Overcoming Explicit Environment Representations with Geometric Fabrics

Spahn, Max; Bakker, Saray; Alonso-Mora, Javier

DOI
10.1109/LRA.2025.3570891

Publication date
2025

Document Version
Final published version

Published in
IEEE Robotics and Automation Letters

Overcoming Explicit Environment Representations With Geometric Fabrics

Max Spahn, Saray Bakker, *Graduate Student Member, IEEE*, and Javier Alonso-Mora, *Senior Member, IEEE*

***Abstract*—Deployment of robots in dynamic environments requires reactive trajectory generation. While optimization-based methods, such as Model Predictive Control focus on constraint verificaction, Geometric Fabrics offer a computationally efficient way to generate trajectories that include all avoidance behaviors if the environment can be represented as a set of object primitives. Obtaining such a representation from sensor data is challenging, especially in dynamic environments. In this letter, we integrate *implicit* environment representations, such as Signed Distance Fields and Free Space Decomposition into the framework of Geometric Fabrics. In the process, we derive how numerical gradients can be integrated into the push and pull operations in Geometric Fabrics. Our experiments reveal that both, ground robots and robotic manipulators, can be controlled using these implicit representations. Moreover, we show that, unlike the explicit representation, implicit representations can be used in the presence of dynamic obstacles without further considerations. Finally, we demonstrate our methods in the real-world, showing the applicability of our approach in practice.**

***Index Terms*—Collision avoidance, motion and path planning, reactive and sensor-based planning.**

I. Introduction

As ROBOTS make their way into human shared environments, fast reactive behavior is needed to ensure that collisions are avoided at all time. Trajectory Generation (TG) is commonly formulated as a receding horizon optimization problem, where the robot's trajectory is optimized over a short time horizon. Such methods are known as Model Predictive Control (MPC) and have shown great success for autonomous vehicles and drones where the dimension of the configuration space remains small. For higher dimensional configuration spaces, e.g. manipulators and mobile manipulators, the computational cost of Model Predictive Control (MPC) scales poorly with the number of obstacles, leading to slower computation cycles and ultimately to a less reactive behavior [1]. To realize real-time control often requires simplifications of the constraints, e.g., using *kineostatic danger fields* [2] or limiting collision avoidance to the end-effector [3]; achieving compute times of around 50 ms and 20 ms respectively. Geometric Fabrics (fabrics) offer an alternative to these approaches. Based on differential geometry, policies are composed of several components to form a highly reactive and fast behavior [4]. However, the composition of Geometric Fabrics (fabrics) of individual obstacle avoidance geometries is limited to simple geometric shapes, such that a differentiable distance function can easily be formulated. This *explicit* environment representation sets a challenging requirement on the perception part of the motion generation pipeline.

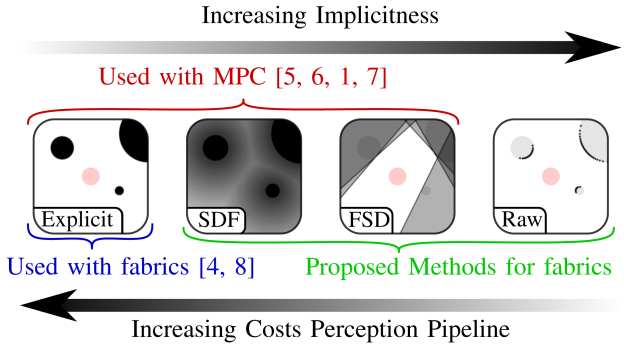


Fig. 1. Different levels of implicitness for environment representations.

In this work, we present and analyze three different representations of the environment to overcome this drawback, namely Free Space Decomposition (FSD), Signed Distance Fields (SDFs) and raw sensor data, from visual sensors, such as cameras or lidars, into the framework of fabrics. We refer to these representations as *implicit*. Generally, the more implicit an environment representation is, the more computational costs are moved from the perception pipeline to the planner, see Fig. 1. In the process, we derive essential extensions to the framework and analyze strengths and weaknesses of the individual methods. To summarize, this letter makes the following contributions:

- We integrate implicit representation of the environment into the framework of fabrics, namely Free Space Decomposition (FSD), Signed Distance Field (SDF) and raw sensor data.
- We derive how numerical gradients can be used for pull-back operations which are essential in the composition of fabrics.
- We analyze the strengths and weaknesses of the three representations in different environments, including moving obstacles, and with two different robot morphologies.

Received 28 October 2024; accepted 8 May 2025. Date of publication 16 May 2025; date of current version 10 June 2025. This article was recommended for publication by Associate Editor W. Thomason and Editor A. Bera upon evaluation of the reviewers' comments. This work was supported by European Union through ERC, INTERACT, under Grant 101041863. *(Corresponding author: Max Spahn.)*

The authors are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: max.spahn@protonmail.com; s.bakker-7@tudelft.nl; j.alonsomora@tudelft.nl).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3570891, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3570891

- We present initial real-world experiments illustrating the power of our open-source implementation.

II. Related Works

Initially, we provide an overview of recent developments in geometric control applied to Trajectory Generation (TG). Subsequently, we recall implicit environment representations employed within alternative TG approaches.

A. Geometric Control for Trajectory Generation

Operational space control emerged as a pioneering control approach, entailing the imposition of desired dynamical systems onto robotic systems [9]. The foundation of this concept was generalized under the name of geometric control, where the exploration of differential geometry offers stability and convergence through geometric prerequisites [10].

More recently, Riemannian Motion Policies (RMPs) for manipulation tasks have introduced a highly responsive TG methodology [11], [12]. This approach attains composability through a separation between the importance metric and the forcing term. Leveraging the *pullback* and *pushforward* operators to navigate across configuration space manifolds, distinct behaviors such as collision avoidance and goal attraction can be systematically designed. Nevertheless, the design of RMPs necessitates an intuitive understanding and experiential experience, with convergence being conditionally established [13].

