Avoiding Dense and Dynamic Obstacles in Enclosed Spaces: Application to Moving in Crowds

Lukas Huber, Jean-Jacques Slotine, and Aude Billard, *Fellow, IEEE*

***Abstract*—This article presents a closed-form approach to constraining a flow within a given volume and around objects. The flow is guaranteed to converge and to stop at a single fixed point. The obstacle avoidance problem is inverted to enforce that the flow remains enclosed within a volume defined by a polygonal surface. We formally guarantee that such a flow will never contact the boundaries of the enclosing volume or obstacles. It asymptotically converges toward an attractor. We further create smooth motion fields around obstacles with edges (e.g., tables). Both obstacles and enclosures may be time-varying, i.e., moving, expanding, and shrinking. The technique enables a robot to navigate within enclosed corridors while avoiding static and moving obstacles. It was applied on an autonomous robot (QOLO) in a static complex indoor environment and tested in simulations with dense crowds. The final proof of concept was performed in an outdoor environment in Lausanne. The QOLO-robot successfully traversed a marketplace in the center of town in the presence of a diverse crowd with a nonuniform motion pattern.**

***Index Terms*—Autonomous agents, collision avoidance, crowd navigation, dynamical systems, mobile robots.**

I. Introduction

ROBOTS navigating in human-inhabited environments will encounter disturbances constantly, for instance, when autonomous delivery robots drive around pedestrians. The robot must have a flexible control scheme to avoid collisions. As the number of obstacles increases and their motion becomes less predictable, the robot needs to reevaluate its path within milliseconds to prevent a crash while moving actively toward its goal.

Control using dynamical systems (DS) is ideal for addressing such situations. In contrast to classical path planning, the control law is closed-form, hence requires no replanning, and can ensure impenetrability of obstacles [1], [2]. DS offer stability and convergence guarantees in addition to the desired on-the-fly reactivity.

This article addresses the need for a reactive (closed-form) obstacle avoidance approach with formal guarantees of impenetrability to handle highly dynamic environments and realistic obstacles, such as obstacles with sharp edges. To this end, this article extends our previous work [3], in which we presented a closed-form obstacle avoidance approach guaranteed to not penetrate smooth concave, albeit star-shaped, obstacles. We present three novel theoretical contributions as follows:

1) We extend the modulation parameters of the obstacle avoidance with surface friction and repulsive value (Section III). They allow an agent to move slower and further away from obstacles, respectively. The extensions result in more *cautious* behavior.
2) We invert the obstacle description to ensure that the robot moves within the enclosed space defined by the boundary, while preserving convergences guarantees toward an attractor. This boundary may represent walls, furniture, or even joint limits of manipulators (Section IV).
3) We extend the approach to handle *nonsmooth* surfaces, i.e., obstacles with sharp edges. The novelty comes from creating a smooth dynamical system around an obstacle without approximation of the curvature (Section V).
4) We show that the approach can be extended to tackle dynamic environments, with obstacles that have *deforming* shapes (Section VI).
5) We propose an approach on how to use the dynamical obstacle avoidance in combination with a robot arm (Section VII). The safe joint velocity is evaluated based on the proximity function combined with the obstacle avoidance algorithm.

We validate these contributions with a wheelchair robot moving in a simulated crowd of pedestrians and an office environment with real furniture (Section IX).

II. Related Work

A. Sampling Based Exploration

Sampling algorithms, such as probabilistic road map (PRM) [4] or the rapidly exploring random trees (RRT) [5] can find a path in cluttered environments. They are computationally intensive, and early approaches were limited to static environments. In [6], online (partial) replanning and elastic-band methods deform the path locally. This allowed adapting to dynamic environments. Reinforcement learning allowed Zhang

Manuscript received December 14, 2021; revised March 23, 2022; accepted March 28, 2022. This work was supported by the EU ERC under Grant SAHR. This paper was recommended for publication by Associate Editor F. Stulp and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: Lukas Huber.)*

Lukas Huber and Aude Billard are with the LASA Laboratory, Swiss Federal School of Technology in Lausanne - EPFL, CH-1015 Lausanne, Switzerland (e-mail: lukas.huber@epfl.ch; aude.billard@epfl.ch).

Jean-Jacques Slotine is with the Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: jjs@mit.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2022.3164789.

Digital Object Identifier 10.1109/TRO.2022.3164789

et al. [7] to learn from previously explored PRM-paths and efficiently adapt to dynamic obstacles. However, the switching often comes with the theoretical loss of global convergence [8]. Recent work uses customized circuitry on a chip for onboard, fast global sampling and evaluation [9]. This method allows fast replanning, but a customized chip design is required for each robot configuration.

B. Real-Time Optimization

With improvements in hardware and computational speed, optimization algorithms, such as model predictive control (MPC) have become feasible for onboard use in dynamic path planning, and obstacle avoidance [10]. MPC has been used to control nonholonomic robots in environments with multiple convex obstacles [11]. Most optimization methods cannot guarantee convergence to a feasible solution at runtime.

Power diagrams were used to identify the robot's collision-free, convex neighborhood, and an associated, well-known convex optimization problem generates a continuous flow [12]. This method is limited to convergence for convex obstacles with almost spherical curvature.

Control barrier functions (CBFs) and control Lyapunov functions (CLFs) were united through the use of quadratic programming (QP) to create collision-free paths in [13]. The convergence constraint was softened to ensure the feasibility of the optimization problem. As a result, full convergence cannot be guaranteed anymore. In [14], local minima were overcome by introducing a (virtual) orientation state. This orientation state introduces a dependence on the history of the QP problem. In addition, the authors do not address the challenge of finding an appropriate Lyapunov candidate.

In [15] a diffeomorphic transformation to a sphere-world is used, which is based on [16]. The method introduces a dependence on its history through virtual obstacle positions and radii. Furthermore, the construction of the diffeomorphism requires full knowledge about the space.

C. Learning-Based Approaches

With the rising popularity of machine learning in the past years, these algorithms have been applied to sensor data to infer data-driven control [17] but this method cannot ensure impenetrability. Other approaches use neural networks on a circular representation of crowds to create steering laws but cannot guarantee convergence [18].

D. Velocity Obstacles

Velocity obstacles extend the obstacle shape by its potential future positions based on the current velocity [19]. Velocity obstacles allow safe navigation in dynamic environments. They were successfully applied in multiagent scenarios [20] and extended to include acceleration, and nonholonomic constraints of the agent [21]. The velocity obstacle approach often conservatively limits the workspace.

E. Artificial Potential Fields and Navigation Functions

Artificial potential fields were used to create collision-free trajectories [22], but they are prone to local minima. In [23], artificial potential fields for sphere-worlds were designed to have only a global minimum. A diffeomorphic transformation was introduced to map *star-worlds* to *sphere-worlds* [16], and extended to include *trees of stars* [24]. The tuning of critical parameters needs the knowledge of the whole space. Hence, in practice, full convergence is difficult to achieve.

More recent approaches introduce artificial potential fields with only the global minimum for more general environments [25], [26]. Automated approximation of the tuning parameter has been proposed [27], but it does not generalize easily to dynamic environment. In [28], full convergence is ensured around ellipse obstacles through quadratic potential functions. Learning methods were used to tune the hyperparameters of potential fields to obtain human-inspired behavior for obstacle avoidance [29].

Dangerfields were used in [30] to ensure collision avoidance for robot arms through repulsion from dynamic obstacles. The approach was extended by guiding the motion through an artificial potential field in [31]. The design of the artificial potential field remains a challenge for this method. Dynamic reference points help to reduce the probability of converging to a local minimum for potential fields [32].

F. Harmonic Potential Fields

Harmonic potential functions are interesting as they guarantee that no topologically critical points arise in free space. In [33], the harmonic potential functions were evaluated numerically to overcome the challenge of finding them analytically. Closed-form harmonic potential functions can be generated by approximating the obstacles through linear *panels* [34]. This linear approximation applies to concave obstacles but is limited to 2-D environments [1]. In [35], known harmonic potential functions are interpolated to navigate in more complex environments. A closed-form solution to harmonic potential flow around simple obstacles was presented in [36]. The work allows to avoid moving obstacles but is limited to convex. In [37], the approach was extended to concave obstacles by using a discrete, sensor-based representation. Closed-form approaches using harmonic potential fields are often simplified to a circular world or require high (close to circular) curvature.

In [3], a dynamic reference point was introduced to ensure convergence for star-shaped environments. However, the approach was not able to handle boundaries or deforming obstacles.

III. Obstacle Avoidance Formulation

Dynamical system-based obstacle avoidance has been proposed by the authors in [3]. We restate previous definitions, and introduce two new extensions: a *friction parameter* (Section III-F) and a *repulsion parameter* (Section III-G).

A. Notation

The state variable $\xi \in \mathbb{R}^d$ defines the state of a robotic system. If not mentioned otherwise, ξ will refer to the Cartesian position of the agent.

The variable π is used as the circle constant throughout the article.

Superscripts are used to denote the name of variables and subscripts for enumeration.

Bold-face Latin characters describe vectors and matrices.

The brackets $\langle \cdot, \cdot \rangle$ denote the dot product of two vectors.

The $\times$-operator indicates the cross product.

The square brackets indicate elements of a vector, e.g., $\xi_{[1]}$ denotes the first element of ξ. Double dots within the brackets indicate a subvector up to the specified number, e.g., $\xi_{[1:2]}$ is a vector of the first two elements of ξ.

B. Dynamical Systems

This work focuses on motion toward a goal of an autonomous dynamical system, i.e., $\lim_{t\to\infty} f(\xi_a) = 0$. The most direct dynamics toward the attractor is a linear dynamical system of the form

$$\mathbf{f}(\xi) = -k(\xi - \xi^a) \tag{1}$$

where $k \in \mathbb{R}$ is a scaling parameter. The attractor ξ^a is visualized throughout this work as a star: $*$ and set $k = 1$.

C. Obstacle Description

Each obstacle has a continuous distance function $\Gamma(\xi) : \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$, which allows to distinguish three regions as follows:

$$\begin{aligned} &\text{Free points:} && \mathscr{X}^f = \{\xi \in \mathbb{R}^d : \Gamma(\xi) > 1\} \\ &\text{Boundary points:} && \mathscr{X}^b = \{\xi \in \mathbb{R}^d : \Gamma(\xi) = 1\} \\ &\text{Interior points:} && \mathscr{X}^o = \{\xi \in \mathbb{R}^d \setminus (\mathscr{X}^f \cup \mathscr{X}^b)\}. \end{aligned} \tag{2}$$

1) Reference Point: For each obstacle i a reference point is chosen such that it lies within the kernel of the obstacle: $\xi_i^r \in \mathscr{X}_i^o$.[1] The reference direction is defined as

$$\mathbf{r}_i(\xi) = (\xi - \xi_i^r) \,/ \|\xi - \xi_i^r\| \quad \forall \xi \in \mathbb{R}^d \setminus \xi_i^r. \tag{3}$$

The reference point is visualized throughout this work as a cross $+$.

2) Distance Function: By construction, the distance function $\Gamma(\cdot)$ increases monotonically in radial direction and has a continuous first-order partial derivative (C^1 smoothness). Here, we define the general distance function as

$$\Gamma^o(\xi) = (\|\xi - \xi^r\| / R(\xi))^{2p} \quad \forall \xi \in \mathbb{R}^d \setminus \xi^r \tag{4}$$

with the power coefficient $p \in \mathbb{N}_+$. The local radius $R(\xi) = \|\xi^b - \xi^r\|$ is a function of the local boundary point, which is defined as

$$\xi^b = b\mathbf{r}(\xi) + \xi^r \quad \text{such that} \quad b > 0\,, \quad \xi^b \in \mathscr{X}^b. \tag{5}$$

[1]The kernel of a star-shaped obstacle defines the region from which any surface point is visible [38].

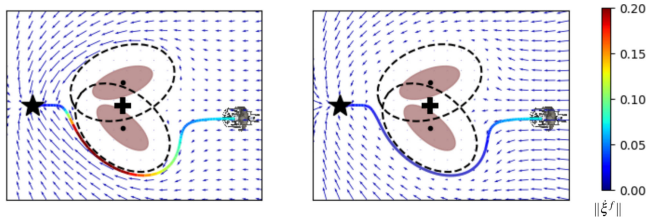


Fig. 1. Velocity obtained with the isometric-inspired eigenvalue (left) results in accelerations close to the obstacle. The modulation inspired by surface friction (right) reduces the velocity with decreasing distance to the obstacle. The black star is the attractor ξ^a, and the black cross is the (shared) reference point ξ^r.

D. Obstacle Avoidance Through Modulation

Real-time obstacle avoidance is obtained by applying a dynamic modulation matrix to a dynamical system $\mathbf{f}(\xi)$

$$\dot{\xi} = \mathbf{M}(\xi)\mathbf{f}(\xi) \qquad \text{with} \quad \mathbf{M}(\xi) = \mathbf{E}(\xi)\mathbf{D}(\xi)\mathbf{E}(\xi)^{-1}. \tag{6}$$

The modulation matrix is composed of the basis matrix

$$\mathbf{E}(\xi) = [\mathbf{r}(\xi) \;\; \mathbf{e}_1(\xi) \;\; .. \;\; \mathbf{e}_{d-1}(\xi)] \tag{7}$$

which has the orthonormal tangent vectors $\mathbf{e}_i(\xi)$ evaluated at the boundary point ξ^b given in (5).

The diagonal eigenvalue matrix is given as

$$\mathbf{D}(\xi) = \mathbf{diag}\left(\lambda^r(\xi), \lambda^e(\xi), \ldots, \lambda^e(\xi)\right). \tag{8}$$

We set the eigenvalues to

$$\lambda^r(\xi) = 1 - 1/\Gamma(\xi)^{1/\rho} \qquad \lambda^e(\xi) = 1 + 1/\Gamma(\xi)^{1/\rho} \tag{9}$$

with the reactivity factor $\rho \in \mathbb{R}_{>0}$ and the distance function $\Gamma(\xi)$ from (4). In this work, we simply choose $\rho = 1$.

E. Multiple Obstacles

In the presence of multiple obstacles, the velocity is modulated for each obstacle individually as described in (6). The final velocity is obtained by taking the weighted directional mean of the individual velocities; see Appendix-A.

F. Surface Friction Imitation

The choice of eigenvalues in (9) is inspired by the harmonic potential flow, i.e., the description of the potential flow of an incompressible fluid. The incompressibility constraint forces the velocity to increase in regions where the flow is pushed around the obstacle, i.e., the eigenvalues in the tangent direction increase. This leads to acceleration close to the surface (see Fig. 1).

