (instances B and C), or by bringing the robot to a full stop (instance A) when the trees form a wall-like obstacle, blocking the half-space where the velocity projection would yield a cost decrease. Additionally, Figure 13 also illustrates that the VAE is capable of successfully reconstructing the range image despite the significant amount of invalid pixels in the input image (both in the long-range regions and within four vertical clusters obscured by the propellers and the outer cage of the AR).

6.6.4 Experiment with Drifting Odometry This last experiment implements the drifting odometry case discussed in Section 6.5. The experimental settings are similar to the previous one, i.e., the LiDAR range image is used for maintaining collision avoidance in the presence of adversarial velocity reference. However, instead of relying on the accurate LiDAR-based odometry, we implement an Frequency Modulated Continuous Wave (FMCW) radar-based velocity estimator which provides, through integration, a drifting position and heading odometry. Indeed, while the LiDAR-based position estimate is accurate in well-structured environments (including forests), the radar measurements have several orders of magnitude fewer points with generally greater noise, thus resulting in scan-to-scan matching approaches being impractical. However, the availability of Doppler measurements enable reliable velocity estimation,

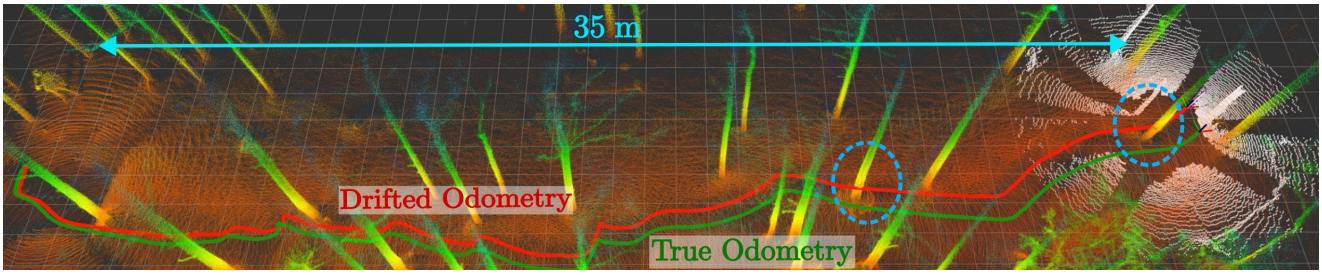

Figure 14. Bird's eye view of the trajectory with both odometries (drifted in red, ground truth in green). The white pointcloud depicts a single observation given to the NMPC. The controller is agnostic of the map, and operates in the local frame, and therefore maintains collision avoidance despite the drift.

though without position corrections, inevitably leading to drift over time.

Specifically, we implement the radar-inertial part of the factor-graph estimator presented in Nissov et al. (2024), whose details are reported in Appendix D. The odometry has an RPE of 3% and a heading RPE of $2°$ against the LiDAR odometry from Khattak et al. (2020), which we use as ground truth.

Figure 14 shows a bird's eye view of the trajectory. Both odometries (red for radar, green for ground truth) are reported, showing the progressive drift between the two. This experiment further verifies the associated results on drifting odometry previously presented in simulation within Section 6.5.

7 Conclusion

This work contributes an NMPC framework for collision avoidance using a neural SDF as a differentiable, volumetric approximation of exteroceptive data. This representation enables the optimal controller to enforce position constraints, thus enforcing collision avoidance. The neural architecture is divided into two parts, trained sequentially: a VAE which encodes the image in a low-dimensional latent space, and a 4-layer coordinate-based MLP which approximates the corresponding SDF. We further perform a theoretical analysis of the recursive feasibility and stability properties of the proposed NMPC, and derive corresponding terminal conditions. The neural SDF encoding is first evaluated, then the NMPC controller is evaluated through several ablation studies and comparisons. Furthermore, it is validated through outdoor experiments at $1.5\,\mathrm{m/s}$, demonstrating the avoidance of trees against adversarial inputs and drifting odometry.

This method results in a tightly integrated control and collision-avoidance policy that achieves competitive navigation performance with low computational cost. Additionally, the approach exhibits strong resilience – within identified thresholds – in scenarios where odometry estimation drifts. This addresses the primary motivation for adopting mapless navigation policies. Finally, our results demonstrate the effectiveness of neural SDF encoding as a foundation for mapless navigation. Future work includes seven key directions. First, improving the framework's implementation allows for reduced sensing-to-action delay. This could be achieved by optimizing neural network architecture and weights, and improving CPU-GPU memory transfer. Second, investigating energy-based approaches to ensure that the NMPC retains a dissipative behavior when the active collision constraints prevent matching the reference velocity. Third, extending the neural representation to consider the orientation would relax the spherical robot assumption and enhance maneuverability. Fourth, incorporating visual sensing alongside or instead of range data could mitigate sensor degradation and better capture fine features, with a bimodal VAE fusing sensor inputs. Fifth, relaxing the 0-memory assumption through temporal observation windows, key-frame selection, and uncertainty-aware filtering. Another avenue for this task is the use of neural world models, encoding a temporal sequence of image data into a latent representation. Sixth, relaxing the static world assumption by predicting obstacle motion over time would further benefit from world modeling. Finally, future work could focus on the uncertainty awareness, w.r.t. both the noisy state information and the coarsely approximated SDF from the (also noisy) observations, e.g. by extending the proposed method to an uncertainty-informed stochastic MPC, or using a robust formulation that incorporates error bounds.

Acknowledgements

We thank Morten Nissov for his help in the setup of the radar and the corresponding estimation software.

Author contributions

Statements and declarations

Ethical considerations

This article does not contain any studies with human or animal participants.

Consent to participate

Not applicable.

Consent for publication

Not applicable.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

This work was partially supported by the European Commission Horizon projects DIGIFOREST (EC 101070405) and SPEAR (EC 101119774).

References

Adhau S, Gros S and Skogestad S (2024) Reinforcement learning based MPC with neural dynamical models. *European Journal of Control* : 101048DOI:10.1016/j.ejcon.2024.101048.

Alhaddad M, Mironov K, Staroverov A and Panov A (2024) Neural potential field for obstacle-aware local motion planning. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 9313–9320. DOI:10.1109/ICRA57147.2024.10611635.

