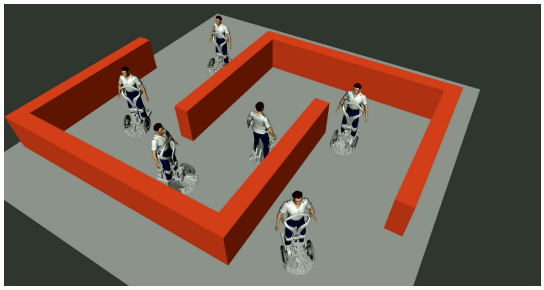
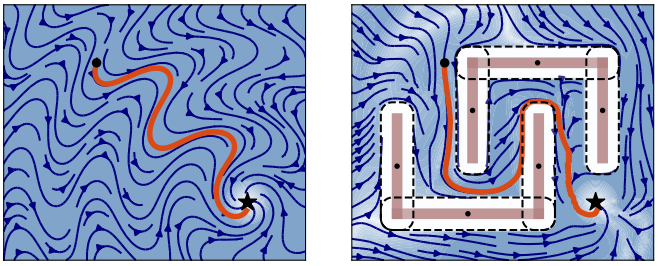
Avoidance of Concave Obstacles through Rotation of Nonlinear Dynamics

Lukas Huber[1], Jean-Jacques Slotine[2], Aude Billard[1]

***Abstract*—Controlling complex tasks in robotic systems, such as circular motion for cleaning or following curvy lines, can be dealt with using nonlinear vector fields. In this paper, we introduce a novel approach called rotational obstacle avoidance method (ROAM) for adapting the initial dynamics when the workspace is partially occluded by obstacles. ROAM presents a closed-form solution that effectively avoids star-shaped obstacles in spaces of arbitrary dimensions by rotating the initial dynamics towards the tangent space. The algorithm enables navigation within obstacle hulls and can be customized to actively move away from surfaces, while guaranteeing the presence of only a single saddle point on the boundary of each obstacle. We introduce a sequence of mappings to extend the approach for general nonlinear dynamics. Moreover, ROAM extends its capabilities to handle multi-obstacle environments and provides the ability to constrain dynamics within a safe tube. By utilizing weighted vector-tree summation, we successfully navigate around general concave obstacles represented as a tree-of-stars. Through experimental evaluation, ROAM demonstrates superior performance in terms of minimizing occurrences of local minima and maintaining similarity to the initial dynamics, outperforming existing approaches in multi-obstacle simulations. The proposed method is highly reactive, owing to its simplicity, and can be applied effectively in dynamic environments. This was demonstrated during the collision-free navigation of a 7 degree-of-freedom robot arm around dynamic obstacles.**



(a) QOLO-robot navigating within small labyrinth



(b) Initial dynamics **(c)** Rotated dynamics

Fig. 1: An autonomous wheelchair is guided by obstacle avoidance to navigate in an environment of complex obstacles (c). The intensity of the shading in (a) and (b) indicates the magnitudes of the velocity.

I. Introduction

Reactive motion plays a crucial role in numerous real-world robotics applications. When operating outside the controlled environments of factory floors, robots are exposed to unpredictable and dynamic surroundings, making precise estimation challenging. As a result, real-time adaptive controllers are essential to enable robots to adapt and reevaluate their actions in response to changing conditions.

A primary constraint when navigating in dynamic and cluttered environments is to ensure the safety of individuals moving around the robot. This requires the robot to constantly and rapidly replan its path to avoid collisions while making every effort to maintain its intended task and adhere to the originally intended movement dynamics. Furthermore, to protect physical hardware from potential damage caused by high accelerations, it is crucial to design a smooth system. Additionally, collision avoidance needs to seamlessly integrate with reactive control techniques (Figure 1).

Control methods based on vector fields [1] and dynamical systems [2] have proven to be well-suited for addressing the challenges posed by such scenarios. Rather than precomputing a trajectory for the robot, these methods generate a (nonlinear) control field that is evaluated in real-time at the robot's position. This allows the robot to react instantaneously to disturbances and perceive changes in its environment. In this work, we leverage the framework of dynamical systems and vector fields to develop an adaptive obstacle avoidance approach capable of modifying nonlinear motion in dynamic and complex environments (Fig. 1).

Dynamical systems represent the evolution of a system's state without considering its history, effectively capturing the system's dynamics as a vector field. In robotics, dynamical systems have been employed for learning complex dynamics [3] and enforcing stability guarantees through techniques such as Lyapunov Stability [4] or Contraction Theory [2]. Force control in robotics often utilizes second-order dynamical sys-

This work was supported by EU ERC grant SAHR.

[1] LASA Laboratory, Swiss Federal School of Technology in Lausanne - EPFL, Switzerland. {lukas.huber;aude.billard}@epfl.ch

[3] Nonlinear Systems Laboratory, Massachusetts Institute of Technology, USA. jjs@mit.edu

tems [5], while obstacle avoidance typically relies on first-order systems that output desired velocities based on the current position [6]. Consequently, additional controllers are employed to ensure force- and torque-controlled robots follow the desired motion of the dynamical system [7].

Analogously, desired dynamics based on position can be represented as a vector field [1]. These nonlinear vector fields can be designed to ensure convergence and stability properties. Nonlinear vector fields designed for path following, known as "vector-field-guided path following," enable smooth convergence towards a desired path, reducing path-following errors [8].

In this paper, we develop an adaptive obstacle avoidance framework capable of modifying nonlinear motion in response to dynamic and complex environments. The proposed approach aims to navigate the control system safely while maintaining the integrity of the intended motion dynamics. The effectiveness of our approach is demonstrated through experiments.

A. Related Work

Obstacle avoidance methods are often categorized into local and global strategies [9]. Global methods focus on convergence to the goal, while local methods prioritize reactive collision avoidance in dynamic scenarios.

Global sampling-based methods, such as Rapidly-Exploring Random Trees (RRT), have been widely utilized to populate the search space by branching out a space-filling tree in order to find a feasible path [10]. Building upon RRT, the RRT* algorithm was introduced to automatically scale the sampling length during the search, resulting in improved performance and convergence rate [11]. Similarly, Probabilistic Roadmaps (PRM) employ space sampling to determine the collision-free status of points. By connecting the collision-free samples, local connectivity graphs are formed, enabling graph-search algorithms to find an optimal path from the initial position to the goal [12]. Although PRM and RRT are probabilistic-complete, meaning they can find a feasible path if one exists [13], the sampling-based approach incurs a significant computational cost. This renders it unsuitable for real-time recomputation, which is essential for dynamic environments. Sampling algorithms like RRT and PRM exhibit a complexity that is linear to the number of nodes n in the sampled tree, represented as $\mathcal{O}(n) = \mathcal{O}(exp(N))$ [14]. Even in two-dimensional environments, the trees need to cover the entire workspace, leading to a large number of nodes. To address the computational challenges and facilitate fast computation in dynamic environments, initial paths are often reshaped. One approach involves interpreting trajectories as *elastic strips*, enabling efficient adaptation and obstacle avoidance [15]. However, this comes at the loss of convergence guarantees.

Motion Optimization (MO) has been employed to dynamically adapt the originally sampled trajectories, analogous to stretching an elastic band around an object [16]. However, MO tends to perform poorly in highly non-convex problems and often converges to suboptimal local minima. To alleviate this issue, Markov chains have been utilized for joint-space control [17], while other approaches incorporate Riemann geometry and geometric constraints to improve convergence heuristics [18], [19]. Nonetheless, MO remains local in nature and often fails to converge when dealing with non-convex motions.

Model Predictive Control (MPC) reduces the optimization problem to a finite time horizon, thereby reducing the convergence time [20]. The increased computational power in recent years has facilitated the utilization of MPC for real-time collision avoidance [21]. Sampling-based MPC has been employed for dynamic configuration-space collision avoidance in scenarios where an analytic cost function is absent [22], [23]. However, the limited time horizon of MPC often limits the problem to local optimality, and global convergence cannot always be guaranteed.

In recent years, the application of Machine Learning (ML) in collision avoidance for robotics has gained significant traction. End-to-end learning approaches have enabled collision-free multi-robot navigation [24], while fuzzy artificial potential fields augmented with neural networks have been utilized for motion planning of mobile robots [25]. Conversely, RRT has served as a foundation for training reinforcement learning algorithms [26]. While learning-based methods demonstrate impressive performance, they often lack collision avoidance guarantees and cannot ensure convergence to a reliable solution. Deep learning methods, such as multi-layer neural networks, incur a computational cost that scales with the number of parameters p, denoted as $\mathcal{O}(p)$ [27]. Given that modern neural networks frequently comprise millions of parameters, the inference process becomes the most time-consuming component of the control loop. In numerous robotic implementations, the vision-based neural network constitutes the most time-intensive aspect of the control loop [28]. However, employing neural networks to output control commands introduces substantial computation time to the control loop [23], [26].

While most global methods lack the ability to adapt to dynamic environments, this limitation can be overcome by employing local methods. Many local methods can be interpreted as nonlinear feedback controllers.

One of the early collision avoidance methods in robotics is Artificial Potential Fields (APF). APF represents each obstacle as a repulsive field through which the agent navigates [29]. The reactive nature of APF makes it suitable for dynamic environments, such as tracking moving targets [30]. APF has also been integrated with closed-feedback loops for real-time collision avoidance of robotic manipulators [31]. While APF is efficient and easy to implement, it is prone to local minima in multi-obstacle environments [32].

Navigation functions (NFs) were developed to address the issue of multiple local minima by constructing a potential field with a single minimum [33]. While initially designed for sphere worlds, NFs utilize diffeomorphic mappings between *star-worlds* and *sphere-worlds* to extend their applicability to more general scenarios [34]. These functions can also be employed to navigate around inverted obstacles that represent space boundaries, and their scope has been further expanded to include navigation around *trees-of-stars* [35]. However, achieving optimal performance with NFs requires compre-

hensive knowledge of the obstacle distribution, making the tuning of critical parameters a challenging task. Efforts have been made to enhance NFs by eliminating critical parameters in sphere worlds [36], and alternative approaches have been proposed to automate the tuning process [37]. NFs offer the advantage of quick re-evaluation, making them robust against disturbances. Nonetheless, due to the inherent difficulty in tuning them for general obstacle configurations, NFs pose challenges when applied in dynamic environments. Furthermore, as NFs rely on the gradient of the potential function to guide the system dynamics, they are unable to effectively track arbitrary nonlinear dynamics.

Vector fields (VF) employ highly nonlinear vector fields to navigate paths while ensuring collision avoidance. They have been successfully applied to evade multiple circular obstacles in two-dimensional space using local repulsive fields [38]. VF techniques have also been utilized for controlling fixed-wing aircraft [39] and extended to accommodate initially nonlinear dynamics, such as following a limit cycle [40]. Nonetheless, it involves switching between different dynamics, which can result in high accelerations. However, VF methods typically handle one obstacle at a time, imposing a conservative constraint where the influence regions of obstacles cannot overlap.

Harmonic potential functions (HPF) are intriguing because they guarantee the absence of topologically critical points in free space. While analytical HPFs are often elusive, numerical approximations have been employed [41]. Linear panel representations for obstacles enable the generation of closed-form HPFs [42]. Although this linear approximation can be extended to concave obstacles, its application is limited to two-dimensional environments [43].

Dynamical system modulation (DSM) has emerged as an effective approach for collision avoidance in reactive environments by redirecting initial dynamics away from obstacles [6]. To overcome challenges posed by intersecting, convex obstacles in three dimensions, DSM was extended to incorporate switching to surface following [44]. In our previous research, we successfully imitated the behavior of harmonic potential functions to ensure the convergence of DSM around concave (star-shaped) obstacles [2]. Furthermore, we expanded the scope of DSM to encompass indoor environments, facilitating reactive avoidance based on sensor data [28], [45]. While DSM traditionally assumes a zero-dimensional point, we achieved collision-free rigid-body dynamics by representing mobile agents through multiple control points [46]. However, it is important to note that DSM utilizes straight dynamics towards an attractor as an input, and the convergence guarantees are presently limited to star-shaped obstacles.

B. Contributions

This work significantly expands upon the reviewed DSM approach, enabling obstacle avoidance with nonlinear dynamical systems and for general concave obstacles. The paper's tehcnical contributions are outlined as follows:

- We introduce the Rotational Obstacle Avoidance Method (ROAM), which ensures local minima-free obstacle avoidance for nonlinear dynamics (Section III).
- We present a diffeomorphic mapping that guarantees convergence while maintaining proximity to the general nonlinear dynamics, while trying to preserve the initial flow (Section IV).
- We extend the ROAM formulation to handle obstacle avoidance of multiple dynamic obstacles and enclosed spaces, utilizing the concept of inverted obstacles as introduced in [33], [45] (Section V).
- Finally, we demonstrate the application of ROAM for avoiding general obstacles represented by trees-of-stars in the presence of nonlinear dynamics (Section VI).
- Additionally, we develop a method for vector rotation in general dimensions and the weighted summing of trees of vector rotations (Appendix A).

To validate the properties of ROAM, we conducted following evaluations:

- We performed a quantitative comparison of ROAM against two recent analytical obstacle avoidance methods through simulations, assessing convergence rate, similarity to unperturbed motion, and acceleration along the trajectories (Section VII).
- Furthermore, we conducted a qualitative evaluation of ROAM's application in controlling a 7DoF robot arm to avoid trees-of-stars in three dimensions (Section VII-D2).

The proposed rotation obstacle avoidance method (ROAM) ensures collision avoidance in a local minima-free vector field in the presence of initial nonlinear dynamics. Moreover, the influence regions of the obstacle can overlap, and the method has been extended to inverted obstacles, too, as well as multiple obstacles with overlapping regions of influence, as can be seen in the comparison with similar algorithms in Table I.

	RF [29]	NF [33]	MuMo [45]	VF-CAPF [40]	ROAM
Local minima free		✓	✓	(✓)	✓
Switching free	✓	✓	✓		✓
Overlapping regions	✓	✓	✓		✓
Nonlinear dynamics	✓		(✓)	✓	✓
Dynamic environments	✓		✓	✓	✓
(Optional) repulsion	✓				✓
Inverted obstacle	✓	✓	✓		✓
Trees of Obstacles	✓	✓			✓

TABLE I: Comparison of five different time-invariant algorithms across multiple criteria. All algorithms avoid solving an optimization problem.

II. Preliminaries

A. Notations

In this section, we establish the notations used throughout this paper. We consider a space of general dimension N as the underlying framework for our developments. Vectors are denoted using bold symbols, and the state variable is represented by $\boldsymbol{\xi} \in \mathbb{R}^N$. The time derivative of a vector $\boldsymbol{\xi}$ is denoted as $\dot{\boldsymbol{\xi}}$, assuming $\boldsymbol{\xi}$ is a function that is differentiable with respect to time. Unless explicitly stated otherwise, the paper employs the Euclidean norm, denoted as $|\cdot|$. The circular constant is denoted by π. The matrix $\boldsymbol{I}$ represents the identity matrix of appropriate dimensions. The symbol $\circ$ signifies

the iterative evaluation of a function. For example, when considering two functions $a(\cdot)$ and $b(\cdot)$, the expression $a \circ b(\boldsymbol{\xi})$ denotes the composition $a(b(\boldsymbol{\xi}))$. In the context of this paper, the term *vector field* (VF) refers to the velocity field $\dot{\boldsymbol{\xi}}$, while *dynamical system* (DS) specifically denotes the continuous-time feedback system described in Section II-B.

B. Dynamical Systems

Consider a vector field that governs the evolution of the position $\boldsymbol{\xi} \in \mathbb{R}^N$ of a continuous-time system, described by the state derivative $\dot{\boldsymbol{\xi}} \in \mathbb{R}^N$:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{1}$$

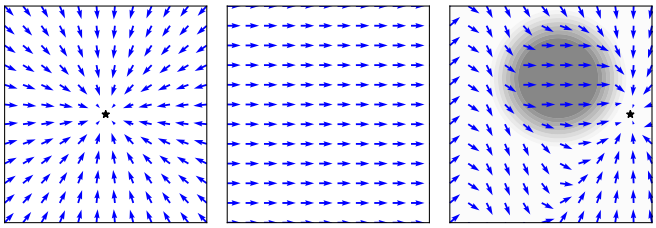
Let us define *straight dynamics*, that do maintain the direction along a trajectory:

Definition II.1 (Straight Dynamics)**.** A dynamical system of the form (1) is referred to as *straight dynamics* if, for all initial conditions $\boldsymbol{\xi}_0$ and $\dot{\boldsymbol{\xi}}_0$, the flow remains collinear with the initial velocity, satisfying $\dot{\boldsymbol{\xi}}_0^T \dot{\boldsymbol{\xi}}_t = \|\dot{\boldsymbol{\xi}}_0\| \|\dot{\boldsymbol{\xi}}_t\|$, for all $t \geq 0$. The flow is said to be *locally straight* if it is straight within a subdomain $\mathscr{X}^s$. In the special case where the attractor is positioned infinitely far away, hence $\exists \boldsymbol{v}^a \in \mathbb{R}^N : \lim_{\|\boldsymbol{\xi}-\boldsymbol{\xi}^a\| \to \infty} \dot{\boldsymbol{\xi}}_t^T \boldsymbol{v}^a = \|\dot{\boldsymbol{\xi}}_t^T\|$, hence the dynamics are called *(locally) collinear*.

Multiple straight dynamics are visualized in Figure 2. In the subsequent sections, we adopt the following parameterization for globally straight dynamics:

$$\boldsymbol{f}^s(\boldsymbol{\xi}) = q(\boldsymbol{\xi}) \left(\boldsymbol{\xi}^a - \boldsymbol{\xi}\right) \tag{2}$$

where $q : \mathbb{R}^N \to \mathbb{R} \setminus 0$ represents a continuous state-dependent scaling function that modulates the speed of the linear system $(\boldsymbol{\xi}^a - \boldsymbol{\xi})$ towards a fixed point $\boldsymbol{\xi}^a$. This scaling allows for the adjustment of the speed of the linear dynamics as the system approaches or moves away from the attractor $\boldsymbol{\xi}^a \in \mathbb{R}^N$, which serves as the unique stable fixed point or *attractor* of the system (Fig. 2a). Importantly, the condition $q(\boldsymbol{\xi}) \neq 0$, combined with the continuity of q, preserves the directionality of the flow. Consequently, the vector field consistently points towards or away from the attractor across the state space.