We propose to mimic surface friction, i.e., slowing down in tangent direction close to an obstacle ($\lim_{\Gamma\to 1} \xi^e = 0$). A friction parameter $\lambda^f(\xi)$ ensures the slowing down close to the surface. The friction dynamics $\dot{\xi}^f$ are obtained by applying the factor to tangent and reference direction as follows:

$$\dot{\xi}^f = \lambda^f(\xi)\frac{\|\mathbf{f}(\xi)\|}{\|\dot{\xi}\|}\dot{\xi} \qquad \text{with} \quad \lambda^f(\xi) = 1 - 1/\Gamma(\xi) \tag{10}$$

where the $\dot{\xi}$ is the modulated velocity from (6).

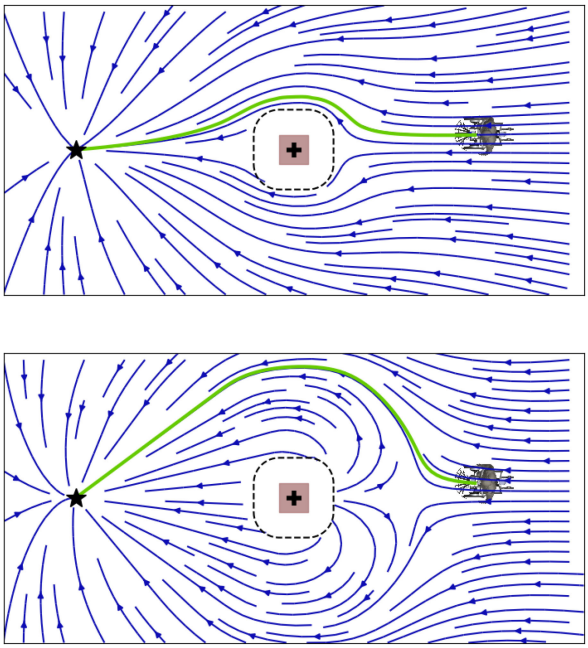


Fig. 2. Repulsion coefficient of $c^{\text{rep}} = 1$ results in strictly positive eigenvalues (top). An increased distance to the obstacle is obtained with higher repulsion coefficients, such as $c^{\text{rep}} = 2$ (bottom).

G. Repulsive Eigenvalue

The eigenvalues in reference direction given in (9) are always positive, as follows from $\Gamma(\xi) \geq 1$. This results in eigenvalues in the range of $\lambda^r(\xi) \in [0, 1]$, hence reduces the velocity in radial direction.

Conversely, active repulsion can be achieved through negative eigenvalues, i.e., $\lambda^r(\xi) < 0$. Active repulsion increases the distance by which the an agents avoids the obstacle (Fig. 2). Since the repulsive eigenvalues are incorporated in the modulation matrix in (6), it is ensured that attractors are preserved.

The eigenvalue in radial direction is defined for repulsive obstacles as

$$\lambda^r(\xi) = \begin{cases} 1 - (c^{\text{rep}}/\Gamma(\xi))^{1/\rho} & \text{if } \langle \mathbf{f}(\xi), \mathbf{r} \rangle < 0 \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

with the repulsive coefficient $c^{\text{rep}} \geq 1$. A repulsive coefficient $c^{\text{rep}} = 1$ corresponds to no repulsion. Note, that the repulsive eigenvalues are coupled with no *tail effect*, i.e., $\lambda^r(\xi) = 0$ in the wake of an obstacle (see [36]).

IV. Inverted Obstacle Avoidance

An autonomous robot often encounters scenarios, where it has boundaries that it cannot pass. This might be a wall for a wheeled robot or the joint limits for a robot arm. These constraints can be interpreted as staying within an obstacle, where the obstacle represents the limits of the free space.

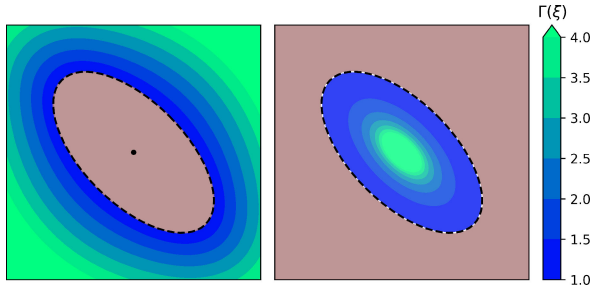


Fig. 3. Forward and inverted Γ-function for the same shape. The brown region marks the inside of the obstacle and wall, respectively.

A. Distance Inversion

The distance function $\Gamma(\xi)$ from (4) can be evaluated within the obstacle $\xi \in \mathscr{X}^o$.[2] For interior points, our boundary function is monotonically decreasing along the radial direction, i.e., the Lie derivative with respect to the reference direction is positive. Furthermore it is bounded. This can be written as

$$L_{\mathbf{r}}\Gamma = \left\langle \frac{\partial \Gamma(\xi)}{\partial \mathbf{r}(\xi)}, \mathbf{r}(\xi) \right\rangle > 0, \qquad \Gamma(\xi) \in [0, 1[\quad \forall\, \mathscr{X}^o \setminus \xi^r.$$

We consider the obstacle boundary $\mathscr{X}^b$ as the description of an enclosing hull. It follows that the interior points of the classical obstacle become points of free space of the enclosing hull and vice versa. Boundary points stay boundary points. We define the distance function of wall obstacles as the inverse of the obstacle distance function

$$\Gamma^w(\xi) = 1/\Gamma^o = (R(\xi)/\|\xi - \xi^r\|)^{2p} \qquad \forall\, \mathbb{R}^d \setminus \xi^r. \tag{12}$$

This new distance function fulfills the condition for the three regions as given in (2). The distance function Γ^w is now monotonically decreasing along radial direction and reaches infinity at the reference point, i.e., $\lim_{\xi \to \xi^r} \Gamma^w(\xi) \to \infty$ (see Fig. 3).

B. Modulation Matrix

The modulation matrix, defined in (6), consists of the eigenvalue $\mathbf{D}(\xi)$ and basis matrix $\mathbf{E}(\xi)$. For inverted obstacles, the diagonal eigenvalue matrix from (8) is a function of the inverted distance function $\Gamma^w(\xi)$.

Conversely, the basis matrix is constant along the radial direction. It can be evaluated everywhere (including the interior of a boundary) except at the reference point ($\xi = \xi^r$). Since the reference direction from (3) is a zero vector, no orthogonal basis is defined. However, the distance value $\Gamma^w(\xi^r)$ reaches infinity, hence from (8) we get that the diagonal matrix is equal to the identity matrix

$$\Gamma^w(\xi^r) \to \infty \quad \Rightarrow \quad \mathbf{D}(\xi^r) = \mathbf{I}. \tag{13}$$

Using (6), it follows for the modulated dynamics at the reference point:

$$\mathbf{M}(\xi^r) = \mathbf{E}(\xi^r)\mathbf{D}(\xi^r)\mathbf{E}(\xi^r)^{-1} = \mathbf{E}(\xi^r)\mathbf{I}\mathbf{E}(\xi^r)^{-1} = \mathbf{I}$$
$$\Rightarrow \dot{\xi}^r = \mathbf{f}(\xi^r). \tag{14}$$

[2] In the classic obstacle avoidance case, this is of no use, since theoretically the obstacle does never reach the boundary [3], and practically an *emergency* control has to be applied in this case.

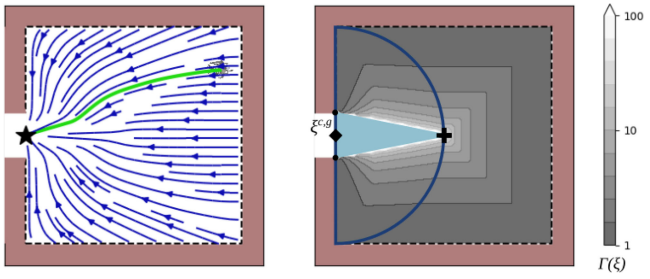


Fig. 5. Dynamical system (left) is not modulated in front of the gap since the Γ-function reaches infinity (right). The *gap region* $\mathscr{X}^g$ is the blue region. The influence of the gap is limited by the blue half-circle around the center of the gap $\xi^{c,g}$ (black square).

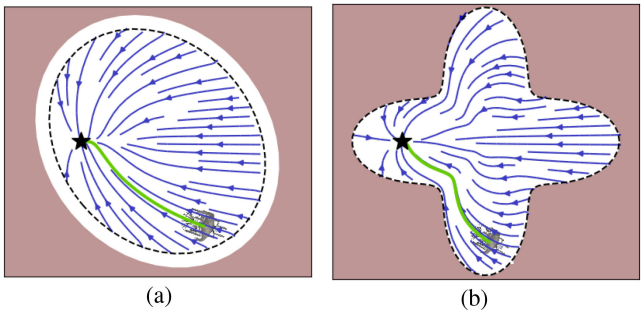


Fig. 4. Smooth flow with full convergence toward the attractor (black star) can be observed within any star-shaped wall with reference point ξ^r (black plus). (a) Ellipsoid boundary. (b) Star-shaped boundary.

The influence of modulation approaches zero when reaching the reference point. Hence the dynamical modulation is continuously defined as can be seen in the the example environments in Fig. 5.

Theorem 1 *Consider a star-shaped enclosing wall in* $\mathbb{R}^d$ *with respect to a reference point inside the obstacle* ξ^r*, and a boundary* $\Gamma^w(\xi) = 1$ *as in (12). Any trajectory* $\{\xi\}_t$*, that starts within the free space of an enclosing wall, i.e.,* $\Gamma(\{\xi\}_0) > 1$ *and evolves on a smooth path according to (6), will never reach the wall, i.e.,* $\Gamma(\{\xi\}_t) > 1,\ t = 0..\infty$ *and converges toward an attractor* $\xi^a \in \mathscr{X}^f$*, i.e.,* $\lim_{t\to\infty} \xi \to \xi^a$. **Proof:** See Appendix B.

C. Guiding Reference Point to Pass Wall Gaps

In many practical scenarios, a hull entails gaps or holes through which the agent enters or exits the space (e.g., door in a room). The modulation-based avoidance slows the agent down when approaching the boundary and does not let it pass through such an exit. We introduce a *guiding reference point* ξ^g for boundary obstacles to counter this effect. We assume convex walls, and the robot size being smaller than the gap width. It is assumed that gap is *a priori* known from a map.

In (14), it was shown that at the center of an inverted obstacle, the influence of the modulation vanishes.

Let us define the *gap region* $\mathscr{X}^g$ enclosed by the lines connecting the gap edges and the reference point ξ^r (see Fig. 5). The guiding reference point is designed in the following manner: close to the gap, the guiding reference point is equal to the position of the evaluation, hence no influence of the modulation. Far away from the gap, the guiding reference point is equal to the reference point; hence the wall has no influence. In between the two regions, the guiding reference point is projected onto the gap region $\mathscr{X}^g$. This can be written as

$$\xi^g = \begin{cases} \xi & \text{if} \quad \xi \in \mathscr{X}^g \\ \text{argmin}_{\hat{\xi}\in\mathscr{X}^g} \|\xi - \hat{\xi}\| & \text{else if } \frac{\|\xi^{c,g}-\xi\|}{\|\xi^{c,g}-\xi^r\|} > 1 \\ \xi^r & \text{otherwise} \end{cases}$$

with $\xi^{c,g}$ the center point of the gap (see Fig. 5).

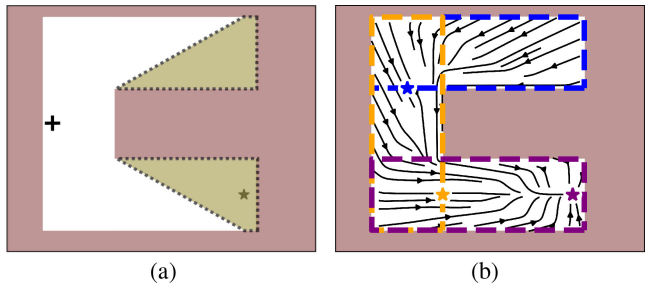


Fig. 6. For the navigation in a concave corridor (brown), the boundary can be extended to be of *star shape*; see the yellow patches in (a). Alternatively, the motion can be divided into partial boundaries, the blue, orange, and purple rectangles. They each have a corresponding dynamical systems with attractors; see the black stars (b). (a) Boundary extensions. (b) Consecutive dynamics.

D. Concave Environments

The presented obstacle avoidance algorithm applies to *star-shaped* environments. The extension of environments with various concave obstacles to fulfill the constraints is described in [3].

Similarly, the presented method for inverted obstacles requires *star-shaped* boundaries. A general concave wall can be extended to meet the constraints [see Fig. 6(a)]. However, this extension results in certain regions not being accessible anymore, i.e., the yellow patches.

Alternatively, we propose to combine the modulation avoidance algorithm with a high-level planner. The planner divides the boundary into several subboundaries with corresponding local dynamics. For this, we use the method to pass gaps in a wall as described in Section IV-C to pass from one local environment to the next one. The modulated DS can be summed according to the local weights

$$w_i^p(\xi) = \begin{cases} \hat{w}_i^p(\xi) / \sum_i \hat{w}_b^p(\xi) & \text{if} \quad \sum_i \hat{w}_i^p(\xi) > 0 \\ \hat{w}_i^p(\xi) = 0 & \text{otherwise} \end{cases}$$

$$\text{with} \quad \hat{w}_i^p(\xi) = \max\left(\Gamma_i(\xi) - 1,\, 0\right).$$

The high-level planners, which decompose the environment autonomously is part of ongoing research.

V. Nonsmooth Surfaces

Human-designed environments often contain obstacles and enclosing walls with nonsmooth surfaces, e.g., a table with edges or a building with corners. An approximation with a high gradient of these surfaces can lead to undesired results.

