Integrated robot planning, path following, and obstacle avoidance in two and three dimensions: wheeled robots, underwater vehicles, and multicopters

Article *in* The International Journal of Robotics Research · May 2019

DOI: 10.1177/0278364919846910

CITATIONS
6

READS
429

1 author:

Antonio Sgorbissa
Università degli Studi di Genova
174 PUBLICATIONS **1,515** CITATIONS

SEE PROFILE

Integrated Robot Planning, Path Following and Obstacle Avoidance in 2D and 3D: wheeled robots, underwater vehicles and multicopters

Journal Title
XX(X):1–27

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

Antonio Sgorbissa[1]

Abstract

The article proposes an innovative, integrated solution to path planning, path following and obstacle avoidance that is suitable both for 2D and 3D navigation. The proposed method takes in input a generic curve connecting a start and a goal position, and is able to find a corresponding path from start to goal in a maze-like environment even in absence of global information, it guarantees convergence to the path with kinematic control, and finally avoids locally-sensed obstacles without being trapped in deadlocks. This is achieved by computing a closed-form expression, in which the control variables are a continuous function of the input curve, the robot's state, and the distance of all the locally-sensed obstacles. Specifically, the article introduces a novel formalism for describing the path in 2D and 3D, as well as a computationally efficient method for path-deformation (based only on local sensor readings) that is able to find a path to the goal even when such path cannot be produced through continuous deformations of the original one. The article provides formal proofs of all the properties above, as well as simulated results in a simulated environment with a wheeled robot, an underwater vehicle, and a multicopter.

Keywords

Autonomous Agents, Motion Control, Wheeled Robots, Marine Robotics, Aerial Robotics

1. Introduction

When designing autonomous robots, the capabilities of i) planning a path to the goal, ii) following the prescribed path, and iii) locally updating the path in order to avoid sensed obstacles, play a fundamental role. Providing these capabilities is especially challenging when the robot is operating in an unknown and cluttered environment, possibly crowded with people or other moving robots. Approaches in the Literature can be classified into different categories, depending on the specific problem (or subset of problems) they focus on.

Traditional global planning approaches (Latombe 1991; LaValle 2006) are able to find a path to the goal, but only if global knowledge about the environment is available: in most cases, such approaches do not provide a solution to locally update the path in presence of obstacles and they do not deal with path following at all. Typically, the integration of path planning, path following and local obstacle avoidance is accomplished according to a multi-tiered architecture: given a path returned by a global planner (tier 1), a controller is implemented to move along the path when the vehicle is in the free-space (tier 2), by switching to obstacle avoidance maneuvering in the presence of obstacles (tier 3). The planner is then invoked to replan a new path (tier 1) if local obstacle avoidance leads to a deadlock situation.

Local obstacle avoidance algorithms are adopted for real-time obstacle avoidance in absence of global information. Among the others, it is worth mentioning the very popular Artificial Potential Fields (Khatib 1986), Vector Field Histograms (Borenstein and Koren 1991; Ulrich and Borenstein 2000) and the Dynamic Window Approach (Fox et al. 1997; Ogren and Leonard 2005), which extends local avoidance strategies with techniques to take into account kinematics constraints. APF, VFH and DWA are similar in that they do not require a path to be followed, since

[1]University of Genova, Italy

Corresponding author:
Antonio Sgorbissa, University of Genova, DIBRIS, Via Opera Pia 13, Genova, 16145, Italy.

Email: antonio.sgorbissa@unige.it

only the target (or the next waypoint) to be reached is needed: when the vehicle is far from obstacles, the path ideally corresponds to a straight line connecting the current position to the target. On the opposite, (Lapierre et al. 2007) proposes an algorithm that drives a unicycle-type robot along a given path with obstacle avoidance capabilities. The path is described in its parametric form, by describing its curvature as a function of the curvilinear abscissa: then, path following is achieved by explicitly controlling the rate of progression of a "virtual target" to be tracked along the path, an idea that was proposed for the first time in (Aicardi et al. 1995) and then applied to AUVs in (Lapierre and Soetanto 2007). Obstacle avoidance is achieved by introducing the Deformable Virtual Zone (DVZ) principle, that induces a local deformation in the path. All local obstacle avoidance algorithms have well known problems in guaranteeing goal reachability, e.g., due to local minima. In principle, DWA and DVZ could be combined with a geometric search strategy, e.g., belonging to the BUG family (Lumelsky and Stepanov 1987), in order to guarantee maze solving capabilities even in absence of global knowledge. However, to the best of our knowledge, this integration has never been attempted. Then, adopting a multi-tiered approach turns out to be – once again – the most comon solution, where local obstacle avoidance algorithms typically constitute the core of tiers 2 and/or 3: they provide path following capabilities and/or a local strategy that switches to obstacle avoidance maneuvering in the presence of obstacles, by demanding the planning problem to a different component of the system.

More recent works have focussed on the problems above. Among the others, (Karaman and Frazzoli 2011) introduces the concept of object shadow, based on modeling the collision states (shadows) created in front of the obstacles for longitudinal motion at constant speeds. The idea is that, as the speed or the obstacle density increases, the free space between the shadows decreases until reaching some critical value beyond which collision-free motion at a constant speed cannot be guaranteed. Based on this concept, the work described in (Shiller et al. 2013) presents an efficient algorithm for online avoidance of static obstacles in very cluttered environments (70 obstacles) that accounts for robot dynamics and actuator constraints. The robot trajectory (path and speed) is generated incrementally by avoiding obstacles optimally one at a time (i.e., by transforming the multi-obstacle problem with m obstacles into m simpler subproblems with one obstacle each), thus yielding a computational complexity that is linear in the number of obstacles. A multi-agent approach to path following and obstacle avoidance for wheeled vehicles is proposed in (Dafflon et al. 2014): here the path is described as a set of straight segments connecting waypoints, and – in presence of obstacles – virtual agents produce a safe path by reacting to environment variations through attraction/repulsion behaviours. A local path planning approach for an independent four-wheel steering mobile base has been recently presented in (Todoran and Bader 2016). The approach smoothly drives and rotates an omnidirectional vehicles to keep a target under observation while still following a given path and avoiding obstacles. Local path planning is performed by sampling the control input space according to the DWA approach, and then a global path planner would be required to guarantee that the goal can be reached in complex environments. The work presented in (Regier et al. 2017) proposes a solution based on the popular DWA that aims at minimizing the estimated completion time instead of the path length, while taking into account the smoothness and the clearance of the path.

The multi-tiered approach has been recently extended to different robotic domains, e.g., flying or underwater robots. In the case of flying robots (Shim et al. 2006; Orsag et al. 2015; Gageik et al. 2015; Lee et al. 2016; Liu et al. 2017), the path may be represented by a set of straight segments connecting waypoints, and one of the many methods available may be adopted for path following with UAVs (Do et al. 2003; Consolini et al. 2010; Nelson et al. 2007). Regarding obstacle avoidance, a large number of heuristic obstacle avoidance methods have been especially developed for UAVs, based on Particle Swarm Optimization (Foo et al. 2009), APFs (Franchi et al. 2012), and probabilistic roadmap-based methods, i.e., by sampling points from the environment's free space, and connecting them to neighboring points if a collision-free path exists between them (Hrabar 2008). (Tomić et al. 2017) presents a unified framework for external wrench estimation, interaction control, and safe collision reaction for flying robots. However, the approach does not deal with the general problem of finding a path to the goal while safely avoiding all obstacles in a maze-like environment. The underwater domain requires facing additional technical challenges (Millar 2014; Zhang et al. 2016; Liu et al. 2016). For instance, (Braginsky and Guterman 2016) proposes a methodology for obstacle avoidance by AUVs that are equipped with forward-looking sonars. Due to the necessity of maintaining constant height when employing sidescan sonar and lower energy consumption, horizontal avoidance maneuvers are preferred over vertical ones, based on a preplanning method and a reactive approach based on APFs and edge detection methods. In case a horizontal approach

cannot find a safe path to avoid the obstacles, a reactive vertical approach is activated.

In this general scenario, a class of approaches exists dealing with path planning, path following and obstacle avoidance in *a more integrated way*: in particular, all the following approaches are based on the idea of introducing an *innovative representation of the path*, which is no longer a curve described in parametric form or a straight-line virtually connecting the start (or the current position) to the goal. To the best of our knowledge, Elastic Bands and Elastic Strips are the first attempt to represent not only the path to be followed, but also a region of the space within which the robot is allowed to deviate from such path (Quinlan and Khatib 1993). The approach includes a method that enables one to deform the path in order to get away from obstacles detected along the motion, and has been extended to the case of a unicycle-type robot in (Khatib et al. 1997) and then to the case of a nine-degree-of-freedom mobile manipulator and a 34-degree-of-freedom humanoid robot in (Brock and Khatib 2002). The problem with Elastic Bands and Elastic Strips is that they can produce only paths that are homotopic to the nominal one (i.e., produced through continuous deformations), and therefore could fail to deform to a collision free-path to the goal even if one exists. This happens, for instance, when closing the door through which a robot has planned to pass: even if another door is open and available to be used, it may be not possible to deform the path as required. In this case, a global planner should be invoked, thus requiring – once again – a multi-tiered approach. In the same spirit, the work presented in (Pathak and Agrawal 2005) describes an integrated path planning and control approach for nonholomic unicycles in an obstacle-ridden environment. A global planner is used to first create a string of variable-sized circular areas – called "bubbles" – which connects the start point to the goal point, with each bubble's size indicative of the radial obstacle clearance available from its center. The robot then moves according to the direction provided by the global plan, while repulsively avoiding unexpected obstacles and keeping itself within the bubbles thanks to two potential-field-based controllers: the first controller drives the unicycle to the center of its bubble, while the second corrects its orientation, by guaranteeing that kinematics and dynamics constraints are not violated. (Sgorbissa and Zaccaria 2012) propose a different approach, in which a purposely-designed planner computes the path as a chain of "Roaming trails", diamond shaped areas that are computed in such a way to guarantee that their intersection with the free-space is always a convex region. During motion, a robot that avoids obstacles by staying within the borders of the diamond shaped area is guaranteed never to be trapped in a concavity made by an unlucky configuration of obstacles, since the next waypoint is always reachable from every point within the Roaming Trail by construction.

Progress beyond state-of-the-art

The contribution of the article is to propose an innovative, integrated solution to path planning, path following and obstacle avoidance that is suitable both for 2D and 3D navigation. The proposed method can:

- take as an input a generic curve connecting a start and a goal location and find a path in a maze-like environment even in absence of global information (the path will lie on the given curve in the free-space, and will diverge from it elsewhere),
- guarantee convergence to the path with kinematic control,
- avoid locally-sensed obstacles without being trapped in deadlocks,

by computing control variables through a closed-form expression, which is a continuous function of

- the input curve,
- the robot's state,
- and the distance of all the locally-sensed obstacles.