Subsequently, fabrics emerged as a novel approach, entirely separating the importance metrics and the defining geometry. By adhering to simple construction principles governing these two components, convergence can be readily verified [13], [14], [15]. In our preceding work on fabrics, this framework was initially applied to mobile manipulation and subsequently extended to encompass more dynamic environments [8].

B. Implicit Environment Represenations

While fabrics mainly focus on explicit environment representation [4], [8], new TG methods lean towards implicit approaches. For TG using short-term optimization, unoccupied space constraints limit robot movement, proven useful for mobile manipulators in cluttered areas [1]. In drone flight, a similar concept generates safe flight zones along a global path [5], [6]. In the context of drone flying, SDF has been utilized with MPC in unknown environments [7]. Raw lidar data has been used in combination with RMPs showing impressively high frequencies when computation is parallized on GPU [16]. Also using GPU-accelerated computing environment representations that do not offer gradients can be used in sampling-based MPC [17], [18]. Raw point cloud data can also be integrated into the global planning phase, where the point cloud is used for collision checking in the configuration validation during the sampling phase [19], [20]. Being global path planning methods, such methods do not take into account the dynamic properties of the environment, such as moving obstacles or the robot's velocity. Tackling implicit representations from the robot's point of view, it is also possible to represent the robot and its occupied space implicitly. For example, the SDF representing the robot as a function of the configuration can be learned to be later exploited for avoidances or reaching tasks that include not only the end-effector of the robot [21], [22].

C. Geometric Fabrics Meet Implicit Representations

As implicit representations have gained popularity to address changing environments, the question arises whether such methods can be integrated into the framework of fabrics, a novel approach for TG that exploits the geometry of the configuration space in a more formel manner. In this work, we give some ideas on how some of the above mentioned ideas tested with different TG methods can be integrated into the framework of fabrics to enable highly reactive behavior without the need for overly complicated obstacle detection. Additionally, we want to exploit the computational speed of fabrics, shown in [8], to derive a set of representations that can be used *without* the need for GPU compute as required in [16], [17], [18]. This constraint is justified by the low computational power on smaller robots, such as autonomous ground vehicles and drones.

III. Background

We introduce some notations and the mathematical operations from differential geometry used in the framework of fabrics. This introduction is rather limited and the reader is referred to [8], [23], [24] for a more in-depth introduction.

A. Configurations and Task Variables

We denote $\boldsymbol{q} \in \mathcal{Q} \subset \mathbb{R}^n$ a configuration of the robot with n its degrees of freedom; $\mathcal{Q}$ is the configuration space of the generalized coordinates of the system. Generally, $\boldsymbol{q}(t)$ defines the robot's configuration at time t, so that $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ define the instantaneous derivatives of the robot's configuration. Similarly, we assume that there is a set of task variables $\boldsymbol{x}_j \in \mathcal{X}_j \subset \mathbb{R}^{m_j}$ with variable dimension $m_j \leq n$. The task space $\mathcal{X}_j$ defines an arbitrary manifold of the configuration space $\mathcal{Q}$ in which a robotic task can be represented. Further, we assume that there is a differential map $\phi_j : \mathbb{R}^n \to \mathbb{R}^{m_j}$ that relates the configuration space to the j^{th} task space. For example, when a task variable is defined as the end-effector position, then ϕ_j is the positional part of the forward kinematics. On the other hand, if a task variable is defined to be the joint position, then ϕ_j is the identity function. In the following, we drop the subscript j in most cases for readability when the context is clear.

We assume that ϕ is in $\mathcal{C}^1$ so that the Jacobian is defined as

$$\boldsymbol{J}_\phi = \frac{\partial \phi}{\partial \boldsymbol{q}} \in \mathcal{R}^{m \times n}, \tag{1}$$

or $\boldsymbol{J}_\phi = \partial_{\boldsymbol{q}} \phi$ for short. Thus, we can write the total time derivatives of $\boldsymbol{x}$ as $\dot{\boldsymbol{x}} = J_\phi \dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{x}} = \boldsymbol{J}_\phi \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_\phi \dot{\boldsymbol{q}}$.

B. Spectral Semi-Sprays

The framework of fabrics designs trajectory generation as second-order dynamical systems $\ddot{\boldsymbol{x}} = \pi(\boldsymbol{x}, \dot{\boldsymbol{x}})$ [4], [12]. The trajectory is defined by the differential equation $\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{f} = 0$,

where $\boldsymbol{M}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ and $\boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ are functions of position and velocity. Besides, $\boldsymbol{M}$ is symmetric and invertible. We denote such systems as $\mathcal{S} = (\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$ and refer to them as *spectral semi-sprays*, or *specs* for short. Often, we drop the subscript $\mathcal{X}$ when the context is clear.

C. Operations on Specs

Trajectory generation requires the combination of multiple components, such as collision avoidance, joint limits avoidance, etc. In the framework of fabrics, these components are represented as specs in different manifolds and combined in a consistent way using a metric-weighted sum. Related operations from differential geometry are recalled here.

Given a differential map $\phi : \mathcal{Q} \to \mathcal{X}$ and a spec $(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$, the *pullback* is defined as

$$\text{pull}_{\phi}(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}} = \left(\boldsymbol{J}_{\phi}^{T} \boldsymbol{M} J, \boldsymbol{J}_{\phi}^{T}(\boldsymbol{f} + \dot{\boldsymbol{J}}_{\phi}\dot{\boldsymbol{q}})\right)_{Q}. \tag{2}$$

The pullback allows converting between two distinct manifolds (e.g. a spec could be defined in the robot's workspace and pulled into the robot's configuration space using the pullback with ϕ being the forward kinematics).