(a) Globally straight **(b)** Globally collinear **(c)** Locally straight

Fig. 2: Dynamical systems can exhibit different straightness characteristics. Such as globally straight with a single, stable attractor $\boldsymbol{\xi}^a$ visualized as a star (a). They can be globally collinear when the attractor is infinitely far away (b). Conversely, dynamics might be globally defined but only locally straight in a subdomain, visualized as the dark gray region (c).

C. Obstacle Description

In accordance with [2], each obstacle is characterized by a continuous distance function $\Gamma(\boldsymbol{\xi}) : \mathbb{R}^N \mapsto \mathbb{R}_{\geq 0}$. This function enables the differentiation of regions outside, on the boundary, or inside the obstacle based on the isolines of Γ.

$$\begin{aligned} &\text{Free space:} && \mathscr{X}^f = \{\boldsymbol{\xi} \in \mathbb{R}^N : \Gamma(\boldsymbol{\xi}) > 1\} \\ &\text{Boundary:} && \mathscr{X}^b = \{\boldsymbol{\xi} \in \mathbb{R}^N : \Gamma(\boldsymbol{\xi}) = 1\} \\ &\text{Interior set::} && \mathscr{X}^i = \{\boldsymbol{\xi} \in \mathbb{R}^N \setminus (\mathscr{X}^f \cup \mathscr{X}^b)\} \end{aligned} \tag{3}$$

1) Star-Shaped Obstacles: For representing star-shaped obstacles, we adopt the notation introduced in [47], [48]. An obstacle is considered star-shaped if there exists a point $\boldsymbol{\xi}^r$ with $\Gamma(\boldsymbol{\xi}^r) < 1$ such that every point on the boundary or in the interior set of the obstacle, i.e., $\boldsymbol{\xi} \in \mathscr{X}^i$ and $\mathscr{X}^i : \boldsymbol{\xi} \in \mathbb{R}^N, \Gamma(\boldsymbol{\xi}) \leq 1$, is connected to $\boldsymbol{\xi}^r$ by a line segment $l(\boldsymbol{\xi}^r, \boldsymbol{\xi})$ contained within $\mathscr{X}^i$.

The set of all such points $\boldsymbol{\xi}^r$ is referred to as the kernel of $\mathscr{X}^i$ and denoted as:

$$\ker(\mathscr{X}^i) = \left\{\boldsymbol{\xi}^r \in \mathscr{X}^i : l(\boldsymbol{\xi}^r, \boldsymbol{\xi}) \subset \mathscr{X}^i, \, \forall \boldsymbol{\xi} \in \mathscr{X}^i\right\} \tag{4}$$

Throughout this paper, a specific choice of $\boldsymbol{\xi}^r$ is referred to as the *reference point*, as commonly used in [2], [45], and represented by a cross symbol $+$. [1]

2) Distance Function: The distance function $\Gamma(\boldsymbol{\xi})$ increases monotonically in the radial direction, i.e., along the vector $\boldsymbol{\xi} - \boldsymbol{\xi}^r$, and has a continuous first-order partial derivative (C^1 smoothness). In this paper, we compute a distance function relative to one boundary point $\boldsymbol{\xi}^b \in \mathscr{X}^b$:

$$\Gamma(\boldsymbol{\xi}) = \|\boldsymbol{\xi} - \boldsymbol{\xi}^b\| / d_0 + 1 \quad \forall \boldsymbol{\xi} \in \mathscr{X}^e \tag{5}$$

where $d_0 \in \mathbb{R}_{>0}$ is a scaling factor, which modulates the obstacle influence. The boundary point $\boldsymbol{\xi}^b$ lies at the intersection between a ball of radius $R(\boldsymbol{\xi})$ around the reference point $\boldsymbol{\xi}^r$:

$$R(\boldsymbol{\xi}) = \|\boldsymbol{\xi}^b - \boldsymbol{\xi}^r\| \quad \boldsymbol{\xi}^b = b\boldsymbol{r}(\boldsymbol{\xi}) + \boldsymbol{\xi}^r \,, \; b > 0 \,, \; \boldsymbol{\xi}^b \in \mathscr{X}^b \tag{6}$$

and the *reference direction* vector $\boldsymbol{r} : \mathbb{R}^N \setminus \boldsymbol{\xi}^r \to \{\boldsymbol{r} \in \mathbb{R}^N : \|\boldsymbol{r}\| = 1\}$:

$$\boldsymbol{r}(\boldsymbol{\xi}) = \left(\boldsymbol{\xi} - \boldsymbol{\xi}^r\right) / \|\boldsymbol{\xi} - \boldsymbol{\xi}^r\|. \tag{7}$$

The boundary point is hence a function of the state, $\boldsymbol{\xi}^b(\boldsymbol{\xi})$.

Further, the normal direction can be defined as the derivative of the distance across space:

$$\boldsymbol{n}(\boldsymbol{\xi}) = d\Gamma(\boldsymbol{\xi}) / d\boldsymbol{\xi} \tag{8}$$

D. Rotated Vector Field

We present a construct that is fundamental to the obstacle avoidance approach proposed here, inspired by the concept of stereographic projection [45]. [2] Consider a unit vector with respect to a basis vector $\boldsymbol{b}$, denoted as $\boldsymbol{k}(\boldsymbol{b}, \boldsymbol{\xi}) : \mathbb{R}_I^N \times \mathbb{R}_I^N \to \mathbb{R}^{N-1}$, under the condition that $\boldsymbol{\xi} : \langle -\boldsymbol{b}, \boldsymbol{\xi} \rangle \neq -1$. This vector extraction captures the rotation of $\boldsymbol{\xi}$ with respect to $\boldsymbol{b}$ and

[1] The reference point is in literature sometimes denoted *kernel point*.

[2] A stereographic projection maps points located on a sphere onto a plane perpendicular to the sphere surface.

has proven to be useful for minima-free vector-summing. The relative rotation satisfies the following properties:

$$\cos\left(\|\boldsymbol{k}(\boldsymbol{b},\boldsymbol{\xi})\|\right) = \langle \boldsymbol{b},\boldsymbol{\xi}\rangle \quad \text{and} \quad \|\boldsymbol{k}(\boldsymbol{b},\boldsymbol{\xi})\| < \pi \tag{9}$$

The basis vector $\boldsymbol{b}$ is utilized as the first column to construct the orthonormal transformation matrix $\boldsymbol{B}$, enabling the transformation into a new basis: $\hat{\boldsymbol{\xi}} = \boldsymbol{B}^T\boldsymbol{\xi}$. In the direction space, the magnitude corresponds to the angle between the original vector and the reference vector. The transformation of the initial vector $\boldsymbol{\xi}$ in the direction space is given by:

$$\boldsymbol{k}(\boldsymbol{b},\boldsymbol{\xi}) = \begin{cases} \arccos\left(\hat{\boldsymbol{\xi}}_{[1]}\right)\hat{\boldsymbol{\xi}}_{[2:]}/\|\hat{\boldsymbol{\xi}}_{[2:]}\| & \text{if } \hat{\boldsymbol{\xi}}_{[1]} \neq 1 \\ \boldsymbol{0} & \text{otherwise} \end{cases} \tag{10}$$

Note that a vector $\boldsymbol{\xi}$, which is anti-collinear to $\boldsymbol{b}$, does not have a unique transformation and is hence excluded; see [45] for further discussion. Correspondingly, we use $\bar{\boldsymbol{k}}(\boldsymbol{b},\bar{\boldsymbol{\xi}}) : \mathbb{R}^N_I \times \mathbb{R}^{N-1} \to \mathbb{R}^N_I$ to describe the inverse mapping, such that:

$$\boldsymbol{\xi} = \bar{\boldsymbol{k}}(\boldsymbol{b},\bar{\boldsymbol{\xi}}) \quad \text{with} \quad \bar{\boldsymbol{\xi}} = \boldsymbol{k}(\boldsymbol{b},\boldsymbol{\xi}) \tag{11}$$

The mapping to the original space is evaluated as follows:

$$\bar{\boldsymbol{k}}(\boldsymbol{b},\bar{\boldsymbol{\xi}}) = \begin{cases} \boldsymbol{B}\begin{bmatrix}1 & 0 & .. & 0\end{bmatrix}^T & \text{if } \|\bar{\boldsymbol{\xi}}\| = 0 \\ \boldsymbol{B}\begin{bmatrix}\cos(\|\bar{\boldsymbol{\xi}}\|) & \sin(\|\bar{\boldsymbol{\xi}}\|)\bar{\boldsymbol{\xi}}/\|\bar{\boldsymbol{\xi}}\|\end{bmatrix}^T & \text{otherwise} \end{cases} \tag{12}$$

It is important to note that the reverse mapping $\bar{\boldsymbol{k}}(\cdot)$ is defined for angles larger than π, but the function is not bijective for these values.

E. Problem Statement

We establish the following requirements for our obstacle avoidance controller:

- **Collision free:** The flow must remain outside the obstacles at all times, i.e., $\{\boldsymbol{\xi}\}_t \in \mathscr{X}^e \; \forall t$ if $\{\boldsymbol{\xi}\}_0 \in \mathscr{X}^e$.
- **State dependent:** The dynamics are history-invariant and depend solely on the current state, i.e. $\dot{\boldsymbol{\xi}}_t = \boldsymbol{f}(\boldsymbol{\xi}_t,\boldsymbol{\xi}_{t-1},..,\boldsymbol{\xi}_0) = \boldsymbol{f}(\boldsymbol{\xi}_t)$.
- **Local minima free:** As demonstrated in [40], each C^1-smooth vector field obstacle introduces at least one stationary point on the surface. However, it must be ensured that (1) this point is a saddle point and not a minimum and (2) there are no additional stationary points in space.
- **Limited and smooth dynamics;** The magnitude of the vector field is upper bounded, i..e., $\exists \boldsymbol{v}^{\max} = \text{const.} : \|\dot{\boldsymbol{\xi}}\| < \boldsymbol{v}^{\max}$. Additionally, the vector field is smooth, meaning that small displacements result in proportionally small velocities: $\lim_{\boldsymbol{\xi}_1 \to \boldsymbol{\xi}_2} \|\dot{\boldsymbol{\xi}}_2 - \dot{\boldsymbol{\xi}}_1\| = 0$
- **General dimensions:** The obstacle avoidance algorithm is applicable in a space of dimensions $N \geq 2$.

Furthermore, we assume that the environment and initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$, as described in (1), possess the following properties:

- **Star-shaped:** All obstacles are star-shaped as defined in Sec. II-C1, or are composed of obstacles (trees-of-stars) with a nonzero intersection regions among the components of a tree, as discussed in Section II-C1.
- **Limited and smooth dynamics:** Analogously to the output, the magnitude of the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ is required to be C^1-smooth and limited, i.e., $\|\boldsymbol{f}(\boldsymbol{\xi})\| < v^{\max}, \; \forall \boldsymbol{\xi} \in \mathbb{R}^N, v^{\max} \in \mathbb{R}_{>0}$.
- **Dynamics as a vector rotation:** The initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ can be evaluated as a local rotation (Section II-D) or a sequence of rotations (Appendix A) of globally straight dynamics with respect to a fixed point $\boldsymbol{\xi}^a$ (Definition II.1).

III. Obstacle Avoidance through Rotation

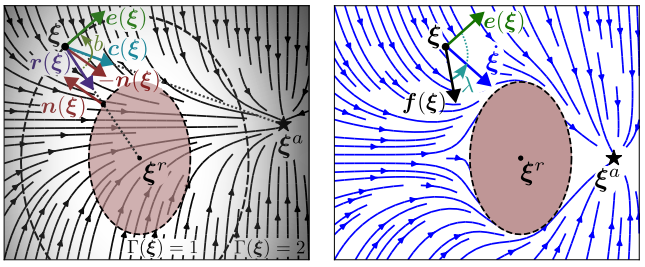
A smooth vector field that effectively avoids an obstacle should have the velocity $\dot{\boldsymbol{\xi}}$ directed away from or perpendicular to the normal $\boldsymbol{n}(\boldsymbol{\xi}) \in \mathbb{R}^N$ as defined in (8). This principle is commonly expressed as follows [6], [43]:

$$\left\langle \boldsymbol{n}(\boldsymbol{\xi}), \dot{\boldsymbol{\xi}} \right\rangle \geq 0 \quad \forall \boldsymbol{\xi} \in \mathscr{X}^b \tag{13}$$

A. Vector Rotation for Collision Avoidance

We propose a method called Rotational Obstacle Avoidance Method (ROAM) to achieve collision-free motion. ROAM smoothly adjusts the initial velocity $\boldsymbol{f}(\boldsymbol{\xi})$ as given in Eq. (1), rotating it towards a feasible half-space as the position approaches the obstacle (Fig. 3). The steps involved in ROAM for avoiding a single obstacle are as follows:

1) Evaluation of pseudo-tangent direction $\boldsymbol{e}(\boldsymbol{\xi})$ using the convergence direction $\boldsymbol{c}(\boldsymbol{\xi})$ (Sec. III-A1)
2) Rotation of the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ towards the pseudo-tangent $\boldsymbol{e}(\boldsymbol{\xi})$ to obtain the collision-free direction $\dot{\boldsymbol{\xi}}$ (Sec. III-A2).
3) Evaluation of the velocity magnitude $h(\boldsymbol{\xi})$ to ensure a smooth vector field (Sec. III-A3)



(a) Initial dynamics **(b)** Rotated system

Fig. 3: A nonlinear dynamical system $\boldsymbol{f}(\boldsymbol{\xi}) = \text{diag}(1 \;\; \Delta\boldsymbol{\xi}_{[2]})\|\Delta\boldsymbol{\xi}\|$ with $\Delta\boldsymbol{\xi} = \boldsymbol{\xi}^a - \boldsymbol{\xi}$ (a) is rotated using ROAM to ensure convergence towards the attractor $\boldsymbol{\xi}^a$ with a single saddle point on the surface (b).

1) Evaluation of Preferred Tangent Direction: At each position on the surface of an obstacle, there can exist multiple tangent directions. In the case of $d = 2$, there are exactly two tangent directions, while for $d \geq 3$, there are infinitely many. To construct a smoothly defined pseudo-tangent $\boldsymbol{e}(\xi)$, we introduce the concept of convergence dynamics, which are used to obtain the pseudo-tangent:

Definition III.1 (Convergence Dynamics)**.** The C^1-smooth convergence-dynamics $\boldsymbol{c} : \mathbb{R}^N \to \mathbb{R}^N$ guides the

initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ around obstacles. The convergence dynamics are locally straight according to Definition II.1, on the surface of the obstacle $\mathscr{X}^b$ as defined in (3) Furthermore, the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ is set to never be anti-collinear to the initial dynamics, i.e., $\langle \boldsymbol{f}(\boldsymbol{\xi}), \boldsymbol{c}(\boldsymbol{\xi}) \rangle \neq -\|\boldsymbol{f}(\boldsymbol{\xi})\| \, \|\boldsymbol{c}(\boldsymbol{\xi})\| \; \forall \boldsymbol{\xi} \in \mathbb{R}^N$.

In this section, we utilize convergence dynamics of the form $\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{\xi} - \boldsymbol{\xi}^a$. For more general convergence dynamics, please refer to Section IV.

The desired pseudo-tangent $\boldsymbol{e}(\boldsymbol{\xi})$ is obtained by rotating the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ away from the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ until it lies in the tangent plane (Fig. 3). Since, in the angular space $\boldsymbol{k}(\boldsymbol{n}, \cdot)$, given in (9), any tangent vector lies at a distance $R^e = \pi/2$ to the normal direction, the tangent hyper-plane $\mathscr{T}$ forms a hyper-sphere in the direction space, as visualized in Figure 4. Hence, the rotation of the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ away from reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ can be evaluated by intersecting the line connecting $\boldsymbol{r}(\boldsymbol{\xi})$ and $\boldsymbol{c}(\boldsymbol{\xi})$ with a circle of radius $R^e \in [\pi/2, \pi]$, see Fig. 4. Thus, $\boldsymbol{e}(\boldsymbol{\xi})$ can be obtained through the following constraints[3]:

$$\begin{aligned} &\text{if} \quad \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c})\| \geq R^e \quad \text{then} \quad \boldsymbol{e} = \boldsymbol{c} \quad \text{otherwise} \\ &\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}) = (1-b)\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) + b\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c}) \\ &\text{such that} \quad \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e})\| = R^e, \quad b \in \mathbb{R}_{>0} \;\; \forall \boldsymbol{\xi} : \boldsymbol{c} \neq \boldsymbol{r} \end{aligned} \tag{14}$$

The solution to the above equality constraints is obtained analytically by solving a quadratic equation with respect to the scalar b.

As a result, the pseudo-tangent $\boldsymbol{e}(\xi)$ can either lie in the tangent plane $\mathscr{T}$ or point away from the obstacle, with a distance to the normal direction contained within the green region depicted in Figure 4. In the special case where $\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{r}(\boldsymbol{\xi})$, the intersection of the hyper-circle in Equation 14 does yield a solution. However, it is ensured that the final vector field $\dot{\boldsymbol{\xi}}$ is continuously defined, as discussed in Section III-A2.