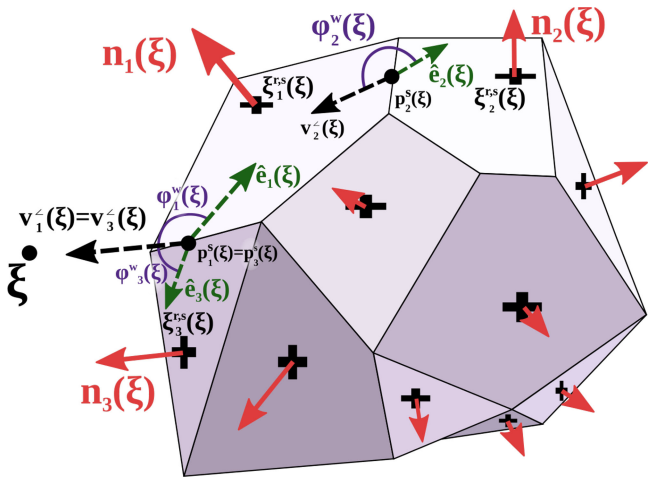


Fig. 7. Variables for the evaluation of the pseudonormal $\hat{\mathbf{n}}(\xi)$ of a nonsmooth star-shaped obstacle are displayed for three surface tiles. Only the surface reference point (cross) and the surface normal (red arrow) are visualized for the other tiles. The angle φ^w is evaluated for each surface at the edge-point closest to ξ.

On the one hand, a smoothing of the edges increases the risk of colliding with them. On the other hand, an increased, smooth hull conservatively increases the boundary region, and certain parts of the space are not reachable with such a controller.

Moreover, the obstacle avoidance algorithm applied to a surface with a high gradient can lead to a fast change of the flow. Even though the trajectories are smooth, the curvature of the flow can be high and induce fast accelerations. This can result in dangerous behavior in the presence of humans or simply exceed the robot's torque limits. We propose an approach to avoid obstacles with nonsmooth surfaces without smoothing the boundary.

A polygonal obstacle consists of $i = 1 \ldots N^s$ individually smooth surface planes which form a star shape in $d-1$ such that

$$\mathscr{X}_i^s = \{\xi, \hat{\xi} \in \mathscr{X}^b,\ \exists\, \mathbf{n}_i\ :\ \langle \mathbf{n}_i,\ (\xi - \hat{\xi})\rangle = 0\}. \tag{15}$$

A. Pseudonormal Vector

The normal to the surface of the obstacle is not defined continuously. As a result, the modulated flow would not be smooth.

We create a smoothly defined pseudonormal $\hat{\mathbf{n}}(\xi)$. It is equal to the normal on the surface of the obstacle. While far away from the obstacle, the pseudonormal approaches the reference direction, i.e.,

$$\hat{\mathbf{n}}(\xi) = \mathbf{n}_i(\xi)\ \ \forall \xi \in \mathscr{X}_i^s \quad \text{and} \quad \lim_{\|\xi - \xi^r\| \to \infty} \hat{\mathbf{n}}(\xi) = \mathbf{r}(\xi). \tag{16}$$

The pseudonormal is the weighted sum of the normals, with the weights being evaluated as follows. At first the vector from the closest edge point $\mathbf{p}_i^s$ (Fig. 7) to the agent's state is created

$$\mathbf{v}_i^{\angle}(\xi) = \xi - \mathbf{p}_i^s \quad \text{with} \quad \mathbf{p}_i^s = \underset{\hat{\xi} \in \mathscr{X}_i^e}{\operatorname{argmin}} \|\xi - \hat{\xi}\| \tag{17}$$

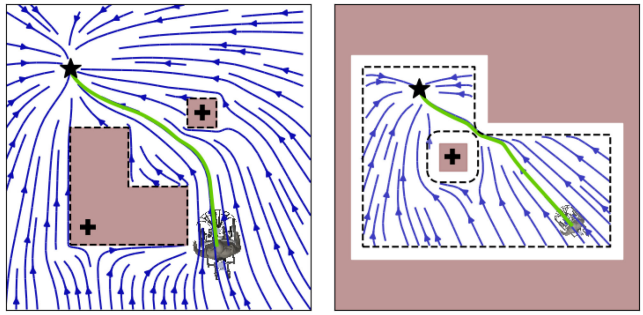


Fig. 8. Nonsmooth inverted obstacles representing rooms or boundary conditions. (a) Nonsmooth obstacles. (b) Nonsmooth boundaries.

with $\mathscr{X}_i^e$ the set of all points at the edge of a surface tile i. This vector is further projected onto the surface plane

$$\hat{\mathbf{e}}_i(\xi) = \left(\mathbf{v}_i^{\angle}(\xi) - \langle \mathbf{n}_i(\xi),\ \mathbf{v}_i^{\angle}(\xi)\rangle\, \mathbf{n}_i(\xi)\right) \operatorname{sign}\langle \mathbf{v}_i^{\angle}(\xi),\ \xi - \xi_i^{r,s}\rangle.$$

The angle to the plane (Fig. 7) in the range $[0, \pi]$ is evaluated as

$$\varphi_i^w(\xi) = \arccos\left(\frac{\langle \hat{\mathbf{e}}_i(\xi),\ \mathbf{v}_i^{\angle}(\xi)\rangle}{\|\hat{\mathbf{e}}_i(\xi)\|\ \|\mathbf{v}_i^{\angle}(\xi)\|}\right) \operatorname{sign}\langle \mathbf{n}_i(\xi),\ \mathbf{v}_i^{\angle}(\xi)\rangle. \tag{18}$$

The edge weight is evaluated as

$$\tilde{w}_i^s(\xi) = \begin{cases} \left(\frac{\pi}{\varphi_i^w(\xi)}\right)^p - 1 & \text{if } \varphi_i^w(\xi) \in]0,\ \pi] \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

with the the weight power $p \in \mathbb{R}$, a free parameter. A lower weight power p results in an increased importance for the closest polygon face compared to the other faces, we simply choose $p = 3$. The final step is the normalization of the weights

$$w_i^s(\xi) = \begin{cases} \tilde{w}_i^s(\xi) / \sum_j \tilde{w}_j^s(\xi) & \text{if } \xi \in \mathbb{R}^d \setminus \mathscr{X}_i^s \\ 1 & \text{otherwise.} \end{cases} \tag{20}$$

The pseudonormal is evaluated as the directional weighted mean (see Appendix A) of normal vectors of the surface tiles $\mathbf{n}_i(\xi)$, the weights w_i^s, and with respect to the reference direction $\mathbf{r}(\xi)$.

The basis matrix from (7) is redefined for nonsmooth surfaces as

$$\mathbf{E}(\xi) = [\mathbf{r}(\xi)\ \ \hat{\mathbf{e}}_1(\xi)\ \ ..\ \ \hat{\mathbf{e}}_{d-1}(\xi)] \tag{21}$$

with $\hat{\mathbf{e}}_i(\xi)$ the orthonormal basis to $\hat{\mathbf{n}}(\xi)$. The resulting smooth vectorfield can be observed in Fig. 8.

B. Inverted Obstacles

For an inverted obstacle (Section IV) the pseudonormal is evaluated at the mirrored position ξ^{mir}. It is obtained by flipping the current robot state ξ along the reference direction $\mathbf{r}(\xi) = \xi - \xi^r$ onto the other side of the boundary (4)

$$\xi^{\text{mir}} = \Gamma(\xi)^2\,(\xi - \xi^r) + \xi^r. \tag{22}$$

The mirrored position allows the evaluation of the distance function, as described in Section V-A. Further, the inverted obstacle is treated, as described in Section IV. This allows avoiding nonsmooth obstacles and boundaries in Fig. 4.

Theorem 2 *Consider a polygon composed of N^s surfaces as given in* (15) *or alternatively an inverted polygon, as described Section V-B. Any trajectory $\{\xi\}_t$, that starts in free space, i.e., $\Gamma(\{\xi\}_0) > 1$ and evolves on a smooth path according to (6), will never reach the surface, i.e., $\Gamma(\{\xi\}_t) > 1, t = 0..\infty$ and will converge toward the attractor as long as it is placed outside all obstacles, i.e., $\lim_{t\to\infty} \xi \to \xi^a \in \mathscr{X}^f$.* **Proof:** See Appendix C.

C. Implementation

Pseudonormals are useful for surroundings with sharp boundaries, for example, pieces of furniture with edges or maps of buildings with sharp corners. These cases consist of polygons with a small number of faces. They allow the evaluation of the algorithm in real time. It is designed for scenarios with obstacles known from previously learned libraries or a known map retrieved at runtime, such as the tracker of [39].

Conversely, if the input data are a point cloud (e.g., Lidar) or an obstacle mesh, the surface can be approximated using a standard regression technique. The surface normal can be obtained directly by taking the derivative or by learning the normal at regression time [40].

VI. Dynamic Environments

In changing environments with moving or deforming obstacles the system is modulated with respect to the relative velocity as

$$\dot{\xi} = \mathbf{M}(\xi)\left(\mathbf{f}(\xi) - \dot{\tilde{\xi}}^{\text{tot}}\right) + \dot{\tilde{\xi}}^{\text{tot}}. \tag{23}$$

The local, relative velocity is summed up over all obstacles

$$\dot{\tilde{\xi}}^{\text{tot}} = \sum_{o=1}^{N_o} w_o^l \dot{\tilde{\xi}}_o \tag{24}$$

with the dynamic weight being a function of the distance $\Gamma(\xi)$

$$w_o^l = \frac{w_o^l}{\sum_o \tilde{w}_o^l} \quad \text{with} \quad \tilde{w}_o^l = \frac{1}{\Gamma_o(\xi) - 1} \quad \forall\, \Gamma_o(\xi) > 1.$$

The relative velocity consists of the obstacle's velocity $\dot{\tilde{\xi}}_o^v$ and deformation $\dot{\tilde{\xi}}_o^d$

$$\dot{\tilde{\xi}}_o = \dot{\tilde{\xi}}_o^v + \dot{\tilde{\xi}}_o^d. \tag{25}$$

Note that avoiding dynamic obstacles is not only a modulation of the DS, i.e., a matrix multiplication. This can result in the velocity at the attractor being nonzero, even though the initial dynamical system has zero value there: $\mathbf{f}(\xi^a) = \mathbf{0}$.

For the rest of this section, we will assume the application to each obstacle implicitly without using the subscript $(\cdot)_o$.

A. Moving Obstacles

The relative velocity of a moving obstacle is obtained similarly to [41]

$$\dot{\tilde{\xi}}^v = \dot{\xi}^{L,v} + \dot{\xi}^{R,v} \times \tilde{\xi}. \tag{26}$$

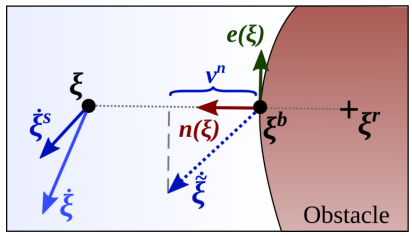


Fig. 9. In order to comply with a velocity limit of the robot, while avoiding a collision with an obstacle of velocity $\dot{\xi}$, the modulated velocity $\dot{\xi}$ might be stretched only in normal direction to obtain the safe velocity command $\dot{\xi}^s$.

The linear velocity $\dot{\xi}^{L,v}$ and angular velocity $\dot{\xi}^{R,v}$ are with respect to the center point of the obstacle ξ^c. The relative position is $\tilde{\xi} = \xi - \xi^c$.

B. Deforming Obstacle

Obstacles and hulls can not only move but they can also change their shape with respect to time, e.g., breathing body for a surgery robot. Conversely, the deformation of the perceived obstacle can be the result of uncertainties in real-time obstacle detection and position estimation.

The deformation velocity of an obstacle is evaluated as

$$\dot{\tilde{\xi}}^d = \dot{\xi}^{L,d} + \dot{\xi}^{R,d} \times \tilde{\xi}^d \tag{27}$$

where the linear velocity $\dot{\xi}^{L,d}$ and angular velocity $\dot{\xi}^{R,d}$ are evaluated on the surface position in reference direction, and $\tilde{\xi}^d = \xi - \xi^b$ is the relative position with respect to the boundary point (see Fig. 9).

The surface deformation should be explicitly given to the algorithm whenever it is known. Alternatively, it can be estimated from sensor readings, such as Lidar or camera.

1) Repulsive Mode: In many scenarios, the consideration of the obstacle's deformation is only of importance when it reduces the robot's workspace, and puts the robot at risk. It is sufficient to consider the deformation only along positive normal direction. For example for a circular object, we have

$$\dot{\tilde{\xi}}^d = \dot{\xi}^{L,d} = \begin{cases} \dot{r}\ \mathbf{n}(\xi) & \dot{r} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

where $\dot{r}$ is the rate of change of the circle radius.

C. Impenetrability With Respect to Maximum Velocity

Many agents have a maximum velocity they can move with, further referred to as $v^{\max}$. This limits the motion of obstacles which an agent can avoid to

$$v^n = \langle \dot{\tilde{\xi}}, \mathbf{n}(\xi) \rangle < v^{\max} \quad \text{as} \quad \Gamma(\xi) \to 1. \tag{29}$$

When close to an obstacle, the agent must prioritize moving away from the obstacle over following the desired motion. Since the modulated velocity $\dot{\xi}$, see (6), does not take into account the maximal velocity, smart cropping needs to be applied. We propose the following method, which prioritizes avoidance in

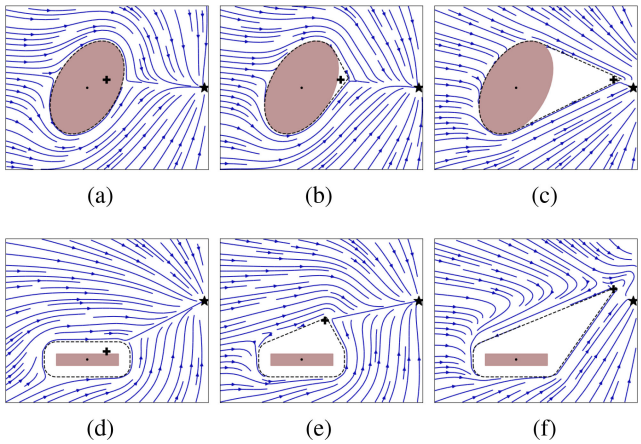


Fig. 10. (a)–(c) Dynamic extension of the hull for an ellipsoid object without margin and (d)–(e) a nonsmooth polygon object with constant margin. (a) Reference inside. (b) Reference close. (c) Reference far. (d) Reference inside. (e) Reference close. (f) Reference far.