For two specs, $\mathcal{S}_1 = (\boldsymbol{M}_1, \boldsymbol{f}_1)_{\mathcal{X}}$ and $\mathcal{S}_2 = (\boldsymbol{M}_2, \boldsymbol{f}_2)_{\mathcal{X}}$, their *summation* is defined by:

$$\mathcal{S}_1 + \mathcal{S}_2 = (\boldsymbol{M}_1 + \boldsymbol{M}_2, \boldsymbol{f}_1 + \boldsymbol{f}_2)_{\mathcal{X}}. \tag{3}$$

Additionally, a spec can be *energized* by a Lagrangian energy. Effectively, this equips the spec with a metric. Specifically, given a spec of form $\mathcal{S}_{\boldsymbol{h}} = (\boldsymbol{I}, \boldsymbol{h})$ and an energy Lagrangian $\mathcal{L}_e$ with the derived equations of motion $\boldsymbol{M}_{\mathcal{L}_e}\ddot{\boldsymbol{x}} + \boldsymbol{f}_{\mathcal{L}_e} = 0$, we can define the operation

$$\begin{aligned} S_{\boldsymbol{h}}^{\mathcal{L}_e} &= \text{energize}_{\mathcal{L}_e}\{S_{\boldsymbol{h}}\} \\ &= (\boldsymbol{M}_{\mathcal{L}_e}, \boldsymbol{f}_{\mathcal{L}_e} + \boldsymbol{P}_{\mathcal{L}_e}[\boldsymbol{M}_{\mathcal{L}_e}\boldsymbol{h} - \boldsymbol{f}_{\mathcal{L}_e}]), \end{aligned} \tag{4}$$

where $\boldsymbol{P}_{\mathcal{L}_e} = \boldsymbol{M}_{\mathcal{L}_e}(\boldsymbol{M}_{\mathcal{L}_e}^{-1} - \frac{\dot{\boldsymbol{x}}\dot{\boldsymbol{x}}^T}{\dot{\boldsymbol{x}}^T \boldsymbol{M}_{\mathcal{L}_e}\dot{\boldsymbol{x}}})$ is an orthogonal projector. The resulting spec is an *energized spec* and we call the operation *energization*.

With spectral semi-sprays and the presented operations, avoidance behavior, such as joint limit avoidance, collision avoidance or self-collision avoidance, can be realized.

D. Optimization Fabrics

In the previous subsection, we recalled how different avoidance behaviors can be combined. Spectral semi-sprays can additionally be *forced* by a potential, denoted as the *forced variant* of form $\mathcal{S}_{\psi} = (\boldsymbol{M}, \boldsymbol{f} + \partial_{\boldsymbol{x}}\psi)$. This forcing term clearly changes the behavior of the system. Under certain conditions, the trajectory $\boldsymbol{x}(t)$ of the forced variant converges towards the minimum of the potential ψ [4].

First, the initial spec that represents an avoidance component is written in the form $\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0$, such that $\boldsymbol{h}$ is *homogeneous of degree 2:* $\boldsymbol{h}(\boldsymbol{x}, \alpha\dot{\boldsymbol{x}}) = \alpha^2\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ (**Creation**). Secondly, the geometry is energized (4) with a Finsler structure [4, Definition 5.4] (**Energization**). The property of homogeneity of degree 2 and the energization with the Finsler structure guarantees, according to [4, Theorem 4.29], that the energized spec forms a *frictionless fabric*. A frictionless fabric is defined to optimize the forcing potential ψ when being damped by a positive definite damping term [4, Definition 4.4]. Thirdly, all avoidance components are combined in the configuration space of the robot using the pullback and summation operation (**Combination**). Note, that both operations are closed under the algebra designed by these operations, i.e. every pulled optimization fabric or the sum of two fabrics is, itself, an optimization fabric. In the last step, the combined spec is forced by the potential ψ with the desired minimum and damped with a positive definite damping term (**Forcing**). This resulting system of form $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \partial_{\boldsymbol{q}}\psi + \beta\dot{\boldsymbol{q}} = 0$ is solved to obtain the trajectory generation policy in acceleration form $\ddot{\boldsymbol{q}} = \pi(\boldsymbol{q}, \dot{\boldsymbol{q}})$.

IV. Methods

As this work does not alter the fundamental principles of fabrics, this section focuses on the integration of three different implicit environment representations into the framework. In the process, we lay out the necessary concepts required for the representations, and provide the tools to combine them with fabrics. The core idea however is identical for all presented methods. Collision avoidance is realized by defining a differentiable mapping ϕ from the configuration space $\mathcal{Q}$ to the distance manifold between the robot and the environment. On that manifold, the desired behavior is encoded using a geometry and a energizing Finsler to form a geometric fabric,

$$\mathcal{S} = (\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}\,.$$

When being combined with other behaviors, this fabric is *pulled* back into the configuration space $\mathcal{Q}$ to be *summed* up with other fabrics. To recall, the *pullback* is defined as

$$\text{pull}_{\phi}(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}} = \left(\boldsymbol{J}_{\phi}^{T} \boldsymbol{M} J, \boldsymbol{J}_{\phi}^{T}(\boldsymbol{f} + \boldsymbol{M}\dot{\boldsymbol{J}}_{\phi}\dot{\boldsymbol{q}})\right)_{Q}.$$

The difference between the methods presented in this work is the definition of the mapping ϕ and therefore the computation of the gradient J. The specifics of the fabric defined on the distance manifold remain unchanged over all methods.