Alsmeier H, Savchenko A and Findeisen R (2024) Neural horizon model predictive control - increasing computational efficiency with neural networks. In: *2024 American Control Conference*. pp. 1646–1651. DOI:10.23919/ACC60939.2024.10644452.

Andersson JA, Gillis J, Horn G, Rawlings JB and Diehl M (2019) CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* 11(1): 1–36. DOI:10.1007/s12532-018-0139-4.

Azinović D, Martin-Brualla R, Goldman DB, Nießner M and Thies J (2022) Neural RGB-D surface reconstruction. In: *2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 6290–6301. DOI:10.1109/cvpr52688.2022.00619.

Barron JT, Mildenhall B, Verbin D, Srinivasan PP and Hedman P (2022) Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In: *2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 5470–5479. DOI:10.1109/cvpr52688.2022.00539.

Bauersfeld L, Kaufmann E, Foehn P, Sun S and Scaramuzza D (2021) NeuroBEM: Hybrid aerodynamic quadrotor model. In: *Robotics: Science and Systems*. p. 42. DOI:10.15607/RSS.2021.XVII.042.

Bloesch M, Burri M, Omari S, Hutter M and Siegwart R (2017) Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The Int. Journal of Robotics Research* 36(10): 1053–1072. DOI:10.1177/0278364917728574.

Brescianini D and D'Andrea R (2020) Tilt-prioritized quadrocopter attitude control. *IEEE Trans. on Control Systems Technology* 28(2): 376–387. DOI:10.1109/TCST.2018.2873224.

Bucki N, Lee J and Mueller MW (2020) Rectangular pyramid partitioning using integrated depth sensors (RAPPIDS): A fast planner for multicopter navigation. *IEEE Robotics and Automation Letters* 5(3): 4626–4633. DOI:10.1109/LRA.2020.3003277.

Cao TT, Tang K, Mohamed A and Tan TS (2010) Parallel banding algorithm to compute exact distance transform with the GPU. In: *2010 ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*. pp. 83–90. DOI:10.1145/1730804.1730818.

Celestini D, Gammelli D, Guffanti T, D'Amico S, Capello E and Pavone M (2024) Transformer-based model predictive control: Trajectory optimization via sequence modeling. *IEEE Robotics and Automation Letters* 9(11): 9820–9827. DOI:10.1109/LRA.2024.3466069.

Chee KY, Jiahao TZ and Hsieh MA (2022) KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots. *IEEE Robotics and Automation Letters* 7(2): 2819–2826. DOI:10.1109/LRA.2022.3144787.

Chen Z and Zhang H (2019) Learning implicit fields for generative shape modeling. In: *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 5939–5948. DOI:10.1109/cvpr.2019.00609.

Criminisi A, Sharp T and Blake A (2008) GeoS: Geodesic image segmentation. In: *2008 European Conf. on Computer Vision*. pp. 99–112. DOI:10.1007/978-3-540-88682-2_9.

Dai A, Ruizhongtai Qi C and Niessner M (2017) Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In: *2017 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 5868–5877. DOI:10.1109/cvpr.2017.693.

Dawson C, Lowenkamp B, Goff D and Fan C (2022) Learning safe, generalizable perception-based hybrid control with certificates. *IEEE Robotics and Automation Letters* 7(2): 1904–1911. DOI:10.1109/LRA.2022.3141657.

De Petris P, Nguyen H, Dharmadhikari M, Kulkarni M, Khedekar N, Mascarich F and Alexis K (2022) RMF-Owl: A collision-tolerant flying robot for autonomous subterranean exploration. In: *2022 Int. Conf. on Unmanned Aircraft Systems*. pp. 536–543. DOI:10.1109/ICUAS54217.2022.9836115.

Dellaert F and GTSAM Contributors (2022) Georgia tech smoothing and mapping library. URL https://github.com/borglab/gtsam.

East S, Gallieri M, Masci J, Koutnik J and Cannon M (2020) Infinite-horizon differentiable model predictive control. In: *2020 Int. Conf. on Learning Representations*. URL https://openreview.net/forum?id=ryxC6kSYPr.

Ebadi K, Bernreiter L, Biggie H, Catt G, Chang Y, Chatterjee A, Denniston CE, Deschênes SP, Harlow K, Khattak S, Nogueira L, Palieri M, Petráček P, Petrlík M, Reinke A, Krátký V, Zhao S, Agha-mohammadi Aa, Alexis K, Heckman C, Khosoussi K, Kottege N, Morrell B, Hutter M, Pauling F, Pomerleau F, Saska M, Scherer S, Siegwart R, Williams JL and Carlone L (2024) Present and future of SLAM in extreme environments: The DARPA subt challenge. *IEEE Trans. on Robotics* 40: 936–959. DOI:10.1109/TRO.2023.3323938.

Fang Z, Yang S, Jain S, Dubey G, Roth S, Maeta S, Nuske S, Zhang Y and Scherer S (2017) Robust autonomous flight in constrained and visually degraded shipboard environments. *JOURNAL of Field Robotics* 34(1): 25–52. DOI:10.1002/rob.21670.

Florence P, Carter J and Tedrake R (2020) Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In: *Algorithmic Foundations of Robotics XII*. pp. 304–319. DOI:10.1007/978-3-030-43089-4_20.

Florence P, Carter J, Ware J and Tedrake R (2018) Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. In: *2018 IEEE Int. Conf. on Robotics and Automation*. pp. 7631–7638. DOI:10.1109/ICRA.2018.8463195.

Forster C, Carlone L, Dellaert F and Scaramuzza D (2017) On-manifold preintegration for real-time visual–inertial odometry. *IEEE Trans. on Robotics* 33(1): 1–21. DOI:10.1109/TRO.2016.2597321.

Frison G and Diehl M (2020) HPIPM: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine* 53(2): 6563–6569. DOI:10.1016/j.ifacol.2020.12.073.

Furrer F, Burri M, Achtelik M and Siegwart R (2016) *RotorS—A Modular Gazebo MAV Simulator Framework*. pp. 595–625. DOI:10.1007/978-3-319-26054-9_23.

Gao F, Wu W, Gao W and Shen S (2019) Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *JOURNAL of Field Robotics* 36(4): 710–733. DOI:10.1002/rob.21842.