Specifically, this is achieved by introducing

- a *novel formalism for describing the path in 2D and 3D* that can be straightforwardly used as a measure of the error in path following, and then can directly be fed to a real time controller;
- a c*omputationally efficient method for path-deformation* (based only on local sensor readings) that produces paths that are not necessarily homotopic to the original path, and is therefore able to find a path to the goal even when such path cannot be produced through continuous deformations of the original one.

Finally notice that the article does not deal with dynamic control. However, most of the solutions proposed in the article are general (with the only exception of Section 4 that deals explicitly with kinematic path following), and may constitute a theoretical framework to design a family of control algorithms for dynamic path following.

Section 2 introduces the novel representation adopted for describing the path in 2D or 3D. Section 3 introduces

the method for updating the path in presence of obstacles. Section 4 describes three case studies in which the method is applied for path following in 2D (wheeled robots) and 3D (underwater vehicles and multicopters). Section 5 discusses the maze-solving capabilities of the approach. Section 6 describes simulated experiments with wheeled robots, underwater robots and multicopters, performed in MATLAB and Simulink. Conclusions follow.

2. Path Definition

In this Section we propose a method to describe a path as a curve in a 3D or a 2D workspace.

In 3D, we assume a fixed frame (*n*–frame) describing North-East-Down positions in Earth-Fixed coordinates, and a body frame (*b*–frame) which moves with tthe vehicle. The vector $(x, y, z, \phi, \theta, \psi)$ describes the position and orientation of the *b*–frame with respect to the the *n*–frame (using Euler angles in the roll-pitch-yaw notation). Since this Section deals only with the definition of the geometric path, linear and angular velocities are not relevant for the discussion and will be introduced later in Section 4. In 2D, the vector describing position and orientation comprises the (x, y, ψ) components only, since all other components can be considered as constantly null.

3D Paths

Differently from other approaches in the Literature, the path is not described as a curve in parametric form. Instead, taking inspiration from previous work with wheeled robots (Morro et al. 2011; Sgorbissa and Zaccaria 2013), N-trailers (Michalek 2014), and Unmanned Underwater Vehicles (Sgorbissa and Zaccaria 2010), paths are described as the intersection of two surfaces in $\Re^3$ represented through implicit equations in the form $f_1(x, y, z) = 0$ and $f_2(x, y, z) = 0$.

Specifically, the set of intersection points $\mathcal{C} = \{(x, y, z)\}$ is given by the solutions of the system

$$\begin{aligned} f_1(x, y, z) &= 0 \\ f_2(x, y, z) &= 0, \end{aligned} \tag{1}$$

where f_1 and f_2 are properly chosen to produce the desired path and meet the following constraints:

C.1) $f_i = f_i(x, y, z) : \Re^3 \rightarrow \Re$, $i = 1, 2$, are twice differentiable functions with first derivatives f_{ix}, f_{iy}, f_{iz};

C.2) $\|\nabla f_i\|^2 = f_{ix}^2 + f_{iy}^2 + f_{iz}^2 > 0$, $i = 1, 2$, in $\Re^3$ possibly deprived of a neighborhood of points D_i where $\|\nabla f_i(x, y, z)\| = 0$ (i.e., the gradient is never null in $\Re^3 \setminus D_i$)*.

C.3) $\nabla f_1 \times \nabla f_2 \neq 0$ in $\Re^3$ possibly deprived of a neighborhood of points D_{12} not belonging to the intersection $\mathcal{C}$ (i.e, the two gradients are always linearly independent in $\Re^3 \setminus D_{12}$).

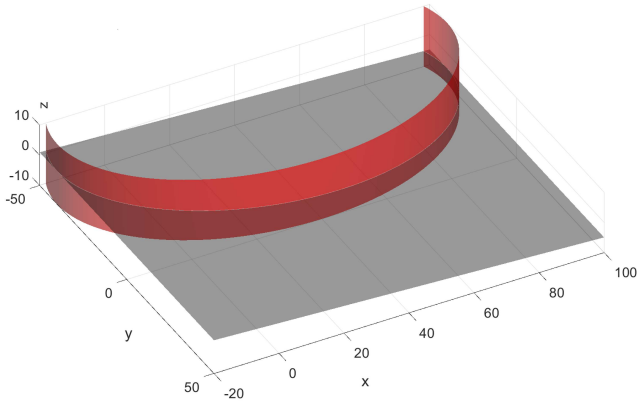

Figure 1. Intersection of a cylinder $f_1(x, y, z) = 0$ (red surface) and a plane $f_2(x, y, z) = 0$ defining an ellipsoidal path.

The intersection $\mathcal{C}$ can consist of multiple curves $\mathcal{C}_j \subseteq \mathcal{C}$.

Remark 1. *Each curve $\mathcal{C}_j \subseteq \mathcal{C}$ is a simple curve, i.e., a curve without self-intersections. This is a well-known consequence of the constraint C.3: since the gradients ∇f_1 and ∇f_2 are linearly independent (i.e., the two surfaces always intersect transversally in $\mathcal{C}_j$), the tangent to the curve is uniquely defined as $\nabla f_1 \times \nabla f_2$ for every $(x, y, z) \in \mathcal{C}_j$, and therefore a self-intersection cannot exist.* □

Remark 2. *Each curve $\mathcal{C}_j$ is either a closed curve or an infinite curve: this follows from Remark 1 and the fact that the gradients ∇f_1 and ∇f_2, and hence the tangent vector $\nabla f_1 \times \nabla f_2$, are continuous in $\Re^3$.* □

Remark 3. *Each surface $f_i(x, y, z) = 0$ divides the space in regions such that $f_i(x, y, z) < 0$ and regions such that $f_i(x, y, z) > 0$. By inverting the sign of $f_i(x, y, z)$, the shape of these regions is unaltered, but the value of $f_i(x, y, z)$ within each region is inverted as well: as it will be clarified later (Remarks 16, 17, 19), this may be used to determine the direction of motion of the vehicle along the path.* □

As an example, consider two non-parallel planes

$$\begin{aligned} f_1(x, y, z) &= a_1 x + b_1 y + c_1 z + d_1 = 0 \\ f_2(x, y, z) &= a_2 x + b_2 y + c_2 z + d_2 = 0, \end{aligned} \tag{2}$$

*In principle, one should add the additional constraint that the neighborhood D_i where the gradient is null cannot include points $\{(x, y, z) | f_i(x, y, z) = 0\}$: however, this can happen only in degenerate cases when $f_i(x, y, z) = 0$ does not describe a surface, e.g., $f_i(x, y, z) = x^2 + y^2 + z^2 = 0$.

with properly chosen coefficients a_i, b_i, c_i, d_i: the intersection defines a straight path. Another example is shown in Figure 1. In order to obtain an ellipsoidal path it is possible to intersect a cylinder with a plane:

$$\begin{aligned} f_1(x,y,z) &= (x-a_1)^2 + (y-b_1)^2 - c_1^2 = 0 \\ f_2(x,y,z) &= a_2 x + b_2 y + c_2 z + d_2 = 0, \end{aligned} \tag{3}$$

given that the center a_1, b_1 and the radius c_1 of the cylinder, as well as the coefficients a_2, b_2, c_2, d_2, are properly chosen.

Multiple parallel paths can be produced by intersecting a sinusoidal profile with a plane:

$$\begin{aligned} f_1(x,y,z) &= z - a_1 \sin(b_1 x + c_1) = 0 \\ f_2(x,y,z) &= a_2 x + b_2 y + c_2 z + d_2 = 0, \end{aligned} \tag{4}$$

by properly choosing coefficients a_1, b_1, c_1, a_2, b_2, c_2, d_2.

Remark 4. *In the article, it is assumed that the robot has no a priori knowledge about the environment: it is only able to acquire in run-time local information about obstacles within a limited sensing range. Under these conditions, a straight line connecting the start and goal positions is likely to be the best choice for the a priori path, given that the robot is capable to avoid locally sensed obstacles as they are encountered. Should a priori knowledge about the environment be available, a method might be applied to find an optimal a priori path among known obstacles (e.g., A^*, Visibility Graph or other graph–based approaches to path–planning (Latombe 1991; LaValle 2006; Sgorbissa and Zaccaria 2012). This may produce a piecewise curve composed of (not necessarily straight) segments connecting adjacent waypoints, by possibly taking into account continuity constraints in the path and its derivatives in proximity of waypoints (e.g., to deal with curvature bounds when concatenating subsequent segments). The approach proposed here is suited to all the alternatives above, but it deals only with the problem of representing the path and then modifying it in run-time in presence of unpredicted obstacles: the article does not deal with the problem of choosing the a priori path, for which the reader can refer to existing approaches in the Literature.* □

The choice of representing the path through the intersection of two surfaces, each defined through an implicit equation $f_i(x,y,z) = 0$, has two strong motivations.

First, given a robot located in (x,y,z), the value returned by $f_i(x,y,z)$ can be taken as a measure of the error between the current position of the robot and the surface itself. Indeed, it holds $f_i(x,y,z) = 0$ only when the vehicle is located on the surface, whereas $f_i(x,y,z) \neq 0$ when the vehicle is not on the surface. Specifically, $f_i(x,y,z)$ increases or decreases depending on which side the vehicle is located with respect to the surface, i.e., depending on the level surface $f_i(x,y,z) = w$, $w \neq 0$, passing through (x,y,z). Notice that, in the case that $f_i(x,y,z) = 0$ defines a plane as in (2), the value w returned in (x,y,z) corresponds to the signed Euclidean distance of the vehicle from the surface*, times the scaling factor $|\nabla f_i(x,y,z)|$. In other cases (e.g., a cylinder), w is not a measure of the distance, but it still has the property that it locally increases or decreases depending on the distance of the vehicle from the surface. This property will be exploited by path following algorithms in Section 4.

Second, the representation introduced in (1) exhibits good properties when updating the path in run-time in presence of obstacles: Section 3 proposes an approach able to produce a deformed, collision avoiding path at a low computational cost (complexity is evaluated in Section 6). Specifically, path deformation can be computed locally by considering each sensor reading individually, i.e., without requiring expensive procedures for segmentation and clustering. In spite of its simplicity, the resulting path takes into account the mutual influence of neighbouring obstacles and drives the vehicle to the goal even in presence of maze-like obstacle configurations: that is, without suffering from the well known problems related to local minima (which, on the opposite, are a common drawback of local approaches).

2D Paths

The special case of a 2D path on a plane can be considered by setting $f_2(x,y,z) = z = 0$ (e.g., to define the ground level), and properly choosing $f_1(x,y,z)$ depending on the path to be followed. In this case, it is possible to make the z variable disappear, and represent the curve through a single implicit equation describing a planar 2D curve:

$$f_1(x,y) = 0. \tag{5}$$

As in the 3D case, the level curve $w = f_1(x,y)$ can be taken as a measure of the error between the current position (x,y) of the vehicle and the path: $f_1(x,y) = 0$ if and only if the vehicle is on the curve, whereas $f_1(x,y)$ locally increases / decreases depending on the position of the vehicle with respect to the curve itself.