A. Signed Distance Fields

a) Representation: Collision avoidance can be realized using SDF [7]. In this approach, the environment is discretized into a grid and the distance to the closest obstacle of the environment is assigned to each voxel in the grid. The distance is zero on the obstacle's surface, negative in its inside and positive outside the obstacle. In this sense, it indicates the distance to the obstacle's frontier. In [7], the SDF is computed based on lidar data in combination with continuous mapping, but other ways to generate SDFs are possible, see [21], [22] for some manipulation examples. In this work, we address dynamic environments, therefore the SDF changes over time, see Fig. 2 for a two-dimensional example.

b) Integration: According to (2), the integration of collision avoidance requires the computation of the gradient J. When the environment is represented as a SDF, the gradient cannot be

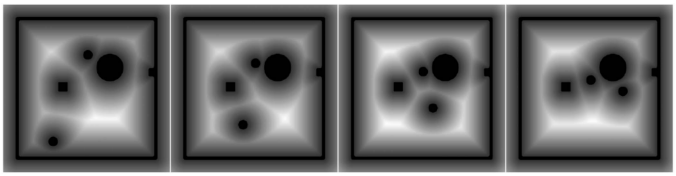


Fig. 2. Changing Signed Distance Field in 2D. .

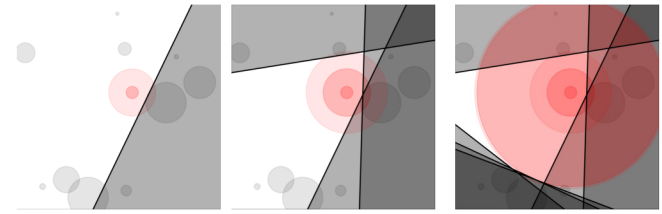


Fig. 3. Iterative Free Space Decomposition in 2D. The $\boldsymbol{x}_{\text{seed}}$ is shown in red, obstacles in light gray, each $\boldsymbol{P}$ by black plane and the block workspace is shaded in gray. The resulting FSD is the remaining white space. .

computed analytically, as it is possible for simple geometric shapes in [8], [23]. To overcome this limitation, we propose to use the numerical gradient as an approximation instead. The SDF is evaluated based on the forward kinematics of the collision link, so $\phi_{sdf}(\text{fk})$ is a function of $\boldsymbol{q}$. As $\partial_{\boldsymbol{q}}\phi_{sdf}$ is not analytically accessible due to the numerical nature of SDFs, we apply the chain rule to obtain

$$\boldsymbol{J}_{\phi,sdf} = \frac{\partial \phi_{sdf}}{\partial \boldsymbol{q}} = \frac{\partial \phi_{sdf}}{\partial \text{fk}} \frac{\partial \text{fk}}{\partial \boldsymbol{q}}.$$

The second term $\partial_{\boldsymbol{q}}\text{fk}$ is the gradient of the forward kinematics, which can be computed analytically. The first part is the gradient of the SDF which can be approximated using finite differences:

$$\left(\frac{\partial \phi_{sdf}}{\partial \text{fk}}\right)_i \approx \frac{\phi_{sdf}(\text{fk} + \Delta_i e_i) - \phi_{sdf}(\text{fk} - \Delta_i e_i)}{2\Delta_i},$$

where Δ is the resolution in the i-th dimension. For the integration into fabrics, the value $\phi_{sdf}(\text{fk})$ and the gradient $\partial_{\text{fk}}\phi_{sdf}$ must be computed at runtime. In contrast, the analytical component $\partial_{\boldsymbol{q}}\text{fk}$ can be precomputed symbolically. Similar to [23], we omit the curvature term $\dot{\boldsymbol{J}}_{\phi}$ in the pullback operation. When working with multi-link robots, such as manipulators, different manifolds are created for the different collision links. Note, that the same SDF can be used for all collision links.

B. Free Space Decomposition

a) Representation: Popular in recent literature [1], [5], [6] is to decompose the environment, with all its obstacles, including dynamic obstacles, into a set of FSD. Then, the workspace is reduced or locally approximated by, typically one, convex regions. The free space is defined by a set of half-planes, each defined by a normal vector $\boldsymbol{n}_{\boldsymbol{P}}$ and a constant $c_{\boldsymbol{P}}$, see Fig. 3. In the following, we describe how the free space decomposition can be computed given a $\mathcal{P}$ of the environment.

The method used in this work is inspired by [5]. We define the $\boldsymbol{x}_{\text{seed}}$, the point in space from which the FSD is computed, as the position of the collision link in question, see Fig. 3. Then, $\mathcal{P}$ is sorted according to the euclidean distance to the $\boldsymbol{x}_{\text{seed}}$. Starting with the closest point, a plane $\boldsymbol{P}$ defined by $\boldsymbol{n}_{\boldsymbol{P}} = \boldsymbol{p} - \boldsymbol{x}_{\text{seed}}$ and $c_{\boldsymbol{P}} = \boldsymbol{p}$ for every $\boldsymbol{p} \in \mathcal{P}$. To further speedup the process, every $\boldsymbol{p}$ in $\mathcal{P}$ that is 'behind' an existing $\boldsymbol{P}$ is removed from $\mathcal{P}$. The method results in set $\mathcal{S}_{\text{planes}} = \{\boldsymbol{P}_i, i \in [0, n]\}$, representing the free space around $\boldsymbol{x}_{\text{seed}}$. The algorithm is visualized in Fig. 3 for a 2D case.

b) Integration: To integrate FSD into the framework of fabrics, we define a differentiable mapping from configuration space $\mathcal{Q}$ to the distance manifold between robot and each constrained plane $\boldsymbol{P}$. Given the fk of a collision link on the robot and the radius of the collision link, the distance to the plane $\boldsymbol{P}$ defined by its normal $\boldsymbol{n}_{\boldsymbol{P}}$ and the constant $c_{\boldsymbol{P}}$, the distance is computed as:

$$\phi(\boldsymbol{P}, \text{fk}, r_{\boldsymbol{L}}) = \frac{\boldsymbol{n}_{\boldsymbol{P}}^T \text{fk} + c_{\boldsymbol{P}}}{\|\boldsymbol{n}_{\boldsymbol{P}}\|} - r_{\boldsymbol{L}}. \tag{5}$$

In contrast to SDFs, the gradients, J and $\dot{\boldsymbol{J}}_{\phi}$, can be computed analytically as a function of $\boldsymbol{q}$, $\boldsymbol{n}_{\boldsymbol{P}}$ and $c_{\boldsymbol{P}}$.