Gao Z, Wen W, Xing Y and Tsourdos A (2024) An integrated framework for autonomous driving planning and tracking based on nnmpc considering road surface variations. *IEEE Trans. on Intelligent Vehicles* DOI:10.1109/TIV.2024.3418951.

Gardner J, Pleiss G, Weinberger KQ, Bindel D and Wilson AG (2018) GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In: *2018 Conf. on Neural Information Processing Systems*, volume 31.

Geiger A, Lenz P and Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: *2012 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 3354–3361. DOI:10.1109/CVPR.2012.6248074.

Gros S and Zanon M (2020) Data-driven economic NMPC using reinforcement learning. *IEEE Trans. on Automatic Control* 65(2): 636–648.

Han L, Gao F, Zhou B and Shen S (2019) FIESTA: Fast incremental euclidean distance fields for online motion planning of aerial robots. In: *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 4423–4430. DOI:10.1109/IROS40897.2019.8968199.

Harms M, Kulkarni M, Khedekar N, Jacquet M and Alexis K (2024) Neural control barrier functions for safe navigation. In: *2024 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. URL https://arxiv.org/abs/2407.19907.

Hewing L, Wabersich KP, Menner M and Zeilinger MN (2020) Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems* 3(1): 269–296. DOI:10.1146/annurev-control-090419-075625.

Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S and Lerchner A (2017) β-VAE: Learning basic visual concepts with a constrained variational framework. In: *2017 Int. Conf. on Learning Representations*.

Hoeller D, Wellhausen L, Farshidian F and Hutter M (2021) Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robotics and Automation Letters* 6(3): 5081–5088. DOI:10.1109/LRA.2021.3068639.

Jacquet M and Alexis K (2024) N-MPC for deep neural network-based collision avoidance exploiting depth images. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 13536–13542. DOI:10.1109/ICRA57147.2024.10610572.

Jang J, Lee I, Kim M and Joo K (2024) AiSDF: Structure-aware neural signed distance fields in indoor scenes. *IEEE Robotics and Automation Letters* 9(5): 4106–4113. DOI:10.1109/LRA.2024.3375117.

Kabzan J, Hewing L, Liniger A and Zeilinger MN (2019) Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters* 4(4): 3363–3370. DOI:10.1109/LRA.2019.2926677.

Kahn G, Abbeel P and Levine S (2021a) BADGR: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters* 6(2): 1312–1319. DOI:10.1109/LRA.2021.3057023.

Kahn G, Abbeel P and Levine S (2021b) LaND: Learning to navigate from disengagements. *IEEE Robotics and Automation Letters* 6(2): 1872–1879. DOI:10.1109/LRA.2021.3060404.

Kamel MS, Stastny T, Alexis K and Siegwart R (2017) *Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System*, volume 707. pp. 3–39. DOI:10.1007/978-3-319-54927-9_1.

Keyumarsi S, Atman MWS and Gusrialdi A (2024) Lidar-based online control barrier function synthesis for safe navigation in unknown environments. *IEEE Robotics and Automation Letters* 9(2): 1043–1050. DOI:10.1109/LRA.2023.3339059.

Khattak S, Nguyen H, Mascarich F, Dang T and Alexis K (2020) Complementary multi–modal sensor fusion for resilient robot pose estimation in subterranean environments. In: *2020 Int. Conf. on Unmanned Aircraft Systems*. pp. 1024–1029. DOI:10.1109/ICUAS48674.2020.9213865.

Kulkarni M and Alexis K (2023) Task-driven compression for collision encoding based on depth images. In: *2023 Int. Symp. on Visual Computing*. DOI:10.1007/978-3-031-47966-3_20.

Kulkarni M and Alexis K (2024) Reinforcement learning for collision-free flight exploiting deep collision encoding. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 15781–15788. DOI:10.1109/ICRA57147.2024.10610287.

Kulkarni M, Nguyen H and Alexis K (2023) Semantically-enhanced deep collision prediction for autonomous navigation using aerial robots. In: *2023 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 3056–3063. DOI:10.1109/ACCESS.2022.3154037.

Kulkarni M, Rehberg W and Alexis K (2025) Aerial gym simulator: A framework for highly parallelized simulation of aerial robots. *IEEE Robotics and Automation Letters* DOI:10.1109/LRA.2025.3548507.

Lee MH and Liu JS (2024) Reliable and accurate implicit neural representation of multiple swept volumes with application to safe human–robot interaction. *SN Computer Science* 5(3): 313. DOI:10.1007/s42979-024-02640-8.

Lee T, Leok M and McClamroch NH (2010) Geometric tracking control of a quadrotor uav on se(3). In: *IEEE Conf. on Decision and Control*. pp. 5420–5425. DOI:10.1109/CDC.2010.5717652.

Liang J, Gao P, Xiao X, Sathyamoorthy AJ, Elnoor M, Lin MC and Manocha D (2024) MTG: Mapless trajectory generator with traversability coverage for outdoor navigation. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 2396–2402. DOI:10.1109/ICRA57147.2024.10611319.

Long K, Qian C, Cortés J and Atanasov N (2021) Learning barrier functions with memory for robust safe navigation. *IEEE Robotics and Automation Letters* 6(3): 4931–4938. DOI:10.1109/LRA.2021.3070250.

Lopez BT and How JP (2017) Aggressive 3-D collision avoidance for high-speed navigation. In: *2017 IEEE Int. Conf. on Robotics and Automation*. pp. 5759–5765. DOI:10.1109/ICRA.2017.7989677.

Loquercio A, Kaufmann E, Ranftl R, Müller M, Koltun V and Scaramuzza D (2021) Learning high-speed flight in the wild. *Science Robotics* 6(59). DOI:10.1126/scirobotics.abg5810.

Lu J, Tian B, Shen H, Zhang X and Hui Y (2023) LPNet: A reaction-based local planner for autonomous collision avoidance using imitation learning. *IEEE Robotics and Automation Letters* 8(11): 7058–7065. DOI:10.1109/LRA.

2023.3314350.

Macklin M (2022) Warp: A high-performance python framework for GPU simulation and graphics. URL https://github.com/nvidia/warp. NVIDIA GPU Technology Conf.

Marić A, Li Y and Calinon S (2024) Online learning of continuous signed distance fields using piecewise polynomials. *IEEE Robotics and Automation Letters* 9(6): 6020–6026. DOI: 10.1109/LRA.2024.3397085.