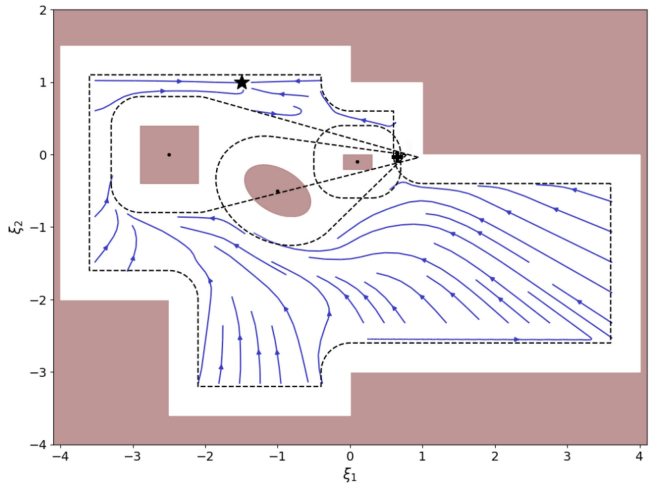


Fig. 11. Full convergence toward the attractor (black star) in an environment of three obstacles intersecting with the boundary.

critical situations but sticks to the initial DS as when safe

$$\dot{\xi}^s = \begin{cases} v^n \mathbf{n}(\xi) + v^e\, \mathbf{e}(\xi) & \text{if } \langle \frac{\dot{\xi}}{\|\dot{\xi}\|}, \mathbf{n}(\xi) \rangle < \frac{v^n}{v^{\max}} \\ v^{\max}\, \dot{\xi}/\|\dot{\xi}\| & \text{else if } \|\dot{\xi}\| > v^{\max} \\ \dot{\xi} & \text{otherwise} \end{cases} \tag{30}$$

with $v^e = \sqrt{(v^{\max})^2 - (v^n)^2}$; see Fig. 9.

Theorem 3 Consider the dynamic environment with N^{obs} obstacles, which have a weighted local velocity $\dot{\tilde{\xi}}^{\text{tot}}$, resulting from the obstacles' movements and surface deformations, defined in (24). An agent is moving in this space and has a maximum velocity of $v^{\max}$, further the obstacles' surface velocities are limited by (29). The agent which starts in free space, i.e., $\Gamma_o(\{\xi\}_0) > 1,\ \forall o \in N^{\text{obs}}$ and moves according to (30), will stay in free space for infinite time, i.e., $\Gamma_o(\{\xi\}_t) > 1$, $t = 0..\infty\ \forall o \in N^{\text{obs}}$.

Proof: See Appendix C. ■

D. Reference Point Placement

1) Dynamic Extension of Hull: Clusters of more than two convex obstacles do often not form a *star shape*. In such cases, we propose to extend the hull of each obstacle such that they all include a common reference point. The new hull is designed to be convex for each obstacle, and hence the cluster is star-shaped. The extended hull creates a cone that is tangent to the obstacle's surface and has the reference point at its tip (Fig. 10). Since the clusters are *star-shaped*, there is a global convergence of the vector field toward the attractor.

The extension of the surfaces can be done dynamically, as a collision-free trajectory is ensured around deforming obstacles (Section VI-B).

2) Cluttered Environments With Obstacles and Boundaries: For obstacles which intersect with the boundary, the reference point has to be placed inside the wall, i.e., $\Gamma_b(\xi_o^r) < 1$ (see Fig. 11). This enforces all trajectories to avoid the intersecting obstacles by moving away from the wall (counterclockwise in this example). The boundary modulates them in the same direction. Hence, there is full convergence of all trajectories toward the attractor. This is true for a boundary-obstacle with a positive (local) curvature

$$c^b(\xi) > 0 \qquad \forall\, \xi \tag{31}$$

with the curvature given as

$$c^{(\cdot)}(\xi) = \lim_{\Delta\xi \to \mathbf{0}} \frac{R(\xi) - R(\xi + \Delta\xi)}{\Delta\xi}$$
$$\forall \xi \in \mathscr{X}^b,\ \ \langle \Delta\xi,\, \mathbf{n}(\xi) \rangle = 0. \tag{32}$$

VII. Obstacle Avoidance With Robots

The algorithm has so far been described for a point mass. It is straightforward to extend this to control robots, which can be approximated by a circle (e.g., drones, wheel-based platforms) by creating a margin around all obstacles wide enough to account for the shape of the robot. The method can also be extended to higher dimensions and multiple degrees-of-freedom robot arms by describing and evaluating the system (robot + obstacle) in joint-space. This, however, requires representing the obstacle in configuration space which is not always easy, especially when the obstacle moves.

Alternatively, in the rest of this section, we introduce a weighted evaluation of the desired dynamics along a robot arm's links to obtain a collision-free trajectory toward the desired goal in Cartesian space.

A. Goal Command Toward Attractor

Consider a robot arm with end-effector's position ξ. The desired velocity toward the attractor is evaluated as described in Sections III to VI and denoted as $\dot{\xi}$. The goal command in joint space is evaluated through inverse-kinematics as

$$\dot{\mathbf{q}}^g = \hat{\mathbf{J}}^{\dagger}(\mathbf{q})\dot{\xi} \tag{33}$$

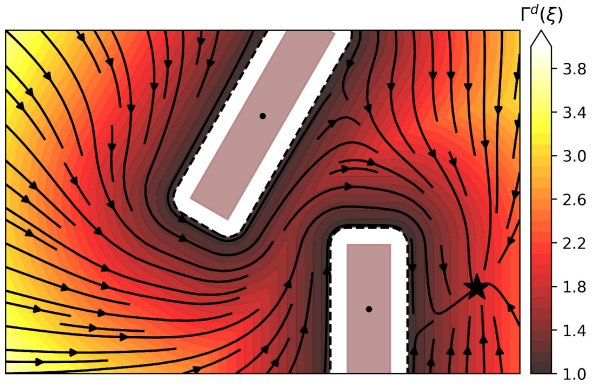


Fig. 12. Γ-field and desired direction in a multiobstacle environment as given in (34).

where $\hat{\mathbf{J}}(\mathbf{q})$ is the Jacobian of the robot arm with respect to position only.[3]

1) Γ-Danger Field: The closer an obstacle is to the robot, the more it is in danger to collide. Based on the $\Gamma(\xi)$-function introduced in (2), we introduce a Γ-danger field with respect to all N^{obs} obstacles as

$$\Gamma^d(\xi) = \min_{o \in [1..N^{\text{obs}}]} \Gamma_o(\xi). \tag{34}$$

The field is displayed in Fig. 12 around two obstacles. This field is used to evaluate the weights to continuously switch between the goal command $\dot{\mathbf{q}}^g$ to the avoidance command $\dot{\mathbf{q}}^m$.

B. Link Avoidance

The avoidance command $\dot{\mathbf{q}}^m$ ensures that each link is avoiding the collision with the environment. To extend the obstacle avoidance algorithm to a rigid body, we introduce N^S section points $\xi^S_{l,s}$ for each link l and $s \in [1..N^s]$. The dynamical system is evaluated at each section point and coupled with a section weight w^S_s. The section point weight w^S_s increases the lower the Γ-danger value (see Appendix C).

1) Rigid Body Dynamics: The linear and angular avoidance velocity for link l are obtained as

$$\begin{aligned} \mathbf{v}^L &= \sum_{s=1}^{N^S} w^S_s \dot{\xi}^S_{l,s} \quad \text{and} \\ \omega^L &= \sum_{s=1}^{N^S} w^S_s (\xi^S_{l,0} - \xi^S_{l,s}) \times \left(\mathbf{v}^S_s - \mathbf{v}^L\right) \end{aligned} \tag{35}$$

where $\xi^S_{l,0}$ is the position of the root of a link l, and $\dot{\xi}^S_{l,s}$ is the dynamical system-based avoidance evaluated as described in in Section III–VI. A single link avoiding a circle can be seen in Fig. 13.

[3]The inverse of the Jacobian $\mathbf{J}^\dagger(\mathbf{q})$ is obtained through the Moore–Penrose pseudoinverse.

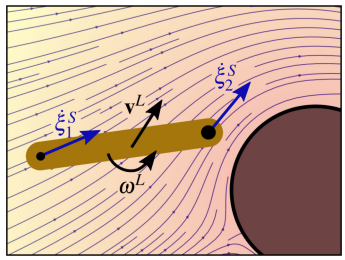


Fig. 13. Resulting linear velocity $\mathbf{v}^L$ and angular velocity ω^L for a rigid body with to sections points.

2) Joint Modulation: The desired joint avoidance command is obtained through inverse kinematics as

$$\dot{\mathbf{q}}^m = \mathbf{J}^\dagger(\mathbf{q}, \xi^S_{l,0}) \begin{bmatrix} \mathbf{v}^L \\ \omega^L \end{bmatrix} \tag{36}$$

where $\mathbf{J}(\mathbf{q}, \xi^S_{l,0})$ is the Jacobian up the root of the link l, i.e., the position of the section point $\xi^S_{l,0}$.

C. Evaluation Along the Robot Arm

The evaluation is performed by iterating over all N^L links, starting from to base of the robot arm. At each iteration, the joint control $\dot{\mathbf{q}}^c$ of all links which are part of the kinematic chain are updated, i.e., if the evaluation is performed for link l, the joint control are updated for $\dot{\mathbf{q}}^c_i \; \forall i \leq l$.

The evaluation is weighted along with the link to ensure collision avoidance of all links while trying to follow the goal command $\dot{\mathbf{q}}^g$. The link weights w^L_l are further described in Section Sec. E2.

1) Initial Joint Control: The joint control is initialized based on the goal control from (33) as

$$\dot{\mathbf{q}}^c \leftarrow \left(1 - \sum_{l=1}^{N^L} w^L_l\right) \dot{\mathbf{q}}^g. \tag{37}$$

The first element of the joint command is then updated based on the avoidance velocity obtained in (36) as

$$\dot{\mathbf{q}}^c_{[1]} \leftarrow \dot{\mathbf{q}}^c_{[1]} + w^L_1 \dot{\mathbf{q}}^m_{[1]}. \tag{38}$$

2) Joint Control Correction: As an effect of influence the obstacle avoidance of each link, the obtained control command $\dot{\mathbf{q}}^c$ differs from the ideal goal command $\dot{\mathbf{q}}^g$. This difference is obtained at link l as

$$\mathbf{v}^\Delta = \hat{\mathbf{J}}_l(\mathbf{q}) \left(\dot{\mathbf{q}}^g_{[1:l]} - \dot{\mathbf{q}}^c_{[1:l]}\right) \quad \forall\, l > 1 \tag{39}$$

where $\dot{\mathbf{q}}^c_{[:l]}$ is the current control command, evaluated up to link $l-1$, and $\hat{\mathbf{J}}_l(\mathbf{q})$ is the arm Jacobian with respect to position up to link l.

The joint speed of link l which would best account for this difference can be evaluated as

$$\dot{q}^\Delta = \langle \omega^\Delta, \mathbf{I}^\omega \rangle \quad \text{with} \quad \omega^\Delta = \mathbf{v}^\Delta \times \mathbf{I}^L \tag{40}$$

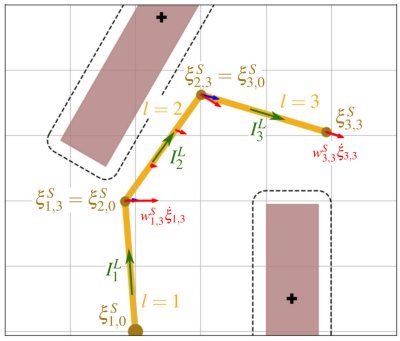


Fig. 14. Robot arm with three links ($N^L = 3$), which each containing three sections points ($N^S = 3$) is surrounded by two obstacles. The blue arrows are the goal velocity obtained (only evaluated at the joints), and the red arrows are the weighted avoidance velocities. The direction of joint rotation $\mathbf{I}^\omega_l$ is pointing out of the plane.

where $\mathbf{I}^\omega$ is the direction of rotation of the joint actuating link l[4] and $\mathbf{I}^L$ is the direction along with the link, i.e., pointing from one joint to the next. The variables can be observed in Fig. 14.

3) Joint Control Update: The velocity of each joint is evaluated one by one, starting at the joint closest to the base of the robot toward the end-effector (see Algorithm 1). The joint command $\dot{\mathbf{q}}^c$ is first updated by applying the correction control from (40) to the current joint l as

$$\dot{\mathbf{q}}^c_{[l]} \leftarrow \dot{\mathbf{q}}^c_{[l]} + \dot{q}^\Delta \sum_{i=1}^{l-1} w^L_i \qquad \forall\, l > 1. \tag{41}$$

The modulation command $\dot{\mathbf{q}}^m$ from (36) is then applied to all underlying joints

$$\dot{\mathbf{q}}^c_{[1:l]} \leftarrow \dot{\mathbf{q}}^c_{[1:l]} + w^L_l \dot{\mathbf{q}}^m_{[1:l]} \qquad \forall\, l > 1. \tag{42}$$

This is executed iteratively for all joints $l > 1$.

D. Validation in Simulation

We applied the algorithm to two scenarios with planar robotic arms (Fig. 15). The scenario in Fig. 15(a) was chosen similar to the one presented in [31]. Our approach can avoid obstacles in a similar setup without a navigation function.

The scenario in Fig. 15(b) includes a more complex robot with three links. Additionally, the skew placement of the obstacle and the position of the start end goal point require the robot to actively go around the obstacle guided by the reference point (see Fig. 12 for the full DS).

VIII. Comparison Algorithms

A. Qualitative Comparison

We have selected multiple time-invariant, local obstacle avoidance algorithms for qualitative comparison with the presented method Table I. The *navigation functions*, *Lyapunov QP*, *sphere-world QP*, and *danger fields* all rely on critical tuning parameters. This is a Lyapunov function for the *Lyapunov QP*,

[4]We assume single-degree-of-freedom joints.

Algorithm 1: Joint Control Command for a Robot Arm.