*This can be easily verified using the standard formula for computing the distance from a point to a plane.

3. Obstacle avoidance in 3D through surface intersection

3D Path deformation

To simplify the discussion, let the following conditions hold concerning $f_1(x,y,z)$ and $f_2(x,y,z)$:

A.1) $f_{1x}^2 + f_{1y}^2 \neq 0$ in $\Re^3$, i.e., the gradient $\nabla f_1(x,y,z)$ is never parallel to the *Down* axis of the world frame. That is, $f_1(x,y,z) = 0$ mostly gives information about the desired direction of motion of the vehicle along the *North-East* axes: a counter example would be a plane parallel to the ground plane.

A.2) $f_{2z} \neq 0$ in $\Re^3$, i.e., the gradient $\nabla f_2(x,y,z)$ always has a non-null component along the *Down* axis. That is $f_2(x,y,z) = 0$ mostly gives information about the desired altitude of the vehicle along the *Down* axis: a counter example would be a plane perpendicular to the ground plane.

The assumptions above are not constraints, since a generic couple of surfaces meeting constraints C.1–C.3 can be used: however, A.1 and A.2 are convenient to illustrate the basic principles of the proposed approach.

Let us now assume that the path intersects a single obstacle $\mathcal{O}_j$, which we initially model as a point in (x_j, y_j, z_j) and therefore it holds $f_1(x_j,y_j,z_j) = 0$ and $f_2(x_j,y_j,z_j) = 0$. In presence of an obstacle, one should first choose how to avoid it, i.e., by mostly operating on the *North-East* direction of motion of the vehicle, determined by the surface $f_1(x,y,z) = 0$, or the altitude, determined by $f_2(x,y,z) = 0$. Once this choice has been performed, the corresponding surface must be modified, with the final results of modifying the intersection in (1) that defines the path.

Suppose to modify $f_1(x,y,z) = 0$: in order to avoid the obstacle $\mathcal{O}_j$, it is sufficient to add a term $O_j(x,y,z)$ to the left side of its implicit equation. In the eventuality that N obstacles $\mathcal{O}_j$ are present, the individual contributions of all the obstacles are summed up. This will produce a deformed intersection $\mathcal{C}'$ given by the solutions of the system:

$$\begin{aligned} f_1'(x,y,z) &= f_1(x,y,z) + \sum_{j=1}^{N} O_j(x,y,z) = 0 \\ f_2(x,y,z) &= 0, \end{aligned} \tag{6}$$

By defining $d_j(x,y,z) = |(x,y,z) - (x_j,y_j,z_j)|$ as the distance between the robot and the obstacle and $\sigma > 0$ the obstacles influence range (which depends on the maximum sensor range), the shape of $O_j(x,y,z)$ must be properly chosen so that:

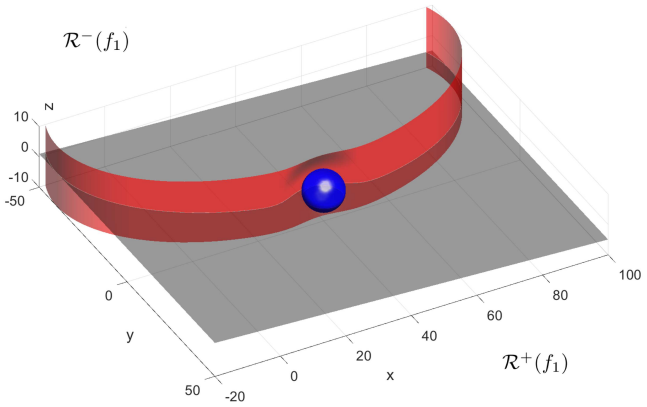

Figure 2. Intersection of a plane and a cylinder $f_1(x,y,z) = 0$ "deformed" by an obstacle. Regions $\mathcal{R}^-(f_1)$ where $f_1(x,y,z) \leq 0$ and $\mathcal{R}^+(f_1)$ where $f_1(x,y,z) \geq 0$ are indicated.

- $f_1'(x,y,z) \neq f_1(x,y,z)$ when $d_j(x,y,z) = 0$, which guarantees that $f_1'(x,y,z)$ does not intersect the obstacle: this can achieved by imposing $O_j(x,y,z) \neq 0$ when the vehicle is close to the point obstacle;
- $f_1'(x,y,z) = f_1(x,y,z)$ when $d_j(x,y,z) \geq \sigma$, which guarantees that obstacles affect the path only within the maximum influence range: this requires $O_j(x,y,z) = 0$ when the vehicle is far from the point obstacle.

To achieve this behaviour, a possible choice is a function with spherical symmetry centered in (x_j, y_j, z_j), such as

$$O_j(x,y,z) = \begin{cases} A_j \left(1 + \cos\left(\pi d_j(x,y,z)/\sigma\right)\right) & d_j < \sigma \\ 0 & d_j \geq \sigma \end{cases} \tag{7}$$

by properly choosing its amplitude A_j*. To better understand the meaning of (7), notice that $O_j(x,y,z)$ was modelled as a Gaussian over a 3-dimensional domain in previous works (Nguyen et al. 2017), with A_j being the Gaussian's amplitude, σ its standard deviation, and $d_j(x,y,z)^2$ the squared distance of (x,y,z) from the center of the Gaussian. The behaviour of the system turns out to be very similar in practice. However, modelling $O_j(x,y,z)$ as a Gaussian unrealistically assumes that $O_j(x,y,z) \neq 0$ in $\Re^3$, which is theoretically incompatible with the assumption that obstacles are locally sensed in run-time, and therefore influence the path only within a finite distance.

To visualize the effect of the deformation induced by an obstacle, consider Figure 2: a cylinder is shown, described

*If $A_j > 0$ (respectively, $A_j < 0$), the obstacle function (7) has spherical level surfaces $O_j(x,y,z) = w$ in $\Re^3$ for $A_j > w > 0$ (respectively, $A_j < w < 0$).

as $f_1(x,y,z) = (x-40)^2 + y^2 + (z+3)^2 - 36 = 0$, deformed by a single obstacle centered in (x_j, y_j, z_j) and lying on that surface (i.e., it holds $f_1(x_j, y_j, z_j) = 0$).

Remark 5. *$O_j(x,y,z)$ has a spherical symmetry and it reaches its maximum (if $A_j > 0$) or minimum (if $A_j < 0$) when $d_j(x,y,z) = 0$, it monotonically tends to zero as $d_j(x,y,z) \to \sigma$, and it is uniformly null for $d_j(x,y,z) \geq \sigma$. $O_j(x,y,z)$ is differentiable in $\Re^3$, and twice differentiable in $\Re^3$ deprived of a spherical surface with radius σ centered in (x_j, y_j, z_j).* □

Remark 6. *The intersection $\mathcal{C}$ in (1) divides the surface $f_2(x,y,z) = 0$ into two regions* $\mathcal{R}^-(f_1) = \{(x,y,z) | f_2(x,y,z) = 0 \wedge f_1(x,y,z) \leq 0\}$ and $\mathcal{R}^+(f_1) = \{(x,y,z) | f_2(x,y,z) = 0 \wedge f_1(x,y,z) \geq 0\}$. Given $\mathcal{O}_j$, if we constrain $A_j > 0$, the deformed intersection $\mathcal{C}'$ lies in $\mathcal{R}^-(f_1)$, otherwise $\mathcal{C}'$ lies in $\mathcal{R}^+(f_1)$: in other words, the sign of A_j determines if obstacles are avoided by moving in $\mathcal{R}^-(f_1)$ or $\mathcal{R}^+(f_1)$. It is straightforward to notice that $\mathcal{R}^-(f_1) = \mathcal{R}^+(-f_1)$ and $\mathcal{R}^+(f_1) = \mathcal{R}^-(-f_1)$.* □

Section 5 discusses how the a priori choice of the sign of A_j may have a significant impact on the possibility to find a path to a goal or not, and how goal reachability can be guaranteed – under some conditions – by switching the sign of A_j in run-time.

Similarly, the choice about the surface to be modified may be crucial, and depends on external factor: a multicopter passing through a door may require to decrease the altitude (i.e., by substituting f_2 with f_2'), whereas a tree can be conveniently avoided by turning around it (i.e., by substituting f_1 with f_1').

Remark 7. *In the rest of the article we suppose, without loosing generality, to modify $f_1(x,y,z) = 0$: avoiding obstacles along the $Down$ axis instead of turning around them along the $North - East$ axes can be obtained, for instance, by switching the analytical expressions of $f_1(x,y,z) = 0$ with $f_2(x,y,z) = 0$ in (6) – thus switching conditions A.1 and A.2.* □

Remark 8. *The deformed intersection $\mathcal{C}'$ can be composed of multiple curves $\mathcal{C}_j' \subseteq \mathcal{C}'$. Notice that, when considering $f_1'(x,y,z)$ in place of $f_1(x,y,z)$, the constraints C.1 and C.2 still hold: the deformed surface $f_1'(x,y,z) = 0$ is twice differentiable in $\Re^3$ (with the limitations in Remark 5) because it results from the sum of twice differentiable functions, and $\|\nabla f_1'\| > 0$ in $\Re^3$ possibly deprived of a neighborhood D_1' (which, in general, is different from D_1). However, we have no formal guarantees that C.3 still holds: in principle, it is possible that the two surfaces $f_1'(x,y,z) = 0$ and $f_2(x,y,z) = 0$ are tangent, i.e., the two gradients $\nabla f_1'$ and ∇f_2 are linearly dependent in a set of points belonging to the intersection $\mathcal{C}$. This event depends on a particular configuration of a finite number N of obstacles (which never occurred in simulations). Should the event occur, it can be easily detected: if the robot reaches a position where $\nabla f_1' \times \nabla f_2 = 0$, it is sufficient to slightly move the centers of obstacles to restore the normal situation. In the following, to simplify the discussion, we assume that C.3 holds also after path deformation for the couple of surfaces $f_1'(x,y,z) = 0$ and $f_2(x,y,z) = 0$: together with C.1 and C.2 this is sufficient to guarantee that each curve $\mathcal{C}_j'$ is a simple closed or infinite curve.* □

Remark 9. *One may consider the possibility to modify both surfaces to avoid the same obstacle $\mathcal{O}_j$, by paying care that the resulting surfaces $f_1'(x,y,z) = 0$ and $f_2'(x,y,z) = 0$ meet constraints C.1–C.3. However, modifying both surfaces around the same obstacle $\mathcal{O}_j$ may increase the chances that $\nabla f_1' \times \nabla f_2' \approx 0$ (i.e., the two deformed surfaces may be close to tangent in the proximity of $\mathcal{O}_j$): a formal analysis of the system's behaviour under these conditions has not been performed yet, and may be subject of future work.* □

2D Path Deformation

In the special case of a 2D path on a plane, the deformed path in presence of N obstacles O_j is given by the first Equation in (6) by making the z variable disappear:

$$f'(x,y) = f(x,y) + \sum_{j=1}^{N} O_j(x,y) = 0 \qquad (8)$$

with (7) analogously modified to describe a 2-dimensional bell-function $O_j(x,y) : \Re^2 \to \Re$.