Lemma III.1. *Let us assume the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$, as given in Definition III.1. The pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi}) \in \mathbb{R}^N$ obtained through* (14) *is smooth and satisfies the boundary inequality $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{e}(\boldsymbol{\xi}) \rangle \geq 0$, stated in* (13)*, at any position on the surface of the obstacle but the saddle point, i.e., $\boldsymbol{\xi} \in \mathscr{X}^b : \langle \boldsymbol{c}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle \neq -1\}$*

Proof. For a starshaped obstacle, we have by definition of the starshaped kernel space in (4), that $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{r}(\boldsymbol{\xi}) \rangle < 0$. Hence, $\|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r})\| < \pi/2$, i.e., $\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r})$ is strictly inside the hyper-sphere $\mathscr{T}$. Furthermore, as long as $\boldsymbol{c}(\boldsymbol{\xi}) \neq \boldsymbol{r}(\boldsymbol{\xi})$, the equality of finding the intersection of a line and a hyper-sphere from (14) has exactly one solution with $b > 0$.

Concerning the boundary condition; looking at the case of $\|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c})\| > R^e$. From the definition of the direction-space in (9), it follows that:

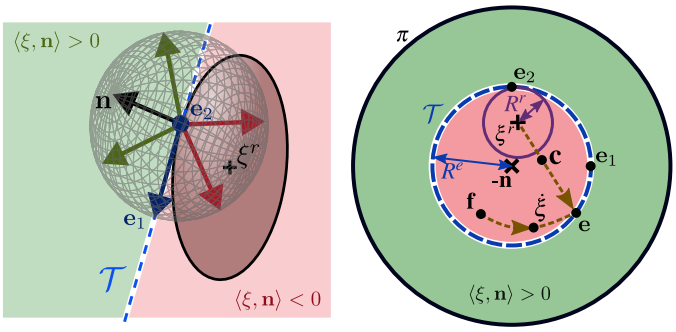
$$\langle \boldsymbol{c}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle > 0 \underset{\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{e}(\boldsymbol{\xi})}{\Rightarrow} \langle \boldsymbol{e}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle > 0 \tag{15}$$

[3]For conciseness dependency on $\boldsymbol{\xi}$ is omitted.

For the case that $\langle \boldsymbol{c}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi}) \rangle < 0$, from (9) and (14) we can conclude:

$$\begin{aligned} \langle \boldsymbol{n}, \boldsymbol{e} \rangle &= \cos\left(\|\boldsymbol{k}(\boldsymbol{n}, \boldsymbol{e})\|\right) = -\cos\left(\|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e})\|\right) \\ &\geq -\cos\left(R^e\right) \geq -\cos\left(\pi/2\right) = 0 \end{aligned} \tag{16}$$

Hence, we have a uniquely defined pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$, which satisfies the boundary condition given in (13). □



(a) Euclidean space **(b)** Directional space $\boldsymbol{k}(-\boldsymbol{n}, \cdot)$

Fig. 4: When approaching the surface of the obstacle, the collision-free vectors are located on one side of the tangent plane $\mathscr{T}$ formed by the linearly independent tangent vectors $\boldsymbol{e}_{(\cdot)}$ (a). These unit vectors can be projected onto a circular hypersphere of dimension $N-1$, where the collision-free pseudo tangent $\boldsymbol{e}$ is calculated (b). Any vector $\boldsymbol{\xi}$ pointing towards the obstacle, i.e., $\langle \boldsymbol{n}, \boldsymbol{\xi} \rangle < 0$ (a), is projected to lie inside the circle of radius $\pi/2$ (b), while a vector pointing away from the obstacle is projected to the outer ring.

2) Rotation Towards Tangent Direction: In a second step, the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ is *rotated* towards the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$. This rotation operation is performed in the direction space, as described in Section II-D, but this time with respect to the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$:

$$\begin{aligned} &\dot{\boldsymbol{\xi}} = \bar{\boldsymbol{k}}\Big(\boldsymbol{c}, \big(1 - \lambda(\boldsymbol{\xi})\big)\boldsymbol{k}(\boldsymbol{c}, \boldsymbol{f}) + \lambda(\boldsymbol{\xi})\boldsymbol{k}(\boldsymbol{c}, \boldsymbol{e})\Big) \quad \text{with} \\ &\lambda(\boldsymbol{\xi}) = \left(\frac{1}{\Gamma(\boldsymbol{\xi})}\right)^q, \;\; R^r = \min\left(R^e - \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r})\|, \frac{\pi}{2}\right) \\ &q = \max\left(1, \frac{R^r}{\Delta\boldsymbol{k}(\boldsymbol{c})}\right)^s, \quad \Delta\boldsymbol{k}(\boldsymbol{c}) = \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) - \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{c})\| \end{aligned} \tag{17}$$

The rotation weight $\lambda(\boldsymbol{\xi}) \in [0, 1]$ is determined based on the inverse of the distance Γ to ensure that the rotation has a decreasing influence as the distance increases. This allows for a smooth transition in the avoidance behavior. Additionally, the smoothing factor q is introduced, which gradually decreases to zero when the convergence dynamics point towards the robot. This effectively cancels the rotation avoidance effect in regions where the tangent $\boldsymbol{e}(\boldsymbol{\xi})$ is not defined, resulting in a smooth avoidance behavior across those positions. The impact of the smoothness constant $s \in \mathbb{R}_{>0}$ can be observed in Figure 5, and unless otherwise specified, we use $s = 0.3$.

Lemma III.2. *The vector field $\dot{\boldsymbol{\xi}}$ obtained through rotation as given in* (17) *satisfies the boundary condition as defined in* (13) *if the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ is tangent or pointing away from the obstacle, i.e., $\langle \boldsymbol{e}, \boldsymbol{n} \rangle \geq 0, \; \forall \boldsymbol{\xi} : \Delta\boldsymbol{k}(\boldsymbol{c}) \neq 0$.*

Proof. When approaching the obstacle, the rotated velocity $\dot{\boldsymbol{\xi}}$ is evaluated as:

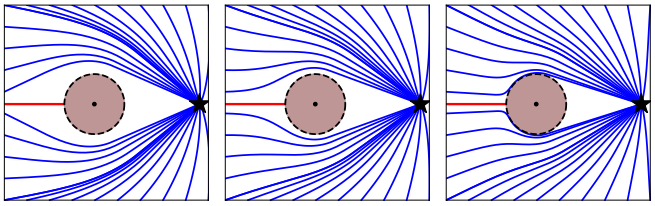
$$\lim_{\Gamma(\boldsymbol{\xi})\to 1,} \lambda(\boldsymbol{\xi}) = 1 \quad \underset{\text{with (11)}}{\Rightarrow} \quad \lim_{\lambda\to 1} \left\langle \boldsymbol{n}, \dot{\boldsymbol{\xi}} \right\rangle = \langle \boldsymbol{n}, \boldsymbol{e} \rangle \geq 0 \tag{18}$$

using Lemma III.1 and that $\Delta\boldsymbol{k}(\boldsymbol{c}) \neq 0 \Rightarrow q > 0$ with the smoothness factor q given in (17).

Following the same logic, we can also analyze the behavior far away from obstacles:

$$\lim_{\Gamma(\boldsymbol{\xi})\to\infty} \lambda(\boldsymbol{\xi}) = 0 \;\Rightarrow\; \lim_{\lambda\to 0} \dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{19}$$

□



(a) s =1e-6 **(b)** $s = 0.3$ **(c)** $s = 1.0$

Fig. 5: A smaller smoothness-constant s increases the reactivity when approaching an obstacle (a), while a larger value ensures a smoother transition across the (red) saddle point trajectory (b, c).

3) Evaluation of Speed: The directional space mapping $\boldsymbol{k}(\cdot)$ and its inverse mapping $\bar{\boldsymbol{k}}(\cdot)$ operate on unit vectors in the original space. As a result, the algorithm in (17) modifies the direction of the initial dynamics, rather than its magnitude. To incorporate magnitude control in a decoupled manner, we introduce an additional component using $h(\boldsymbol{\xi}) : \mathbb{R}^N \to [0,1]$:

$$\|\dot{\boldsymbol{\xi}}\| = h(\boldsymbol{\xi}) \|\boldsymbol{f}(\boldsymbol{\xi})\| \tag{20}$$

The stretching factor $h(\boldsymbol{\xi})$ is designed to slow down when pointing towards an obstacle, and the effect decreases with increasing distance from the obstacle:

$$h(\boldsymbol{\xi}) = \min\left(1, \left(\frac{\|\Delta\boldsymbol{k}(\boldsymbol{c})\|}{R^r}\right)^2 + \left(1 - \frac{1}{\Gamma(\boldsymbol{\xi})}\right)^2\right) \tag{21}$$

where the reference radius R^r and the rotation space distance $\Delta\boldsymbol{k}(\boldsymbol{c})$ are given in (17). In Figure 6, the velocity scaling can be observed to decrease the magnitude of the vectors pointing toward the obstacle.

Theorem III.3. *Consider a vector field $\dot{\boldsymbol{\xi}} \in \mathbb{R}^N$ obtained after a local rotation as defined in* (17) *of an initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ and fixed point at $\boldsymbol{\xi}^a$, with pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ defined in* (14) *with respect to initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$, is locally straight on the surface of the obstacle $\mathscr{X}^b$ according to Definition II.1, and motion scaling $h(\boldsymbol{\xi})$ according to* (21)*. Any motion starting in free space $\{\boldsymbol{\xi}\}_0 \in \mathscr{X}^e$ which evolves according to $\dot{\boldsymbol{\xi}}$ will stay in free space for finite time $\{\boldsymbol{\xi}\}_t \in \mathscr{X}^e$ with $t \in \mathbb{N}_{>0}$ and maintains the stationary point, i.e., $\dot{\boldsymbol{\xi}} = \boldsymbol{0}$ if $\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{0}$*

Proof. From Lemma III.1, we know $\langle \boldsymbol{n}(\boldsymbol{\xi}), \boldsymbol{e}(\boldsymbol{\xi}) \rangle \geq 0$ and Lemma III.2 states that $\langle \boldsymbol{e}, \boldsymbol{n} \rangle \geq 0 \;\; \forall \boldsymbol{\xi} : \Delta\boldsymbol{k}(\boldsymbol{c}) \neq 0$. Additionally, in the latter case, the magnitude scaling from (21) evaluates as

$$\lim_{\|\Delta\boldsymbol{k}(\boldsymbol{c})\|\to 0, \Gamma(\boldsymbol{\xi})\to 1} h(\boldsymbol{\xi}) = 0 \tag{22}$$

Hence, the speed reaches smoothly zero as we approach the saddle point, and its direction does not matter to fulfill smoothness and the boundary condition given in (13).

Furthermore, far away from the obstacle, we have:

$$\lim_{\Gamma(\boldsymbol{\xi})\to\infty} h(\boldsymbol{\xi}) = 1 \tag{23}$$

Hence, the magnitude is equal to the original magnitude and only vanishes around existing stationary points, i.e., $\boldsymbol{f}(\boldsymbol{\xi}) = 0$. □

The spurious stationary point on the surface is given by $\{\boldsymbol{\xi} : \boldsymbol{\xi} \in \mathscr{X}^b, \|\boldsymbol{k}(\boldsymbol{c})\| = 0\}$. Furthermore, by construction, the pseudo tangent $\boldsymbol{e}(\boldsymbol{\xi})$ defined in (14) points away from the reference direction and hence points away from the stationary point on the surface of the obstacle. Hence, it is a saddle point with a single trajectory converging to it $\mathscr{X}^s$.

B. Surface Repulsion

n the vicinity of critical obstacles, it may be desirable to incorporate active repulsion from them without significantly altering their shape. This can be achieved by introducing a behavior similar to artificial potential fields [35], but without introducing spurious attractors in free space. Building upon the developments presented in this section, we can further refine the tangential radius. By increasing the tangential radius such that $R^e \in]\pi/2, \pi]$, values larger than $\pi/2$, leads to a repulsive behavior while maintaining the avoidance properties (Fig. 6).

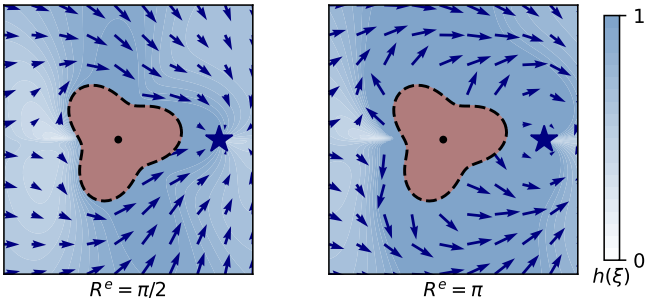


Fig. 6: While a surface repulsion radius of $R^e = \pi/2$ ensures collision avoidance, an increased radius, i.e., $R^e \in]\pi/2, \pi]$ leads to increased repulsion around the obstacle. The initial dynamics are given by $\boldsymbol{f}(\boldsymbol{\xi}) = (\boldsymbol{\xi} - \boldsymbol{\xi}^a)$.

C. Inverted Obstacles

Initial dynamics might be contained within a boundary, such as an agent moving in a room or a robot staying within its joint limits. These constraints can be incorporated by inverting an obstacle and ensuring that the dynamics remain inside a boundary hull (Fig. 7). Analogously to [45], this is achieved by inverting the distance function $\Gamma(\boldsymbol{\xi})$, the reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ and normal vector $\boldsymbol{n}(\boldsymbol{\xi})$, defined in Sec. II. The distance function divides the space into free, boundary, and interior points, as described in (3). Consequently, the distance for the boundary obstacle can be defined as follows:

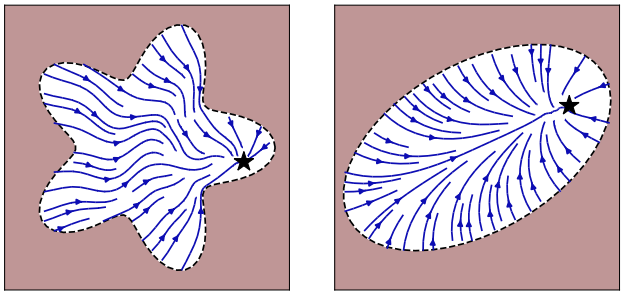
$$\Gamma(\boldsymbol{\xi}) = \left(R(\boldsymbol{\xi}) / \|\boldsymbol{\xi} - \boldsymbol{\xi}^r\|\right)^{2p} \quad \forall \boldsymbol{\xi} \in \mathbb{R}^N \setminus \boldsymbol{\xi}^r \tag{24}$$

with power weight $p \in \mathbb{R}_{>0}$, we choose $p = 1$.

As the normal direction points away from the surface, it naturally points towards the interior of an inverted obstacle, in contrast to the normal direction of a regular obstacle. Consequently, in order to utilize the star-shaped constraint from (17) of $\langle -\boldsymbol{r}(\boldsymbol{\xi}), \boldsymbol{n}(\boldsymbol{\xi})\rangle > 0$ for rotational obstacle avoidance, we need to flip the reference direction:

$$\boldsymbol{r}(\boldsymbol{\xi}) = (\boldsymbol{\xi}^r - \boldsymbol{\xi})/\|\boldsymbol{\xi}^r - \boldsymbol{\xi}\| \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^N \setminus \boldsymbol{\xi}^r \tag{25}$$

Using the *inverted* values, the ROAM can be applied on as described throughout this section (Fig. 7).



(a) Wavy DS in starshape **(b)** Linear DS in ellipse

Fig. 7: The inverted obstacle description is used to keep nonlinear dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ within star-shaped boundaries, in (a) with $\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{R}(\sin(\|\boldsymbol{\xi}^a - \boldsymbol{\xi}\|)(\boldsymbol{\xi}^a - \boldsymbol{\xi})$ where $\boldsymbol{R}(\cdot)$ is a two dimensional rotation matrix and $\boldsymbol{\xi}^a$ the attractor. Further, it can be used to create active repulsion from walls, as in (b) for globall straight dynamics with attractor $\boldsymbol{\xi}^a$.

Lemma III.4. *The rotated dynamics $\dot{\boldsymbol{\xi}}$ evaluated according to* (17) *are C^1-smooth and collision-free as shown in Theorem III.3 when navigating within a boundary described as an inverted obstacle with distance function $\Gamma(\boldsymbol{\xi})$ as defined in* (24)*, and reference vector $\boldsymbol{r}(\boldsymbol{\xi})$ as given in* (25)*.*

Proof. Since the distance function $\Gamma(\boldsymbol{\xi})$, average vector $\boldsymbol{n}(\boldsymbol{\xi})$, and reference direction $\boldsymbol{r}(\boldsymbol{\xi})$ possess all the necessary properties outlined in Theorem III.3, the collision avoidance properties directly carry over.

The distance function $\Gamma(\boldsymbol{\xi})$ from (24) and normal direction (25) are not defined at the reference point $\boldsymbol{\xi}^r$. However, at this point the rotation weight $\lambda(\boldsymbol{\xi})$ reaches zeros:

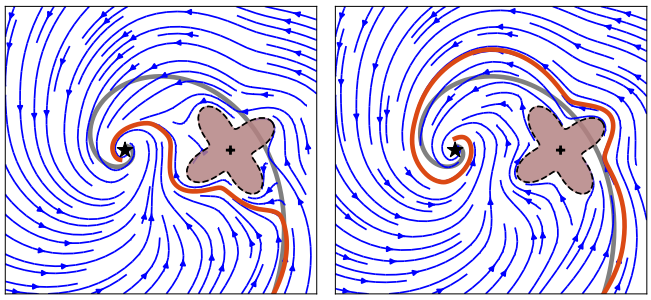
$$\lim_{\xi \to \xi^r} \Gamma(\boldsymbol{\xi}) \to \infty \underset{\text{using (17)}}{\Rightarrow} \lim_{\xi \to \xi^r} \lambda(\boldsymbol{\xi}) = 0 \;\Rightarrow\; \lim_{\xi \to \xi^r} \dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{26}$$

Thus, the rotation has no effect, and the system is smoothly defined. □

A more detailed analysis of inverted obstacles can be found in [45]. Further development applies to both standard and inverted obstacles if not stated otherwise.