Matthies L, Brockers R, Kuwata Y and Weiss S (2014) Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In: *2014 IEEE Int. Conf. on Robotics and Automation*. pp. 3242–3249. DOI:10.1109/ICRA.2014.6907325.

Maurer CR, Qi R and Raghavan V (2003) A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. on Pattern Analysis & Machine Intelligence* 25(2): 265–270. DOI:10.1109/TPAMI.2003.1177156.

Mescheder L, Oechsle M, Niemeyer M, Nowozin S and Geiger A (2019) Occupancy networks: Learning 3d reconstruction in function space. In: *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 4460–4470. DOI:10.1109/CVPR.2019.00459.

Mosbach M and Behnke S (2022) Efficient representations of object geometry for reinforcement learning of interactive grasping policies. In: *IEEE Int. Conf. on Robotic Computing*. pp. 156–163. DOI:10.1109/IRC55401.2022.00034.

Newcombe RA, Izadi S, Hilliges O, Molyneaux D, Kim D, Davison AJ, Kohi P, Shotton J, Hodges S and Fitzgibbon A (2011) Kinectfusion: Real-time dense surface mapping and tracking. In: *IEEE Int. Symp. on Mixed and Augmented Reality*. pp. 127–136. DOI:10.1109/ISMAR.2011.6092378.

Nguyen H, Andersen R, Boukas E and Alexis K (2024) Uncertainty-aware visually-attentive navigation using deep neural networks. *The Int. Journal of Robotics Research* 43(6): 840–872. DOI:10.1177/02783649231218720.

Nguyen H, Fyhn SH, De Petris P and Alexis K (2022) Motion primitives-based navigation planning using deep collision prediction. In: *2022 IEEE Int. Conf. on Robotics and Automation*. pp. 9660–9667. DOI:10.1109/ICRA46639.2022.9812231.

Nissov M, Khedekar N and Alexis K (2024) Degradation resilient lidar-radar-inertial odometry. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 8587–8594. DOI:10.1109/ICRA57147.2024.10611444.

Oleynikova H, Taylor Z, Fehr M, Siegwart R and Nieto J (2017) Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning. In: *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 1366–1373. DOI:10.1109/IROS.2017.8202315.

Ortiz J, Clegg A, Dong J, Sucar E, Novotny D, Zollhoefer M and Mukadam M (2022) iSDF: Real-time neural signed distance fields for robot perception. In: *Robotics: Science and Systems*, volume 30. p. 39. DOI:10.15607/rss.2022.xviii.012.

Pan Y, Zhong X, Wiesmann L, Posewsky T, Behley J and Stachniss C (2024) PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency. *IEEE Trans. on Robotics* 40: 4045–4064. DOI: 10.1109/TRO.2024.3422055.

Park JJ, Florence P, Straub J, Newcombe R and Lovegrove S (2019) DeepSDF: Learning continuous signed distance functions for shape representation. In: *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 165–174. DOI:10.1109/cvpr.2019.00025.

Qin T, Li P and Shen S (2018) VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. on Robotics* 34(4): 1004–1020. DOI:10.1109/TRO.2018.2853729.

Reiter R, Ghezzi A, Baumgärtner K, Hoffmann J, McAllister RD and Diehl M (2024) AC4MPC: Actor-critic reinforcement learning for nonlinear model predictive control. URL https://arxiv.org/abs/2406.03995.

Romero A, Song Y and Scaramuzza D (2024) Actor-critic model predictive control. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 14777–14784. DOI:10.1109/ICRA57147.2024.10610381.

Salzmann T, Arrizabalaga J, Andersson J, Pavone M and Ryll M (2024) Learning for CasADi: Data-driven models in numerical optimization. In: *2024 Learning for Dynamics and Control Conf.* pp. 541–553. URL https://proceedings.mlr.press/v242/salzmann24a.html.

Salzmann T, Kaufmann E, Arrizabalaga J, Pavone M, Scaramuzza D and Ryll M (2023) Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters* 8(4): 2397–2404. DOI:10.1109/LRA.2023.3246839.

Siegwart R, Nourbakhsh IR and Scaramuzza D (2011) *Introduction to autonomous mobile robots*. MIT press. ISBN 9780262295321.

Sitzmann V, Martel J, Bergman A, Lindell D and Wetzstein G (2020) Implicit neural representations with periodic activation functions. In: *2020 Conf. on Neural Information Processing Systems*. pp. 7462–7473. URL https://arxiv.org/abs/2006.09661.

Song Y, Naji S, Kaufmann E, Loquercio A and Scaramuzza D (2021) Flightmare: A flexible quadrotor simulator. In: *2020 Conf. on Robot Learning*. pp. 1147–1157. URL https://proceedings.mlr.press/v155/song21a.html.

Song Y and Scaramuzza D (2022) Policy search for model predictive control with application to agile drone flight. *IEEE Trans. on Robotics* 38(4): 2114–2130. DOI:10.1109/TRO.2022.3141602.

Song Y, Shi K, Penicka R and Scaramuzza D (2023) Learning perception-aware agile flight in cluttered environments. In: *2023 IEEE Int. Conf. on Robotics and Automation*. pp. 1989–1995. DOI:10.1109/ICRA48891.2023.10160563.

Stutz D and Geiger A (2018) Learning 3D shape completion from laser scan data with weak supervision. In: *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 1955–1964. DOI:10.1109/cvpr.2018.00209.

Sucan IA, Moll M and Kavraki LE (2012) The open motion planning library. *IEEE Robotics & Automation Magazine* 19(4): 72–82. DOI:10.1109/MRA.2012.2205651.

Sucar E, Liu S, Ortiz J and Davison AJ (2021) iMAP: Implicit mapping and positioning in real-time. In: *2021 IEEE/CVF Int. Conf. on Computer Vision*. pp. 6229–6238. DOI:10.1109/iccv48922.2021.00617.

Sun S, Romero A, Foehn P, Kaufmann E and Scaramuzza D (2022) A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight. *IEEE Trans. on Robotics* 10: 3357–3373. DOI:10.1109/TRO.2022.3177279.

Syntakas S and Vlachos K (2024) Dropout MPC: An ensemble neural MPC approach for systems with learned dynamics. URL https://arxiv.org/abs/2406.02497.