Shaping the Obstacle Functions - 3D case

The choice of the actual values of A_j deserves a deeper discussion. Consider an obstacle $\mathcal{O}_j$ that is not point-like, and is located in a generic position (x_j, y_j, z_j) that does not lie on the path. It is necessary to take into account both the dimensions of the obstacle and of the vehicle, plus a safety margin: they are summarized by the quantity r_j, in the following referred as the radius of $\mathcal{O}_j$. To avoid dangerous situations, the vehicle shall be constrained to move in such a way that the distance $d_j(x,y,z)$ between its center (x,y,z) and the obstacle center (x_j, y_j, z_j) is greater than r_j. That is, an individual obstacle is defined as a spherical neighborhood

*Each region can be composed of disconnected components, if $\mathcal{C}$ is composed of multiple curves $\mathcal{C}_j$.

$$\mathcal{O}_j = \{(x, y, z) \mid d_j(x, y, z) \leq r_j\}, \tag{9}$$

and the obstacle region $\mathcal{O} \subseteq \Re^3$ is defined as

$$\mathcal{O} = \bigcup_{j=1}^{N} \mathcal{O}_j. \tag{10}$$

The actual values of A_j can be chosen by considering that, to guarantee that collisions are avoided, the path should lie outside $\mathcal{O}$. Obviously, it must hold the additional constraint that, for each obstacle, the influence range $\sigma > r_j$. Otherwise, if $\sigma \leq r_j$ there could be potential collisions that do not have any impact on path deformation, i.e., when $\sigma \leq d_j(x, y, z) \leq r_j$ (7).

To guarantee that, in presence of N obstacles, the path does not intersects $\mathcal{O}$, the following must hold:

$$f_1'(x, y, z) \neq 0, \ \forall (x, y, z) \in \mathcal{O}. \tag{11}$$

From (11) and (6) it follows that

$$f_1(x, y, z) + \sum_{j=1}^{N} O_j(x, y, z) \neq 0, \ \forall (x, y, z) \in \mathcal{O} \tag{12}$$

which has solutions for A_j if and only if holds either

$$f_1(x, y, z) + \sum_{j=1}^{N} O_j(x, y, z) > 0, \ \forall (x, y, z) \in \mathcal{O} \tag{13}$$

or

$$f_1(x, y, z) + \sum_{j=1}^{N} O_j(x, y, z) < 0, \ \forall (x, y, z) \in \mathcal{O}. \tag{14}$$

Suppose that we have made the a priori choice $A_j > 0$ for all obstacles. In this case, instead of computing the absolute value of A_j to satisfy *either* (13) *or* (14), it is convenient to focus exclusively on (13), which has two good properties allowing us to simplify computations:

P.1) for obstacles $\mathcal{O}_j$ lying completely in the semispace with $f_1(x_j, y_j, z_j) > 0$, (13) is always satisfied;

P.2) if (13) is satisfied for each individual obstacle taken separately, it is also verified when considering all the obstacles as a whole.

Both properties above are due to the fact that, when $A_j > 0$, each individual obstacle adds a positive contribution to $f_1'(x, y, z)$. According to the rationale above, we can compute A_j and σ to satisfy (13) for each individual obstacle taken separately. Figure 3 illustrates this concept: $f_1(x, y, z) \geq 0$ in $\mathcal{R}^+(f_1)$ (on the right side) and $f_1(x, y, z) \leq 0$ in $\mathcal{R}^-(f_1)$ (on the left side). Under the condition $A_j > 0$, property P.1 states that an individual obstacle in $\mathcal{R}^+(f_1)$ (blue balls outside the "red envelope") adds a positive contribution in a region $\mathcal{O}_j$ where $f_1(x, y, z) \geq 0$, and then $f_1'(x, y, z) \geq 0$ as well. On the other side, an obstacle in $\mathcal{R}^-(f_1)$ (blue balls inside the "red envelope") adds a positive contribution in a region $\mathcal{O}_j$ where $f_1(x, y, z) \leq 0$: in order to meet (13), it is required to properly compute $A_j > 0$ such that $f_1'(x, y, z) > 0$ in $\mathcal{O}_j$. Property P.2. states that, by iteratively considering additional obstacles, the value of $f_1'(x, y, z)$ in $\mathcal{O}_j$ can only increase: i.e., if (13) is verified in a region $\mathcal{O}_j$, it will be verified in the same region when considering additional obstacles.

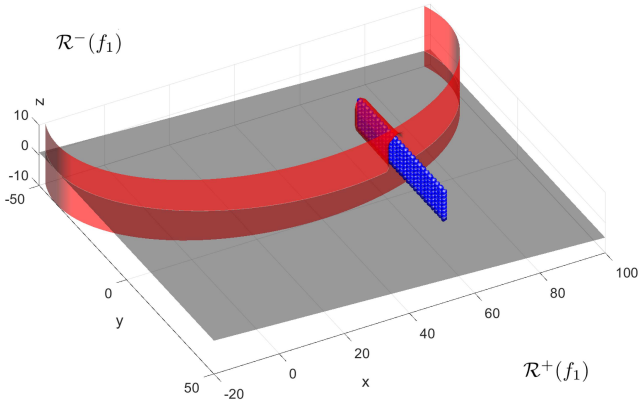

Figure 3. Intersection of a plane and a cylinder "deformed" by the presence of multiple obstacles $\mathcal{O}_j$ with radius $r_j = 1$.

Remark 10. *Obstacles and robots are modelled as having spherical symmetry. Figure 3 shows that, even if we do not explicitly consider the mutual influence of neighbouring obstacles, a visible effect on the shape of* $f_1'(x, y, z) = 0$ *is produced if spherical obstacles are sufficiently close to each others: the "red envelope" that encloses the "lattice of blue obstacles" ultimately produces an asymmetrical, elongated structure to be avoided (see also Remark 20). Extending the approach to asymmetrical robot shapes is not straightforward, and may be subject of future work.* □

The actual value of A_j can be computed by considering that the minima of $O_j(x, y, z)$ in $\mathcal{O}_j$ correspond to the boundary $\partial\mathcal{O}_j$ (7), i.e., the spherical surface with radius r_j centered in (x_j, y_j, z_j), see Remark 5:

$$\min_{(x,y,z)\in\mathcal{O}_j} O_j(x, y, z) = A_j \left(1 + \cos\left(\pi r_j/\sigma\right)\right). \tag{15}$$

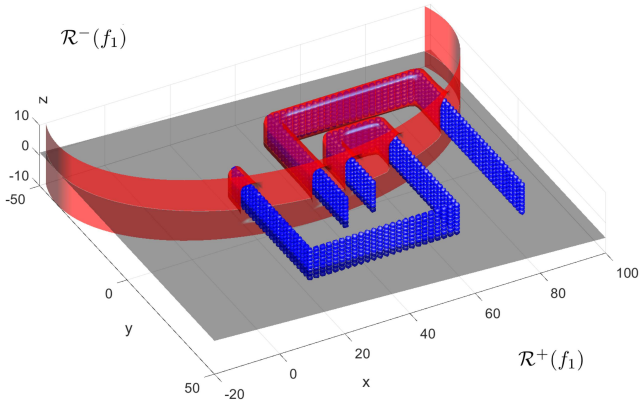

Figure 4. Intersection of a plane and a cylinder "deformed" by the presence of multiple obstacles $\mathcal{O}_j$ with radius $r_j = 1$.

Since we have assumed that the vehicle moves in a subspace of $\Re^3$ such that $\nabla f_1(x,y,z) \neq 0$, also the minima of $f_1(x,y,z)$ in $\mathcal{O}_j$ necessarily lie on the boundary $\partial\mathcal{O}_j$. Then, from (13) it must hold

$$\min_{(x,y,z)\in\partial\mathcal{O}_j} f_1(x,y,z) + A_j\left(1+\cos\left(\pi r_j/\sigma\right)\right) > 0, \quad (16)$$

and therefore

$$A_j > -\min_{(x,y,z)\in\partial\mathcal{O}_j} f_1(x,y,z)/\left(1+\cos\left(\pi r_j/\sigma\right)\right), \quad (17)$$

that allows one to find a value of A_j for any given σ.

Remark 11. *When $A_j > 0$, the condition (14) has not the same properties as (13). First, (14) can never be satisfied for those obstacles $\mathcal{O}_j$ lying in the semispace with $f_1(x_j,y_j,z_j) > 0$. Second, even when all obstacle are in the semispace with $f_1(x_j,y_j,z_j) < 0$ and (14) is verified for each individual obstacle, the same is no more guaranteed when considering more neighbouring obstacles, each adding a positive contribution to $f_1'(x_j,y_j,z_j)$. Symmetrically, if we make the a priori choice $A_j < 0$, it is convenient to focus on (14), which guarantees the properties P.1 and P.2 whereas (13) does not, thus finally requiring to satisfy a condition similar to (17), but with the opposite inequality. Whichever choice is made for the sign of A_j, properties P.1 and P.2 hold if and only if A_j has the same sign for all obstacles.* □

The minimum of $f_1(x,y,z)$ in $\partial\mathcal{O}_j$ can be found using the Lagrange multipliers: this corresponds to finding the two level surfaces $f_1(x,y,z) = w_\alpha$ and $f_1(x,y,z) = w_\beta$ which are tangent to $\mathcal{O}_j$, respectively, in $(x_\alpha,y_\alpha,z_\alpha)$ and $(x_\beta,y_\beta,z_\beta)$, and then taking the minimum between w_α and w_β. This computation is trivial whenever $f_1(x,y,z) = a_1x + b_1y + c_1z + d_1$ describes a plane. We start by defining the constraint corresponding to $\partial\mathcal{O}_j$:

$$g(x,y,z) = (x-x_j)^2 + (y-y_j)^2 + (z-z_j)^2 - r^2 = 0 \quad (18)$$

and then the Lagrange function

$$\Lambda = f_1(x,y,z) + \lambda g(x,y,z). \quad (19)$$

The solutions to the constrained minimization problem can be found by solving the system:

$$\begin{aligned}
\partial\Lambda/\partial x &= a_1 + 2\lambda(x-x_j) = 0\\
\partial\Lambda/\partial y &= b_1 + 2\lambda(y-y_j) = 0\\
\partial\Lambda/\partial z &= c_1 + 2\lambda(z-z_j) = 0\\
\partial\Lambda/\partial\lambda &= (x-x_j)^2 + (y-y_j)^2 + (z-z_j)^2 - r_j^2 = 0.
\end{aligned} \quad (20)$$

From the former three Equations it holds, for $\lambda \neq 0$,

$$\begin{aligned}
(x-x_j) &= -a_1/2\lambda\\
(y-y_j) &= -b_1/2\lambda\\
(z-z_j) &= -c_1/2\lambda
\end{aligned} \quad (21)$$

which can be substituted in the fourth Equation to be solved for λ as

$$\lambda_{\alpha,\beta} = \pm\|\nabla f_1\|/2r_j \quad (22)$$

and finally

$$\begin{aligned}
x_{\alpha,\beta} &= \mp a_1 r_j/\|\nabla f_1\| + x_j\\
y_{\alpha,\beta} &= \mp b_1 r_j/\|\nabla f_1\| + y_j\\
z_{\alpha,\beta} &= \mp c_1 r_j/\|\nabla f_1\| + z_j.
\end{aligned} \quad (23)$$

Equation (23) defines two points $(x_\alpha,y_\alpha,z_\alpha)$ and $(x_\beta,y_\beta,z_\beta)$ where the level surfaces of $f_1(x,y,z)$ are tangent to $g(x,y,z) = 0$. It holds:

$$\begin{aligned}
f_1(x_\alpha,y_\alpha,z_\alpha) &= -\|\nabla f_1\| r_j + a_1x_j + b_1y_j + c_1z_j + d_1\\
f_1(x_\beta,y_\beta,z_\beta) &= \|\nabla f_1\| r_j + a_1x_j + b_1y_j + c_1z_j + d_1,
\end{aligned} \quad (24)$$

from which it can be inferred that the minimum necessarily corresponds to $(x_\alpha,y_\alpha,z_\alpha)$.