C. Raw Sensor Data

a) Representation: As pointclouds generated with either cameras or lidars are usually very large, the direct usage of the raw data is often avoided for TG because of the high computational costs, especially when working on cheaper hardware without access to highly capable GPU hardware. Direct integration refers to create one spherical obstacle for each data point in your raw sensor data. As fabrics are computationally efficient and can handle large amounts of constraints more easily than methods relying on iterative optimization in each time step [8], direct integration becomes feasible. This is also based on the findings by [16] where raw sensor data was used for drone flying with RMPs. This allows to reduce the amount of preprocessing, limiting the amount of uncertainty and room for error in this step. Here, we explain how raw lidar sensor data can be utilized with fabrics. The same approach can be used to directly integrate pointclouds or occupancy grids.

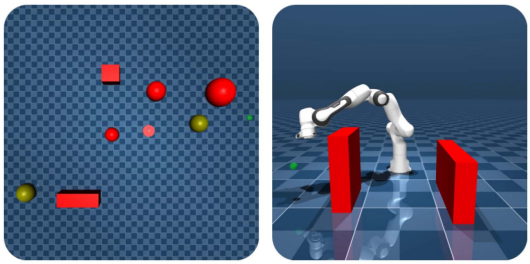
b) Integration: We assume that a sensor outputs a set of n_{points} points $\mathcal{P}$ in the robot's workspace. We integrate this data directly into fabrics by placing a virtual, spherical obstacle at each $\boldsymbol{p} \in \mathcal{P}$. The map is then defined as

$$\phi_{\text{raw}}(\boldsymbol{p}) = \|\boldsymbol{p} - \text{fk}\| - r_{\boldsymbol{L}} - r_{\text{resolution}},$$

The radius $r_{\text{resolution}}$ of these obstacles must be chosen to reflect the resolution of the point cloud.

V. Experimental Results

In this section, we explain our implementation of the presented methods and present quantitative comparisons for two simulation environments, namely a holonomic ground robot without and with moving obstacles, see Fig. 4(a), and a robotic manipulator, see Fig. 4(b). Unless stated otherwise, we evaluate 50 cases with different noise levels σ. The noisy signal is generated using a Gaussian distribution centered around the unoisy sensor data. When using the explicit obstacle positions, the noise is added to the position of the obstacles and is thus measured in m^2. When using FSD or raw sensor data, the noise



(a) Ground robot (b) Robotic manipulator

Fig. 4. Setups for the ground robot and the robotic manipulator.

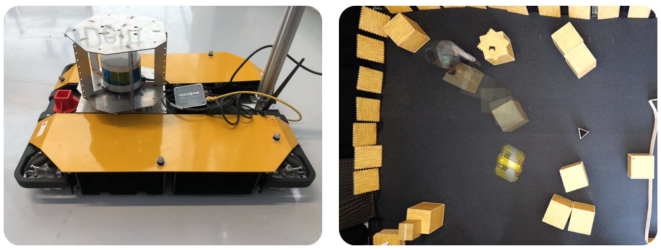


Fig. 5. Real ground robot used to run the experiments. For the real-world experiment, we used a Clearpath Dingo with a Velodyne VLP16 mounted on top (left). The real-world experiment shows the Dingo navigating through an environment where a human throws in obstacles (right).

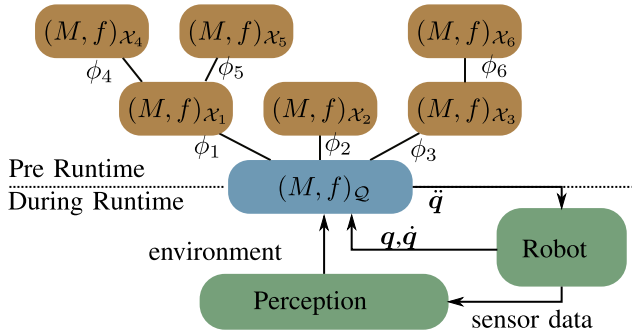


Fig. 6. Composition of symbolic fabrics and runtime loop.

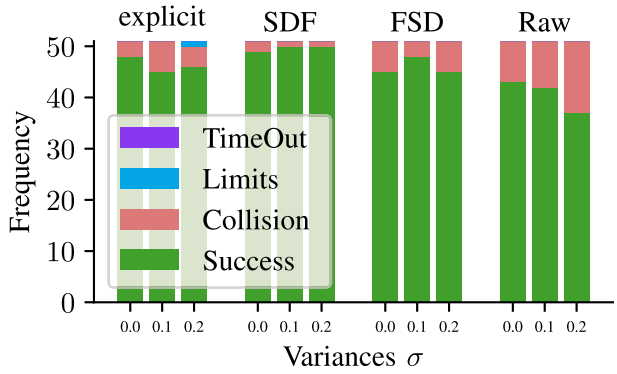


Fig. 7. Success rates for ground robot in **static** environments for different noise level on sensor inputs. .

is applied to the position of the measured points. Its unit is thus m^2 as well. When using SDF, we apply the noise to the computed distance field, but not its gradient, to allow for the same unit, namely m^2, also in this method. For all experiments, except for the real scenario with the manipulator, the implicit representations are computed at every time-step. When using raw sensor data as the representation, geometries and metrics are scaled by a factor of n_{points}^{-1}. The gradient of the SDF is only evaluated on queried points to avoid computational costs. Note that all implementations of the implicit representations are simplistic to promote a fair comparison. For more advanced implementations of these methods, we refer to [5] (FSD) and [7] (SDF).