Tancik M, Srinivasan P, Mildenhall B, Fridovich-Keil S, Raghavan N, Singhal U, Ramamoorthi R, Barron J and Ng R (2020) Fourier features let networks learn high frequency functions in low dimensional domains : 7537–7547.

Tian Y, Liu K, Ok K, Tran L, Allen D, Roy N and How JP (2020) Search and rescue under the forest canopy using multiple UAVs. *The Int. Journal of Robotics Research* 39(10-11): 1201–1221. DOI:10.1177/0278364920929398.

Tolani V, Bansal S, Faust A and Tomlin C (2021) Visual navigation among humans with optimal control as a supervisor. *IEEE Robotics and Automation Letters* 6(2): 2288–2295. DOI:10.1109/LRA.2021.3060638.

Tong M, Dawson C and Fan C (2023) Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields. In: *2023 IEEE Int. Conf. on Robotics and Automation*. pp. 10511–10517. DOI:10.1109/ICRA48891.2023.10161482.

Torrente G, Kaufmann E, Föhn P and Scaramuzza D (2021) Data-driven MPC for quadrotors. *IEEE Robotics and Automation Letters* 6(2): 3769–3776. DOI:10.1109/LRA.2021.3061307.

Tranzatto M, Miki T, Dharmadhikari M, Bernreiter L, Kulkarni M, Mascarich F, Andersson O, Khattak S, Hutter M, Siegwart R and Alexis K (2022) Cerberus in the darpa subterranean challenge. *Science Robotics* 7(66). DOI:10.1126/scirobotics.abp9742.

Ugurlu HI, Pham XH and Kayacan E (2022) Sim-to-real deep reinforcement learning for safe end-to-end planning of aerial robots. *Robotics* 11(5): 109. DOI:10.3390/robotics11050109.

Vasilopoulos V, Garg S, Huh J, Lee B and Isler V (2024) HIO-SDF: Hierarchical incremental online signed distance fields. In: *2024 IEEE Int. Conf. on Robotics and Automation*. pp. 17537–17543. DOI:10.1109/ICRA57147.2024.10610367.

Verschueren R, Frison G, Kouzoupis D, Frey J, van Duijkeren N, Zanelli A, Novoselnik B, Albin T, Quirynen R and Diehl M (2021) acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation* 14: 147–183. DOI:10.1007/s12532-021-00208-8.

Wang H, Wang J and Agapito L (2023) Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In: *2023 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 13293–13302. DOI:10.1109/cvpr52729.2023.01277.

Wang J, Bleja T and Agapito L (2022) Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In: *2022 Int. Conf. on 3D Vision*. pp. 433–442. DOI:10.1109/3DV57658.2022.00055.

Wang P, Liu L, Liu Y, Theobalt C, Komura T and Wang W (2021) NeuS: learning neural implicit surfaces by volume rendering for multi-view reconstruction. In: *2021 Conf. on Neural Information Processing Systems*. pp. 27171–27183. URL https://arxiv.org/abs/2106.10689.

Wang W, Zhu D, Wang X, Hu Y, Qiu Y, Wang C, Hu Y, Kapoor A and Scherer S (2020) TartanAir: A dataset to push the limits of visual SLAM. In: *2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 4909–4916. DOI:10.1109/IROS45743.2020.9341801.

Williams G, Wagener N, Goldfain B, Drews P, Rehg JM, Boots B and Theodorou EA (2017) Information theoretic MPC for model-based reinforcement learning. pp. 1714–1721. DOI:10.1109/ICRA.2017.7989202.

Wu L, Lee KMB, Le Gentil C and Vidal-Calleja T (2023) Log-GPIS-MOP: A unified representation for mapping, odometry, and planning. *IEEE Trans. on Robotics* 39(5): 4078–4094. DOI:10.1109/TRO.2023.3296982.

Yadav I and Tanner HG (2020) Reactive receding horizon planning and control for quadrotors with limited on-board sensing. In: *2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 7058–7063. DOI:10.1109/IROS45743.2020.9341306.

Yan Z, Tian Y, Shi X, Guo P, Wang P and Zha H (2021) Continual neural mapping: Learning an implicit scene representation from sequential observations. In: *2021 IEEE/CVF Int. Conf. on Computer Vision*. pp. 15782–15792. DOI:10.1109/iccv48922.2021.01549.

Yu H, de Croon GCHE and De Wagter C (2023) AvoidBench: A high-fidelity vision-based obstacle avoidance benchmarking suite for multi-rotors. In: *2023 IEEE Int. Conf. on Robotics and Automation*. pp. 9183–9189. DOI:10.1109/ICRA48891.2023.10161097.

Zhang L, Hu Y, Deng Y, Yu F and Zou D (2025) Mapless collision-free flight via MPC using dual KD-Trees in cluttered environments. URL https://arxiv.org/abs/2503.10141.

Zhou X, Wang Z, Ye H, Xu C and Gao F (2021a) EGO-planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters* 6(2): 478–485. DOI:10.1109/LRA.2020.3047728.

Zhou X, Wen X, Wang Z, Gao Y, Li H, Wang Q, Yang T, Lu H, Cao Y, Xu C and Gao F (2022) Swarm of micro flying robots in the wild. *Science Robotics* 7(66). DOI:10.1126/scirobotics.abm5954.

Zhou X, Zhu J, Zhou H, Xu C and Gao F (2021b) EGO-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. In: *2021 IEEE Int. Conf. on Robotics and Automation*. pp. 4101–4107. DOI:10.1109/ICRA48506.2021.9561902.

Zhu Z, Peng S, Larsson V, Xu W, Bao H, Cui Z, Oswald MR and Pollefeys M (2022) NICE-SLAM: Neural implicit scalable encoding for SLAM. In: *2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. pp. 12786–12796. DOI:10.1109/cvpr52688.2022.01245.

A Training Signals

A.1 VAE Training

This section presents the training losses for the VAE networks used in the paper. The networks are trained with a latent dimension of 128, as recalled in Section 6.1. Figures 15 report the MSE and KLD losses for both depth camera and LiDAR VAEs, using the datasets presented in Table 1.