In order for the path not to intersect $\mathcal{O}_j$ it must hold:

$$A_j > -f_1(x_\alpha,y_\alpha,z_\alpha)/\left(1+\cos\left(\pi r_j/\sigma\right)\right). \quad (25)$$

Among the possible choices, we can compute A_j such that (25) tends to be an equality, i.e., the deformed intersection is tangential to the spherical surface $\partial\mathcal{O}_j$. Since we have assumed $A_j > 0$, this finally yields:

$$A_j = \max\left(0, -f_1(x_\alpha, y_\alpha, z_\alpha)/\left(1+\cos\left(\pi r_j/\sigma\right)\right)\right). \quad (26)$$

The procedure above must be reiterated for all obstacles, thus guaranteeing that the deformed surface $f_1'(x,y,z)=0$ does not collide with any of them.

Remark 12. *Depending on the value of σ and A_j different paths are obtained: all of them guarantee that the constraint in (11) is met, but have different shapes. When σ is higher, the vehicle is influenced by obstacles at a greater distance, thus avoiding the obstacle along a lower curvature path.* □

In the case that $f_1(x,y,z)=0$ does not define a plane, the computation can be ideally made in the same way as above, using Lagrange multipliers: however solving the resulting system is not as computationally efficient as in the linear case (20). Since we may require to deal with a huge number of obstacles in real-time*, this solution cannot be pursued. Therefore, a different procedure is adopted, which requires only that $f_1(x,y,z)$ is locally convex (respectively, concave if we have assumed $A_j<0$) in $\mathcal{O}_j$. Notice that, according to properties P.1 and P.2, the local convexity (respectively, concavity) of $f_1(x,y,z)$ shall be guaranteed for each region $\mathcal{O}_j$ taken separately, without explicitly taking into account the presence of neighbouring obstacles, Remark 10.

Consider, for instance, a quadric defined as

$$f_1(x,y,z) = (x,y,z)Q(x,y,z)^T + P(x,y,z)^T + R = 0 \quad (27)$$

where $Q\in\Re^{3\times3}$, $P\in\Re^{1\times3}$, and R is a scalar. Please remark that the intersection of a quadric $f_1(x,y,z)=0$ with $f_2(x,y,z)=0$ can produce a wide class of different paths in $\Re^3$, which meet most requirements of robotics applications.

First of all, it is necessary to check if $f_1(x,y,z)$ is locally convex in $\mathcal{O}_j$. To this purpose, since $f_1(x,y,z)$ is twice differentiable, it is sufficient that its Hessian $\nabla^2 f_1(x,y,z)$

$$\nabla^2 f_1(x,y,z) = \begin{bmatrix} 2Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & 2Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & 2Q_{33} \end{bmatrix} \quad (28)$$

is positive semidefinite in the interior of $\mathcal{O}_j$.

The Hessian is positive definite on $\Re^3$ if, for instance, $f_1(x,y,z)=0$ defines an ellipsoid, e.g.,

$$f_1(x,y,z) = Q_{11}x^2 + Q_{22}y^2 + Q_{33}z^2 + R = 0 \quad (29)$$

with $Q_{11}, Q_{22}, Q_{33} > 0$ and $R<0$ (in this case it is aligned with the *North*, *East*, *Down* axes, as in Figures 1, 2, 3 4).

Instead of using Lagrange multipliers to find the minimum of $f_1(x,y,z)$ in $\partial\mathcal{O}_j$, and consequently compute a proper value for A_j through (25), we search for an approximated solution. Specifically, as long as $f_1(x,y,z)$ is convex in $\mathcal{O}_j$, it can be locally approximated by a linear function $\hat{f}_1(x,y,z) = a_1x + b_1y + c_1z + d_1$ tangent to $f_1(x,y,z)$ in (x_j,y_j,z_j), so that the following property holds:

$$\hat{f}_1(x,y,z) \le f_1(x,y,z),\ \forall (x,y,z)\in\mathcal{O}_j. \quad (30)$$

The parameters a_1, b_1, c_1, d_1 of $\hat{f}_1(x,y,z)$ are computed as:

$$\begin{aligned} a_1 &= \partial f(x_j,y_j,z_j)/\partial x \\ b_1 &= \partial f(x_j,y_j,z_j)/\partial y \\ c_1 &= \partial f(x_j,y_j,z_j)/\partial z \\ d_1 &= -a_1x_j - b_1y_j - c_1z_j + f(x_j,y_j,z_j) \end{aligned} \quad (31)$$

where the former three Equations impose that the gradients $\nabla f_1(x,y,z)$ and $\nabla\hat{f}_1(x,y,z)$ must be identical in (x_j,y_j,z_j), whereas the fourth Equation imposes that the two functions have the same value in (x_j,y_j,z_j).

After the linearization, the condition in (25) can be satisfied by finding the minimum point $(x_\alpha,y_\alpha,z_\alpha)$ of $\hat{f}_1(x,y,z)$ in $\mathcal{O}_j$ through the Lagrange multipliers (using the same procedure as above) and by finally computing

$$A_j = \max\left(0, -\hat{f}_1(x_\alpha,y_\alpha,z_\alpha)/\left(1+\cos\left(\pi r_j/\sigma\right)\right)\right), \quad (32)$$

choosing the equality as in (26). Notice in fact that, as long as (32) is satisfied, (25) is necessarily satisfied as well, given how $\hat{f}_1(x,y,z)$ has been chosen to satisfy (30),(31).

Figure 4 shows the deformed intersection $\mathcal{C}'$ resulting from a number of obstacles arranged as to produce the walls of a maze. Since $A_j>0$ in the example, $\mathcal{C}'$ lies completely in $\mathcal{R}^-(f_1)$, whereas the obstacles in the semi-space with $f_1(x,y,z)>0$ do not have any impact.

Remark 13. *If $f_1(x,y,z)$ is concave in $\Re^3$, two approaches can be adopted: i) using the function $-f_1(x,y,z)$, which describes the same surface as $f_1(x,y,z)=0$ but is convex in $\Re^3$; ii) using the concave function $f_1(x,y,z)$ by making the a priori choice $A_j<0$, which finally leads to conditions similar to (32) and (25), but with the opposite inequality.* □

*In a real implementation, the method to compute A_j may be reiterated for each individual point in a point cloud.

Shaping the Obstacle Functions - 2D case

The concepts above can be better visualized by referring to a 2D path espressed as $f_1(x,y)=0$. Let us define, for each obstacle, a circular neighborhood $\mathcal{O}_j$ where the radius r_j takes into account both the dimensions of the robot and the obstacle. The value of A_j must be chosen to satisfy the following condition (by assuming $A_j>0$ for all obstacles):

$$f_1(x,y)+\sum_{j=1}^{N} O_j(x,y)\neq 0, \forall (x,y)\in\mathcal{O}. \tag{33}$$

As in 3D, the condition above can be verified for each individual obstacle taken separately. Since the minima of $O_j(x,y)$ correspond to the boundary $\partial\mathcal{O}_j$, and the minima of $f_1(x,y)$ lie on $\partial\mathcal{O}_j$ as well, (33) yields:

$$A_j > -\min_{(x,y)\in\partial\mathcal{O}_j} f_1(x,y)/\left(1+\cos\left(\pi r_j/\sigma\right)\right). \tag{34}$$

By focusing on the general case that $f_1(x,y)$ is not linear, we start by checking if $f_1(x,y)$ is locally convex in $\mathcal{O}_j$ (respectively concave, in the case that $A_j<0$). If the answer is positive, $f_1(x,y)$ can be locally approximated by a plane $\hat{f}_1(x,y)$ tangent to $f_1(x,y)$ in (x_j,y_j), so that the property $\hat{f}_1(x,y)\leq f_1(x,y)$ necessarily holds in $\mathcal{O}_j$.

This concept is shown in Figure 5. In this particular case, the path $\mathcal{C}$ is a straight line corresponding to the level curve $w=0$ of a paraboloid $w=f_1(x,y)=ax^2+bx+c$: since y does not appear in the equation, the y-axis and the path are represented as being perpendicular to the page, and $\mathcal{C}$ consequently looks like a single point. Following the rationale of the previous Section, the convex paraboloid $w=f_1(x,y)$ can be approximated with a tangent plane $w=\hat{f}_1(x,y)$: as long as $\hat{f}_1(x,y)+O_j(x,y)>0$ in a neighborhood $\mathcal{O}_j$, $f_1(x,y)+O_j(x,y)>0$ as well.

Then, after finding the minimum (x_α,y_α) of $\hat{f}_1(x,y)$ on $\partial\mathcal{O}_j$ by using the Lagrange multipliers, we can choose A_j to satisfy the following condition:

$$A_j=\max\left(0,-\hat{f}_1(x_\alpha,y_\alpha)/\left(1+\cos\left(\pi r_j/\sigma\right)\right)\right). \tag{35}$$

4. Path following

Let the vector (u,v,w,p,q,r) describe the linear and angular velocities of the vehicle expressed in the b–frame: u, v, and w are the linear velocities along the x_b, y_b, and z_b axes of the b–frame; p, q, and r are the angular velocities describing rotations around x_b, y_b, and z_b. In the case of a differentially driven vehicle moving on the 2D plane, the vector describing linear and angular velocities needs to include the (u,r) components only. Also, assume that obstacles are detected by proper sensors and represented as spherical regions as in 10. For each obstacle $\mathcal{O}_j$ in (x_j,y_j,z_j) a radius r_j is defined, taking into account the robot and obstacles dimensions, safety margins, etc.