Input: N^L, N^S, $f(\xi)$, obstacle environment
Output: $\dot{\mathbf{q}}^c$

$\xi^r_o \;\; \forall o \in 1..N^{\text{obs}}$ {update dynamic obstacles as in Section VI}
$\dot{\mathbf{q}}^g \leftarrow (\mathbf{J}(\mathbf{q}))^\dagger \dot{\xi}$ {compute goal command as in (33)}
for $l = 1\text{to}N^L$ **do**
  for $s = 1\text{to}N^S$ **do**
    $\Gamma(\xi_{s,l})$ {compute *dangerfield* as in (34)}
    w^Γ {compute danger weight as in (55)}
  end for
  w^L_l {compute link weight as in (57)}
end for
$\dot{\mathbf{q}}^c \leftarrow (1 - \sum_l w^L_l)\dot{\mathbf{q}}^g$ {initialize control command}
$\dot{\mathbf{q}}^c_{[1]} \leftarrow \dot{\mathbf{q}}^c_{[1]} + w^L_1 \dot{\mathbf{q}}^m_{[1]}$
for $l = 2\text{to}N^L$ **do**
  if $w^L_l > 0$ **then**
    for $s = 1\text{to}N^s$ **do**
      w^S_s {compute section weight as in (58)}
    end for
    $\mathbf{v}^L_l, \omega^L_l$ {compute avoidance velocities as in (35)}
    $\dot{\mathbf{q}}^m_l$ {compute modulation command as in (36)}
    $\dot{\mathbf{q}}^c_{[1:l]} \leftarrow \dot{\mathbf{q}}^c_{[1:l]} + w^L_l \dot{\mathbf{q}}^m_{[1:l]}$
  end if
  $\mathbf{v}^\Delta \leftarrow \mathbf{J}_l(\mathbf{q})(\dot{\mathbf{q}}^g_{[1:l]} - \dot{\mathbf{q}}^c_{[1:l]})$
  $\omega^\Delta \leftarrow \mathbf{v}^\Delta \times \mathbf{I}^L$
  $\dot{q}^\Delta \leftarrow \langle \omega^\Delta, \mathbf{I}^\omega \rangle$
  $\dot{\mathbf{q}}^c_{[l]} \leftarrow \dot{\mathbf{q}}^c_{[l]} + \dot{q}^\Delta \sum_i w^L_i$
end for

and parameters for the diffeomorphic transformation or navigation function for the other three methods. While the parameters and functions can be obtained through manual tuning, no solution exists to set them in real-time automatically. Hence, these methods cannot be easily applied to dynamic environments.

Navigation functions and the diffeomorphic transform are defined globally. Their tuning parameter depends on the distribution of the obstacles all across space. Two obstacles close together far from an agent will determine the possible tuning parameters and influence the avoidance behavior. As a result, the methods cannot be transferred easily to cluttered dynamic environments.

While other methods have already allowed navigation inside walls, this has not been done in combination with proven *star-shaped* world convergence and dynamic surroundings.

B. Quantitative Comparison

For a quantitative comparison, we chose algorithms that can function in cluttered, dynamic environments and can handle external hulls (see Table I).

The method for the modulation algorithm in dynamic environments is presented in this article (referred as *Dynamic* during this section). It is compared to [36], which uses modulation matrix based on an orthogonal decomposition matrix $\mathbf{E}(\xi)$ (referred as

TABLE I
PROPOSED DYNAMIC OBSTACLE AVOIDANCE IS COMPARED TO DIFFERENT STATE-OF-THE-ART METHODS

	Dynamic *(presented)*	**Reference** [3]	**Orthog.** [36]	**Repulsion** [22]	**Navigation Functions** [16], [23], [24]	**Lyapunov QP** [14]	**Sphere World QP** [15]	**Danger Fields** [30], [31]
Time invariant	✓	✓	✓	✓	✓	✓	✓	✓
History invariant	✓	✓	✓	✓	✓			✓
Convergence: convex	✓	✓	(✓)		✓	✓	✓	✓
Convergence: star-shape	✓	✓			✓	(✓)	✓	✓
No critical tuning parameter (incl. Lyapunov function)	✓	✓	✓	✓				
Avoidance behavior independent of global distribution	✓	✓	✓	✓		✓		
Closed from solution (no optimization)	✓	✓	✓	✓	✓			✓
Considering initial dynamics (not goal position only)	(✓)	(✓)	(✓)	✓		✓	✓	(✓)
Navigation inside walls	✓		✓	✓	✓	✓	✓	✓
Smooth motion around corners	✓			✓	✓	✓	✓	✓
Cluttered dynamic environment	✓	(✓)	(✓)	✓				

The last three items refer to the main contributions of this work.

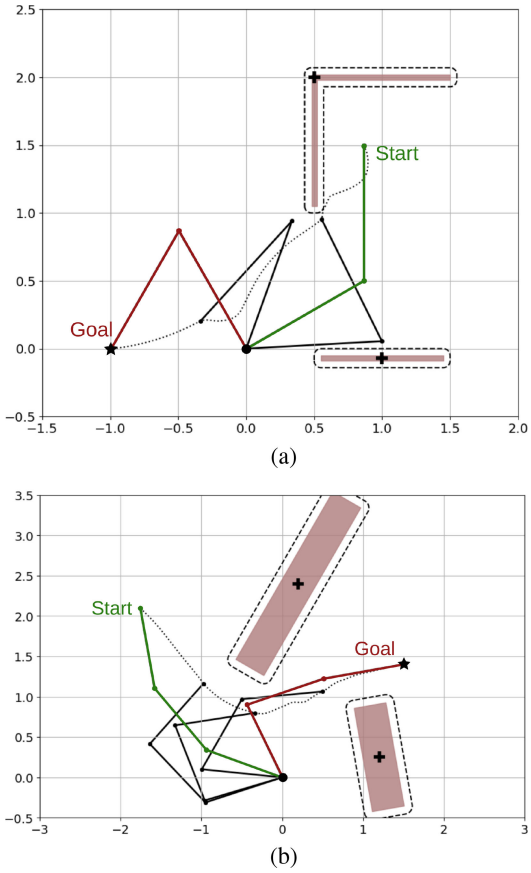


Fig. 15. Time sequence of robot arms navigating in two different environments. (a) 2 DoF arm. (b) 3 DoF arm.

Orthogonal) and the potential field algorithm [22] (referred to as *Repulsion*).

The comparison is made in a simulated environment (Fig. 16). Two ellipse-shaped obstacles randomly change shape, and the movement is inspired by *random walk*. The combined maximum expansion and obstacle's velocity are lower than the maximum

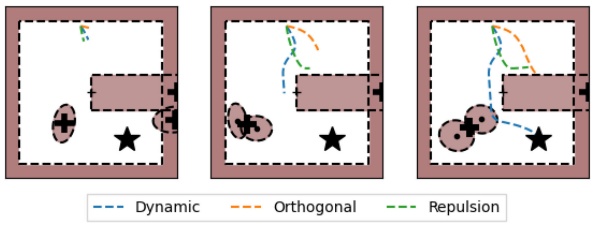


Fig. 16. Three snapshots of one experimental run are placed from left to right. The obstacles are randomly initialized and move according to a random walk. One obstacle moved from the right to the left, while the other one was stationary. The three algorithms start at the same randomly chosen position and move toward the attractor.

TABLE II
OUTCOME OF THE 300 TRIALS RUNS WERE EITHER FULL CONVERGENCE, COLLISION WITH AN OBSTACLE OR GETTING STUCK IN A LOCAL MINIMUM

	Converged	Collided	Minimum
Dynamic	77%	23%	0%
Orthogonal	20%	23%	57%
Repulsion	39%	1%	60%

speed of the agents of $v^{\max} = 1\,\mathrm{m/s}$. The three algorithms are given the same attractor as a goal. They start at the same time and encounter the same environment.

The *Dynamic* algorithm is observed to have the highest rate of convergence (Table II) resulting from the increased environment information through the reference point. The *Repulsion* has a preferable behavior on avoiding collisions because of its conservative behavior around obstacles (moving far away and only approaching them slowly). This influences the distance traveled and the time needed to reach a goal (Table III). The mean of the velocity is lower for the *Dynamic* algorithm. This is a result of no *tail effect* behind the obstacles (see Section III-G). The variance of the velocity is similar for the three algorithms.[5]

[5]The source code of the implementation is [Online]. Available: https://github.com/epfl-lasa/dynamic_obstacle_avoidance

TABLE III
MEAN AND THE STANDARD DEVIATION (AFTER THE ±) ARE COMPARED FOR THE THREE ALGORITHMS FROM THE 54 TRIALS, WHERE ALL THREE AGENTS CONVERGED

	d $[m]$	t $[s]$	$\bar{v}$ $[m/s]$	σ_v $[m/s]$
Dynamic	9.69 ± 1.19	1.03 ± 0.13	0.61 ± 0.05	0.34 ± 0.02
Orthogonal	10.06 ± 1.53	1.17 ± 0.21	0.55 ± 0.05	0.34 ± 0.02
Repulsion	9.8 ± 1.29	1.39 ± 0.2	0.46 ± 0.03	0.21 ± 0.02

The metrics of distance (d), duration of the Run (t), the mean velocity ($\bar{v}$), and the standard deviation of the velocity (σ_v) are listed.

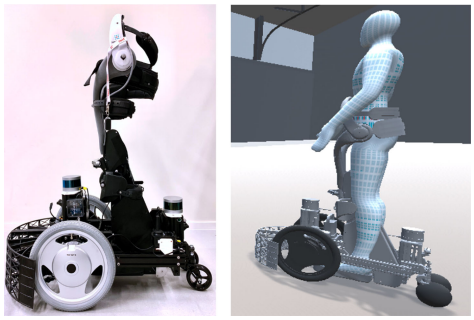


Fig. 17. Picture of the semiautonomous wheelchair real (left) and the simulation rendering including an operator (right).

C. Computational Complexity

The presented algorithm is closed-form, whereas the *matrix inverse* is the most complex computation. Since it is applied to all obstacles, the complexity follows as $\mathcal{O}(d^{2.4} N^{\text{obs}})$, a function of the number of dimensions d and the number of obstacles N^{obs}. It had an average time of 3.48 ms on a computer with 8 *Intel Core i7-6700 CPU @ 3.40 GHz*.

This is more complex than the *Orthgonal*, where the matrix multiplication is the most complex operation. The complexity follows as $\mathcal{O}(d^2\, N^{\text{obs}})$. This reflects in the slightly faster evaluation time of 2.84 ms.

The potential field is the simplest of the three algorithms, with the norm being the most complex operation and a complexity $\mathcal{O}(dN^{\text{obs}})$. It has the lowest evaluation time of 1.75 ms.

The search for the optimal reference point is the computationally most extensive calculation because it requires the (iterative) closest-distance evaluation between obstacles. The Python library *shapely*[6] was used for the implementation, and an evaluation time of 5.14 ms was obtained.

IX. Empirical Validation

The empirical validation is performed with the mobile robot *QOLO* [42], see Fig. 17; first in simulation and then on a real robot platform. QOLO, a semiautonomous wheelchair, is designed to navigate in pedestrian environments and indoors. This platform is hence suited to test our algorithm's ability to avoid moving obstacles (pedestrians) and nonconvex obstacles (walls, indoor furniture) containing sharp edges (tables, shelves). In all our experiments, we assume that QOLO has information about the goal, i.e., the attractor ξ^a of our nominal DS.[7]

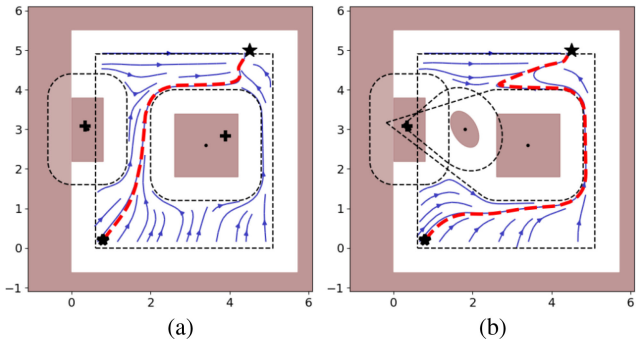


Fig. 18. Two static office environments with two tables in a rectangular room. The center table divides the room into two passages in (a). In (b) one of the passages is blocked by a static person. Convex expansion of the hull around the reference point (black cross) ensures convergence toward the attractor (black star). The path followed by the agent is visualized in red. (a) Office with two passages. (b) Office with passage at side.

A. Static Environment

We task QOLO to navigate in an office-like environment. The room is a square (5 × 5 m), modeled as a boundary obstacle. Further, two tables are located in the room, one at the side and one at the center. The robot starts from the bottom left, and the attractor ξ^a is placed at the opposite side of the room (illustrated with a star). All objects, including the wall, are static and known *a priori*, and the localization is performed using the SLAM algorithm. The robot evaluates the modulated avoidance in real-time. We run the following two scenarios:

1) QOLO is in the room, and there are two possible paths to go around the center table. The dynamical system is split by the obstacle at the center [Fig. 18(a)]. The robot chooses its preferred trajectory at runtime.
2) In addition, there is a (static) person in the room, which blocks the center passage.
   The reference point of the obstacles is automatically placed inside the wall. The robot finds its path around the obstacles [Fig. 18(b)].

B. Dense Crowd (Simulation)

The robot is navigating in a corridor within a dense, simulated crowd. The motion of the crowd is created according to [43]. Two hundred people are moving uniformly along the 6 m wide corridor, either in the same or opposite direction as the robot (see Fig. 19).

QOLO is tasked to travel from one end of the corridor to the other. It is guided by the attractor of the nominal DS. All pedestrians are modeled as circular obstacles with radius of 0.6 m [Fig. 20(a)].

At each timestep, the problem is reduced to avoiding a subset of the pedestrians. Due to the crowd's density, the robot could

[6][Online]. Available: https://github.com/Toblerity/Shapely

[7]A video of the experiments is [Online]. Available: https://youtu.be/WKsowu68v8

Fig. 19. QOLO moving in a crowd.

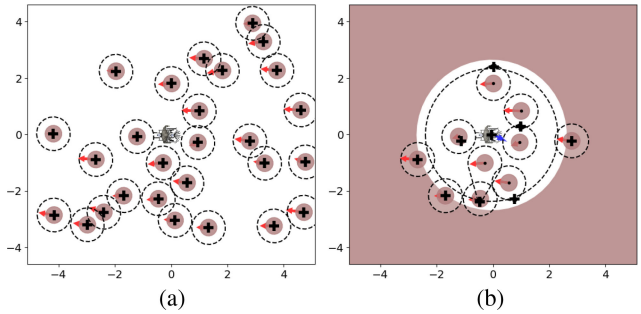


Fig. 20. Environment with many agents (left) is reduced to a scenario with 10 obstacles and an enclosing hull (right). (a) All crowd agents. (b) Local Obstacle Hull.