Figure 15 illustrates that the depth camera VAE converges to a lower MSE value than the LiDAR VAE. This correlates with the results presented in Section 6.1 and is explained

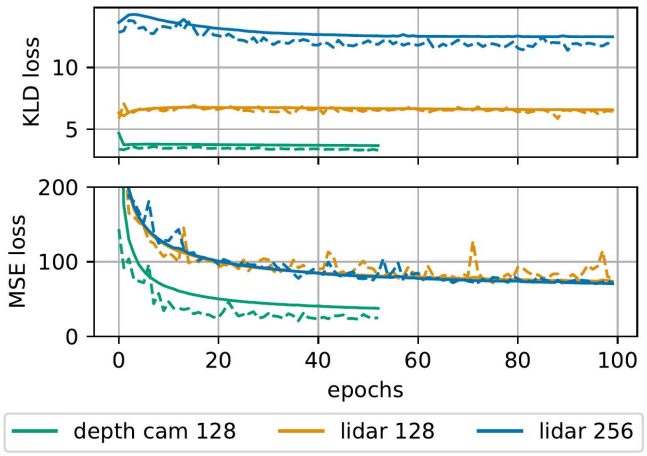

Figure 15. Training (solid lines) and validation (dashed lines) losses for 2 different VAEs input image types (LiDAR and Depth Camera). In the LiDAR case, we report the training signals for 2 different latent sizes.

by the larger spatial frequency of the signal to encode. The figure also reports the training loss for the LiDAR VAE trained with a latent dimension of 256. The results indicate that a latent dimension larger than 128 brings minimal improvements to the reconstruction, and correlates with the results reported in Kulkarni and Alexis (2023).

We note that the KLD is higher for the 256-latent training as the β weight is scaled as a function of the latent size Higgins et al. (2017).

The training duration for 100 epochs is $\sim 9\,\mathrm{h}$ on an NVIDIA GeForce RTX 3090 GPU, using the full dataset from Table 1.

A.2 SDF MLP Training

The section presents the training losses for the various SDF networks evaluated in Section 6.2. Figure 16 reports the two training losses, i.e., the SDF MSE as well as the SDF gradient MSE.

Additionally, we report therein (as dashed lines) the training losses using the gradient direction and eikonal losses, as presented in, e.g., Ortiz et al. (2022).

The *direction+eikonal* training presents a faster convergence rate of the training signals in the early epochs compared to the *MSE* training. However, both trainings converge to similar values.

In both cases, the eikonal loss remains relatively high, in particular with smaller networks. This is due to the complex topology of the learned SDF network, which has several angular points at which the gradient is discontinuous. Because the neural approximation is smooth, the eikonal loss remains quite high in those regions.

The training duration for 40 epochs is $\sim 12\,\mathrm{h}$ on an NVIDIA GeForce RTX 3090 GPU. We note that the main bottleneck in terms of training time is the computation of the target SDF values for the sampled points, which is largely impacted by the chosen resolution of the grid as presented in Section 4.1. Solutions to speed up the training include pre-computing the SDF values on pre-sampled data points, and improving the distance transform approximation algorithm by avoiding redundant computation.

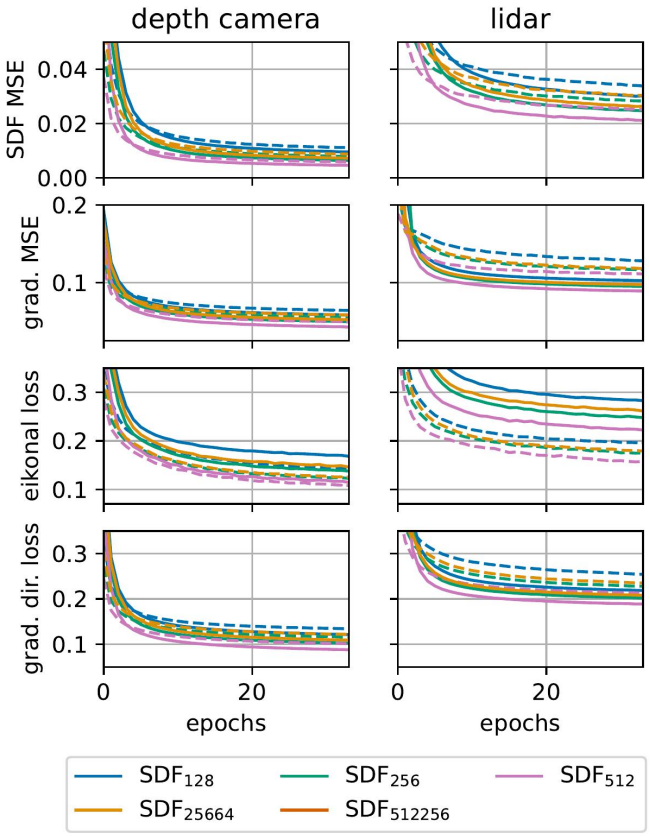

Figure 16. Color-coded validation losses and metric for the 6 evaluated neural network sizes in Section 6.2. The solid lines depict the signals for the training scheme proposed in Section 4.3.2. Dashed lines report training signals for optimization on the eikonal loss and the gradient direction loss.

B Choice of Dynamic Model

In order to evaluate the impact of different system dynamics representations in the NMPC, we conduct a comparative study using various levels of simplification for the multirotor dynamics. Specifically, we assess:

- a double integrator (2^{nd} order linear system);
- the corresponding 2^{nd} nonlinear representation with thrust/attitude mapping to the acceleration (used throughout this paper);
- a model extended with a simplified, 1^{st} order response for the roll and pitch (as commonly done in NMPC for quadrotors, see, e.g., Kamel et al. (2017));
- and a 4^{th} order dynamics representation, detailed, e.g., in Lee et al. (2010).

For fair comparison, the four NMPC are tuned for good tracking of a lemniscate trajectory with high accelerations, without any perception constraints.

We design trajectories aiming to excite the higher order dynamics of the system with higher reference speed ($3\,\mathrm{m/s}$) and sharp turns in cluttered environments, similar to those presented in Section 6.5. Results, summarized in Table 9, indicate only minor improvements in success rate as model complexity increases. This suggests that performance is more constrained by perception and the collision avoidance system than by the limits of the dynamics representation. However, we observe that more accurate dynamics models result in smoother trajectories with fewer acceleration spikes.