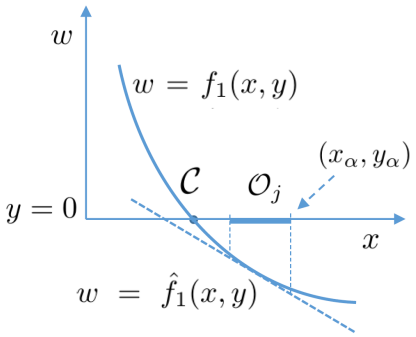

Figure 5. A straight path $\mathcal{C}$ in 2D produced as the intersection of a plane and a paraboloid in presence of an obstacle $\mathcal{O}_j$.

The following remarks are in order.

Remark 14. *Obstacles affect the path only at a finite distance determined by the parameter σ whereas farther obstacles can be ignored. The approach described in the previous Sections still works if the cloud of sensor readings acquired in run-time are used to build a local occupancy grid to reduce the impact of perceptual errors. In this perspective, only occupied grid cells at a maximum distance σ from the robot should be considered as an individual obstacle.* □

Remark 15. *Problems related to robot localization are not considered. Indeed, since localization returns an approximate estimate of the robot's pose, the vehicle moves on a path whose distance from the desired path depends also on localization accuracy. However, obstacle avoidance is guaranteed, also in the presence of inaccurate localization, by the fact that range measurements (and then the position of obstacles) are expressed relatively to the robot, and not with respect to an absolute reference frame (i.e., only local information is required for obstacle avoidance).* □

Since the control algorithms to guarantee convergence to the path defined in (1) are partially dependent on the robot kinematic and dynamic properties, we consider three case studies separately: a wheeled robot, an underwater robot, and a multicopter. Specifically, we propose controllers based on our previous results (Morro et al. 2011; Sgorbissa and Zaccaria 2010; Nguyen et al. 2016, 2017), and therefore we will not report an analysis on the effect of control parameters: the reader can refer to our previous work. Moreover, even if we explore only kinematics control, some of the simulated

experiments in Section 6 are based on Simulink models taking into account dynamic effects as well.

Wheeled robots

Consider a differentially driven vehicle whose kinematics can be modelled as a unicycle, with the b-frame origin in the midpoint between the two driving wheels, and the x-axis pointing along the direction of motion. The system is governed by the kinematic equations

$$\begin{aligned} \dot{x} &= u\cos\psi \\ \dot{y} &= u\sin\psi \\ \dot{\psi} &= r. \end{aligned} \tag{36}$$

(Morro et al. 2011) presented a method that provably achieves convergence to a curve $f_1(x,y)=0$ by setting the forward velocity u and the steering velocity r as follows:

$$\begin{aligned} &u = u(t), \quad u(t) > 0, \\ &r = K_1(-\|\nabla f_1\| u S(f_1) - \dot{f}_1) + \dot{\psi}_c, \end{aligned} \tag{37}$$

where

- $u(t)$ is a positive velocity profile;
- K_1 is a gain;
- $\dot{f}_1 = \dot{f}_1(x,y,\psi) = f_{1x}u\cos\psi - f_{1y}u\sin\psi$ describes how f_1 varies with time, i.e., it is a measure of how fast the vehicle is getting closer to / farther from the path*;
- $S(f_1)$ is the C^n sigmoid function $S(f_1) = K_2 f_1/\sqrt{1+{f_1}^2}$; K_2, with $0 < K_2 \leq 1$, determines the shape of the sigmoid;
- $\psi_c = \arg(f_{1y} - i f_{1x})$ is the orientation of the vector $(f_{1y}, -f_{1x})$ normal to ∇f_1 in (x,y), i.e., tangent to the level curve, and $\dot{\psi}_c$ is its time derivative that takes into account the curvature of the path.

The control law in (37) can be intuitively interpreted as follows. If the vehicle is in (x,y,ψ) and it is moving along a level curve $w = f_1(x,y)$ with $w > 0$, it holds $\dot{f}_1 = 0$ and $\dot{\psi} = \dot{\psi}_c$: in this case, the controller sets $\dot{\psi} = \dot{\psi}_c - K_1 \|\nabla f_1\| u S(w)$, and the vehicle approaches the path by leaving the level curve with $w > 0$ on its left side. This follows the fact that $\dot{\psi} < \dot{\psi}_c$ since the second term is negative, i.e., $\dot{\psi}$ is set to a lower value than required to move on the level curve. Symmetrically, when the vehicle is moving along a level curve with $w < 0$, the controller sets $\dot{\psi} = \dot{\psi}_c + K_1 \|\nabla f_1\| u S(w)$ since $S(-w) = S(w)$, and the vehicle tends to the path by leaving the level curve on its right side as $\dot{\psi} > \dot{\psi}_c$.

Remark 16. *In accordance with the considerations above, a vehicle moving along the path $f_1(x,y)=0$ with the control law in (37) keeps the region $\mathcal{R}^-(f_1)$ on its right. Then, it is possible to invert its direction of motion by inverting the sign of $f_1(x,y)$, since $\mathcal{R}^-(-f_1) = \mathcal{R}^+(f_1)$ (Remark 6).* □

In presence of obstacles, the control law in (37) can be straightfowardly used to follow the deformed path $f_1'(x,y)=0$. Then, the control input r at each step can be computed as a closed-form expression (37)(8) that is a continuous function of f_1, its first and second order derivatives, the robot's pose, as well as the distance from all locally sensed obstacles.

Underwater robots

Kinematics equations must be properly reformulated for 3D path tracking. Assume that the b-frame origin is located in the center of mass of the vehicle, and that it is possible to control only the linear velocity u of the vehicle along the x_b-axis of the b–frame (using rear thrusts), as well as its rotational velocities q and r about the y_b- and the z_b-axis. This choice is very common in AUVs with glider capability to guarantee energetic efficiency: see for instance (Alvarez et al. 2009) where u is obtained through propulsion jet-pumps at the vehicle stern, r is obtained through two power jet-pumps at the vehicle bow, and the diving attitude q is controlled through the internal displacement of the battery pack. Also, assume that the AUV is stabilized in roll by a separate mechanism†, which guarantees that $\dot{\phi} = \phi = 0$.

Kinematics equations can be written as follows:

$$\begin{aligned} \dot{x} &= u\cos\psi\cos\theta \\ \dot{y} &= u\sin\psi\cos\theta \\ \dot{z} &= -u\sin\theta \\ \dot{\phi} &= 0 \\ \dot{\theta} &= q \\ \dot{\psi} &= r\tfrac{1}{\cos\theta} \\ & \qquad \theta \neq \pm\tfrac{\pi}{2}. \end{aligned} \tag{38}$$

(Sgorbissa and Zaccaria 2010) have shown that the system (38) can be driven to converge to the curve defined by two intersecting surfaces $f_1(x,y,z)=0$ and $f_2(x,y,z)=0$ by setting control inputs u, q and r as follows:

*The absolute value of the velocity $|u|$ can be used instead of u, which guarantees convergence to the path even when moving backward. Here the analysis is limited to positive values for sake of simplicity.

†The Fòlaga underwater vehicle, used as a reference in this work, implements this strategy: see the experimental Section 6.

$$
\begin{aligned}
&u = u(t), \quad \lim_{t\to\infty} u(t) > 0 \\
&r = K_{11}(-\|\nabla \hat{D}_1\| u \cos^2\theta S(f_1) - \cos\theta \dot{f}_1) + \dot{\psi}_c \\
&q = K_{21}(-\|\nabla \hat{D}_2\| u S(f_2) - \dot{f}_2) + \dot{\theta}_c \\
&\qquad\qquad \|\nabla f_1\| \neq 0, f_{2z} \neq 0
\end{aligned} \tag{39}
$$

where $\|\nabla f_1\| \neq 0$ and $f_{2z} \neq 0$ necessarily hold under the conditions A.1, A.2 in Section 2, and the following quantities are introduced for ease of notation:

- $\hat{D}_{1\hat{x}} = (f_{1x} f_{2z} - f_{1z} f_{2x})/f_{2z}$
- $\hat{D}_{1\hat{y}} = (f_{1y} f_{2z} - f_{1z} f_{2y})/f_{2z}$;
- $\hat{D}_{2\hat{x}} = f_{2x}\cos\psi + f_{2y}\sin\psi$;
- $\hat{D}_{2\hat{y}} = -f_{2z}$;
- $\|\nabla \hat{D}_i\| = (\hat{D}_{i\hat{x}}^2 + \hat{D}_{i\hat{y}}^2)^{1/2}$, $i = 1, 2$;

and

- $u(t)$ is a positive velocity profile*;
- $K_{i1} > 0$, $i = 1, 2$ are two gains which weight the errors relative to the two surfaces;
- $S(f_i)$ is the C^n sigmoid function $S(f_i) = K_{i2} f_i / \sqrt{1 + f_i^2}$; K_{i2}, with $0 < K_{i2} \leq 1$, determines the shape of the sigmoid;
- $\psi_c = \arg(\hat{D}_{1\hat{y}} - i\hat{D}_{1\hat{x}})$, and $\dot{\psi}_c$ is its time derivative;
- $\theta_c = \arg(\hat{D}_{2\hat{y}} - i\hat{D}_{2\hat{x}})$, and $\dot{\theta}_c$ is its time derivative.

The equations in (39) are very similar to (37). The basic idea is to decouple 3D path tracking into two disjoint problems. In particular, by controlling r in order to converge to the surface $f_1(x, y, z) = 0$ and, at the same time, controlling q in order to converge to the surface $f_2(x, y, z) = 0$, the system is expected to converge to the desired path, i.e., expressed as the intersection of the two surfaces.

Please notice that, similarly to the 2D case, the controller sets the angular velocity $q < \dot{\theta}_c$ (respectively, $q > \dot{\theta}_c$) when the vehicle is moving tangentially to the surface level $w = f_2(x, y, z)$ with $w > 0$ (respectively, $w < 0$). Also, as long as we keep θ far from singularities as stated in (38), it holds $\cos\theta > 0$: the controller sets the angular velocity $r < \dot{\psi}_c$ (respectively, $r > \dot{\psi}_c$) when the vehicle is moving tangentially to the surface level $w = f_1(x, y, z)$ with $w > 0$ (respectively, $w < 0$).

Remark 17. *A vehicle moving along the path with the control law in (39) keeps the regions $\mathcal{R}^-(f_1)$ and $\mathcal{R}^-(f_2)$ on its right. Then, it is possible to invert its direction of motion by inverting the sign of either $f_1(x, y, z)$ or $f_2(x, y, z)$, since $\mathcal{R}^-(-f_1) = \mathcal{R}^+(f_1)$ and $\mathcal{R}^-(-f_2) = \mathcal{R}^+(f_2)$ (Remark 6).* □

Remark 18. *By setting $f_2(x, y, z) = z = 0$ and assuming that it constantly holds $\theta = 0$, the control law in (39) reduces to the one proposed for wheeled robots.* □

In presence of obstacles the control law in (39) can be straightforwardly used to follow the deformed path determined by the intersection of $f_1'(x, y, z) = 0$ and $f_2(x, y, z) = 0$. Then, each control input q and r can be computed at each step as a closed-form expressions (39)(6) that is a continuous function of f_1 and f_2, their first and second order derivatives, the robot's pose, as well as the distance from all locally sensed obstacles.