The performance is measured in terms of success, solver time and execution time to reach the goal. Importantly, the solver time reported in this work does not include the computation of the environment representation. For SDF however, we included the computation of the numerical gradient because it is fabrics-specific. Lastly, we evaluate the methods in the real world using a manipulator, see Fig. 13, and a holonomic ground robot, see Fig. 5.

a) Details on implementation: The implementation used in this work uses symbolic pre-computation of fabrics. In this symbolic interpretation, the composition of the different behaviors is performed before runtime in a symbolic way, see Fig. 6. The implementation is identical to the one in [25]. The code can be found at www.github.com/tud-amr/fabrics. For collision avoidance, we used a soft-max function to avoid approaching infinity close to the collision frontier, which is of the form,

$$\boldsymbol{h} = \frac{-\theta_1}{1+e^{\theta_2 \boldsymbol{x}-\theta_3}}\dot{\boldsymbol{x}}^2, \mathcal{L}_e = \frac{\theta_4}{1+e^{\theta_5 x-\theta_6}}\text{sgn}(\dot{\boldsymbol{x}})\dot{\boldsymbol{x}}^2,$$

where the parameters θ_i were tuned differently for all methods. The simulation environment as well as the algorithm for computing the FSD, SDF and the raw lidar data can be found as part of urdfenvs. For the real world experiments, we used a ROS bridge and used the same implementation to process the point clouds, generated by either a Velodyne VLP-16 mounted on the robot, see Fig. 5, or by an occupancy grid build using the octomap package [26].

B. Ground Robot

For the experiments with the ground robot, 11 obstacles (cuboids or spheres), out of which at most 2 are moving, were randomized in their sizes and positions. When comparing explicit environment representations with the proposed techniques using noise-free sensor data ($\sigma = 0.0$ m), SDFs demonstrate the highest success rate. Using raw sensor data or FSD shows lower success rates, see Fig. 7. FSD is more sensible to changes in sensor data, because large areas of points are combined. When the constraining planes are close to the robot, minor changes (either by noise in the sensor data or by changes in the robot's position) can result in substantial changes in the constraints. This is not the case for SDF as the field has a global character where the magnitude of changes are independent of the distance to the robot. To explain the collisions with the explicit representation, we must emphasize that, unlike constrained optimization problem formulations for TG, fabrics do not provide hard constraint-satisfaction guarantees [23]. Also, the theory is built in the continuous time domain, such that numerical inaccuracies

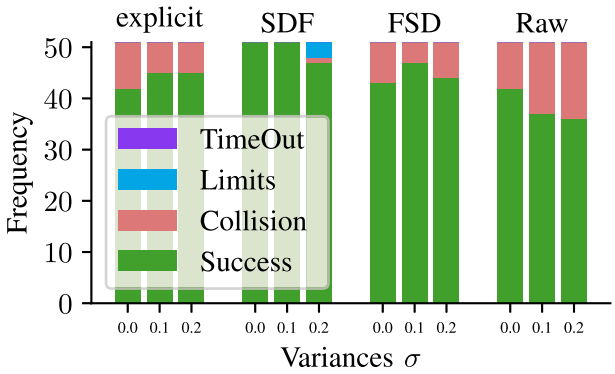


Fig. 8. Success rates for ground robot in **dynamic** environments for different noise level on sensor inputs. .

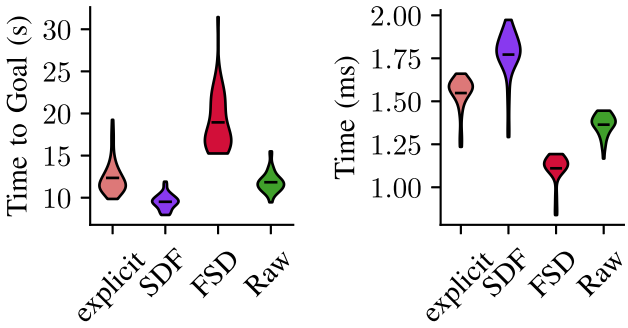


Fig. 9. Evaluation metrics, goal reaching (left) and solvertimes (right), for the ground robot in simulation. .

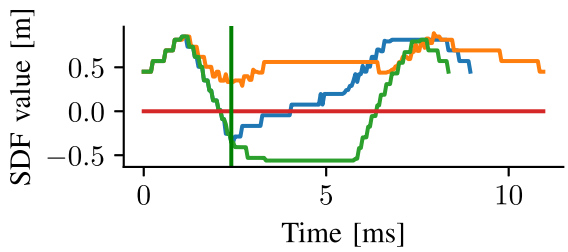


Fig. 10. When the robot is in collision, e.g. because of sensor inaccuracies or sensor noise, implicit representanions (here using SDF) allow it to navigate out of collision smoothly. The SDF is only activated at $t_1 = 2.4$s (blue), always active (orange), never active (green). .

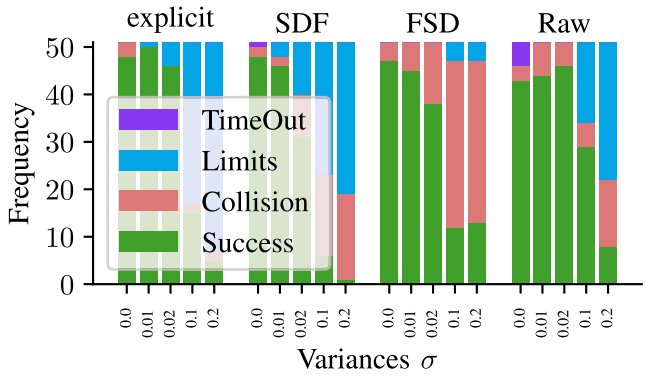


Fig. 11. Success rates for robotic manipulator in static environments for different noise level on sensor inputs. .