	2nd order linear	2nd order nonlinear	2nd order nonlinear with 1st order att.	4th order
Success	17/20	17/20	18/20	18/20
Path excess [%]	7.9	6.2	5.8	4.9
Avg. velocity	2.27	2.3	2.28	2.26
Jerk integral	21.25	18.51	17.03	15.84
Comp. time [ms]	10.78	11.08	11.29	11.73

Table 9. Evaluation metrics for different dynamics models with increasing accuracy.

Finally, as expected, we observe that the computation time increases by 9% with the model complexity. While this augmentation is non-negligible, it does not change the order of magnitude of the optimization complexity, confirming numerical tractability with all evaluated models.

This study highlights that, in the considered scenarios, the coarseness of the dynamics model is not a limiting factor. We support this claim through two main considerations. First, the perception constraints prevent aggressive tilting (to ensure that the trajectory remains within the FoV) and large velocities (because of the maximum range in the SDF, which enforces the predicted positions to remain within close range). As a result, our framework effectively restricts the operational domain of the quadrotor dynamics, justifying the relevance of our approximation. Second, as in many perception-driven tasks, the collision-avoidance system itself remains the primary performance bottleneck. Failures to prevent collisions are more likely to result from its limitations than from the inaccuracies of the dynamics model.

Therefore, we argue that the model presented in Section 5.2 provides a relevant description of the dynamics for navigation without agile maneuvering. Finally, we note that our framework can accommodate any dynamics representation – linear or nonlinear – to suit the specific use case.

C Lyapunov Stability

This section derives the condition on the terminal cost defined in Eq. (26) such that the Lyapunov stability condition (24) is satisfied.

We recall that the terminal cost is defined as

$$V(\mathbf{x}) = p\,\mathbf{v}^\top \mathbf{v}, \tag{28}$$

where $p > 0$, and the Lyapunov stability is verified if

$$V(\mathbf{x}_{N+1}) - V(\mathbf{x}_N) \leq -\ell(\mathbf{x}_N, \pi_\mathrm{b}(\mathbf{x}_N)), \tag{29}$$

where $\ell(\mathbf{x}_N, \pi_\mathrm{b}(\mathbf{x}_N))$ is the stage cost, evaluated in $\mathbf{x}_N$ applying the braking policy (19).

Writing the discrete-time control policy for (19) and accounting for the edge case causing overshoot when getting close to standstill $\mathbf{v} = 0$, the policy becomes:

$$\pi_\mathrm{b}(\mathbf{x}) = \begin{cases} \zeta(-\mathbf{a}_\mathrm{b}(\mathbf{v})) & \text{if } \|\mathbf{v}\| > \|\mathbf{a}_\mathrm{b}(\mathbf{v})\|\,\delta t \\ \zeta(-\frac{\mathbf{v}}{\delta t}) & \text{otherwise} \end{cases}, \tag{30}$$

where ζ denotes the nonlinear mapping from the acceleration to the thrust and attitude input (9c), and δt is the discretization time step.

The left-hand side in Eq. (29) is therefore written

$$\begin{aligned} V(\mathbf{x}_{N+1}) - V(\mathbf{x}_N) &= p(\|\mathbf{v}_{N+1}\|^2 - \|\mathbf{v}_N\|^2) \\ &= -p\Delta v(2\,\|\mathbf{v}_N\| - \Delta v) \end{aligned}, \tag{31}$$

where $\Delta v = \|\mathbf{v}_N\| - \|\mathbf{v}_{N+1}\|$ is the absolute velocity decrease induced by applying $\pi_\mathrm{b}(\mathbf{x}_N)$. By construction of this maximum braking policy, a constant deceleration is applied such that the velocity decrease can be expressed by forward Euler integration over δt. Thus, by Eq. (30) it holds

$$\Delta v = \begin{cases} \|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t & \text{if } \|\mathbf{v}_N\| > \|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t \\ \|\mathbf{v}_N\| & \text{otherwise} \end{cases}. \tag{32}$$

Hence, Eq. (31) becomes

$$V(\mathbf{x}_{N+1}) - V(\mathbf{x}_N) = \begin{cases} -p\,\|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t(2\,\|\mathbf{v}_N\| - \|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t) & \text{if } \|\mathbf{v}_N\| > \|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t \\ -p\,\|\mathbf{v}_N\|^2 & \text{if } \|\mathbf{v}_N\| \leq \|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t \end{cases} \tag{33}$$

We consider each case separately.

First case: $\|\mathbf{v}_N\| > \|\mathbf{a}_b(\mathbf{v}_N)\|\,\delta t$ Considering the first case, inserting the inequality $\|\mathbf{v}_N\| > \|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|\,\delta t$ in Condition (29) and reformulating yields

$$p\,\|\mathbf{a}_\mathrm{b}(\mathbf{v}_N)\|^2\,\delta t^2 \geq \ell(\mathbf{x}_N, \pi_\mathrm{b}(\mathbf{x}_N)). \tag{34}$$

The left-hand side of the inequality is upper-bounded since $\|\mathbf{a}_\mathrm{b}(\mathbf{v})\|$ is bounded due to input constraints. Further, ℓ is quadratic in $\|\mathbf{v_N}\|$, Eq. (34) can not be verified globally with any finite p. Instead, we find p such that the inequality holds for all $\mathbf{x}_N$ within a predefined area of operation since arbitrarily large velocities are not intended states for autonomous navigation. Specifically, we want the inequality (34) to hold for $\|\mathbf{v}_N\| \leq \overline{v}$. Assuming that the reference velocity is also bounded, an upper bound on the stage cost $\overline{\ell}$ can be expressed.

Similarly, a lower bound on $\|\mathbf{a}_\mathrm{b}(\mathbf{v})\|$, denoted $\underline{\mathbf{a}_\mathrm{b}} > 0$, can be obtained as

$$\begin{aligned} \underline{\mathbf{a}_\mathrm{b}} = \min_{\mathbf{v}}\ & \|\mathbf{a}_\mathrm{b}(\mathbf{v})\| \\ s.t.\quad & \|\mathbf{v}\| \leq \overline{v} \end{aligned} \tag{35}$$

Rewriting Eq. (34), the corresponding condition on p is thus given by

$$p \geq \frac{\overline{\ell}}{\underline{\mathbf{a}_\mathrm{b}}^2 \delta t^2}. \tag{36}$$

Second case: $\|\mathbf{v}_N\| \leq \|\mathbf{a}_b(\mathbf{v})\|\,\delta t$ The second case in Eq. (33) corresponds to the specific case when the velocity approaches the discontinuity in π_b for $\mathbf{v} = 0$.