IV. General Nonlinear Motion

For systems characterized by small nonlinearities and a single attractor, the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ exhibit global straightness, as defined in Definition II.1, resulting in desirable behavior. However, in systems with high nonlinearities, this can lead to avoidance patterns that do not accurately reflect the global dynamics, as illustrated by the example shown in Figure 8a. To address this issue, this section introduces modified convergence dynamics that aim to maintain similarity with the original dynamics while being locally straight, as depicted in Figure 8b.



(a) Globally straight $\boldsymbol{c}(\boldsymbol{\xi})$ **(b)** Locally straight $\boldsymbol{c}(\boldsymbol{\xi})$

Fig. 8: For motion with initial dynamics of $\boldsymbol{f}(\boldsymbol{\xi}) = \begin{bmatrix} -1 & 2 \\ -2 & -1 \end{bmatrix} \boldsymbol{\xi}$ (gray) and a single stationary point (black star), employing a globally straight convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ results in a trajectory (orange) that deviates significantly from the original motion (a). In contrast, by utilizing locally straight convergence dynamics, the trajectory (b) maintains similarity to the original motion throughout its course.

A. Nonlinear Motion without Stationary Point

Let us first focus on initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ which do not have any directional singularity point, i.e., $\nexists \boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0$. The convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ of such initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ is constructed by evaluating as a weighted sum of the initial velocity at the reference point of the obstacle $\boldsymbol{\xi}^r$ and at position $\boldsymbol{\xi}$:

$$\boldsymbol{c}(\boldsymbol{\xi}) = w^c(\boldsymbol{\xi})\boldsymbol{f}(\boldsymbol{\xi}^r) \hat{+} (1 - w^c(\boldsymbol{\xi}))\boldsymbol{f}(\boldsymbol{\xi}) \tag{27}$$

The convergence weight is chosen such that the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ is straight on the surface of the obstacle:

$$w^c(\boldsymbol{\xi}) = \begin{cases} 1 & \text{if } \Gamma(\boldsymbol{\xi}) < 1 \\ 1/\Gamma(\boldsymbol{\xi}) & \text{otherwise} \end{cases} \tag{28}$$

where $\hat{+}$ describes the rotational summing described in Appendix A. Note that the rotation summing from (71) is not defined for two anti-collinear vectors. Hence, we cannot apply this summing at a directional singularity point, as will be further discussed in the next subsection.

Lemma IV.1. *The convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi}) : \mathbb{R}^N \to \mathbb{R}^N$ as proposed in* (27) *for initial dynamics without directional singularity points, i.e., $\{\boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0\} = \emptyset$, are straight according to Definition II.1 on the surface and inside the obstacle, i.e, $\boldsymbol{c}(\boldsymbol{\xi}) \in \mathscr{F}^s$, $\boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i$.*

Proof. Inside and on the surface of the obstacle, we have:

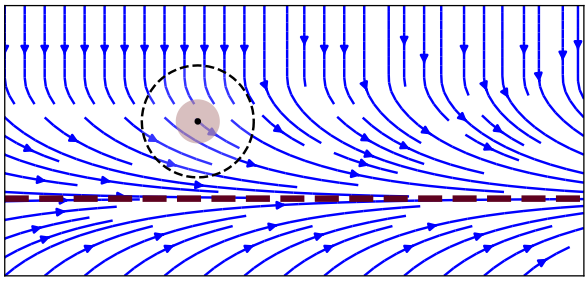
$$\boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i \quad \Rightarrow \quad \Gamma(\boldsymbol{\xi}) \leq 1 \quad \Rightarrow \quad w^c(\boldsymbol{\xi}) = 1 \tag{29}$$

Hence the dynamics in this region are given as:

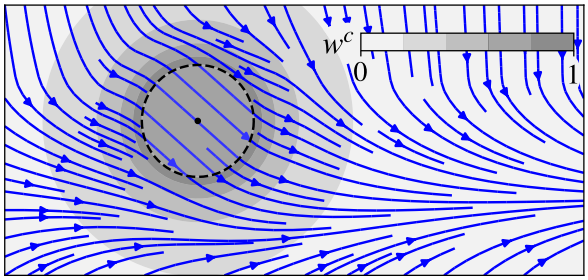
$$\boldsymbol{c}(\boldsymbol{\xi}) = 1\boldsymbol{f}(\boldsymbol{\xi}^r) \hat{+} 0\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}^r) \qquad \boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i \tag{30}$$

Thus, we have locally collinear dynamics. □

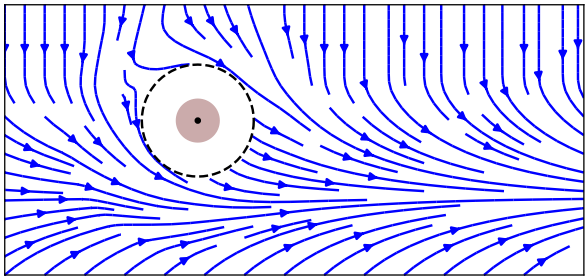
As the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ are straight on the surface of the obstacle, they can be used in the rotational obstacle avoidance method defined in Section III, see Figure 9.



(a) Nonlinear dynamics to follow the line at $y = 0$ in red



(b) Convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ are locally straight in the gray subset



(c) Rotated avoidance without local minima

Fig. 9: The line following dynamics $\boldsymbol{f}(\boldsymbol{\xi}) = [1 \;\; -\boldsymbol{\xi}_2]^T$ shown in (a) uses locally straight convergence dynamics (b) to ensure the absence of local minima when avoiding the obstacle (c).

B. Nonlinear Motion in the Presence of a Stationary Point

Many initial dynamics can be characterized by a motion with a single directional singularity point, i.e., $\{\boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla \boldsymbol{f}(\boldsymbol{\xi}) \neq 0\} = \{\boldsymbol{\xi}^a\}$. This singularity point may correspond to a desired attractor point or arise from a limit cycle where an unstable stationary point resides at its center. In the presence of such a singularity point, the directional summing approach employed in (27) cannot be directly applied. This is because we cannot smoothly define local rotations around $\boldsymbol{\xi}^a$. However, we can overcome this limitation by *unfolding* the space using a sequence of mappings (Fig. 10).

1) Shrinking and Inverse Mapping: The first step is to shrink the obstacles to a single point, i.e., all boundary points of the obstacle $\mathscr{X}^b$ are mapped to the reference point $\boldsymbol{\xi}^r$.

Definition IV.1 (Shrinking Mapping)**.** The shrinking mapping $\boldsymbol{m}^s(\boldsymbol{\xi}) : \mathscr{X}^e \to \mathbb{R}^N \setminus \boldsymbol{\xi}^r$ defined as

$$\boldsymbol{m}^s(\boldsymbol{\xi}) = \boldsymbol{\xi}^r + \boldsymbol{r}(\boldsymbol{\xi}) \left(\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\| - \|\boldsymbol{\xi}^b - \boldsymbol{\xi}^r\| \right) \;\; \forall \boldsymbol{\xi} \in \mathscr{X}^e \quad (31)$$

is a bijection, and it maps the point on the obstacle's surface to the obstacle's reference direction, i.e., $\lim_{\Gamma(\boldsymbol{\xi}) \to 1} \boldsymbol{m}^s(\boldsymbol{\xi}) \to \boldsymbol{\xi}^r$.

Analogously, we define the inverse of this mapping:

Definition IV.2 (Inflating Mapping)**.** The inflating mapping $\boldsymbol{m}^i(\boldsymbol{\xi}) : \mathbb{R}^N \setminus \boldsymbol{\xi}^r \to \mathscr{X}^e$ defined as:

$$\boldsymbol{m}^i(\boldsymbol{\xi}) = \boldsymbol{\xi}^r + \boldsymbol{r}(\boldsymbol{\xi}) \left(\|\boldsymbol{\xi} - \boldsymbol{\xi}^r\| + \|\boldsymbol{\xi}^b - \boldsymbol{\xi}^r\| \right) \;\; \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^r \quad (32)$$

is a bijection and the inverse function of the shrinking mapping defined in (31), i.e., $\boldsymbol{m}^i \circ \boldsymbol{m}^s(\boldsymbol{\xi}) = \boldsymbol{\xi} \;\; \forall \boldsymbol{\xi} \in \mathscr{X}^e$

The effect of shrinking and inflation mapping can be observed in Figure 10.

2) Folding Mapping: Let us introduce a *folding* mapping $\boldsymbol{m}^f(\boldsymbol{\xi})$, which moves the stationary point infinitely far away. Hence, the dynamics in the mapped space are without directional singularity point, which can be treated with the method introduced in Section IV-A. Conversely, the surface of the obstacle should not be affected by the mapping. The desired properties of this folding mapping are given as follows:

1) the stationary point gets mapped infinitely far away

$$\lim_{\boldsymbol{\xi} \to \boldsymbol{\xi}^a} \Gamma\left(\boldsymbol{m}^f(\boldsymbol{\xi})\right) \to \infty \quad (33)$$

2) at the reference point (which represents the surface of the obstacle after the shrinking mapping), the effect of the mapping vanishes

$$\boldsymbol{\xi}^r = \boldsymbol{m}^f(\boldsymbol{\xi}^r) \quad (34)$$

Let us first define the unit directions in the coordinate system, which has its center at the stationary point and the first axis points towards the reference direction of the obstacle:

$$\hat{\boldsymbol{\xi}} = \boldsymbol{B}^T(\boldsymbol{\xi} - \boldsymbol{\xi}^a)/\|\boldsymbol{\xi} - \boldsymbol{\xi}^a\| \quad (35)$$

where $\boldsymbol{B}$ is the orthonormal matrix of which the first row aligns with $\boldsymbol{\xi}^r - \boldsymbol{\xi}^a$

The desired constraints of the mapping from (33) and (34) can be achieved by uniformly stretching along dimensions $i \in [2..N]$ of $\hat{\boldsymbol{\xi}}$ as:

$$\begin{aligned} \boldsymbol{s}_i &= (2/(1+p) - 1)^g \frac{\hat{\boldsymbol{\xi}}_i}{\|\hat{\boldsymbol{\xi}}_{[2:N]}\|} \quad p \in]-1, 1], \; \boldsymbol{\xi} \neq \boldsymbol{\xi}^a \\ \text{with} \quad p &= \frac{\langle \boldsymbol{\xi} - \boldsymbol{\xi}^a, \boldsymbol{\xi}^r - \boldsymbol{\xi}^a \rangle}{\|\boldsymbol{\xi} - \boldsymbol{\xi}^a\| \, \|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\|} \end{aligned} \quad (36)$$

where $g \in \mathbb{R}_{>0}$ is the power factor, we choose $g = 2$. The above equation pushes points opposite the attractor relative to the obstacle infinitely far away, i.e., $\lim_{\boldsymbol{\xi}_1 \to -1} \boldsymbol{s}_i \to \infty$, see the green line in Figure 10.

Finally, the stretching of the *folding* mapping along dimension $i = 1$ is constructed as follows:

$$\boldsymbol{s}_1 = \|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\| \left(1 + \ln \left(\frac{\|\boldsymbol{\xi} - \boldsymbol{\xi}^a\|}{\|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\|} \right) \right) \quad (37)$$

Using this, the folding mapping $\boldsymbol{m}^f(\boldsymbol{\xi}) : \{\boldsymbol{\xi} \in \mathbb{R}^N : \hat{\boldsymbol{\xi}}_1 \neq -1, \boldsymbol{\xi} \neq \boldsymbol{\xi}^a\} \to \mathbb{R}^N$ is defined as:

$$\boldsymbol{m}^f(\boldsymbol{\xi}) = \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^T(\boldsymbol{\xi} - \boldsymbol{\xi}^a) + \boldsymbol{\xi}^a \quad \forall \boldsymbol{\xi} \in \mathbb{R} \setminus \boldsymbol{\xi}^a \tag{38}$$

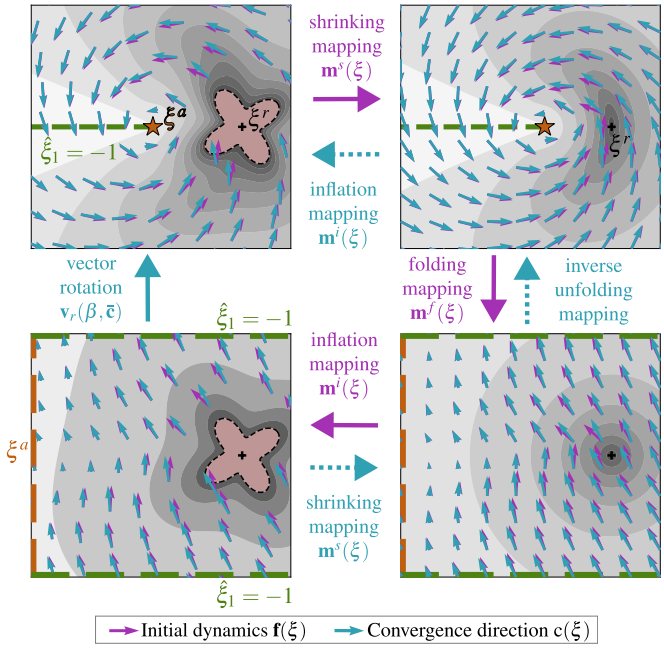


Fig. 10: The initial dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ undergo a series of three consecutive mappings, depicted in a clockwise manner starting from the top left. The convergence weight $w^c(\boldsymbol{\xi})$ is evaluated in the transformed space, where its value is represented by the gray shading. Notably, the weight diminishes to zero at the attractor point $\boldsymbol{\xi}^a$. The directional summing operation from (27) is performed to obtain the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$. This step is carried out in the mapped space. To obtain the reverse mapping, the vector rotation $\boldsymbol{v}_r(\cdot)$ is simply evaluated in the original space. The folding mapping $\boldsymbol{m}^f(\boldsymbol{\xi})$ maps the attractor infinitely far away in the $-\hat{\boldsymbol{\xi}}_1$ direction, whereas the line with $\hat{\boldsymbol{\xi}}_1 = -1$ is folded out to be infinitely far in the directions of $\pm\hat{\boldsymbol{\xi}}_{[2..d]}$.

Lemma IV.2. *The mapping $\boldsymbol{m}^f(\boldsymbol{\xi}) : \boldsymbol{m}^f(\boldsymbol{\xi}) : \{\boldsymbol{\xi} \in \mathbb{R}^N : \hat{\boldsymbol{\xi}}_1 \neq -1, \boldsymbol{\xi} \neq \boldsymbol{\xi}^a\} \to \mathbb{R}^N$ as given in (38) is a bijection, and smoothly defined, i.e., $\lim_{\boldsymbol{\xi} \to \boldsymbol{\xi}_j} \boldsymbol{m}^f(\boldsymbol{\xi}_i) = \boldsymbol{m}^f(\boldsymbol{\xi}_j)$. Furthermore, the mapping has no effect at the reference point, i.e., $\boldsymbol{m}^f(\boldsymbol{\xi}^r) = \boldsymbol{\xi}^r$, and the attractor is mapped infinitely far away, i.e., $\lim_{\boldsymbol{\xi} \to \boldsymbol{\xi}^a} \Gamma\left(\boldsymbol{m}^f(\boldsymbol{\xi})\right) \to \infty$.*

Proof. The system is made up of the two transformations, one along the $\boldsymbol{\xi}^r - \boldsymbol{\xi}^a$ as given in (37), and one perpendicular to, given in in (36). Since all the underlying functions involved are smooth and bijective, the resulting transformation is also smooth and bijective. However, it is important to note that there are discontinuities at $\boldsymbol{\xi} = \boldsymbol{\xi}^a$ and $\hat{\boldsymbol{\xi}}_1 = -1$. This points are excluded from the input set (in Eq. 42 we will additionally ensure that the corresponding weight goes to zero, for a smooth effect).

Furthermore for an input value of $\boldsymbol{\xi}^r$ in (37), we get that $\boldsymbol{s}_1 = \|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\|$, and using (36) we get $\boldsymbol{s}_i = 0$ with $i \in [2..d]$. Hence, the transformation in (38) yields,

$$\begin{aligned} \boldsymbol{m}^f(\boldsymbol{\xi}^r) &= \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^T(\boldsymbol{\xi}^r - \boldsymbol{\xi}^a) + \boldsymbol{\xi}^a \\ &= \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\begin{bmatrix}1 & 0 & .. & 0\end{bmatrix}^T + \boldsymbol{\xi}^a \\ &= \boldsymbol{B}\begin{bmatrix}\|\boldsymbol{\xi}^r - \boldsymbol{\xi}^a\| & 0 & .. & 0\end{bmatrix}^T + \boldsymbol{\xi}^a \\ &= \boldsymbol{\xi}^r - \boldsymbol{\xi}^a + \boldsymbol{\xi}^a = \boldsymbol{\xi}^r \end{aligned} \tag{39}$$

For the region $\hat{\boldsymbol{\xi}}_1 = -1$, we get for the stretching factors $s_i \to \infty$ for $d \in [2..d]$ from (36). □

3) Evaluation of the Relative Rotation: The total mapping can be written as follows:

$$\boldsymbol{m}(\boldsymbol{\xi}) = \boldsymbol{m}^i \circ \boldsymbol{m}^f \circ \boldsymbol{m}^s(\boldsymbol{\xi}) \tag{40}$$

Since this mapping transforms the attractor infinitely far away, it produces a vector field without directional singularity, as Section IV-A requires. Thereof, we can evaluate the convergence dynamics in the mapped space:

$$\bar{\boldsymbol{c}}(\boldsymbol{\xi}) = w^m \circ \boldsymbol{m}(\boldsymbol{\xi})\boldsymbol{f}(\boldsymbol{\xi}^r) \hat{+} \left(1 - w^m \circ \boldsymbol{m}(\boldsymbol{\xi})\right)\boldsymbol{f}(\boldsymbol{\xi}) \tag{41}$$

the symbol $\hat{+}$ implies the directional summing as defined in Appendix A.