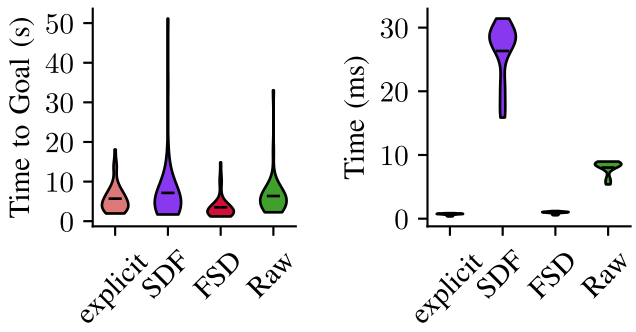
in the integration step and the low-level controller can lead to violations of collision avoidance. Interestingly, the success rate slightly increases with added noise for explicit environment representations. As the noise for the obstacle's position follows a Gaussian distribution in our experiments, the distance between robot and obstacle can also be approximated with a Gaussian for small variances σ. However, after applying the geometries to the Gaussian-distributed distances, the repelling accelerations average higher than for noise-free obstacle positions. That eventually results in more conservative behavior with fabrics, explaining the decrease in number of collisions. Moreover, terminal velocities at time of collision for all methods are usually approaching zero on collision. In such local minima, the formulation looses its ability to avoid collisions as the repulsive acceleration is strictly proportional to the square of the speed of approaching [8]. When exposing all methods to an environments with moving obstacles ($v_{\max} = 0.5$ m/s), the explicit representation has degrading performance, e.g. without noise 47/50 (static) to 42/50 (dynamic), see Fig. 8. On the other hand, implicit representations do not suffer much for this change, 49/50 successes for SDF and 46/50 successes for SDF with $\sigma = 0.2$. This confirms our hypothesis that implicit representations are a suitable approach in human-shared, changing environments. Moreover, this finding is in line with the findings in [8] on the inability for the explicit representation to avoid moving obstacles. The time it takes to reach the target is higher when using FSD, see Fig. 9. Decomposing the free space into hyperplanes tends to be more conservative, because forming corridors between two obstacles in front of the robot are approximated with a artificial wall blocking the way forward. Only on moving forward this wall is eventually split into two walls, one for each side of the corridor. This problem is also reported in the literature on MPC formulations for drone flying [6]. Computation times are low for all methods (between 0.5 ms and 2.0 ms) Fig. 9. As all experiments were conducted on simple CPU's, this highlights the usability for simple hardware setups that require fast compute times for reactive behaviors.

When using implicit environments in the real simulator, errors in the sensors may lead to brief periods of collision violations for the method that do not correspond to an actual collision. In a qualitative experiment, we show that our method is able to smoothly move out of collision in that case. Specifically, we used a signed distance field that was only activated, once the robot is in collision. The value of the SDF at the robot's position over time shows that the robot moves out of the collision right after the activation, see Fig. 10.

C. Robotic Manipulator

The experiments with the robotic manipulator were obtained by randomly selecting goal locations.

High success rates are achieved for all methods in static, noise-free environments, as shown in Fig. 11. As the noise level increases, success rates decrease for all methods. Unlike implicit representations, an explicit environment representation does not suffer from collisions as sensor noise increases, but leads to higher limit-violation rates. Limit-violations are usually caused by a negative x value in the collision spec, leading to infinitely high repulsive accelerations. The reason is likely the inability of



(a) Goal reaching time (b) Solvertimes

Fig. 12. Evaluation metrics for robot manipulator in simulation.

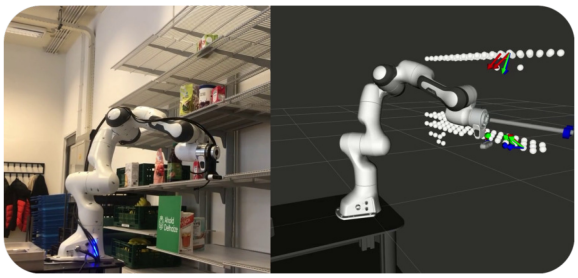


Fig. 13. Real robotic manipulator used to run the experiments. A Franka Emika Panda is confronted with an unknown shelf environment in a supermarket. Arrows indicate the normals of the planes obtained by the FSD. Different colors indicate different links on the robotic arm. .

the low-level controller to accurately track the commanded acceleration. More advanced control techniques or the integration of the robot's dynamics could help prevent this in future works. Up to a noise level of $\sigma = 0.02$, success rates are still between 60% (SDF) and 90% (raw sensor data). In terms of execution time, all methods perform similarly, as expected from the tuning process. Solver times are highest for SDF (≈ 25.0 ms) and below 10 ms for the other methods, see Fig. 12(b). The significantly higher solver times for SDF are due to the computation of the numerical gradient, which is fabrics-specific and thus included in the solver time. This makes all methods suitable for real-time applications.