In this case, Condition (29) becomes

$$p\,\|\mathbf{v}_N\|^2 \geq \|\mathbf{Q}\|_2\,\|\mathbf{v}_N - \mathbf{v}_\mathrm{ref}\|^2 + \|\pi_\mathrm{b}(\mathbf{x}_N)\|^2_\mathbf{R}. \tag{37}$$

For a nonzero $\mathbf{v}_\mathrm{ref}$, the above inequality cannot hold since the left-hand side vanishes for $\mathbf{v}_N$ approaching zero,

while the term $\|\mathbf{v}_N - \mathbf{v}_{\text{ref}}\|^2$ does not. In this case, the competing objectives in the terminal cost (reduce velocity to zero) and stage cost (track $\mathbf{v}_{\text{ref}}$) do not allow for showing asymptotic stability without further assumptions. In this case, we assume the presence of a reference governor, which modifies $\mathbf{v}_{\text{ref}}$ in a suitable manner to retain stability. This can easily be achieved by setting $\mathbf{v}_{\text{ref}} = 0$ for all $t_i \geq t_N$ in the case when $\|\mathbf{v}_N\| \leq \|\mathbf{a}_{\text{b}}(\mathbf{v})\|\,\delta t$. This simple modification of future $\mathbf{v}_{\text{ref}}$ essentially results in a reduction of the reference velocity beyond the prediction horizon, whenever $\|\mathbf{v}_N\| \leq \|\mathbf{a}_{\text{b}}(\mathbf{v})\|\,\delta t$.

Inserting $\mathbf{v}_{\text{ref}} = 0$ into (37) results in

$$p\,\|\mathbf{v}_N\|^2 \geq \|\mathbf{Q}\|_2\,\|\mathbf{v}_N\|^2 + \|\pi_{\text{b}}(\mathbf{x}_N)\|^2_{\mathbf{R}}\,. \tag{38}$$

Inserting the Cauchy-Schwarz inequality, an upper bound of the stage cost can be found as

$$\ell(\mathbf{x}_N, \pi_{\text{b}}(\mathbf{x}_N)) \leq \|\mathbf{Q}\|\,\|\mathbf{v}_N\|^2 + \|\pi_{\text{b}}(\mathbf{x}_N)\|^2_{\mathbf{R}}\,, \tag{39}$$

where $\|\mathbf{Q}\|$ is the 2-norm of the diagonal matrix $\mathbf{Q}$.

From Eq. (30), we have that $\pi_{\text{b}}(\mathbf{x}_N) = \zeta(\frac{\mathbf{v}_N}{\delta t})$. Since the mapping ζ is nonlinear in $\mathbf{v}_N$, the term $\|\pi_{\text{b}}(\mathbf{x}_N)\|^2_{\mathbf{R}}$ is not simply quadratic in $\mathbf{v}_N$. Instead, we find a quadratic upper bound on $\|\pi_{\text{b}}(\mathbf{x}_N)\|^2_{\mathbf{R}}$ by defining a scalar $\tilde{r} > 0$ such that

$$\left\|\zeta(\frac{\mathbf{v}_N}{\delta t})\right\|^2_{\mathbf{R}} \leq \tilde{r}\,\|\mathbf{v}_N\|^2\,. \tag{40}$$

Such scalar $\tilde{r}$ is obtained by inverting ζ and finding the maximum over braking actions $\left[T,\ \phi,\ \theta\right]$, i.e.

$$\tilde{r} = \max_{T,\,\phi,\,\theta} \frac{m^2\left(r_1\phi^2 + r_2\theta^2 + r_3\left(T - gm\right)^2\right)}{\delta_t^2\left(T^2 - 2Tgm\cos\phi\cos\theta + g^2m^2\right)}, \tag{41}$$

where $\mathbf{R} = \text{diag}(r_1, r_2, r_3)$.

Thus, a sufficient condition for Condition (29) to hold is

$$p \geq \|\mathbf{Q}\| + \tilde{r}. \tag{42}$$

Combining the first and second cases, a sufficient condition to ensure the Lyapunov stability condition (24) is provided by selecting the terminal cost

$$V(\mathbf{x}_N) = \max\left(\frac{\bar{\ell}}{\underline{\mathbf{a}_{\text{b}}}^2\delta t^2},\ \|\mathbf{Q}\| + \tilde{r}\right)\ \mathbf{v}_N^\top\mathbf{v}_N. \tag{43}$$

D Radar-Inertial Odometry

The radar-inertial odometry follows Nissov et al. (2024), but with slight enhancements. The core structure is as follows: a factor graph-based smoother that fuses Frequency Modulated Continuous Wave (FMCW) radar pointclouds and IMU measurements of angular velocity and specific force. For each radar measurement, a node is created in the graph and connected to the graph with an IMU pre-integration factor, following Forster et al. (2017). Afterwards, a radial speed factor, linking the navigation states to the radar Doppler measurements, is connected to the newly created node. To maintain computational efficiency, the graph size is limited by duration, and sufficiently old nodes are periodically marginalized following the approach in Dellaert and GTSAM Contributors (2022).

Furthermore, a secondary integration routine provides IMU-rate odometry estimates for use by the proposed method. This architecture enables high-rate, accurate velocity estimation, with inevitable position drift resulting from the integration of noisy signals.

E Index to multimedia Extensions

Extension	Media type	Description
1	Video	A visualization of both the ablation and comparative studies presented in Sections 6.3 and 6.4, providing insights into the evaluated environments.
2	Video	The recording and relevant visualizations of the experiment presented in Section 6.6.2, where the constrained-FoV depth camera is used for collision avoidance.
3	Video	The recording and relevant visualizations of the experiment presented in Section 6.6.3, where a human operator provides adversarial velocity reference to the controller.
4	Video	The presentation of the simulations conducted in Section 6.5, as well as the recording and relevant visualizations of the experiment presented in Section 6.6.4, where a radar-based velocity estimator is used to assess resilience to position drift

Table 10. Index of multimedia extensions.