Multicopters

Consider the rotation matrix ${}^w R_b$ describing the orientation of the b-frame located in the center of mass of the vehicle with respect to the n-frame, whose elements are a function of the pitch, roll and yaw angles ϕ, θ, ψ†:

$$
{}^w R_b = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi + s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}. \tag{40}
$$

The traslational dynamics of the vehicle in the world frame can be described through Newton's equations as

$$
m(\ddot{x}, \ddot{y}, \ddot{z})^T = (0, 0, mg)^T - {}^w R_b (0, 0, T_1)^T, \tag{41}
$$

where $\ddot{x}, \ddot{y}, \ddot{z}$ describe the acceleration of the origin of the b-frame in the n-frame, g is the gravitational acceleration, m is the vehicle's mass, and T_1 is the overall force produced by propellers along the z_b axis of the b-frame (also referred to as thrust). The angular dynamics can be described through the Euler's equations as

$$
\dot{\omega} = \mathcal{I}^{-1}[-\omega \times \mathcal{I}\omega + [T_2\ T_3\ T_4]^T], \tag{42}
$$

where $\mathcal{I}$ is the moment of inertia matrix referenced to the center of mass, $\omega = p x_b + q y_b + r z_b$ denotes the angular velocity of the vehicle with x_b, y_b, x_b the unit vectors oriented along the axes of the b-frame, and finally T_2, T_3, T_4 are the three torques about the x_b, y_b and z_b axes.

*As in the 2D case, we limit our analysis to positive velocity profiles for sake of simplicity.

†$c\alpha$ and $s\alpha$ are the abbreviations for $\cos(\alpha)$ and $\sin(\alpha)$, respectively

According to the Newton-Euler equations, it is possible to achieve a non-null acceleration along the North, East, Down directions of the *n*-frame by properly operating on the overall thrust T_1 and the roll, pitch and yaw angle (ϕ, θ, ψ) (41), which can be controlled by operating on the three torques T_2, T_3, T_4 (42)*.

In this situation, as well as in all situations in which it is possible to control the vehicle's accelerations along the three axis of the n-frame, the approach proposed here is the following (Nguyen et al. 2016, 2017).

First, we compute the desired heading vector expressed in the *n*-frame, to be used as a reference:

$$h = -w_1 \frac{f_1 \nabla f_1}{\|\nabla f_1\|} - w_2 \frac{f_2 \nabla f_2}{\|\nabla f_2\|} + w_3 \frac{\nabla f_1 \times \nabla f_2}{\|\nabla f_1 \times \nabla f_2\|} \quad (43)$$

where w_i, $i = 1, 2, 3$, weight the relative importance of the three components, i.e., the weights w_1, w_2, and w_3 determine if the vehicle should give more importance to either approaching $f_1(x, y, z) = 0$, $f_2(x, y, z) = 0$, or moving along the surface intersection.

Second, h is properly scaled depending on the desired velocity $u(t)$ at time t, and projected onto the *b*-frame:

$$h_b = {}^b R_w h \frac{u(t)}{\|h\|}. \quad (44)$$

The velocity vector h_b at each step can be computed as a closed-form expression (44)(43)(6) that is a continuous function of f_1 and f_2, their first and second order derivatives, the robot's pose, as well as the distance from all locally sensed obstacles.

Third, a cascaded control can then be adopted to compute the thrust and the three torques (and, ultimately, the speeds of the propellers) to achieve the desired velocity h_b (e.g., based on feedback linearization as in (Nguyen et al. 2016) or a standard PID-based control).

Remark 19. *A vehicle moving along the path keeps the regions $\mathcal{R}^-(f_1)$ and $\mathcal{R}^-(f_2)$ on its right: its direction of motion is determined by the vector $\nabla f_1 \times \nabla f_2$ in (43), and can be inverted by inverting the sign of either $f_1(x, y, z)$ or $f_2(x, y, z)$ (as in Remark 17).* □

5. Path Planning

This Section discusses the maze-solving capabilities of the approach: that is, under which conditions the deformed intersection (6) includes a path from a start to a goal position not colliding with obstacles (11), given that an obstacle-free path exists lying on the undeformed surface $f_2(x, y, z) = 0$.

To this end, the first Subsection determines the conditions (Theorem 2) such that the proposed method is able to find a path through an arbitrarily complex maze, without being trapped into deadlocks. If the conditions above do not hold, the second Subsection provides additional rules such that the method behaves in a similar way as the BUG2 algorithm (Lumelsky and Stepanov 1987), and provides general conditions for goal reachability (Theorem 3).

Intrinsic Maze-solving Capabilities

In order to prove the maze-solving capabilities of the approach, let†:

- $\mathcal{R} = \mathcal{R}^-(f_1) \cup \mathcal{R}^+(f_1) = \{(x, y, z) | f_2(x, y, z) = 0)$ be the undeformed surface defined by $f_2(x, y, z) = 0$;
- $\mathcal{C} = \bigcup_{j=1}^{C} \mathcal{C}_j$ be the intersection produced in (1);
- $\mathcal{S}_j = \{(x, y, z) | O_j(x, y, z) \geq 0\} \cap \mathcal{R}$ be the area of influence of $\mathcal{O}_j$ on the surface $\mathcal{R}$;
- $\mathcal{S}_j^-$ and $\mathcal{S}_j^+$ be the intersections of $\mathcal{S}_j$ with $\mathcal{R}^-(f_1)$ and $\mathcal{R}^+(f_1)$, respectively;
- $\mathcal{S} = \bigcup_{j=1}^{N} \mathcal{S}_j$ be the overall area of influence of obstacles;
- $\mathcal{C}' = \bigcup_{j=1}^{C} \mathcal{C}'_j$ be the deformed intersection, i.e., the set of curves generated in (6).
- $\{start, goal\}$ be a couple of points belonging to a curve $\mathcal{C}_j \subseteq \mathcal{C}$;
- $t = (\nabla f_1 \times \nabla f_2)/\|\nabla f_1 \times \nabla f_2\|$ be a unit vector tangent to $\mathcal{C}$, with sign properly chosen such that it is possible to move on $\mathcal{C}_j$ from $start$ to $goal$ along t, henceforth referred to as the direction of the curve (Remarks 16, 17, 19);
- $\mathcal{L}_j$, also referred to as "islands", be the union of regions $\mathcal{S}_i$ that are in a given topological relation with each other. Specifically, each region $\mathcal{S}_i$ belongs to an island; $\mathcal{S}_i$ and $\mathcal{S}_j$ belong to the same island if $\mathcal{S}_i \cap$

*More precisely, the angular accelerations produced by T_2, T_3, T_4 can be integrated to obtain the three velocities p, q, r expressed in the *b*-frame (41), whereas the relation with $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$ expressed in the *n*-frame is more complex. As long as the roll and pitch angles are small enough, angular velocities expressed in the *n*-frame and the *b*-frame tend to coincide.

†All new quantities introduced here, including "islands" of obstacles, do not need to be computed for obstacle avoidance. Everything which is needed has already been introduced in the previous Sections: their definition is only required to discuss the properties of the approach.

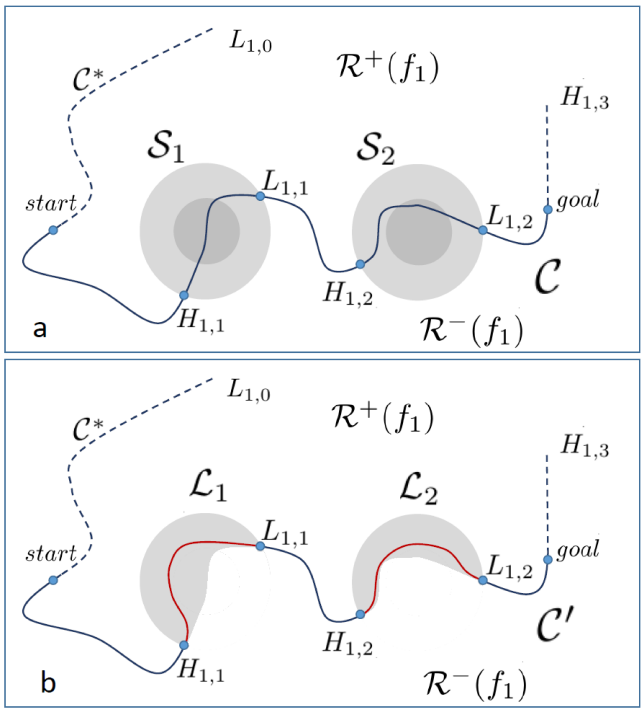

Figure 6. a) The curve $\mathcal{C}$ (before deformation) with $start$ and $goal$, hit and leave points; b) the curve $\mathcal{C}'$ (after deformation) lying in $\mathcal{R}^+(f_1)$: two "islands" $\mathcal{L}_1$ and $\mathcal{L}_2$ in $\mathcal{R}^+(f_1)$ are visible.

$\mathcal{S}_l \neq \emptyset$ *. When the sign of A_j is set a priori for all $\mathcal{O}_j$, and then system avoids obstacles only in $\mathcal{R}^-(f_1)$ (respectively, $\mathcal{R}^+(f_1)$), $\mathcal{L}_j$ are defined by considering only subregions $\mathcal{S}_i^-$ (respectively, $\mathcal{S}_i^+$).

Some of these concept are shown in Figure 6a and b, by assuming that $f_2(x, y, z) = 0$ is a plane for ease of representation: in this case only one curve $\mathcal{C}_1 = \mathcal{C}$ is produced, intersecting the areas of influence $\mathcal{S}_1$ and $\mathcal{S}_2$ of two obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$. The deformed intersection $\mathcal{C}'$ lies in $\mathcal{R}^+(f_1)$: two islands $\mathcal{L}_1$ and $\mathcal{L}_2$ are visible.

In general, each curve $\mathcal{C}_j \subseteq \mathcal{C}$ is either a closed or an infinite curve (Remark 2), and then it necessarily intersects the boundary of $\mathcal{S}$ an even number of times $2I_j$, where I_j depends on the relative position of obstacles with respect to $\mathcal{C}_j$†. By referring to the direction of the curve t and using a terminology that is common in local path planning algorithms (Lumelsky and Stepanov 1987), let:

- $H_{j,i}$ be the i^{th} hit point of C_j, i.e., the $(2i-1)^{th}$ intersection of C_j with the boundary of $\mathcal{S}$;
- $L_{j,i}$ be the i^{th} leave point of C_j, i.e., the $2i^{th}$ intersection of C_j with the boundary of $\mathcal{S}$.