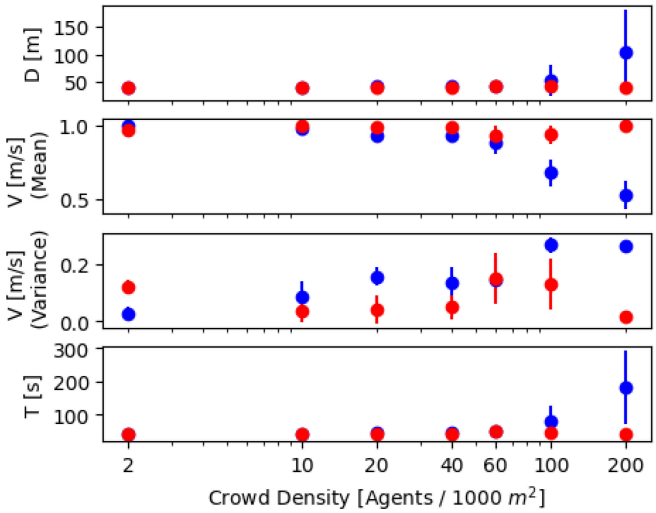


Fig. 21. QOLO agent is moving in parallel (red) and opposite direction (blue) to the crowd. When the robot moves with the crowd, the density of the crowd has a negligible effect. When the robot moves in the opposite direction, the denser the crowd, the larger the cumulative distance (D) and mean velocity (V), as well as time (T), needed to reach the end of the corridor. The standard deviation of the velocity (V Variance) increases for dense crowds with counterflow.

realistically perceive only a subset of the pedestrians in real-time. The number of perceived people is set to $N^c = 10$. The rest of the people are hidden behind a virtual, circular wall. The center of the circular wall $\xi^{c,w}$ is displaced from the position of the robot ξ^Q based on the remaining obstacles

$$\xi^{c,w} = \xi^Q + \sum_{i=N^c+1}^{N^{\text{obs}}} \frac{\xi_i^c - \xi^Q}{\|\xi_i^c - \xi^Q\|} e^{-(\|\xi_i^c - \xi^Q\| - r^p - r^Q)} \tag{43}$$

where i is iterating over the list of the obstacle which are ordered based on their distance to the robot. The displacement factor is with respect to the radius of each pedestrian, $r^p = 0.6\,\text{m}$, and the robot radius, $r^Q = 0.5\,\text{m}$.[8]

The radius of the hull is chosen such that the next closest obstacle $N^c + 1$ is fully within the hull. The resulting environment has a dynamic hull with changing center-position and radius (Fig. 20).

Reducing the environment to only sphere obstacles decreases the computational time since there is a closed-form solution for the closest distance between two spheres. The evaluation on ROS1[9] and Python 2.7 run at around 200 Hz on a *Up Board: Intel Celeron N3350* with 2.4 GHz (CPU) and 8 GB of RAM. The number of nearby obstacles (including the wall) was eleven, while the wall remained far from the agent, it helped guide the robot around the local crowd (see Section VI-D2). This is done by placing a reference point inside the wall if a crowd cluster is touching the wall [see small cluster at the bottom in Fig. 20(b)]. Further, fast contraction of the boundary can happen when the local crowd density is high. This forces the obstacle to stay away from surrounding obstacles.

1) Quantitative Analysis: We evaluate the effect of the crowd size on the time it takes for the robot to travel through the corridor. The crowd moves along the (infinite) corridor at an average velocity of $1\,\text{m/s}$. The simulation runs with a steady-state crowd flow. QOLO is tasked to move in the same or opposing direction crowd's flow with the desired velocity of $1\,\text{m/s}$.

We assess the time, speed, and distance traveled by the robot when moving with and opposite to the flow, see Fig. 21. When moving with the flow (parallel flow), the crowd has no significant effect on the distance traveled by the agent or the velocity. When moving against the flow of the crowd, a decrease of the robot's velocity can be observed for crowds denser than 20 agents per 1000 square meters. (No effect on the crowd was observed since only a single robot was moving against a large crowd.) The distance traveled increases significantly for densities above 100 agents per 1000 square meters. As a result, the average time needed to reach the goal more than doubles for a crowd size of 100 people compared to 2 people.

In counterflow scenarios, the standard deviation of the flow increases. This results from situations where the robot has to slow down or stop to avoid the upcoming agents.

C. Proof of Concept: Outdoor Environment

A qualitative proof of concept was performed in an outdoor environment. We brought the QOLO robot to the center of

[8]For a real-world implementation sensory distance measurements in the horizontal plane can be used to create the virtual circular wall and its displacement, since the detection of people is still a time intensive task.

[9][Online]. Available: https://www.ros.org/

Fig. 22. Desired path of the robot in the outdoor environment is the direct line from the initial position of the robot (right) to the target position on the left.

(a) (b)

Fig. 23. (a) Camera and the LIDAR of the robot are interpreted by the (b) detector, which is used for the obstacle avoidance algorithm. (a) Camera view. (b) Detector interpretation.

Lausanne, Switzerland.[10] The robot was tasked to travel back and forth across a small marketplace (Fig. 22). The location is restricted to pedestrians only, and a total of six streets meet at the crossing. This results in a large diversity in both the pedestrians' speed and direction of movement. The robot's controller is initialized with a linear DS to reach a goal 20-m away from the onset position. Pedestrians are detected with a camera and Lidar-based tracker developed by Jia *et al.* [39]. The output of the tracker is displayed in Fig. 23. Recordings were taken on Saturday morning when the market was ongoing, and the crowd had a high density.

The nonholonomic constraints of QOLO are taken into account by evaluating the dynamical system 0.53 m in front of the center of the wheel-axes. The linear command of the robot is the velocity part in the moving direction. The angular velocity follows the perpendicular part of the velocity. These velocities are provided to the low-level controller of the robot.

The geometry of QOLO is taken into account by placing a margin of 0.5 m around each pedestrian.

A total of five runs were executed. The robot reached its goal autonomously without intervention. The driver reported *high angular acceleration* during parts of the trip.

[10]Approval was obtained from the EPFL Ethics board and the police of Lausanne city. A driver was on board the robot during the experiment. He could start and stop at all times. A second experimenter was watching the scene and verified the output of the tracker. This was necessary in case the detector/tracker disfunctioned.

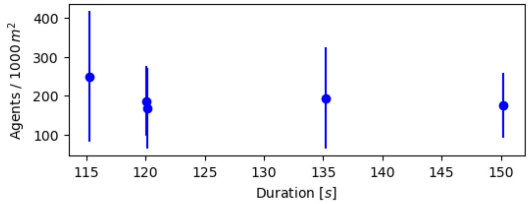


Fig. 24. Crowd-density is highly varying during the five runs.

Post-hoc analysis of the video recordings revealed that the crowd density varied with a mean between 150 and 260 people per 1000 square meters (Fig. 24). The time to complete the runs ranged from 115–150 s. No correlation was observed between the density of the crowd and the time taken to reach the goal. We expect this to result from external factors influencing the run, such as the speed and direction of the crowd.

We see this as a successful proof-of-concept of the obstacle avoidance algorithm in a real crowd scenario. The crowd motion was more complex than the streamline simulation, as people would come from all directions and would not group in a uniform flow. Moreover, the crowd included diverse pedestrians, from families with small children to elderly people.

X. Discussion

In this article, we have introduced a dynamical system-based obstacle avoidance algorithm. The modulation-based approach has a theoretical proof of convergence and can be applied to higher dimensional space. The implementations presented in this work focus on collision avoidance in two dimensions, such as navigation of mobile robots and obstacle avoidance for simplified robot arms.

The inverted obstacle has shown to be a great representation of workspace boundaries of mobile robots, such as walls of a room or the local window in dense crowd navigation. These boundaries could be interpreted as control constraints similar to CBFs. Recent works have used such control barrier function in the context of safe learning by demonstration [44], [45] or reinforcement learning [46]. Other than existing methods, we propose a closed-form solution for star-shaped barriers.

Since many human-made environments contain nonsmooth surfaces (e.g., tables, corners of rooms), the solution for providing a smooth flow without extending the hull has shown suitable for practical implementations.

The concept of dividing dynamical systems into direction and magnitude and the presented method summing vectors to avoid local minima has been used throughout: 1) to create a smooth pseudonormal for polygon obstacles and walls and from that a smooth flow in their presence; 2) to sum the flow created by several obstacles without creating local minima; and 3) to solve the optimization problem to find the closest distance for pairs of obstacles.

Finally, the algorithm has been successfully applied to a nonholonomic robot in a static indoor environment and dynamic outdoor crowds.

A. Scalability and Speed

The implementation used for the experiments on QOLO was running on Python 2.7. Note that already the update to Python >3.6 gives around a 1.5-speed improvement. Additionally, switching to a compiled language like C++ can increase the speed by a factor of around 10 [47]. The proposed algorithm can run at a frequency of >1 kHz onboard of a mobile robot. This is well above the human reaction time of around 250 ms.

The bottleneck of the current implementation is the image recognition/tracking since it was only able to run at an average frequency of around 5 Hz. This is due to the computationally heavy evaluation of the deep-neural networks used for obstacle recognition. Nevertheless, we believe that the approach of separating perception and motion-planning is favorable, supported by current trends in self-driving cars and other autonomous vehicles [48], [49].

B. Contribution

The proposed method adds value to the current state of the art, not for global path planning but for reactive obstacle avoidance with convergence proofs. The modulation-based avoidance algorithm fully converges in (local) *star-shaped* environments toward the desired attractor ξ^a. The behavior is similar to approaches using potential artificial fields. However, the presented approach does not require finding and deriving an artificial potential function, but the modulation directly outputs the desired velocity.

Compared to other closed-form and QP-based obstacle avoidance algorithms, our method does not require any tuning of critical parameters for its convergence, nor the finding of a Lyapunov candidate function. All parameters presented in this article can be chosen within the defined range, and theoretical convergence is ensured.

In the presented article (and initially introduced in [3]), we presented several methods of how to place the reference point (i.e., tune its position). Even though its position is critical for convergence, it is known for *star worlds.*

1) *Convex obstacles:* The reference point can be placed anywhere within the obstacle, i.e., $\xi^r \in \mathscr{X}^i$.
2) *Star shapes:* The reference point has to be placed within the kernel of the obstacle. Most of the time, this is just the geometrical center of the obstacle. For polygons, an algorithm to find the kernel has been described in [38].
3) *Intersecting convex obstacles:* If several convex obstacles intersect and form a *star shape*, the reference point can be placed anywhere at the center of all intersecting obstacles o, i.e., $\xi^r \in \mathscr{X}_o^i \ \ \forall o$.
   If the convex obstacles do not form a *star-shape* or are intersecting with the boundary, the hull can be extended dynamically, as described in Section VI-D.

The placement of the reference point only becomes challenging when obstacles merge or separate dynamically. While we propose approximations for many cases (see Appendices-A-2 and A-3), we do not provide a solution for all scenarios.

To the best of the authors' knowledge, there currently exists no closed-form method to generate a flow around merging and dividing obstacles, which has convergence guarantees. The placement of the reference point for such dynamic scenarios is ongoing research.

C. Future Work

Future work can extend the proposed work in the following areas:

1) *Low-Level Controller:* The low-level controller used in crowd navigation displaced the evaluation point away from the center of the robot. This resulted in an increased (conservative) margin around the robot. The way the shape of the robot is considered in the algorithm should be improved.
2) *Environment Recognition:* The update rate of the (deep-learning-based) tracker was approximately 5 Hz, while the present avoidance algorithm ran at a frequency of 50 to 100 Hz. The obstacle avoidance was often evaluated with old environment information (but updated robot position). An intermediate estimator could predict how the crowds move in-between.
3) *High-Level Planning:* The combination of the fast obstacle avoidance controller with slower planning algorithms could allow to handle more complex environments, i.e., including the avoidance of surrounding (nonstar-shaped) environments.
4) *Evaluation in High Dimensional Space:* The experimental implementation is executed on an autonomous wheelchair (a 2-D scenario). However, this work provides a theoretical solution, which can be applied to 3-D and higher dimensional spaces. The next step will be the implementation and evaluation in a higher dimensional space.

XI. Conclusion

This article presents a dynamical system-based algorithm for local navigation under convergence constraint. The work has provided and tested the solution for local crowd navigation. It ensures certain convergence constraints to not only safely navigate but also reach the goal in local scenarios. The advantage of the method comes from the low complexity and speed of the algorithm. Furthermore, tuning free convergence is obtained in star-world scenarios. This will allow to scale to higher dimensions and transfer to various scenarios.

Acknowledgment

The authors would like to thank the support of D. Paez-Granados and D. Gonon for the running of the experiments. Their effort and insight have contributed to the implementation using QOLO and the qualitative analysis of the results.

Appendix

A. Directional Weighted Mean

The weighted summation of vectors can result in a zero-sum (e.g., two vectors opposing each other with equal weight) and lead to undesired local-minima of dynamical systems.

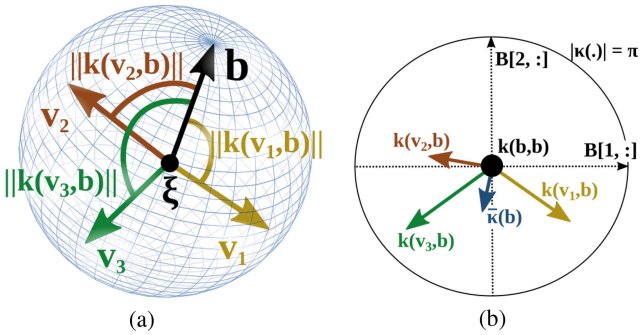


Fig. 25. Various directions (here three) are described with respect to a (a) basis direction $\mathbf{r}^0$. The directions are transformed to the (b) direction-space $\mathcal{K}$, where the weighted mean, $\bar{\kappa}$, is obtained. (a) Three initial vectors. (b) Direction-space.

We extend here the directional weighted summing introduced in [3] for more general application. It ensures that the summed vector field is free of local minima

$$\{\mathbf{v} \in \mathbb{R}^d \ : \ \|\mathbf{v}\| = 1\}. \tag{44}$$

The transformation into direction space $\mathscr{K}$is given by

$$\mathcal{K} = \{\kappa \in \mathbb{R}^{d-1} \ : \ \|\kappa\| < \pi\}. \tag{45}$$

The direction space is with respect to a base vector $\mathbf{b}$ which is the first column of the orthonormal transformation matrix $\mathbf{B}$. This allows the transformation into the new basis

$$\hat{\mathbf{v}}_i = \mathbf{B}^T \mathbf{v}_i. \tag{46}$$

The magnitude of the transformed vector in direction space is equal to the angle between the original vector and the reference vector. The transformation of the initial vector $\mathbf{v}_i$ in the direction space is

$$\kappa_i(\mathbf{b}) = \mathbf{k}(\mathbf{v}_i, \mathbf{b}) = \begin{cases} \arccos\left(\hat{\mathbf{v}}_{i[1]}\right) \frac{\hat{\mathbf{v}}_{i[2:]}}{\|\hat{\mathbf{v}}_{i[2:]}\|} & \text{if } \ \hat{\mathbf{v}}_{i[1]} \neq 1 \\ \mathbf{0} & \text{if } \ \hat{\mathbf{v}}_{i[1]} = 1. \end{cases} \tag{47}$$

The mean is evaluated as a function of the weight w_i of all N^v vectors

$$\bar{\kappa} = \sum_{i=1}^{N^v} w_i \kappa_i. \tag{48}$$

The mapping into original space is evaluated as

$$\bar{\mathbf{v}}(\bar{\kappa}) = \begin{cases} \mathbf{B} \begin{bmatrix} 1 & 0 & .. & 0 \end{bmatrix}^T & \text{if } \|\bar{\kappa}\| = 0 \\ \mathbf{B} \begin{bmatrix} \cos\left(\|\bar{\kappa}\|\right) & \sin\left(\|\bar{\kappa}\|\right) \frac{\bar{\kappa}}{\|\bar{\kappa}\|} \end{bmatrix}^T & \text{otherwise.} \end{cases} \tag{49}$$

1) Intuition: In the 2-D case, this hypersphere is a line that represents the angle between the initial DS $\mathbf{f}(\xi)$ and the modulated DS $\dot{\xi}$. It has a magnitude strictly smaller than π, the directional space is a vector space, where the weighted mean is taken (see for the 3-D case in Fig. 25).