The mapping weight is defined as:

$$w^m(\boldsymbol{\xi}) = 1/\sqrt{\left(\Gamma(\boldsymbol{\xi}) - 1\right)\left(\Gamma(m(\boldsymbol{\xi})) - 1\right) + 1} \tag{42}$$

This weight function is designed to ensure a decreasing influence as the distance from the obstacle increases, as well as a decreasing influence when the position is opposite to the singularity point relative to the obstacle (green line in Figure 10). Additionally, it ensures that the weight approaches one when the position lies on the surface of the obstacle.

Finally, the mapping into the original space can be made through a vector rotation

$$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{v}_r(\beta, \bar{\boldsymbol{c}}) \tag{43}$$

where the vector rotation $\boldsymbol{v}^r$ and angle *beta* are obtained according to Eq. (71), using input vector $\boldsymbol{v}_i = \boldsymbol{m}(\boldsymbol{\xi}) - \boldsymbol{\xi}^a$ and output vector $\boldsymbol{v}_o = \boldsymbol{\xi} - \boldsymbol{\xi}^a$.

Theorem IV.3. *The smoothly defined convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ as given in (43) for obstacles in a vector field with a directional singularity point $\boldsymbol{\xi}^a$, i.e., $\{\boldsymbol{\xi} : \|\boldsymbol{f}(\boldsymbol{\xi})\| = 0, \nabla\boldsymbol{f}(\boldsymbol{\xi}) \neq 0\} = \{\boldsymbol{\xi}^a\}$ is ensured to be straight according to Definition II.1 on the surface and inside the obstacle, i.e., $\boldsymbol{c}(\boldsymbol{\xi}) \in \mathscr{F}^s$, $\boldsymbol{\xi} \in \mathscr{X}^b \cup \mathscr{X}^i$.*

Proof. Lemma IV.2 states that the unfolding has no effect at $\boldsymbol{\xi}^r$ in the shrunk space, which is the surface of the obstacle in the original space as stated in Definition IV.1. Further, the inflating mapping is the inverse of the shrinking as stated in Definition IV.2. Thus, when approaching the surface, i.e., $\Gamma(\boldsymbol{\xi}) \to 1$, Lemma IV.1 transfers to Theorem IV.3. □

4) Dynamical Systems with Multiple Attractors: A dynamical system can exhibit multiple attractors [49]. However, if the system is continuous across space, there must exist a region known as the *saddle boundary*. Points starting on the *saddle boundary* do not converge to any specific attractor, and this boundary divides the space into regions that contain only a

single attractor each. By interpreting the *saddle boundary* as an inverted obstacle, as described in Section III-C, the method presented in this section can be applied to the local regions. This allows for effective obstacle avoidance in the presence of multiple attractors, ensuring that the system remains within specific regions corresponding to individual attractors.

5) Obstacles Across the Attractor: When an obstacle spans across the attractor $\boldsymbol{\xi}^a$, i.e., $\Gamma(\boldsymbol{\xi}^a) \leq 1$, the unfolding mapping is not defined. Hence, the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ must approach globally straight dynamics according to Definition II.1. To account for this, the initial dynamics are updated as follows:

$$\boldsymbol{f}(\boldsymbol{\xi}^r) \leftarrow w^\Gamma \text{sign}\left(\nabla \boldsymbol{f}(\boldsymbol{\xi})|_{\xi=\xi^a}\right)(\boldsymbol{\xi}^r - \boldsymbol{\xi}^a) + (1-w^\Gamma)\boldsymbol{f}(\boldsymbol{\xi}^r) \tag{44}$$

The weight is designed to reach one when the obstacle reaches the attractor, e.g., $w^\Gamma = 1/\Gamma(\boldsymbol{\xi}^a)$.

V. Multi-Obstacle Environments

In the presence of multiple obstacles, the rotational obstacle avoidance method for a single obstacle can be extended using a weighted rotational summing. First, the weighted convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ as introduced in (27) is averaged as follows:

$$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o \boldsymbol{c}_o(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o w_o^c \boldsymbol{f}(\boldsymbol{\xi}_o^r) \tag{45}$$

where $\hat{+}$ denotes the rotational summing as defined in Appendix A.

The obstacle weights have been proposed in [45] as:

$$w_o(\boldsymbol{\xi}) = \frac{\tilde{w}_o(\boldsymbol{\xi})}{\sum_{i=1}^{N^{\text{obs}}} \tilde{w}_i(\boldsymbol{\xi})} \text{ with } w_o(\boldsymbol{\xi}) = \frac{1}{\Gamma_o(\boldsymbol{\xi})} \quad \forall \boldsymbol{\xi} \in \mathscr{X}^e \tag{46}$$

The weights associated with each obstacle ensure that their sum is at most one and converge to zero as the distance from the respective obstacle increases. Furthermore, on the surface of an obstacle o, the corresponding weight equals 1. Moreover, the weighting allows for overlapping of the influence regions of the obstacles.

These convergence dynamics are used to evaluate the preferred pseudo tangent for each obstacle $\boldsymbol{e}_o(\xi)$ as defined in (14). Finally, the rotation of the initial dynamics from (17) can be restated for multi-obstacle scenarios as:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o \dot{\boldsymbol{\xi}}_o = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} \lambda_o w_o m(\boldsymbol{\xi}) \boldsymbol{e}_o(\boldsymbol{\xi}) \tag{47}$$

The method is summarized in Algorithm 1 and handles star-shaped obstacles and boundaries, as shown in Figure 11.

Lemma V.1. *The weighted rotation of the convergence dynamics $\boldsymbol{c}(\boldsymbol{\xi})$ from (45) and the final dynamics $\dot{\boldsymbol{\xi}}$ from (47) conserves impenetrability and saddle-point properties introduced in Theorem III.3.*

Proof. When approaching an obstacle o, the weight defined in (46) results in $\lim_{\Gamma_o(\boldsymbol{\xi})\to 1} w_o(\boldsymbol{\xi}) = 1$. Hence the evaluation of the multi-obstacle avoidance converges to the single obstacle scenario of obstacle o. It follows that the properties of the single obstacle case are conserved locally. □

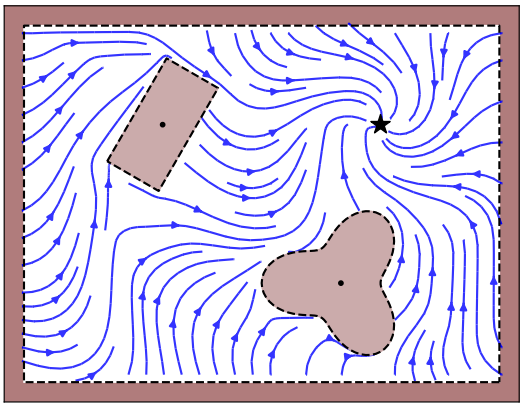


Fig. 11: The rotational obstacle avoidance with respect to wavy dynamics as used in Figure 7a moves towards the attractor (black star) while avoiding collisions inside a rectangular hull with two obstacles.

Algorithm 1 Rotational Obstacle Avoidance Method (ROAM)

Input: $\boldsymbol{f}(\boldsymbol{\xi})$, N^{obs} obstacles
Output: $\dot{\boldsymbol{\xi}}$

for $o = 1$ **to** N^{obs} **do**
  $\tilde{w}_o(\boldsymbol{\xi}) = 1/\Gamma_o(\boldsymbol{\xi})$ {Evaluate weights (46)}
end for
$w_o(\boldsymbol{\xi}) = \tilde{w}_b(\boldsymbol{\xi}) / \sum_{i=1}^{N^{\text{obs}}} \tilde{w}_i(\boldsymbol{\xi})$ {Normalize weights (46)}
for $o = 1$ **to** N^{obs} **if** $w_o(\boldsymbol{\xi}) > 0$ **do**
  $\boldsymbol{f}(\boldsymbol{\xi}_o^r)$ {Compute DS at reference point}
end for
$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} w_o w_o^c \boldsymbol{f}(\boldsymbol{\xi}_o^r)$ {Rotational average 45}
for $o = 1$ **to** N^{obs} **if** $w_o(\boldsymbol{\xi}) > 0$ **do**
  $\boldsymbol{e}_o(\boldsymbol{\xi})$ {Compute pseudo tangent (14)}
  $h(\boldsymbol{\xi})$ {Compute magnitude (21)}
end for
$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{o=1}^{N^{obs}} \lambda_o w_o m(\boldsymbol{\xi}) \boldsymbol{e}_o(\boldsymbol{\xi})$ {Rot. average (47)}

A. Dynamic Environments

v The closed-form description of the algorithm enables short computation time without the need for offline trajectory planning. Dynamic obstacles can be considered by transposing the avoided direction into the moving reference frame:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\boldsymbol{\xi})\boldsymbol{f}(\boldsymbol{\xi}) + \dot{\tilde{\boldsymbol{\xi}}} \quad \text{with} \quad \dot{\tilde{\boldsymbol{\xi}}} = \sum_{o=1}^{N^{\text{obs}}} w_o \dot{\tilde{\boldsymbol{\xi}}}_o \tag{48}$$

The method's application to dynamics obstacles is experimentally evaluated in Section VII-D2.

B. Motion within Multiple Enclosing Hulls

The outer boundary might not be star-shaped, for example, when a robot moves through a curvy corridor. However, there such a general space can be divided into multiple stars-shapes [50], which can be used by ROAM to guide a motion to stay within the boundary. This requires convergence dynamics, which transition between the obstacles as:

$$\boldsymbol{c}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) \hat{+} \sum_{b=1}^{N^{bnd}} w_b(\boldsymbol{\xi}) \boldsymbol{c}_b(\boldsymbol{\xi}) \tag{49}$$

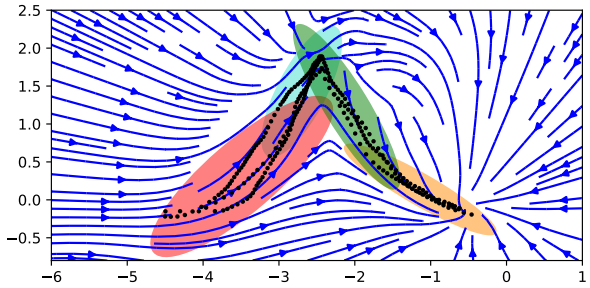
where $N^{bnd} \in \mathbb{N}_{>0}$ is the number of boundaries. Furthermore, the weights of the multi-boundary environment are set to:

$$w_b(\boldsymbol{\xi}) = \frac{\max\left(\Gamma_b(\boldsymbol{\xi}),1)\right)-1}{\sum_{i=1}^{N^{bnd}} \max\left(\Gamma_i(\boldsymbol{\xi}),1\right)-1} \tag{50}$$

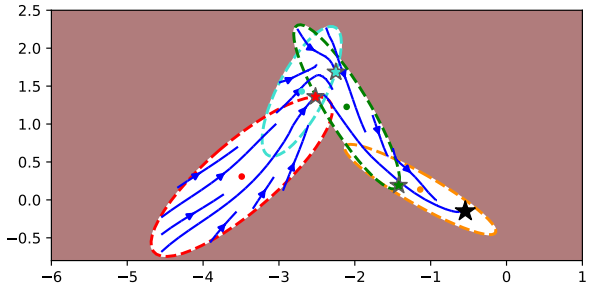
Where the local dynamics $\boldsymbol{c}_b(\boldsymbol{\xi})$ are locally straight according to Definition II.1 in the subdomain of the surface of each boundary. The attractor of each hull $\boldsymbol{\xi}_b^a$ is placed such that it lies outside of the corresponding boundary b for all boundaries that do not contain the global attractor $\boldsymbol{\xi}^a$, i.e.,

$$\begin{cases} \boldsymbol{\xi}_b^a = \boldsymbol{\xi}^a & \text{if } \Gamma_b(\boldsymbol{\xi}^a) > 1 \\ \boldsymbol{\xi}_b^a \in \mathscr{X}_b^i \cup \mathscr{X}_b^b & \text{otherwise} \end{cases} \tag{51}$$

The boundary encapsulation can be seen in Figure 12.



(a) The data (position and velocity) are fit using Gaussian Mixture Model (GMM) with four components [51], which is used to predict the velocity.



(b) The GMM components are used as a multi-hull environment to bind the dynamics and enforce leaving the boundary in the direction of the local attractors $\boldsymbol{\xi}_b^a$ (colored stars).

Fig. 12: Approaches for learning nonlinear motion from demonstration can guarantee stability by ensuring that the system asymptotically converges towards the attractor $\boldsymbol{\xi}^a$ [52]. However, they do not prohibit the dynamics of taking an undesired shortcut or moving away from the data, which is the known region (a). Convex hulls can be obtained through trajectory learning methods, by interpreting the Gaussian Mixture Model applied to the data as ellipse-shaped obstacles [3]. ROAM can enforce the final dynamics to stay close to the data (b).

Lemma V.2. *A motion starting within multiple boundaries* $\{\boldsymbol{\xi}\}_0 \in \bigcup_{b\in N^{bnd}} \mathscr{X}_b^e$ *and evolving according to* $\dot{\boldsymbol{\xi}}$ *as described in* (47) *and with convergence dynamics* $\boldsymbol{c}(\boldsymbol{\xi})$ *defined in* (49) *will stay within the boundaries, i.e,* $\{\boldsymbol{\xi}\}_t \in \bigcup_{b\in N^{bnd}} \mathscr{X}_b^e \ \forall\, t \in \mathbb{N}_{>0}$.

Proof. If $\boldsymbol{\xi}$ is within multiple boundaries, and it approaches boundary b, the weight given in in (50) evaluates to:

$$\lim_{\Gamma_b(\xi)\to 1} w_b(\boldsymbol{\xi}) = \frac{0}{0+\sum_{i\neq b}^{N^{bnd}} \max\left(\Gamma_b(\boldsymbol{\xi}),1)\right)-1} = 0 \tag{52}$$

Hence, boundary b has no effect, and the motion can traverse any boundary $b = 1..N^{\text{bnd}}$. Until, $\boldsymbol{\xi}$ is within only one boundary b. In this case, the corresponding boundary weights simplify to:

$$w_b(\boldsymbol{\xi}) = \frac{\max\left(\Gamma_b(\boldsymbol{\xi}),1)\right)-1}{\max\left(\Gamma_b(\boldsymbol{\xi}),1)\right)-1+\sum_{i\neq b}^{N^{bnd}} 0} = 1 \tag{53}$$

given that $\Gamma_b(\boldsymbol{\xi}) > 1$. Thus, the algorithm evolves according to the single-boundary case, i.e., collision avoidance with the boundary b is ensured. □

VI. General Concave Obstacles

A general obstacle can be described as a union of multiple star-shaped obstacles [34], also referred to as trees of stars. Extending the algorithm to such shapes enables navigating in many more environments. Let us for this introduce a general obstacle using nomenclature from graph theory [53]:

Definition VI.1 (Tree of Obstacles)**.** A tree of star-shaped obstacles represents a shape without holes. Each obstacle (node) in the tree can have multiple children (successors), but have exactly one single parent (predecessor), except for the root obstacle which does not have a parent. All obstacles have a non-zero intersection with their parent and children. Obstacles are assigned a level l in the tree, starting from $l = 0$ at the root.

A. Velocity Propagation through Obstacle Tree

The avoidance velocity $\dot{\boldsymbol{\xi}}$ in the presence of trees-of-obstacles is obtained through the summed average of a rotation-tree as described in Appendix A. The tree is constructed as described in Algorithm 2, and the individual steps are detailed below.

1) Surface Point Propagation: The rotational avoidance of each obstacle $o = 1..N^{\text{com}}$ of the tree is obtained at the corresponding surface point $\boldsymbol{s}_o$. The surface points are obtained by propagating the position $\boldsymbol{\xi}$ through the obstacle tree, starting from the an obstacle o down to the root r:

$$\boldsymbol{s}_p = b(\boldsymbol{\xi}_c^r - \boldsymbol{s}_c) + \boldsymbol{s}_c \quad \text{such that} \quad \Gamma_p(\boldsymbol{s}_p) = 1 \tag{54}$$

where p is the parent of the component c. The factor $b \in \mathbb{R}_{>0}$ is evaluated such that $\boldsymbol{s}_p$ lies on the parent's surface. Note, that the obstacles are intersecting, hence we have $\Gamma(\boldsymbol{s}_p) < 1$. See the surface points of a three-component tree in Figure 13.