D. Real-World Experiments

We tested the methods presented in this letter in the real-world using a Clearpath Dingo and a Franka Emika Panda.

a) Dingo: For the Clearpath Dingo, we used a Velodyne VLP16 to generate the point cloud data in the ground plane, effectively discarding information for higher z-values. We use a resolution of 1 ray/degree. The method is wrapped into a ros-node where new control actions are commanded at 40 Hz. We test the methods in an arbitrary setup environment where obstacles are placed and thrown in front of the robot by a human, see Fig. 5. For detailed understanding of the setup, we refer to the accompanied video material. The robot is able to quickly adapt to the obstructions and safely navigates the environment. However, similar to the findings in simulation, when exposed to local minima the robot is not able to escape. This is due to the fact that all methods presented in this letter are highly reactive, exhibiting no time-horizon planning into the future. This is well in line with existing literature on the framework of fabrics and emphasizes the ideal usage of fabrics as a safe medium on which attractor policies can act [27].

b) Panda: For the Franka Emika Panda, we used the octomap package to generate the occupancy grid. The method is wrapped into a ros-node where new control actions are commanded at 40 Hz. We used three collision links on the robots for collision avoidance, see Fig. 13 for the realization of FSD. The experiments reveal that implicit environment representations allow for collision-free motion of robotics arms without the need for a complex perception pipeline.

VI. Conclusion

This letter introduces several approaches to surpass explicit environment representations, employing three distinct implicit representations within the fabrics framework. The study demonstrates that these techniques notably reduce demands and constraints on perception pipelines, while maintaining a low computational load on the planner. Consequently, the proposed methods enable the application of analogous strategies in both dynamic and static environments. The outcomes underline the successful integration of numerical gradients, frequently accessible in trained models, into the symbolic implementation of fabrics detailed in [8]. Additionally, the real-world experiments show that the methods can be transferred physical robots. Integration of established methods for generating implicit environment representations, e.g., [5] for FSD and [7] for SDF, could be considered. Their usage may lead to even better performance in real-world experiments. Implicit representations, especially those using raw point clouds, for motion planning or TG is often tackled using GPU-accelerated algorithms [16], [17], [18]. Although, this presented work explicitly avoids using GPU acceleration, a quantitative comparison would further increase the understanding of strengths and weaknesses of both approaches. While this work refrains from delving into dynamic environmental representations, an exploration of the potential synergies between the implicit representations introduced here and the findings from [8] is a promising avenue for investigation. Finally, the results in this letter reveal that implicit environment representations can be straight-forward integrated into fabrics and show competitive computational costs without heavy hardware requirements.

Acknowledgment

Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] M. Spahn, B. Brito, and J. Alonso-Mora, "Coupled mobile manipulation via trajectory optimization with free space decomposition," in *Proc. 2021 IEEE Int. Conf. Robot. Automat.*, 2021, pp. 12759–12765.

[2] M. Wonsick, P. Long, A. Ö. Önol, M. Wang, and T. Padir, "A holistic approach to human-supervised humanoid robot operations in extreme environments," *Front. Robot. AI*, vol. 8, 2021, Art. no. 550644.
[3] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robot. Automat. Lett.*, vol. 8, no. 12, pp. 7986–7993, Dec. 2023.
[4] N. D. Ratliff, K. Van Wyk, M. Xie, A. Li, and M. A. Rana, "Optimization fabrics," 2020, *arXiv:2008.02399v2*.
[5] S. Liu et al., "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1688–1695, Mar. 2017.
[6] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2019, pp. 1934–1940.
[7] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "VoxBlox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.
[8] M. Spahn, M. Wisse, and J. Alonso-Mora, "Dynamic optimization fabrics for motion generation," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 2684–2699, Apr. 2023.
[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. - IEEE Int. Conf. Robot. Automat.*, 1985, pp. 500–505.
[10] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Control Systems*, vol. 49. Berlin, Germany: Springer, 2019.
[11] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," 2018, *arXiv:1801.02854*.
[12] C.-A. Cheng et al., "RMP flow: A computational graph for automatic motion policy generation," in *Proc. 13th Int. Workshop Algorithmic Found. Robot.*, Springer, 2020, pp. 441–457.
[13] M. Xie et al., "Geometric fabrics for the acceleration-based design of robotic motion," 2021, *arXiv:2010.14750*.
[14] A. Li et al., "RMP2: A structured composable policy class for robot learning," 2021, *arXiv:2103.05922*.
[15] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Neural autonomous navigation with Riemannian motion policy," in *Proc. 2019 Int. Conf. Robot. Automat.*, 2019, pp. 8860–8866.
[16] M. Pantic et al., "Obstacle avoidance using raycasting and Riemannian motion policies at kHz rates for MAVs," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1666–1672.
[17] C. Pezzato, C. Salmi, M. Spahn, E. Trevisan, J. Alonso-Mora, and C. H. Corbato, "Sampling-based model predictive control leveraging parallelizable physics simulations," *IEEE Robot. Automat. Lett.*, vol. 10, no. 3, pp. 2750–2757, 2025.
[18] B. Sundaralingam et al., "Curobo: Parallelized collision-free robot motion generation," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 8112–8119.
[19] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 8749–8756.
[20] C. W. Ramsey, Z. Kingston, W. Thomason, and L. E. Kavraki, "Collision-affording point trees: SIMD-amenable nearest neighbors for fast collision checking," in *Proc. Robot.: Sci. Syst.*, *arXiv:2406.02807*.
[21] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized deep signed distance fields for reactive motion generation," in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 6673–6680.
[22] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 480–487, Feb. 2023.
[23] N. D. Ratliff, K. Van Wyk, M. Xie, A. Li, and M. A. Rana, "Generalized nonlinear and finsler geometry for robotics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 10206–10212.
[24] K. Van Wyk et al., "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3202–3209, 2022.
[25] M. Spahn and J. Alonso-Mora, "Autotuning symbolic optimization fabrics for trajectory generation," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 11287–11293.
[26] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.
[27] K. Van Wyk, A. Handa, V. Makoviychuk, Y. Guo, A. Allshire, and N. D. Ratliff, "Geometric fabrics: A safe guiding medium for policy learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 6537–6543.