Theorem A *Consider a unit vector* $\mathbf{b}$ *as the basis for the projection given in (47) and the corresponding reconstruction function defined in (49). The resulting transformation of unit vector* $\mathbf{k}(\mathbf{v}, \mathbf{b}) : \{\mathbf{v} \in \mathbb{R}^d \setminus -\mathbf{b} \ : \ \|\mathbf{v}\| = 1\} \rightarrow \mathcal{K}$ *defined in (45) is a bijection and the basis vector projects to the origin, i.e.,* $\mathbf{b} \rightarrow \mathbf{0}$.

Proof: The proof is divided into three parts as follows: *(I)* Showing that transformation and reconstruction are the inverse functions of each other, *(II)* any unit vector is transformed to the direction space $\mathcal{K}$, and *(III)* is reconstructed to a unit vector.

(I) Inverse Functions: To be the inverse functions, applying one after the other onto a vector, must results in the original vector. (Vectors $\mathbf{b}$ will be treated separately below).

Let us apply the forward transformation based on (46) and (47) to a unit vector $\mathbf{v}_1$. It can be summarized to

$$\kappa(\mathbf{b}) = \arccos\left(\langle \mathbf{b}, \mathbf{v}_1 \rangle\right) \frac{\hat{\mathbf{B}}^T \mathbf{v}_1}{\|\hat{\mathbf{B}} \mathbf{v}_1\|} \tag{50}$$

with $\hat{\mathbf{B}}$ the matrix $\mathbf{B}$ without the first row.

We apply it to a single vector, hence the corresponding weight is $w_1 = 1$. From (48), we get $\bar{\kappa} = \kappa_1$. The reconstruction follows with (49):

$$\bar{\mathbf{v}} = \mathbf{B} \begin{bmatrix} \cos\left(\|\bar{\kappa}\|\right) \\ \sin\left(\|\bar{\kappa}\|\right) \frac{\bar{\kappa}}{\|\bar{\kappa}\|} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \cos\left(\arccos\left(\langle \mathbf{b}, \mathbf{v}_1 \rangle\right)\right) \\ \frac{\sin(\arccos(\langle \mathbf{b}, \mathbf{v}_1 \rangle))\left(\hat{\mathbf{B}}\right)^T \mathbf{v}_1}{\|\left(\hat{\mathbf{B}}\right)^T \mathbf{v}_1\|} \end{bmatrix}$$

$$= \mathbf{B} \begin{bmatrix} \langle \mathbf{b}, \mathbf{v}_1 \rangle \\ \left(\hat{\mathbf{B}}\right)^T \mathbf{v}_1 \end{bmatrix} = \mathbf{B} \left(\mathbf{B}\right)^T \mathbf{v}_1 = \mathbf{v}_1$$

by using

$$\sin(\arccos\left(\langle \mathbf{b}, \mathbf{v}_1 \rangle\right)) = \sqrt{1 - \langle \mathbf{b}, \mathbf{v}_1 \rangle^2}$$

$$= \sqrt{\|\mathbf{B}\mathbf{v}_1\|^2 - \langle \mathbf{b}, \mathbf{v}_1 \rangle^2} = \| \left(\hat{\mathbf{B}}\right)^T \mathbf{v}_1 \|.$$

For the case that $\mathbf{v}_1 = \mathbf{b}$, we get from (47) that $\bar{\kappa} = \kappa_1 = \mathbf{0}$. And with (49), we get $\bar{\mathbf{v}}(\bar{\kappa}) = \mathbf{b}$, i.e., the original vector.

Hence for all cases, the transformation is bijective.

(II) Transformation Domain: From definition of the transform in (47), the maximum magnitude is the arccos, which is $\|\kappa_i\| < \pi$. Hence it lies in the domain of (45). (The magnitude π is only reached for a vector $-\mathbf{b}$, which is excluded from the transform).

(III) Reconstruction Domain: From the inverse transform (49), we have that norm of any transformed vector $\|\bar{\mathbf{v}}(\bar{\kappa})\| = 1$, this follows from the fact that $\mathbf{B}$ is orthonormal. As a result they all lie in the domain (44). ■

2) Pairwise Closest Distance in Direction Space: The directional space can be used for gradient descent to find the closest distance between two (convex) obstacles. This is done by moving along the surface of the obstacle in direction space. Since the direction space is of dimension $d-1$, finding the closest point of each obstacle has only $2(d-1)$ degrees of freedom (instead of $2\,d$ in the Cartesian space). The problem converges to the global minimum when the points start on the line which connects the two center points.

The optimization problem is given as

$$\min_{\Phi} \ \mathbf{f}^{\,b}(\Phi) \quad \text{with } \mathbf{f}^{\,b}(\Phi) = \|\xi_1^b(\phi_1) - \xi_2^b(\phi_2)\|, \quad \Phi = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}$$

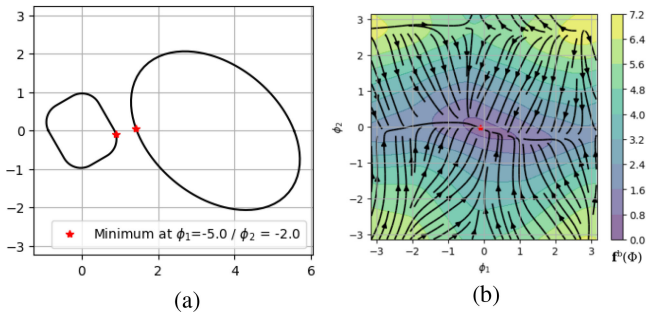


Fig. 26. Minimum distance problem for a rectangular object (with margin) and an ellipsoid. The boundary-reference-point which corresponds to the closes point is marked in red (a) and the corresponding gradient descent problem in direction space (b). (a) Minimum Distance. (b) Value Function.

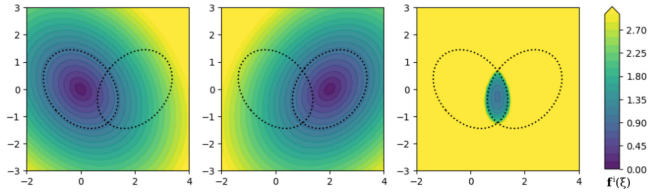


Fig. 27. Individual Γ-function from the obstacle in (a) and (b) are summed as described in (Appendix-A-3). The resulting value function (c) is used for the angle based gradient-descent.

where the $\xi_i^b(\phi_i)$ denotes the boundary point in the direction ϕ_i for the obstacle i with respect to its reference point ξ_i^r. The direction space of each obstacle is created such that the null-direction points toward the other obstacles center. An example is visualized in Fig. 26.

3) Intersecting Obstacle Descent: Two intersecting obstacles need to share one reference point, which lies within both obstacles. The simplification of the problem to surface points only is not of use anymore. Hence, the optimization problem is evaluated in Cartesian space to find a common point that lies inside of the two obstacles' boundaries

$$\min_{\xi \in \mathcal{X}_1^b \cap \mathcal{X}_2^b} \mathbf{f}^i(\xi) \quad \text{with} \quad \mathbf{f}^i(\xi) = \frac{\Gamma^b}{\Gamma^b - \Gamma_1(\xi)} + \frac{\Gamma^b}{\Gamma^b - \Gamma_2(\xi)}$$

with Γ^b the base distance value, i.e., where the value function reaches infinity. It is chosen slightly larger than the Γ-value on the surface, we simply choose $\Gamma^b = 1.1$. The step size can be optimized based on the gradient. The optimization problem is convex, and points starting within the intersection region will stay inside due to the infinite repulsion at the boundary as $\Gamma \to 1$. The value function of two ellipses can be found in Fig. 27.

4) Closest Distance for Mixed Environments: The above method for gradient descent in directional space to find the closest distance between two objects, can be applied to an object-boundary pair, if obstacle's curvature c^o is larger than boundary's curvature c^b at any position

$$c^o(\xi_1) < c^b(\xi_2) \qquad \forall\, \xi_1,\, \xi_2 \tag{51}$$

with the local curvature being defined in (32).

Note that for noncircular obstacles, the condition might locally not hold mainly if the space contains polygon obstacles with local flat regions ($c = 0$). This can lead to a locally nonoptimal solution when placing the reference point based on the closest distance.

B. Proof of Theorem 1

1) Applicability of General Proofs: In [3] convergence has been proven for star-shaped obstacles. The proof was developed based on the distance function $\Gamma(\xi)$. Due to inverting the distance function for enclosing wall obstacles and a continuous definition of the modulation, the proof of star-shaped obstacles applies to the case of enclosing walls.

2) Continuity Across Reference Point: In Section IV-A the inverted distance function $\Gamma^w(\xi)$ was not defined at the reference point, as it reaches an infinite value. The continuous definition for the eigenvalue is a unit value, i.e., $\lambda^e(\xi^r) = \lambda^r(\xi^r) = 1$, it follows that the diagonal matrix is equal to the identity matrix $D(\xi^r) = I$. As shown in (14): we get

$$\xi = \xi^r \quad \rightarrow \quad \dot{\xi} = \mathbf{E}\,\mathbf{D}\,\mathbf{E}^{-1}\mathbf{f}(\xi^r) = \mathbf{E}\,\mathbf{I}\,\mathbf{E}^{-1}\mathbf{f}(\xi^r) = \mathbf{f}(\xi^r) \tag{52}$$

that is, no modulation of the initial DS. In fact, this is equivalent to the case far away for a classical obstacle with $\lim_{\|\xi - \xi^r\| \to \infty} \Gamma^o(\xi) \to \infty$.

Even though the basis matrix $\mathbf{E}(\xi)$ is not defined at ξ^r, the DS is continuously defined across this point since the modulation has no effect.

The trajectory that traverses the reference point ξ^r of the inverted obstacle corresponds to the trajectory that gets stuck in a saddle point for a common obstacle. As a result, there is full convergence for the inverted obstacles. ■

C. Proof of Theorem 2

We show first that the modulation has full rank and hence that the dynamics does not vanish outside the attractor and that it is smooth.

1) Full Rank: The basis matrix from (21) has full rank everywhere outside of the obstacle, if the following condition holds:

$$\arccos\left(\langle \mathbf{r}(\xi),\, \hat{\mathbf{n}}(\xi) \rangle\right) < \pi/2. \tag{53}$$

The angle between the normal to each surface i, $\mathbf{n}_i(\xi)$ and the reference direction $\mathbf{r}(\xi)$ can be evaluated by defining an vector $\tilde{\mathbf{n}}_i(\xi) = \mathbf{n}_i(\xi) + \sum_{j=1}^{d-1} k_i^e \mathbf{e}_j(\xi)$ with $\langle \mathbf{e}_j(\xi),\, \mathbf{r}(\xi) \rangle = 0$, $k_i^e \in \mathbb{R}$ such that $\mathbf{p}_i^s(\xi) + \tilde{\mathbf{n}}_i(\xi)$ intersects with $\xi^r + k^r \mathbf{r}(\xi)$ at $\mathbf{q}_i^s(\xi)$ with $k^r \in \mathbb{R}$ (Fig. 28).

This allows to create a triangle spanned by the lines ξ, $\mathbf{p}_i^s(\xi)$ and $\mathbf{q}_i^s(\xi)$, colored in blue in Fig. 28. Using the associative law of the dot product, the geometry constraint of the blue triangle and (19), the maximum angle results in

$$\langle \mathbf{n}_i, \mathbf{r} \rangle = \langle \tilde{\mathbf{n}}_i, \mathbf{r} \rangle \geq \langle \xi - \mathbf{p}_i^s, \mathbf{r} \rangle \geq 0 \quad \forall\, w_i(\xi) > 0. \tag{54}$$

Hence the directional transformation of (47) results in $\|\kappa_i\| < \pi/2\|$, $\forall\, w_i(\xi) > 0$. Using additionally the *triangle equality* for vectors: $\|\kappa_1 + \kappa_2\| \leq \|\kappa_1\| + \|\kappa_2\|$ applied to all surface

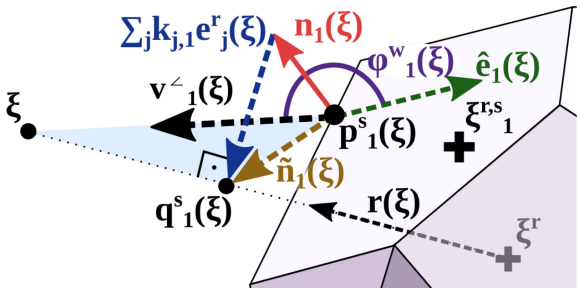


Fig. 28. Visualization of variables used for the weighted directional mean.

directions, it follows with (48) that:

$$\|\bar{\kappa}\| = \|\sum_{i=1}^{N^v} w_i \kappa_i\| \leq \sum_{i=1}^{N^v} w_i \|\kappa_i\|$$
$$\leq \left(\sum_{i=1}^{N^v} w_i\right) \|\max_{i \text{ with } w_i>0} \kappa_i\| \leq \frac{\pi}{2}.$$

Since the basis vector of the directional mean is $\mathbf{r}(\xi)$, with (49) condition (53) holds true.

2) Smooth Vector Field: The continuous extension across the reference point is defined in Appendix -2 and applicable, too.

The reference direction $\mathbf{r}(\xi)$ and the distance function $\Gamma(\xi)$ does not have any other discontinuity.

The pseudonormal $\hat{\mathbf{n}}(\xi)$ is smoothly defined across space. Even in the case when the edge point with the minimum is switching (17), no discontinuity occurs since the angle will stay the same due to the flat surface.