2) Velocity Propagation: We can now propagate the velocity $\boldsymbol{f}(\boldsymbol{\xi}^r)$ iteratively from parent $p(c)$ to the component c until we reach the respective obstacle o. The iteration starts at the root r and is done for each obstacle, except the root. The propagated velocity $\boldsymbol{f}_c$ is obtained as follows:

$$\boldsymbol{f}_c = \boldsymbol{v}^r(\boldsymbol{f}_{p(c)}, \phi_c) \qquad o = 1..N^{\text{obs}} \setminus r \tag{55}$$

where $\boldsymbol{v}^r(\cdot)$ is the vector rotation as described in (72). The vector rotation is obtained with respect to the vectors $\boldsymbol{v}_i = (\boldsymbol{s}_c - \boldsymbol{\xi}_c^r)$ and $\boldsymbol{v}_o = (\boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r)$ as described in (71). When constructing the obstacle tree, the parent of an obstacle needs to be chosen such that $\boldsymbol{v}_i$ and $\boldsymbol{v}_o$ are not anti-collinear. This is ensured if the direction from the component to the parent

Algorithm 2 Avoidance of Tree-of-Obstacles

Input: $\boldsymbol{f}(\cdot)$, N^{com} obstacle-components
Output: $\dot{\boldsymbol{\xi}}$

$\boldsymbol{t}^r(\cdot)$ {Create rotation tree, see Algo. 4}
for $o = 1$ **to** N^{com} **do**
  $\boldsymbol{s}_0 = \boldsymbol{r}(\boldsymbol{\xi})$ {Set initial surface point}
  $w_o^h \leftarrow \Gamma_o(\boldsymbol{s}_0)$ {Compute hiding-weight (58)}
  $w_o \leftarrow \Gamma_o(\boldsymbol{\xi})$ {Compute obstacle weights}
  $c = o$ {Initialize c to obstacle o}
  for $l = l(n)$ **to** 0 **do** {From node to root}
    $\boldsymbol{s}_{p(c)} = b(\boldsymbol{\xi}_c^r - \boldsymbol{s}_c) + \boldsymbol{s}_c$ {Propagate reference (54)}
    $c = p(c)$ {Set iterator c to parent p}
  end for
  $\boldsymbol{f}_0 = \boldsymbol{f}(\boldsymbol{\xi}^r)$ {Set initial tangent}
  for $c = 1$ **to** $l(n)$ **do** {From root to node}
    $\boldsymbol{v}^r(\cdot), \beta \leftarrow (\boldsymbol{s}_c - \boldsymbol{\xi}_c^r), (\boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r)$ {Get rot. (71)}
    $\boldsymbol{f}_c = \boldsymbol{v}^r(\boldsymbol{f}_{p(c)}, \beta)$ {Propagate rotated velocity (72)}
    $\boldsymbol{f}_c \Rightarrow \boldsymbol{t}^r(\cdot)$ {Append to tree}
  end for
  $\dot{\boldsymbol{\xi}}_o \leftarrow \boldsymbol{f}_o$ {Avoidance with propagated velocity (57)}
end for
$w_f = 1 - \sum_o w_o^h w_o$ {Weight of initial velocity $\boldsymbol{f}(\boldsymbol{\xi})$}
$\dot{\boldsymbol{\xi}} \leftarrow \boldsymbol{t}^r(w_f, w_o^h(\boldsymbol{\xi}))$ {Evaluate rotation tree, Algo. 4}

is never opposing the direction from the parent to the *grandparent* (parent of the parent):

$$\frac{\left\langle \boldsymbol{\xi}_{p(c)}^r - \boldsymbol{\xi}_c^r, \boldsymbol{\xi}_{p(p(c))}^r - \boldsymbol{\xi}_{p(c)}^r \right\rangle}{\|\boldsymbol{\xi}_{p(c)}^r - \boldsymbol{\xi}_c^r\| \, \|\boldsymbol{\xi}_{p(p(c))}^r - \boldsymbol{\xi}_{p(c)}^r\|} \neq -1 \tag{56}$$

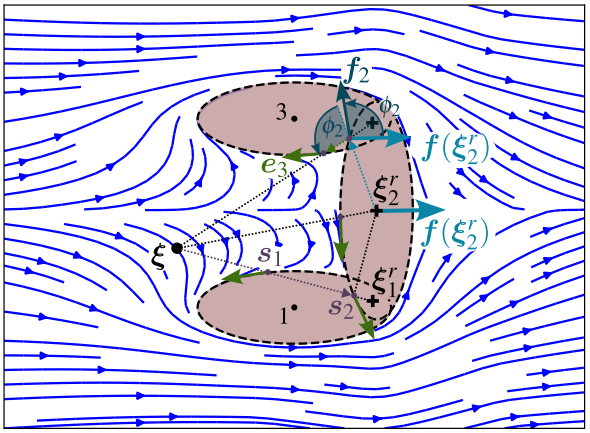


Fig. 13: The surface points $\boldsymbol{s}_{(\cdot)}$ (purple) are evaluated for each obstacle up to the root, to then propagate the desired tangent velocity $\boldsymbol{e}$ (green) from the root to each obstacle.

3) Tangent Evaluation: Finally, after propagating the velocity to the obstacle o, the pseudo tangent is evaluated. For trees-of-stars, we want to enforce the pseudo tangent to be parallel to the surface at all times, hence we adopt (17) as follows:

$$\begin{aligned}
&\text{if} \quad \frac{\langle -\boldsymbol{n}, \boldsymbol{f} \rangle}{\|-\boldsymbol{n}\| \, \|\boldsymbol{f}\|} > \cos(R^e): \\
&\quad \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}_o) = \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r}) + b\,(\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}_p) - \boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{r})) \\
&\quad \text{such that} \quad \|\boldsymbol{k}(-\boldsymbol{n}, \boldsymbol{e}_p)\| = R^e, \quad b \in \mathbb{R}_{>0} \\
&\text{otherwise} \\
&\quad \boldsymbol{k}(\boldsymbol{n}, \boldsymbol{e}_o) = \boldsymbol{k}(\boldsymbol{n}, -\boldsymbol{r}) + b\,(\boldsymbol{k}(\boldsymbol{n}, \boldsymbol{e}_p) - \boldsymbol{k}(\boldsymbol{n}, -\boldsymbol{r})) \\
&\quad \text{such that} \quad \|\boldsymbol{k}(\boldsymbol{n}, \boldsymbol{e}_p)\| = \pi - R^e, \quad b \in \mathbb{R}_{>0}
\end{aligned} \tag{57}$$

4) Hiding Weights: An obstacle o which is occluded by its parent $p(o)$ should not influence the avoidance velocity. For this reason, we introduce the *hiding weight* which reaches zero at full occlusion:

$$w_o^h = \begin{cases} 1 & \text{if } \Gamma(\boldsymbol{s}_o) > 1 \\ \Gamma(\boldsymbol{s}_o)^{\frac{1}{1-b}} & \text{if } b < 1 \\ 0 & \text{otherwise} \end{cases} \qquad b = \frac{\left\langle \boldsymbol{\xi} - \boldsymbol{\xi}_o^r, \boldsymbol{\xi}_{p(o)}^r - \boldsymbol{\xi}_o^r \right\rangle}{\|\boldsymbol{\xi} - \boldsymbol{\xi}_o^r\| \, \|\boldsymbol{\xi}_{p(o)}^r - \boldsymbol{\xi}_o^r\|} \tag{58}$$

Lemma VI.1. *Let us assume a tree with obstacle components $o = 1..N^{\text{com}}$ and the corresponding reference points $\boldsymbol{\xi}_o^r$, which all lie within obstacle o and the corresponding parent p, i.e., $\boldsymbol{\xi}_o^r \in \mathscr{X}_o^i \cap \mathscr{X}_{p(o)}^i$. Conversely, the reference of each parent lies outside of the obstacle, i.e., $\boldsymbol{\xi}_{p(o)}^r \in \mathscr{X}_{p(o)}^i \setminus \mathscr{X}_o^i$. Moreover, the direction from obstacle to parent is never opposing the direction from the parent to the grandparent as described in* (56)*. Let us assume the dynamics $\boldsymbol{f}(\boldsymbol{\xi})$ are locally straight in the surrounding of the obstacle according to Definition II.1. The vector field $\dot{\boldsymbol{\xi}}$ obtained through the propagation of the velocity $\boldsymbol{f}(\boldsymbol{\xi})$ as described in* (55)*, with the rotation of the final velocity given by* (57)*, and vector tree summing using the weights w_o^h given in* (58) *ensures collision avoidance according to the boundary condition* (13) *with the absence of local minima in free-space.*

Proof. As shown in Lemma III.1, the tangent is defined everywhere as long as the reference direction $\boldsymbol{r}_o(\boldsymbol{\xi})$ is not parallel to the propagated velocity $\boldsymbol{f}_o(\boldsymbol{\xi})$. Moreover, smoothness is ensured due to the adaptable rotation weight $\lambda(\boldsymbol{\xi})$ as shown in Theorem III.3.

Furthermore, the rotation defined in (55) is smoothly defined for obstacle-parent-pair, due to the fact that the reference point of the parent is required to lay outside of the child obstacles, and the parent-opposing inequality from (56).

This is also ensured for the last surface point s_o, i.e., the projection of the position on the obstacle's surface since the hiding weight w_o^h goes to zero if the two vectors are opposing:

$$\frac{\left\langle \boldsymbol{s}_c - \boldsymbol{\xi}_c^r, \boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r \right\rangle}{\|\boldsymbol{s}_c - \boldsymbol{\xi}_c^r\| \, \|\boldsymbol{s}_{p(c)} - \boldsymbol{\xi}_{p(c)}^r\|} \;\Rightarrow\; b = 1 \;\Rightarrow\; w^h = 0 \tag{59}$$

Hence, the corresponding tangent and the vector can be omitted.

Since all weights and corresponding vectors are smoothly defined, according to Lemma A.1 the vector tree evaluation leads to continuous and minima-free dynamics.

Furthermore, from Lemma V.1 we know that the impenetrability of the obstacles is preserved as a single component dominates when approaching the corresponding surface. □

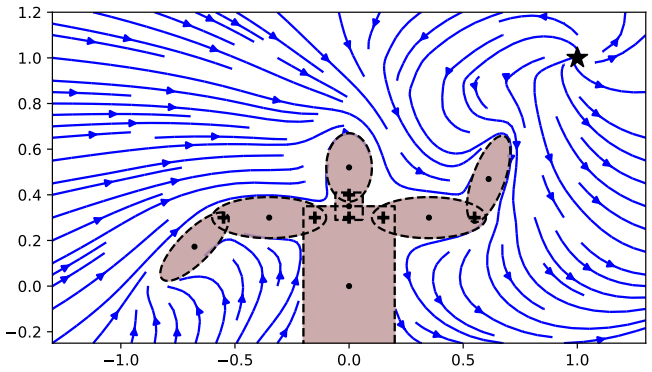


Fig. 14: Obstacle avoidance of a two-dimensional human with a total of seven sub-obstacles (corresponding reference points as black crosses) of a circular motion with a single stationary point (black star).

B. Convergence Sequence

The obstacle avoidance algorithm is dependent on locally straight dynamics on the surface of the tree-of-obstacles. Since the general shapes described can be encapsulating directional singularity points, such as attractors, the design of convergence direction requires special care. The computation has to ensure smoothness when getting closer to the directional singularity point $\boldsymbol{\xi}^a$. It is detailed in Algorithm 3.

Algorithm 3 Convergence Direction for Tree-of-Obstacles. The abbreviation pred. refers to the predecessor of the direction tree.

Input: $\boldsymbol{f}(\boldsymbol{\xi})$, N^{com} obstacle-components
Output: $\boldsymbol{c}(\boldsymbol{\xi})$ Locally straight dynamics

$w_1^m(\boldsymbol{\xi})$ {Mapping weight for root component (42)}
$\boldsymbol{f}(\boldsymbol{\xi}), (\boldsymbol{\xi} - \boldsymbol{\xi}^a) \Rightarrow \boldsymbol{v}^{r,i}(\cdot), \beta_i$ {Initial rotation (71)}
$\boldsymbol{v}^{r,c}(\cdot), \beta_c \leftarrow \boldsymbol{f}(\boldsymbol{\xi}^r), (\boldsymbol{\xi}^r - \boldsymbol{\xi}^a)$ {Convergence rot. (71)}
$\boldsymbol{v}^{r,a}(\cdot), \beta_a \leftarrow (\boldsymbol{\xi} - \boldsymbol{\xi}^a), (\boldsymbol{\xi}^r - \boldsymbol{\xi}^a)$ {Get rotation (71)}
$\boldsymbol{t}_1^r(\cdot) \leftarrow \boldsymbol{v}^{r,i}(\cdot), \boldsymbol{v}^{r,a}(\cdot), \boldsymbol{v}^{r,c}(\cdot)$ {Create rotation tree}
{Tree reduction with Algo. 1 using vector-weight pairs:}
$\boldsymbol{t}_1^r\left([\boldsymbol{f}(\boldsymbol{\xi}^r), w_1^m], [\boldsymbol{f}(\boldsymbol{\xi}), (1 - w_1^m)]\right) \Rightarrow \boldsymbol{t}^r(\cdot)$
for $n = 1$ **to** $N^{\text{com}}, i \neq r$ **do** {Iteration over components}
  $c \leftarrow n$ {Set initial node}
  $w_i^m(\boldsymbol{\xi})$ {Mapping weight (42)}
  for $l = l(n)$ **to** 0 **do** {From node to root}
    $(\boldsymbol{\xi}_{p(c)}^r - \boldsymbol{\xi}_c^r) \Rightarrow \boldsymbol{t}^r(\cdot)$ {Append to tree with pred. c}
    $c \leftarrow p(c)$ {Set current node iterator to its parent}
  end for
  $\boldsymbol{v}^{r,c}(\cdot) \Rightarrow \boldsymbol{t}^r(\cdot)$ {Append to tree with pred. $c = r$}
end for
$w_o^n \leftarrow w_o^m$ {Weight normalization (60)}
$\boldsymbol{c}(\boldsymbol{\xi}) \leftarrow \boldsymbol{t}^r(w_o^n)$ {Tree reduction, Algo. 4}

C. Mapping Weight Normalization

A mapping weight w_o^m as introduced in (60) approaching one indicates that the $\boldsymbol{\xi}$ is on the surface of the corresponding obstacle o. To ensure that this weight remains high while taking into account other obstacles, the normalized weights are defined as follows:

$$w_o^n = \begin{cases} \hat{w}_o^n / \sum_i^{N^{\text{com}}} \hat{w}_i^n & \text{if } \sum_i^{N^{\text{com}}} \hat{w}_i^n > 1 \\ \hat{w}_o^n & \text{otherwise} \end{cases}, \quad \hat{w}_o^n = 1/(1 - w_o) \tag{60}$$

for all obstacle $o = 1...N^{\text{obs}}$.

D. Practical Considerations

As for each obstacle, the algorithm needs to propagate the whole obstacle tree. The presence of multiple obstacles and long trees can lead to many computations. Therefore, it is advised to adapt the distance function Γ_o to approach infinity after a certain distance for outer leaves, as proposed in (70). However, the distance function of the root Γ_r should decrease slower, i.e.,

$$\left\| \frac{d}{d\boldsymbol{\xi}} \Gamma_o(\boldsymbol{\xi}) \right\| \geq \left\| \frac{d}{d\boldsymbol{\xi}} \Gamma_r(\boldsymbol{\xi}) \right\| \tag{61}$$

This has the effect that the algorithm only considers the obstacle core far away, while when approaching the obstacle, the more detailed leaf structure is considered. This leads to a sparse evaluation tree and enables real-time applicability.

VII. Evaluation

A. Computational Cost

The most computationally intensive part of the ROAM algorithm is the matrix-vector multiplication in N dimensions for the stereographic projections and unfolding mappings. Therefore, the algorithm's complexity, given O obstacles, K components, and an obstacle tree of level L, can be expressed as $\mathcal{O}(N^2OKL)$.

B. Obstacle Avoidance While Following a Stable Limit Cycle

1) Setup: We compare the three algorithms MuMo [45], VF-CAPF [40], and ROAM (proposed approach). The chosen scenario uses initial dynamics and the obstacle distribution as proposed in [40]. The initial dynamics represent a circular limit cycle of the form:

$$\boldsymbol{f}(\boldsymbol{\xi}) = \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + 2(R_0 - \|\boldsymbol{\xi}\|)\boldsymbol{I} \right) \boldsymbol{\xi} \tag{62}$$

where the circle radius is $R_0 = 2$, and $\boldsymbol{I}$ is the two-dimensional identity matrix. The environment contains six convex obstacles, and the agent is aware of their location at all times (see Fig. 15). A grid of 10x10 evenly distributed starting points was constructed, of which 93 were in free space. The trajectory is evaluated for these starting points through Euler integration with a time step $dt = 0.01s$ and a maximum of 500 iterations.[4]

[4]Source code on https://github.com/hubernikus/nonlinear_obstacle_avoidance.git, 2022/02/31

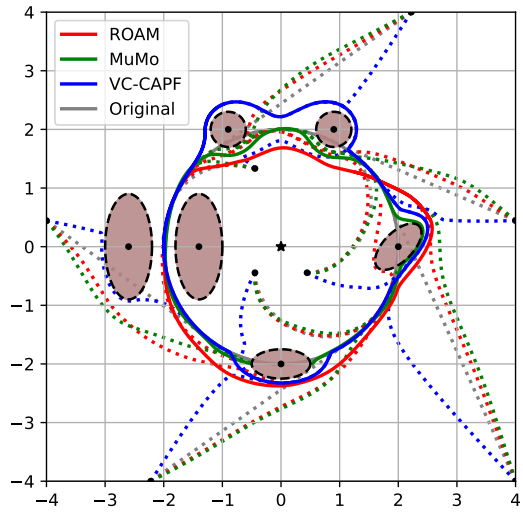


Fig. 15: MuMo [45], VF-CAPF [40] and ROAM are used to guide a nonlinear, limit-cycle-following vector field around six concave obstacles. The resulting limit cycles are visualized as a solid line in the respective color. Additionally, the trajectories from specific starting points (black circles) are visualized with a dashed line.

2) Metrics: The trajectories are compared by observing the local minima, the distance to the desired limit cycle, and the similarity between the velocity after obstacle avoidance and the initial velocity. Furthermore, we look at the evolution of the velocity over time, i.e., the discrete acceleration.

The root mean square error (RMSE), i.e., $\text{RMSE} = \sum_{m=1}^{M} \|\boldsymbol{v}_i - \boldsymbol{u}_i\|$ is used, as well as the normalized inverted cosine similarity (NICS)

$$\text{NICS} = \frac{1}{2}\left(1 - \frac{1}{M}\sum_{m=1}^{M} \frac{\langle \boldsymbol{v}_i, \boldsymbol{u}_i \rangle}{\|\boldsymbol{v}_i\| \, \|\boldsymbol{u}_i\|}\right) \tag{63}$$

with $\text{NICS} \in [0,1]$. Note that the smaller NICS, the higher the similarity of the two vectors.

3) Results: In Table II, it can be observed that MuMo is the only method among the three that results in trajectories ending up in local minima on the surface of an obstacle in almost half of the cases. These local minima lie on the limit cycle, suggesting that by increasing the step number, all MuMo trajectories would eventually converge to these local minima. Due to this behavior, MuMo is deemed inappropriate for the proposed scenario. Its limited applicability to nonlinear initial dynamics as pointed out in Table I.

The focus of the comparison is thus between ROAM and VF-CAPF. ROAM demonstrates better path following, with trajectories maintaining a shorter distance to the reference circle throughout the motion. Moreover, ROAM exhibits lower acceleration along the path, indicating smoother motion with fewer abrupt changes. Additionally, ROAM shows higher similarity to the initial dynamics compared to VF-CAPF.

These findings can be qualitatively observed in Figure 15, where VF-CAPF closely follows the initial path (gray) outside the obstacle region and deviates only when in close proximity to the obstacle. The metrics indicate that this behavior leads to higher accelerations and more significant deviations from the initial trajectory overall. On the other hand, MuMO moves more directly towards the limit cycle compared to ROAM, but at the expense of creating local minima on the surface of the bottom obstacle.

	ROAM (proposed)	MuMo	VF-CAPF	Original dynamics
N^m	**0%**	48%	**0%**	0%
RMSE($\boldsymbol{\xi}, R_0$)	1.46 ± 1.11	**1.24 ± 1.15**	1.48 ± 0.87	0.93 ± 0.96
RMSE($\boldsymbol{\xi}, \boldsymbol{f}_0$)	0.15 ± 0.07	**0.04 ± 0.09**	0.47 ± 0.43	0.00 ± 0.00
NICS($\boldsymbol{\xi}, \boldsymbol{f}_0$)	0.04 ± 0.02	**0.01 ± 0.02**	0.12 ± 0.11	0.00 ± 0.00
RMSE($\boldsymbol{\xi}_t, \boldsymbol{\xi}_{t+1}$)	2.51 ± 2.66	**2.04 ± 5.98**	3.47 ± 2.14	0.27 ± 0.17
NICS($\boldsymbol{\xi}_t, \boldsymbol{\xi}_{t+1}$)[$10^{-4}$]	0.63 ± 0.66	**0.51 ± 1.49**	1.08 ± 0.82	0.07 ± 0.04

TABLE II: The different trajectories are compared in (1) the ratio of trajectories which end up in a local minimum on the obstacle N^m, (2) the distance to the desired trajectory, i.e., the difference to radius R_0. Furthermore, (3) RMSE and NICS between all approaches to the original DS are evaluated, as well as (4) the change of the dynamics over time (corresponds to acceleration). The mean and standard deviation are evaluated over the 93 trajectories.

C. Nonlinear Path Following with Autonomous Wheelchair

1) Experimental Setup: In the second evaluation, the autonomous wheelchair QOLO was employed, and various initial vector fields combined with avoidance methods were tested. The control point of the wheelchair was positioned in front of the wheel axis to ensure effective maneuverability. To account for the shape and size of the QOLO wheelchair, a margin of 0.7 meters was added to the obstacles during the evaluation [45].

The path followed by the wheelchair consisted of three road segments, denoted as $s = 1$, the segment closest to the goal, up to $s = 3$. Although there was no strict constraint to remain within the road boundaries, the initial dynamics were specifically designed to guide the wheelchair towards the center of the road, promoting adherence to the desired path (see Fig. 16a).

Seven tables were randomly placed along the path, with their centers positioned on the path itself. The tables were positioned away from the starting point and the attractor. Additionally, no more than two tables intersected, including the margin, allowing all table shapes to be represented as star-shapes, as required by MuMo [45].

2) Navigation Algorithms: Three approaches are used to navigate in this environment.

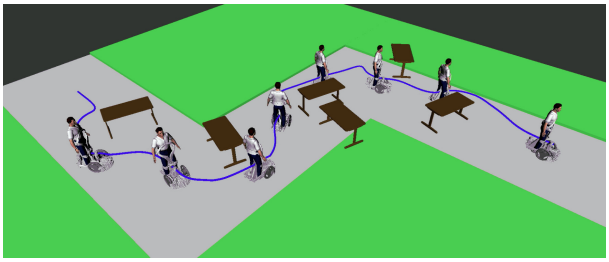
Local straight dynamics (2) are combined with MuMo [45]. The initial dynamics consist of three separate vector fields with corresponding local attractors (star), see Fig. 16b. The attractor switches when transitioning from one region to the next (crossing the line).

Local PF (path following) is combined with ROAM. The local PF dynamics are given as:

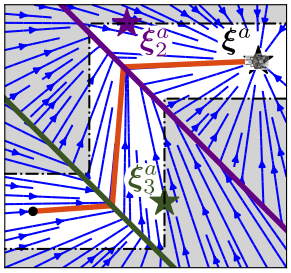
$$\boldsymbol{f}_s(\boldsymbol{\xi}) = \boldsymbol{u}_s + \langle \Delta\boldsymbol{\xi}, \boldsymbol{u}_s \rangle \, \boldsymbol{u}_s - \Delta\boldsymbol{\xi} \quad \text{with} \quad \Delta\boldsymbol{\xi} = (\boldsymbol{\xi} - \boldsymbol{\xi}_s^a) \tag{64}$$

for all segments $s = 1..3$, where $\boldsymbol{u}_s \in \mathbb{R}^2$ is the (local) nominal direction pointing along the road segment, and $\boldsymbol{\xi}_s^a \in \mathbb{R}^2$ is the local attractor.

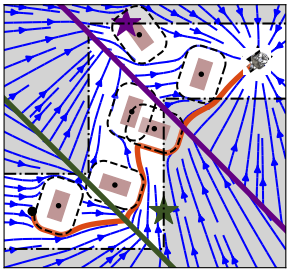
Global PF dynamics is evaluated by using directional-tree averaging as described in Appendix A. The root of the



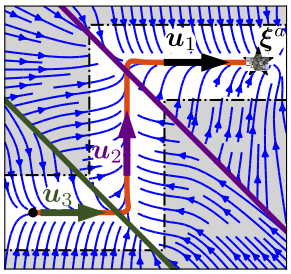
(a) QOLO-robot navigating along a wavy road using global PF



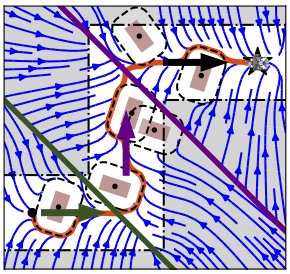
(b) Local straight - initial



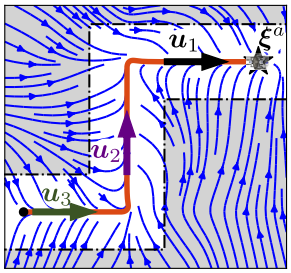
(c) Local straight - avoiding



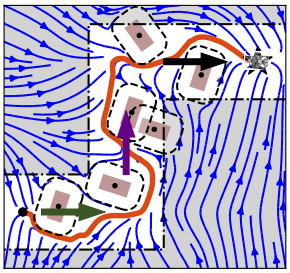
(d) Local PF - initial



(e) Local PF - avoiding



(f) Global PF - initial



(g) Global PF - avoiding

Fig. 16: The robot is navigating between static tables on a wavy road (gray) through a grass field (green). The initial dynamics and the reference trajectory (orange) are on the left, with the corresponding obstacle dynamics on the right. The local attractors (colored starts) and switching regions (colored lines) are used to create global dynamics (a, c). The global path following (f) uses a single attractor only.

direction-tree is given as $\boldsymbol{v}_{0,1} = \boldsymbol{\xi} - \boldsymbol{\xi}^a$. The direction tree is populated iteratively:

- $\boldsymbol{v}_{s,1}(\boldsymbol{\xi}) = \boldsymbol{u}_s$ with respective parent direction $\boldsymbol{v}_{s-1,1}$
- $\boldsymbol{v}_{s,2}(\boldsymbol{\xi}) = \boldsymbol{f}_s(\boldsymbol{\xi})$ with respective parent direction $\boldsymbol{v}_{s,1}$

for all segments $s = 1..3$. The final dynamics are obtained through the weighted evaluation described Algorithm 4, using the following weights

$$w_{s,1}(\boldsymbol{\xi}) = 0, \;\; w_{s,2}(\boldsymbol{\xi}) = \frac{1}{d_s}(1 + \min(\langle \boldsymbol{v}_s, \boldsymbol{\xi}_s^a - \boldsymbol{\xi} \rangle, 0)) \quad (65)$$

where $d_s \in \mathbb{R}_{\geq 0}$ the distance to the line segment s. The segment weights $w_s(\boldsymbol{\xi}) \in [0,1]$ are normalized if their sum exceeds one.

3) Results: The combination of the global PF with ROAM ensures the convergence of all trajectories, as shown in Table III. The other two methods achieve a convergence rate of approximately 92 %. This can be attributed to the use of a high-level planner, specifically switching between the local dynamics. Since this conflicts with the guarantees of absences of local minima. While more sophisticated switching or transitioning methods may exist, to the best of our knowledge, there is no global path sequencer that can guarantee these convergence properties within a finite time. Furthermore, when using the local potential field (PF) with ROAM, the robot spends less time on the desired path and has a greater average distance to the path boundaries compared to the local straight algorithm combined with MuMo. However, the average distance traveled remains approximately the same across all methods.

	Local straight [45]	Local PF	Global PF
Converged []	92%	92%	**100%**
Off-track [%]	1.49 ± 0.00	**0.79 ± 0.00**	1.89 ± 0.00
Δd [m]	1.54 ± 0.04	**1.73 ± 0.02**	1.65 ± 0.04
Distance [m]	**20.2 ± 3.4**	21.0 ± 3.8	20.8 ± 2.8

TABLE III: The three approaches for following the local path are compared based on the following metrics: the convergence ratio to the attractor, the ratio of trajectories deviating from the path, the distance to the road border Δd, and the total length of the trajectory. The reported values represent the mean and standard deviation calculated from 100 runs with randomly distributed furniture, while keeping the start and endpoints consistent.

D. Obstacle Avoidance in Three Dimensions

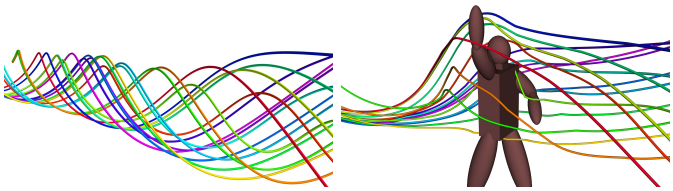
1) Spiraling Motion Around Human in Simulation: Inspired by (62), we propose spiraling dynamics as:

$$\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{B}^T \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + 2\left(R_0 - \|\boldsymbol{B}\tilde{\boldsymbol{\xi}}\|\right)\boldsymbol{I} \right) \boldsymbol{B}\tilde{\boldsymbol{\xi}} + \boldsymbol{p}(\boldsymbol{\xi}) \quad (66)$$
$$\text{with } \boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{p}(\boldsymbol{\xi}) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$$

with the spiraling radius of $R_0 = 0.1\,\text{m}$ and $\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{\xi}^a$ the relative position with respect to the center $\boldsymbol{\xi}^a$.

The obstacle tree representing the human in ROAM consists of components corresponding to the limbs and main body, with the core serving as the root of the tree. When applying ROAM for collision avoidance, see Figure 17, it can be observed that all trajectories successfully avoid the human without becoming trapped in local minima. Furthermore, the dynamics of the system return to the initial state, far away from the obstacle, both at the beginning and after successfully avoiding the collision. This behavior highlights the effectiveness of ROAM in generating smooth and convergent trajectories while maintaining the desired dynamics of the system.

2) Qualitative Evaluation on Robot Arm: Experiments with the real robot were performed using the 7 DoF Panda robot by Franka Emika on a fixed base (see Fig. 18). The scenario chosen is the automated disinfecting of a running conveyor belt, which transports various parcels. The initial dynamics



(a) Initial spiraling motion $\boldsymbol{f}(\boldsymbol{\xi})$ **(b)** Rotated motion around human $\boldsymbol{\xi}$

Fig. 17: ROAM is used to guide trajectories from 16 different initial positions and ensures that all trajectories successfully avoid the static human in simulation.

$\boldsymbol{f}(\boldsymbol{\xi})$ are similar to the spiral motion in (66), but the basis $\boldsymbol{B}$ and perpendicular dynamics $\boldsymbol{p}(\boldsymbol{\xi})$ given by:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{p}(\boldsymbol{\xi}) = \begin{bmatrix} 0 & 0 & \left(\boldsymbol{\xi}^a(t) - \boldsymbol{\xi}\right) \end{bmatrix}^T \tag{67}$$

Moreover, the attractor is dynamic and moves back and forth the conveyer belt:

$$\boldsymbol{\xi}^a(t) = \begin{bmatrix} 0.5 & 0.6 + 0.1 \sin\left(\frac{\pi}{10} t\right) & 0.3 \end{bmatrix}^T \tag{68}$$

As a result, the initial dynamics $\boldsymbol{f}(\boldsymbol{\xi}, t)$ are time-varying, too.

It is assumed that the robot has complete knowledge of the relative position and shape of the conveyor belt. The position and velocity of the parcel are determined using reflective markers (Optitrack). The analytical shape of the objects is known analytically. Additionally, an operator is handling parcels on the conveyer belt, but the robot is not directly aware of its presence. However, since an impedance controller is employed [7], the robot is capable of adapting to physical interactions.

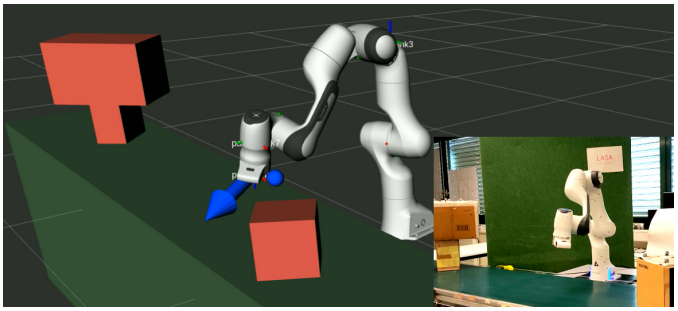


Fig. 18: The robot is aware of the two obstacles (brown shapes) as well as the conveyor belt (green block) to obtain the avoidance dynamics (blue arrow). The center of the initial dynamics (blue dot) is moving across the conveyor belt.

The robot successfully avoids both the parcels and the conveyor belt while maintaining adherence to the initial dynamics whenever feasible. By utilizing trees-of-stars to represent the concave obstacle and positioning the reference point of the root-component on the conveyor belt, the robot effectively avoids the parcel from above (see Fig. 19). Notably, comparable methods such as MuMo [45] or VF-CAPF [40] do theoretically not permit the placement of a reference point in trees of obstacles that facilitates collision avoidance in such environments.

VIII. Discussion

The proposed rotational obstacle avoidance method (ROAM) has successfully addressed the challenge of avoiding collisions with initially nonlinear dynamics around general concave obstacles without holes. It stands out as the first state-dependent solution for trees-of-stars obstacles that is free from local minima without the need for hyperparameter tuning. Furthermore, ROAM enables obstacle avoidance while attempting to maintain nonlinear dynamics, which is a significant advantage. The algorithm has demonstrated improved convergence and motion similarity compared to the baseline methods in experimental evaluations. Moreover, its low computational cost has allowed its application to dynamic obstacle avoidance scenarios involving a robotic arm.

A. Stationary Points

ROAM introduces a new stationary point for each obstacle (or obstacle tree). However, due to the topological properties of smooth vector fields, at least one fixed point is created for each obstacle (a hole in space). These fixed points are observed to be saddle points and have no practical significance as they are unstable and the probability of reaching them is effectively zero. Additionally, any noise or perturbation in the system pushes the motion away from these unstable points. It is worth noting that while ROAM is used for dynamic scenarios, the trajectories of these saddle points should be preserved as they reflect the smoothness of the velocity. Smoothness is crucial as it ensures that even with uncertainties in perception or unexpected disturbance, there will be no discontinuity in the desired velocity command $\dot{\vec{\xi}}$. Nevertheless, the removal of saddle points could be achieved by setting the smoothness factor q to 1 in Equation (17) and selecting any tangent direction for $\vec{e}(\vec{\xi})$ when $\vec{c}(\vec{\xi})$ and $\vec{\xi}^r$ are collinear.

B. Trees of Stars

ROAM relies on a star-shaped (or trees-of-stars) obstacle environment. While in certain real-world scenarios, such division can be achieved based on the rigid-body features of the surroundings (e.g., dividing a human into limbs and core or a table into plate and legs), it is often challenging to determine such divisions for more complex obstacles or in higher-dimensional spaces, such as joint space collision avoidance. Some propositions for extracting star-shaped environments exist [54]; however, further research is needed to extend these approaches for real-time applications.

However, there exist algorithms to simplify general shapes by approximating them as a union of overlapping spheres [55], which could serve as the basis for constructing the obstacle tree in future work. Additionally, the circular shape of these components reduces the computational complexity of tasks such as checking for component intersections and evaluating the normal direction and distance to the obstacles.

C. Tangent Following on the Surface

Each tree of stars has exactly two saddle points, which lie on the convergence direction line passing through the root of the

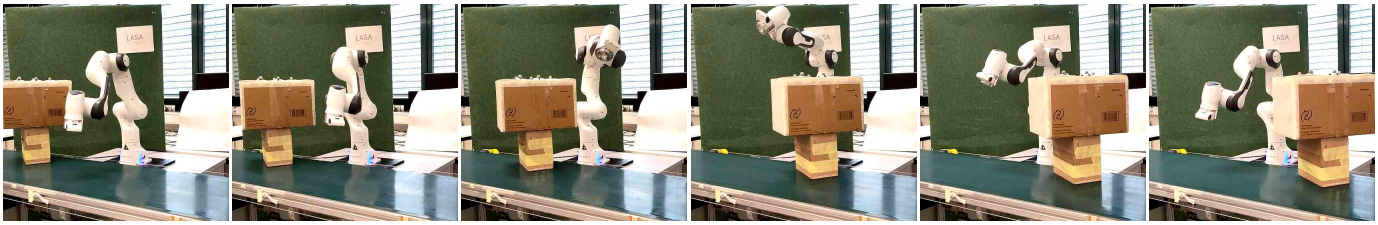


Fig. 19: The 7DoF robot arm adeptly avoids the star-shaped parcels that are being transported on the conveyor belt.

tree. At any other surface position, the velocity is tangent to the surface. This can lead to extensive wall-following behaviors for obstacles with multiple levels. To address this, future work should focus on optimizing the selection of saddle points and allowing the rotated velocity to deviate from the obstacle's tangent plane. However, this would require a combination with high-level planning.