3) Applicability of General Proofs: Since we have a smooth field of normal vectors $\mathbf{n}(\xi)$, we further need to define any smooth distance function which decreases its value with increasing distance. The two properties are sufficient to comply with the proof of Appendix-B ■

D. Proof Theorem 3

As an agent approaches the surface of an obstacle, the importance weight from this obstacle is approaching one, i.e., $\lim_{\Gamma_{\hat{o}}(\xi)\to 1} w_{\hat{o}} = 1$, see (VI). It follows with (24) that the relative velocity is $\dot{\tilde{\xi}} = \dot{\tilde{\xi}}^{\text{tot}} = \dot{\tilde{\xi}}_{\hat{o}}$.

As a result it is sufficient to analyze impenetrability for each obstacle individually. The next step is to ensure impenetrability of the three cases in (30).

1) Evaluation in Moving Frame: $\dot{\xi} = \dot{\xi}$

The simplest case comes with no stretching, but the evaluation in the local frame of the moving boundary of the obstacle. It follows that the Neuman-boundary condition for impenetrability holds (see also [3]).

2) Contraction Within Margin: $\dot{\xi} = v^{\max}\|\dot{\xi}\|\dot{\xi}$

This contraction is only performed, if it results in a normal velocity which is larger than the velocity of the obstacle $\dot{\tilde{\xi}}$, i.e., the evaluation in the moving frame results $\langle(\dot{\xi} - \dot{\tilde{\xi}}, \mathbf{n}(\xi)\rangle \geq 0$, hence ensuring impenetrability.

3) Contraction in Tangent Direction:

$$\dot{\xi} = v^n \mathbf{n}(\xi) + \sqrt{(v^{\max})^2 - \|\dot{\xi}^n\|^2}\, \mathbf{e}(\xi).$$

This limited contraction along the normal direction ensures that the velocity in normal direction remains equal to the obstacles' velocity. The evaluation of the Neuman boundary condition in the moving frame leads to

$$\langle(v^n \mathbf{n}(\xi) + \sqrt{(v^{\max})^2 - \|\dot{\xi}^n\|^2}\, \mathbf{e}(\xi)) - \dot{\tilde{\xi}}, \mathbf{n}(\xi)\rangle = v^n - v^n = 0$$

using the definition of (29) and the fact that the normal $\mathbf{n}(\xi)$ and the tangent $\mathbf{e}(\xi)$ are orthogonal.

Hence, we have impenetrability for multiple obstacles. ■

E. Evaluation Weights for a Robot Arm

We introduce link weights w^L_l for each link $l \in [1..N^L]$, and section weights $w^S_{l,s}$ for each section point $\xi^S_{l,s}$ with $s \in [1, ..N^S]$. The weights determine how much the obstacle avoidance command influences the control, i.e., zero weights indicate following the goal velocity only, whereas a weight of 1 means full avoidance at this specific position.

These weights are designed, such that their product is smaller than one, i.e.,

$$0 \leq \sum_l \left(w^L_l \sum_s w^S_{l,s}\right) \leq 1.$$

Furthermore, if a section point approaches the surface of an obstacle, it should dominate, i.e.,

$$\Gamma^d(\xi^S_{l,s}) \to 1 \quad \Rightarrow \quad w^L_l w^S_{l,s} \to 1.$$

And when the robot arm is far away from any surface, all weights should go to zero

$$\min_{l,s}\left(\Gamma(\xi^S_{l,s})\right) \to \infty \quad \Rightarrow \quad \sum_l \left(w^L_l \sum_s w^S_{l,s}\right) \to 0.$$

1) Base Weight: For each link l and section point s, we define a Γ-danger weight $w^\Gamma(\Gamma^d) :]1, \infty[\to]\infty, 0[$ to represent the danger of colliding for a specific point: the higher the weight, the greater the chance of collision. It is defined as

$$w^\Gamma(\Gamma^d) = \begin{cases} \frac{\Gamma^c - \Gamma^{\min}}{\Gamma^d(\xi^S_{s,l}) - \Gamma^{\min}} - 1 & \text{if} \quad \Gamma^d(\xi^S_{s,l}) < \Gamma^c \\ 0 & \text{otherwise} \end{cases} \tag{55}$$

where $\Gamma^{\min} = 1$ is the lower bound, $\Gamma^c > 1$ the cutoff value and $\Gamma^d(\xi)$ defined in 34.

2) Link Weights: The influence of each link l is evaluated as

$$\hat{w}^L_l = c^L |\dot{\mathbf{q}}^g_{[l]}| \frac{l}{N^L} \max_{s\in[1..N^S]} w^\Gamma(\xi^S_{s,l}) \quad \forall\, l \in [1..N^L] \tag{56}$$

where $c^L \in \mathbb{R}_{>0}$ is a constant link-weight factor and $\dot{\mathbf{q}}^g_{[l]}$ the lth element of the goal vector defined in (33). The l-factor in the equation gives an increasing importance for links closer to the end effector.

The link weights are obtained through normalization

$$w_l^L = \begin{cases} \hat{w}_l^L / \hat{w}^{\text{sum}} & \text{if } \hat{w}^{\text{sum}} > 1 \\ \hat{w}_l^L & \text{otherwise.} \end{cases} \quad \text{with } \hat{w}^{\text{sum}} = \sum_{l=1}^{N^L} \hat{w}_l^L \tag{57}$$

3) Section Weights: For all links l with $w_l^L > 0$, the influence weight of each section point is evaluated as

$$w_s^S = \frac{\hat{w}_s^S}{\sum_{s=1}^{N^S} \hat{w}_s^S} \quad \text{with} \quad \hat{w}_s^S = \frac{s}{N_S} w^\Gamma(\xi_{l,s}^S) \quad \forall\, s \in [1..N_S]. \tag{58}$$

REFERENCES

[1] H. J. S. Feder and J.-J. Slotine, "Real-time path planning using harmonic potentials in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1997, vol. 1, pp. 874–881.

[2] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

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

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

[5] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics*, Baco Raton, FL, USA: CRC, 2000.

[6] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. J. Robot. Res.*, vol. 21, no. 12, pp. 1031–1052, 2002.

[7] Y. Zhang, N. Fattahi, and W. Li, "Probabilistic roadmap with self-learning for path planning of a mobile robot in a dynamic and unstructured environment," in *Proc. IEEE Int. Conf. Mechatron. Automat.*, 2013, pp. 1074–1079.

[8] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1199–1212, Oct. 2008.

[9] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. Konidaris, "Robot motion planning on a chip," in *Proc. Robot.: Sci. Syst.*, 2016, pp. 1–9.

[10] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[11] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.

[12] O. Arslan and D. E. Koditschek, "Exact robot navigation using power diagrams," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1–8.

[13] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53 rd IEEE Conf. Decis. Control*, 2014, pp. 6271–6278.

[14] M. F. Reis, A. P. Aguiar, and P. Tabuada, "Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria," *IEEE Contr. Syst. Lett.*, vol. 5, no. 2, pp. 731–736, Apr. 2021.

[15] G. Notomista and M. Saveriano, "Safety of dynamical systems with multiple non-convex unsafe sets using control barrier functions," *IEEE Control Syst. Lett.*, vol. 6, pp. 1136–1141, 2021. [Online] Available: https://scholar.google.com/scholar?&q=Safety%20of%20dynamical%20systems%20withmultiple%20non%2Dconvex%20unsafe%20sets%20using%20control%20barrier%20functions

[16] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Trans. Amer. Math. Soc.*, vol. 327, no. 1, pp. 71–116, 1991.

[17] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 593–600.

[18] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 656–663, Apr. 2017.

[19] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[20] J. Van den, M. Berg Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 1928–1935.

[21] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proc. IEEE /RSJ Int. Conf. Intell. Robot. Syst.*, 2009, pp. 5573–5578.

[22] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[23] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Adv. Appl. Math.*, vol. 11, no. 4, pp. 412–442, 1990.

[24] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Automat.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

[25] S. G. Loizou, "Closed form navigation functions based on harmonic potentials," in *Proc. IEEE 50th Conf. Decis. Control Eur. Control Conf.*, 2011, pp. 6361–6366.

[26] S. Paternain, D. E. Koditschek, and A. Ribeiro, "Navigation functions for convex potentials in a space with convex obstacles," *IEEE Trans. Automat. Control*, vol. 63, no. 9, pp. 2944–2959, Sep. 2018.

[27] S. G. Loizou, "The navigation transformation," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1516–1523, Dec. 2017.

[28] H. Kumar, S. Paternain, and A. Ribeiro, "Navigation of a quadratic potential with ellipsoidal obstacles," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 4777–4784.

[29] A. Duan *et al.*, "Learning to avoid obstacles with minimal intervention control," *Front. Robot. AI*, vol. 7, 2020, Art. no. 60.

[30] B. Lacevic, P. Rocco, and A. M. Zanchettin, "Safety assessment and control of robotic manipulators using danger field," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1257–1270, Oct. 2013.

[31] A. M. Zanchettin, B. Lacevic, and P. Rocco, "Passivity-based control of robotic manipulators for safe cooperation with humans," *Int. J. Control*, vol. 88, no. 2, pp. 429–439, 2015.

[32] L. Zhang, J. Wang, Z. Lin, L. Lin, Y. Chen, and B. He, "Distributed cooperative obstacle avoidance for mobile robots using independent virtual center points," *J. Intell. Robotic Syst.*, vol. 98, no. 3, pp. 791–805, 2020.

[33] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using Laplace's equation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1990, pp. 2102–2106.

[34] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Trans. Robot. Automat.*, vol. 8, no. 3, pp. 338–349, Jun. 1992.

[35] J. Guldner and V. I. Utkin, "Sliding mode control for an obstacle avoidance strategy based on an harmonic potential field," in *Proc. 32nd IEEE Conf. Decis. Control*, 1993, pp. 424–429.

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

[37] M. Saveriano and D. Lee, "Distance based dynamical system modulation for reactive avoidance of moving obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 5618–5623.

[38] D.-T. Lee and F. P. Preparata, "An optimal algorithm for finding the kernel of a polygon," *J. ACM*, vol. 26, no. 3, pp. 415–421, 1979.

[39] D. Jia, A. Hermans, and B. Leibe, "Dr-spaam: A spatial-attention and auto-regressive model for person detection in 2 D range data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10270–10277.

[40] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "Spinnet: Learning a general surface descriptor for 3 D point cloud registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11753–11762.

[41] S. Khansari-Zadeh, "A dynamical system-based approach to modeling stable robot control policies via imitation learning," PhD Thesis, Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, 2012.

[42] D. F. P. Granados, H. Kadone, and K. Suzuki, "Unpowered lower-body exoskeleton with torso lifting mechanism for supporting sit-to-stand transitions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2755–2761.

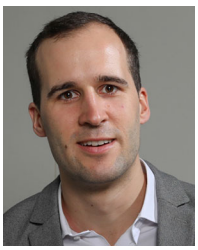
[43] F. Grzeskowiak, M. Babel, J. Bruneau, and J. Pettré, "Toward virtual reality-based evaluation of robot navigation among people," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces*, 2020, pp. 766–774.

[44] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. IEEE 18th Eur. Control Conf.*, 2019, pp. 3420–3431.

[45] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Proc. Learn. Dyn. Control*, 2020, pp. 708–717.

[46] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3387–3395.

[47] M. Fourment and M. R. Gillings, "A comparison of common programming languages used in bioinformatics," *BMC Bioinf.*, vol. 9, 2008, Art. no. 82.
[48] C. Badue *et al.*, "Self-driving cars: A survey," *Expert Syst. with Appl.*, vol. 165, 2021, Art. no. 113816.
[49] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 1, pp. 187–210, 2018.



Lukas Huber received the B.Sc. degree in mechanical engineering from the Federal Institution of Technology, Zurich (EPFL), Zurich, Switzerland, in 2015, and the M.Sc. degree in mechanical engineering with a focus on Robotics from the Technical Institution of Technology, Lausanne (EPFL), Lausanne, Switzerland, in 2018.

He visited the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2018, to further focus on nonlinear control and contraction theory. He is currently a doctoral Researcher with Learning Algorithm and System Laboratory, EPFL. His research interests include obstacle avoidance, motion learning, artificial intelligence, dynamical systems, and control.

Jean-Jacques Slotine received the Ph.D. degree in estimation and control from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1983.

He is currently a Professor of Mechanical Engineering and Information Sciences, Professor of Brain and Cognitive Sciences, and Director of the Nonlinear Systems Laboratory with the MIT, and a Distinguished Faculty with Google AI, Mountain View, CA, USA. After working with Bell Labs with the Computer Research Department, he joined the MIT faculty, in 1984. Professor Slotine is the co-author of two graduate textbooks, "Robot Analysis and Control" (Asada and Slotine, Wiley, 1986), and "Applied Nonlinear Control" (Slotine and Li, Prentice-Hall, 1991) and is one of the most cited researchers in systems science and robotics. His research interests include developing rigorous but practical tools for nonlinear systems analysis and control. These have included key advances and experimental demonstrations in the contexts of sliding control, adaptive nonlinear control, adaptive robotics, machine learning, and contraction analysis of nonlinear dynamical systems.

Dr. Slotine was a Member of the French National Science Council, from 1997 to 2002, and of Singapore's A*STAR SigN Advisory Board, from 2007 to 2010, and currently is on the Scientific Advisory Board of the Italian Institute of Technology. He was the recipient of the 2016 Oldenburger Award.

Aude Billard (Fellow, IEEE) received the M.Sc. degree in physics from the Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, in 1995, and the M.Sc. degree in knowledge-based systems and Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K., in 1996 and 1998, respectively.

She is currently a Professor of Micro and Mechanical Engineering and the Head of the Learning Algorithms and Systems Laboratory, School of Engineering, EPFL. Her research interests include machine learning tools to support robot learning through human guidance. This also extends to research on complementary topics, including machine vision and its use in human-robot interaction and computational neuroscience to develop models of motor learning in humans.

Dr. Billard was the recipient of the Intel Corporation Teaching Award, the Swiss National Science Foundation Career Award, in 2002, the Outstanding Young Person in Science and Innovation from the Swiss Chamber of Commerce, and the IEEERAS Best Reviewer Award, in 2012. She served as an Elected Member of the Administrative Committee of the IEEE Robotics and Automation Society (RAS) for two terms, from 2006 to 2008 and 2009 to 2011, and is the Chair of the IEEE-RAS Technical Committee on Humanoid Robotics.