D. Higher Dimensional Applications: Joint Limits

During the experiments with the robotic arm, there were instances where obstacle-free motions led the arm to reach its joint limits or encounter singularities. Future research will investigate a unified framework that combines task-space collision avoidance with joint-space constraint avoidance. ROAM is well-suited for evaluating the desired motion in both representations due to its low computational cost and applicability in higher dimensions.

E. Region of Influence

Navigation functions for obstacle avoidance [33], [36] and vector field avoidance algorithms [38], [40] often require parameter tuning based on the obstacle distribution to ensure the absence of local minima. In contrast, the introduced method, ROAM, can handle intersecting obstacle regions, and its absence of local minima is guaranteed as long as each distance function $\Gamma_o(\vec{\xi})$ for each obstacle o individually satisfies the limitations outlined in Section II-C2. The distance function proposed in Equation (5), which was used throughout this work, is globally active. However, when dealing with dense environments, the linear scaling of ROAM with the number of obstacles can result in high computational costs. To address this, the influence region can be reduced by introducing a distance threshold $d^{\max} \in \mathbb{R}$, such that:

$$\lim_{\|\boldsymbol{\xi}-\boldsymbol{\xi}^b\|\to d^{\max}} \Gamma(\boldsymbol{\xi}) \to \infty \quad \text{and} \quad \lim_{\|\boldsymbol{\xi}-\boldsymbol{\xi}^b\|\to 0} \Gamma(\boldsymbol{\xi}) = 1 \tag{69}$$

For example, the distance function can be set as follows:

$$\Gamma(\boldsymbol{\xi}) = \begin{cases} d^{\max}/(d^{\max} - \|\boldsymbol{\xi}-\boldsymbol{\xi}^b\|) & \text{if } \|\boldsymbol{\xi}-\boldsymbol{\xi}^b\| < d^{\max} \\ \infty & \text{otherwise} \end{cases} \tag{70}$$

F. Application to Robotic Systems

The proposed algorithm assumes that the robot is a point mass. However, it can be extended to real robots by introducing a margin around the obstacles or using multiple control points, as proposed in [45]. It is important to note that the latter method does not guarantee full convergence in general scenarios. Alternative methods employ a full-body representation of the agent [19], [23], which involves sampling or optimization. However, they cannot guarantee finding a feasible path in a finite time. Therefore, future work should focus on extending ROAM to handle analytic avoidance scenarios where trees of stars represent both the agent and the obstacle.

APPENDIX

A. Vector Rotations

The rotational obstacle avoidance method (ROAM) requires weighted summing between vectors as well as partial (sequential) rotations applied to vector fields. Herefore, we introduce vector math concepts to simplify these operations.

1) Perpendicular Rotation Base: Let us consider two initial directions $\boldsymbol{v}_i, \boldsymbol{v}_o \in \mathbb{R}^N \setminus \boldsymbol{0}$. They are used to construct the base vectors $\boldsymbol{b}_i, \boldsymbol{b}_o \in \{\boldsymbol{b} \in \mathbb{R}^N : \|\boldsymbol{b}\| = 1\}$:

$$\boldsymbol{b}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|}, \; \boldsymbol{b}_o = \frac{\hat{\boldsymbol{b}}_o}{\|\hat{\boldsymbol{b}}_o\|}, \quad \beta = \arccos\left(\frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_o\rangle}{\|\boldsymbol{v}_i\|\,\|\boldsymbol{v}_o\|}\right) \tag{71}$$
$$\text{with} \quad \hat{\boldsymbol{b}}_o = \boldsymbol{v}_o - \boldsymbol{v}_i \langle \boldsymbol{v}_i, \boldsymbol{v}_o\rangle \;, \quad \forall \boldsymbol{v}_i, \boldsymbol{v}_o : \frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_o\rangle}{\|\boldsymbol{v}_i\|\,\|\boldsymbol{v}_o\|} \neq -1$$

The angle β and two vectors $\boldsymbol{b}_i$, $\boldsymbol{b}_o$ represent the rotation of a vector in N dimensions, this is equivalent to $2N+1$ parameters.[5]

2) Rotating a Vector: Rotating a vector $\boldsymbol{v}$ entails rotating the component which lies in the plane spanned by $\boldsymbol{b}_i$ and $\boldsymbol{b}_o$, while conserving the part orthogonal to the plane.

$$\boldsymbol{v}_r\left(\boldsymbol{v}, \beta, \boldsymbol{b}_{\{i,o\}}\right) = p_0 \boldsymbol{b}_i \cos(\phi) + \boldsymbol{b}_o \sin(\phi) + \boldsymbol{v} - \sum_{j\in\{i,o\}} \boldsymbol{b}_j \langle \boldsymbol{b}_j, \boldsymbol{v}\rangle$$
$$\text{with} \quad \phi = \phi_0 + \beta \;, \; \tan(\phi_0) = \frac{\langle \boldsymbol{b}_i, \boldsymbol{v}\rangle}{\langle \boldsymbol{b}_o, \boldsymbol{v}\rangle}, \tag{72}$$
$$p_0 = \sqrt{\langle \boldsymbol{b}_i, \boldsymbol{v}\rangle^2 + \langle \boldsymbol{b}_o, \boldsymbol{v}\rangle^2}$$

Note that by changing the rotation angle β, a partial rotation or over-rotation can be applied to a vector, too. In this work, the basis is only explicitly stated if not clear from the context.

Moreover, as each rotation basis is defined by the two vectors $\boldsymbol{b}_i$ and $\boldsymbol{b}_o$, a rotation can be rotated, too. For example, a rotation expressed in a relative reference frame can be rotated to the global frame. This allows for the evaluation of weighted rotation sequences, as will be discussed later in this section.

[5]This compares to $N(N-1)/2$ parameters required to describe rigid body rotation in N dimensions. Hence, already at $N=4$ the rigid body orientation uses more parameters to describe vector rotation, and stays more efficient for higher dimensions.

3) Weighted Rotation Summing: Let us assume that we have $N^{\text{vec}} \in \mathbb{N}_{>0}$ vector rotations which are defined with respect to a shared basis vector $\boldsymbol{b}_i$. The rotations have a respective second basis vector $\boldsymbol{b}_{o,v}$ and rotation angle β_v with $v \in 1..N^{\text{vec}}$. The basis and angle of the weighted average rotation is defined as:

$$\hat{\boldsymbol{b}}_o = \boldsymbol{B}(\boldsymbol{b}_i)\left(\sum_{v=1}^{N^{\text{vec}}} \boldsymbol{B}^T(\boldsymbol{b}_i) w_v \beta_v \boldsymbol{b}_{o,v}\right) \quad \text{and} \quad \hat{\beta} = \|\hat{\boldsymbol{b}}_o\| \tag{73}$$

where $w_v \in [0,1]$ are the vector weights, and $\boldsymbol{B}(\boldsymbol{b}_i)$ is an orthonormal basis with respect to the vector $\boldsymbol{b}_i$.

We further define the rotational sum using the symbol $\hat{+}$ as:

$$\hat{\boldsymbol{v}} = \boldsymbol{v}_0 \hat{+} \sum_{v=1}^{N^{\text{vec}}} w_v \boldsymbol{v}_v \tag{74}$$

where the summation is performed according to (73), with normalization of the vector pairs $[\boldsymbol{v}_0, \boldsymbol{v}_v]$ to obtain the rotations as described in (71).

4) Weighted Rotation Sequence: Let us assume that we have N^{rot} sequential rotations such that the output vector $\boldsymbol{b}_{o,n}$ of the element n is equal to the input vector of the next in the sequence $n+1$, i.e., $\boldsymbol{b}_{o,n} = \boldsymbol{b}_{i,n+1}$ with $n = 1..N^{\text{rot}}-1$. To evaluate the weighted sequence each basis is adapted with respect to all the parent rotations as:

$$\boldsymbol{b}_{\{i,o\},c} \leftarrow \boldsymbol{v}_r((w_n - 1)\beta_n, \boldsymbol{b}_{\{i,o\},c}) \quad n = 1..N^{\text{rot}}-1,\ c > n \tag{75}$$

The full weighted rotation of the sequence is obtained using the vector rotation as defined in (71) with input vector $\boldsymbol{b}_{i,1}$ and output $\boldsymbol{v}_{o,n} = \boldsymbol{v}_r((1-w_n)\beta_n, \boldsymbol{b}_{o,n})$ with $n = N^{\text{rot}}$.

5) Direction Tree Evaluation: Let us define a direction tree with nodes $n = 1..N^{\text{node}}$. There exists a single (unique) root node r, which has no parent. All other nodes n have single parent $p(n)$, but a node can have multile children, given by the set $\mathscr{C}_n$. The set of all offspring, i.e., including the children of children etc., is referred to as $\mathscr{C}_n^{\text{tot}}$. Each node has a level l, which indicates the discrete distance to the root. The value of each node corresponds to a unit vector $\boldsymbol{v}_i$, and hence the vector rotation between the parent and the node can be evaluated using (71). For given weights w_n, the weighted direction is evaluated as described in Algorithm 4. The rotation tree is interpreted as a combination of weighted summations and sequences, as described earlier in this section.

Lemma A.1. *The vector rotation as described in Algorithm 4 ensures a smooth vector summing for weights $w_n \in [0,1]$ such that $\sum_{n=1}^{N^{\text{node}}} w_n = 1$, such that we reach converge to a vector when its corresponding weight is approaching one, i.e.,* $\lim_{w_i \to 1} \boldsymbol{v}^* = \boldsymbol{v}_i$.

Proof. When a specific direction-node n has weight $w_n = 1$, it follows from Algorithm 4 all the child-nodes weight one and zero otherwise:

$$w_c = \begin{cases} 1 & \text{if } c \in \mathscr{C}_n \\ 0 & \text{otherwise} \end{cases} \tag{76}$$

Hence, we have a rotation sequence of the form (75). Due to the sum of the weights of all sequence elements being equal to one, the summed weight equals $\boldsymbol{b}_{o,e}$ where e is the index of the ending element. Note successive vectors cannot be anti-parallel, as denoted in (71). □

Algorithm 4 Rotation Tree Summing

Input: w_n, $\boldsymbol{v}_n$ for $n = 1..N^{\text{node}}$
Output: Averaged vector-rotation $\boldsymbol{v}^*$

for $l = L^{\max}..1$ **do** {Reverse iteration over levels}
  for $n : l_n = l$ **do** {For all nodes of the same level}
    $w_{p(n)} \leftarrow w_{p(n)} + w_n$ {Update cumulative weight}
  end for
end for
$\boldsymbol{v}_1^* = \boldsymbol{v}_r$ {Initialize base of root r}
for $l = 2..L^{\max}$ **do** {Iteration over levels}
  $\boldsymbol{v}_l^* = \boldsymbol{v}_{l-1}^* \hat{+} \sum_c^{l_c==l} \boldsymbol{v}_c$ {Weighted average from (74)}
  $\boldsymbol{v}_r(\cdot), \beta \leftarrow \boldsymbol{v}_{l-1}^*, \boldsymbol{v}_l^*$ {Rotation from vectors (71)}
  for $c \in \mathscr{C}_l^{\text{tot}}$ **do** {For all offsprings of l}
    $\boldsymbol{v}_{r,c}(\cdot), \beta_c \leftarrow \boldsymbol{v}_l^*, \boldsymbol{v}_c$ {Rotation from vectors (71)}
    $\boldsymbol{b}_{j,c} \leftarrow \boldsymbol{v}_{r,c}(-\beta_c, \boldsymbol{b}_{j,c})$ {Sequence inversion (75)}
    $\boldsymbol{b}_{j,c} \leftarrow \boldsymbol{v}_r(\beta, \boldsymbol{b}_{j,c})$ {Apply rotation to $\boldsymbol{v}_l^*$ as (72)}
  end for
end for
$\boldsymbol{v}^* = \boldsymbol{v}_{L^{\max}}^*$

Note, that the weighted evaluation of the direction tree can be reduced to a single vector $\boldsymbol{v}^*$, or the whole sequence can be kept by storing all $\boldsymbol{v}_{(\cdot)}^*$. If all node-parent pair had an angle $\beta < \pi$, then this will be the case for the resulting sequence, too.

REFERENCES

[1] V. M. Goncalves, L. C. Pimenta, C. A. Maia, B. C. Dutra, and G. A. Pereira, "Vector fields for robot navigation along time-varying curves in n-dimensions," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 647–659, 2010.

[2] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1462–1469, 2019.

[3] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, vol. 87, 2018.

[4] J.-J. E. Slotine, W. Li, *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199.

[5] S. S. M. Salehian and A. Billard, "A dynamical-system-based approach for controlling robotic manipulators during noncontact/contact transitions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2738–2745, 2018.

[6] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[7] K. Kronander and A. Billard, "Passive interaction control with dynamical systems," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 106–113, 2015.

[8] Y. A. Kapitanyuk, A. V. Proskurnikov, and M. Cao, "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1372–1385, 2017.

[9] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[10] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

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

[13] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[14] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[15] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[16] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, IEEE, 1993, pp. 802–807.

[17] M. Zucker, N. Ratliff, A. D. Dragan, *et al.*, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[18] N. Ratliff, M. Toussaint, and S. Schaal, "Understanding the geometry of workspace obstacles in motion optimization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 4202–4209.

[19] J. Mainprice, N. Ratliff, M. Toussaint, and S. Schaal, "An interior point method solving motion planning problems with narrow passages," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2020, pp. 547–552.

[20] I. Sánchez, A. D'Jorge, G. V. Raffo, A. H. González, and A. Ferramosca, "Nonlinear model predictive path following controller with obstacle avoidance," *Journal of Intelligent & Robotic Systems*, vol. 102, pp. 1–18, 2021.

[21] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[22] M. Bhardwaj, B. Sundaralingam, A. Mousavian, *et al.*, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*, PMLR, 2022, pp. 750–759.

[23] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2022.

[24] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE robotics and automation letters*, vol. 5, no. 3, pp. 4249–4256, 2020.

[25] D. Wang, S. Chen, Y. Zhang, and L. Liu, "Path planning of mobile robot in dynamic environment: Fuzzy artificial potential field and extensible neural network," *Artificial Life and Robotics*, vol. 26, pp. 129–139, 2021.

[26] M. Cai, E. Aasi, C. Belta, and C.-I. Vasile, "Overcoming exploration: Deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications," *IEEE Robotics and Automation Letters*, 2023.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] L. Huber, J.-J. Slotine, and A. Billard, "Fast obstacle avoidance based on real-time sensing," *IEEE Robotics and Automation Letters*, 2022.

[29] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[30] L. Huang, "Velocity planning for a mobile robot to track a moving target—a potential field approach," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 55–63, 2009.

[31] A. Tulbure and O. Khatib, "Closing the loop: Real-time perception and control for robust collision avoidance with occluded obstacles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5700–5707.

[32] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, IEEE Computer Society, 1991, pp. 1398–1399.

[33] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in applied mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[34] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Transactions of the American Mathematical Society*, vol. 327, no. 1, pp. 71–116, 1991.

[35] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[36] S. Paternain, D. E. Koditschek, and A. Ribeiro, "Navigation functions for convex potentials in a space with convex obstacles," *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 2944–2959, 2017.

[37] S. G. Loizou, "The navigation transformation," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1516–1523, 2017.

[38] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 2513–2518.

[39] J. P. Wilhelm and G. Clem, "Vector field uav guidance for path following and obstacle avoidance with minimal deviation," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1848–1856, 2019.

[40] W. Yao, B. Lin, B. D. Anderson, and M. Cao, "Guiding vector fields for following occluded paths," *IEEE Transactions on Automatic Control*, 2022.

[41] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, IEEE, 1990, pp. 2102–2106.

[42] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3 Ju, 1992.

[43] H. J. S. Feder and J.-J. Slotine, "Real-time path planning using harmonic potentials in dynamic environments," in *Proceedings of International Conference on Robotics and Automation*, IEEE, vol. 1, 1997, pp. 874–881.

[44] D. Zheng, X. Wu, Y. Liu, and J. Pang, "A dynamical system approach to real-time three-dimensional concave obstacle avoidance," in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2020, pp. 1082–1087.

[45] L. Huber, J.-J. Slotine, and A. Billard, "Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds," *IEEE Transactions on Robotics*, 2022.

[46] F. M. Conzelmann, L. Huber, D. Paez-Granados, A. Bolotnikova, A. Ijspeert, and A. Billard, "A dynamical system approach to decentralized collision-free autonomous coordination of a mobile assistive furniture swarm," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 7259–7265.

[47] H. Brunn, "Über kerneigebiete," *Mathematische Annalen*, vol. 73, no. 3, pp. 436–440, 1913.

[48] G. Hansen, I. Herburt, H. Martini, and M. Moszyńska, "Starshaped sets," *Aequationes mathematicae*, vol. 94, pp. 1001–1092, 2020.

[49] X. Luo, M. Small, M.-F. Danca, and G. Chen, "On a dynamical system with multiple chaotic attractors," *International Journal of Bifurcation and Chaos*, vol. 17, no. 09, pp. 3235–3251, 2007.

[50] S. R. Lindemann and S. M. LaValle, "Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 600–621, 2009.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Systems & Control Letters*, vol. 96, pp. 51–59, 2016.

[53] D. E. Knuth, *The art of computer programming*. Pearson Education, 1997, vol. 3.

[54] A. Dahlin and Y. Karayiannidis, "Creating star worlds–modelling concave obstacles for reactive motion planning," *arXiv preprint arXiv:2205.09336*, 2022.

[55] P. M. Hubbard, "Approximating polyhedra with spheres for time-critical collision detection," *ACM Transactions on Graphics (TOG)*, vol. 15, no. 3, pp. 179–210, 1996.