If $\mathcal{C}_j$ is a closed curve, the first hit point $H_{j,1}$ is arbitrarily chosen. Also, since all hit / leave points are periodically encountered while looping along a closed curve, we assume that L_{j,I_j+i} and H_{j,I_j+i} identify the same points as $L_{j,i}$ and $H_{j,i}$, respectively. If, on the opposite, $\mathcal{C}_j$ is an open curve with both endpoints at infinity, its first and second endpoints are referred to as $L_{j,0}$ and $H_{j,I+1}$.

Each curve $\mathcal{C}_j$ is ideally composed of:

- I_j segments with endpoints $H_{j,i}$, $L_{j,i}$ that lie completely in $\mathcal{S}$: these segments are indicated with the notation $\mathcal{C}(H_{j,i}, L_{j,i})$;
- I_j or $I_j + 1$ segments (depending on whether $\mathcal{C}_j$ is a closed or an infinite curve) with endpoints $L_{j,i}$, $H_{j,i+1}$ that do not intersect the interior of $\mathcal{S}$: these segments are indicated with the notation $\mathcal{C}(L_{j,i}, H_{j,i+1})$.

In Figure 6a, the curve $\mathcal{C}_1$ intersects $\mathcal{S}_1$ and $\mathcal{S}_2$ and produces 5 segments ($I_1 = 2$ and $\mathcal{C}_1$ is an infinite curve). The first segment $\mathcal{C}(L_{1,0}, H_{1,1})$ connects the point at infinity $L_{1,0}$ to the hit point $H_{1,1}$, and it is followed by $\mathcal{C}(H_{1,1}, L_{1,1})$, $\mathcal{C}(L_{1,1}H_{1,2})$, $\mathcal{C}(H_{1,2}, L_{1,2})$, and finally by $\mathcal{C}(L_{1,2}, H_{1,3})$, with $H_{1,3}$ a point at infinity‡.

Theorem 1. *Given the intersection $\mathcal{C}$ and the deformed intersection $\mathcal{C}'$. Given that the intersection of each curve $\mathcal{C}_j \subseteq \mathcal{C}$ with the boundary of $\mathcal{S}$ produces a set of leave points $L_{j,i}$ and hit points $H_{j,l}$.*

Then it holds that:

(i) the intersection of $\mathcal{C}'$ with the boundary of $\mathcal{S}$ produces the same hit and leave points as C;

(ii) for every segment $\mathcal{C}(L_{j,i}, H_{j,i+1}) \subseteq \mathcal{C}$ not intersecting the interior of $\mathcal{S}$, a segment $\mathcal{C}'(L_{j,i}, H_{j,i+1}) \subseteq \mathcal{C}'$ exists completely overlapping with $\mathcal{C}(L_{j,i}, H_{j,i+1})$;

(iii) for every segment $\mathcal{C}(H_{j,i}, L_{j,i}) \subseteq \mathcal{C}$ lying inside an island $\mathcal{L}_l$, a segment $\mathcal{C}'(H_{j,i}, L_{k,l}) \subseteq \mathcal{C}'$ exists that lies completely in $\mathcal{L}_l \setminus \mathcal{O}$.§

Proof. Statements i) and ii) directly follow the fact that the contribution of obstacles $O_j(x, y, z)$, $i = 1, \ldots, N$, is null on the boundary and in the exterior of $\mathcal{S}$, and therefore hit and leave points, as well as segments $\mathcal{C}(L_{j,i}, H_{j,i+1})$ not intersecting the interior of $\mathcal{S}$, are unaffected by obstacles.

*The relation has obvious transitive properties: if $\mathcal{S}_i \cap \mathcal{S}_l \neq \emptyset$ and $\mathcal{S}_l \cap \mathcal{S}_k \cap \neq \emptyset$, $\mathcal{S}_i$ and $\mathcal{S}_k$ necessarily belong to the same island even if $\mathcal{S}_i \cap \mathcal{S}_k = \emptyset$. Islands ideally model "chains" of obstacles that are within perceptual range from each others.

†If $\mathcal{C}_j$ is tangent to $\mathcal{S}$, then it intersect its boundary in two coincident points.

‡In principle, a segment $\mathcal{C}(H_i, L_i)$ can collapse to a point, e.g., if $\mathcal{C}$ is tangent to $\mathcal{S}$.

§That is, $\mathcal{C}'(H_{j,i}, L_{k,l})$ does not intersect any obstacle; the first endpoint $H_{j,i}$ of both segments is identical, but the second endpoint $L_{k,l}$ of the latter can be different from the second endpoint $L_{j,i}$ of the former.

To proof Statement iii), consider once again a generic segment $\mathcal{C}'(L_{j,i-1}, H_{j,i}) = \mathcal{C}(L_{j,i-1}, H_{j,i})$ belonging to $\mathcal{C}_j$ and not intersecting the interior of $\mathcal{S}$. By moving from the first endpoint $L_{j,i-1}$ along the direction t, the second endpoint $H_{j,i}$ lying on the boundary of an island is met, say $\mathcal{L}_l$. Since $\mathcal{C}'$ is a simple closed or infinite curve (Remark 8), $\mathcal{C}'(L_{j,i-1}, H_{j,i})$ must necessarily be connected to a second segment, whose first endpoint is $H_{j,i}$. Even if we do not know its exact shape, this second segment cannot have its second endpoint in the interior of $\mathcal{L}_l$, once again as a consequence of the facts that $\mathcal{C}'$ is a simple closed or infinite curve. Then, its second endpoint necessary lies on the boundary of $\mathcal{L}_l$, and it must correspond to the first endpoint $L_{k,l}$ of a segments $\mathcal{C}'(L_{k,l}, H_{k,l+1}) = \mathcal{C}(L_{k,l}, H_{k,l+1})$, possibly belonging to a different curve $\mathcal{C}_k$, $k \neq j$.

The resulting segment $\mathcal{C}'(H_{j,i}L_{k,l})$ lies completely in $\mathcal{L}_l$ and it is guaranteed not to intersect the obstacle region $\mathcal{O}$ according to (11), i.e., it completey lies in $\mathcal{L}_l \setminus \mathcal{O}$. □

Remark 20. *Given two obstacles $\mathcal{O}_j$ and $\mathcal{O}_l$ and their areas of influence $\mathcal{S}_j$ and $\mathcal{S}_l$ on $\mathcal{R}$, from Theorem 1 it follows that: (i) if $\mathcal{O}_j \cap \mathcal{O}_l \cap \mathcal{R} \neq \emptyset$, the deformed intersection cannot not pass through the obstacles on $\mathcal{R}$, since this would generate a collision – which is prevented by (11); (ii) if $\mathcal{S}_j \cap \mathcal{S}_l = \emptyset$, the deformed intersection can pass through the two areas of influence; (iii) if $\mathcal{O}_j \cap \mathcal{O}_l \cap \mathcal{R} = \emptyset$ and $\mathcal{S}_j \cap \mathcal{S}_l \neq \emptyset$, knowing in advance if the deformed intersection passes through them or not is not straightforward – as this is not prevented from (11).* □

Now let

- $\mathcal{C}^p \subseteq \mathcal{C}$ be a continuous segment of $\mathcal{C}$ whose endpoints are $start$ and $goal$, henceforth referred to as the "nominal path";
- $\mathcal{C}^* = \mathcal{C} \setminus \mathcal{C}^p$ be the set of points that belong to $\mathcal{C}$ but are not part of the nominal path $\mathcal{C}^p$ (Figure 6).

The following Theorem demonstrates that, under proper conditions, if a nominal path $\mathcal{C}^p$ exists leading from $start$ to $goal$ in absence of obstacles, then a path $\mathcal{C}'^p \subset \mathcal{C}'$ exists leading from $start$ to $goal$ in presence of obstacles, henceforth referred to as the "deformed path".

Theorem 2. *Given the intersection $\mathcal{C}$, a nominal path $\mathcal{C}^p \subseteq \mathcal{C}$ with endpoints start and goal, and the deformed intersection $\mathcal{C}'$. Given a finite set of obstacles $\mathcal{O}_j$ whose positions (x_j, y_j, z_j) can assume arbitrary values in $\Re^3$, subject to the following constraints holding for each island $\mathcal{L}_l$, $l = 1, \ldots, L$:*

- $Int(\mathcal{S}) \cap \{start, goal\} = \emptyset$
- $\mathcal{L}_l \cap \mathcal{C}^p \neq \emptyset \implies \mathcal{L}_l \cap \mathcal{C}^* = \emptyset$

Then, it holds that:

(i) $\{start, goal\} \in \mathcal{C}'$;

(ii) a continuous segment $\mathcal{C}'^p \subset \mathcal{C}'$ with endpoints start and goal necessarily exists.

Proof. Statement i) directly follows from the first constraint: since $\{start, goal\}$ cannot be in the interior of $\mathcal{S}$, they are not affected by the presence of obstacles, and therefore necessarily belong to $\mathcal{C}'$ as well.

Concerning statement ii), if $\mathcal{L}_l \cap C^p = \emptyset$, $l = 1, \ldots, L$ (i.e., the nominal path does not intersect any island and the second constraints is necessarily verified), then $\mathcal{C}'^p = \mathcal{C}^p$ is not affected by obstacles and the demonstration is concluded. Otherwise, statement ii) can be demonstrated according to the following rationale.

Let the nominal path $\mathcal{C}^p$ be part of a curve $\mathcal{C}_j \subseteq \mathcal{C}$. Let $H_{j,i}$ be the first hit point encountered after $start$ when moving in $\mathcal{C}_j$ along t, and $L_{j,k}$ the last leave point encountered before $goal$. According to Theorem 1, it holds that $\mathcal{C}'(start, H_{j,i}) = \mathcal{C}(start, H_{j,i})$ and $\mathcal{C}'(L_{j,k}, goal) = \mathcal{C}(L_{j,k}, goal)$.

Now consider the segment $\mathcal{C}'(start, H_{j,i}) = \mathcal{C}(start, H_{j,i})$, with $H_{j,i}$ on the boundary of an island $\mathcal{L}_l$ such that $\mathcal{L}_l \cap \mathcal{C}^p \neq 0$: the segment is necessarily followed by a segment $\mathcal{C}'(H_{j,i}, L_{j,l})$ that lies completely in $\mathcal{L}_l$, and the leave point belongs to the nominal path since the island $\mathcal{L}_l$ cannot intersect $\mathcal{C}$ outside the nominal path ($\mathcal{L}_l \cap \mathcal{C}^* = 0$ according to the second constraint). That is, the second endpoint $L_{j,l}$ is necessarily located somewhere in $\mathcal{C}^p$, and this result can iterated for subsequent segments encountered while moving in $\mathcal{C}'$ along t: all of these segments have both endpoints in $\mathcal{C}^p$. Since the number of leave points $L_{j,l} \in \mathcal{C}^p$ is necessarily finite and the same leave point cannot be encountered twice because $\mathcal{C}'$ is a simple curve without self intersections, it necessarily holds that the segment $\mathcal{C}'(L_{j,k}, goal)$ is finally considered in this process. □

In general, leave / hit points are not necessarily encountered in the same order along $\mathcal{C}$ and $\mathcal{C}'$. Consider Figure 7a, in which the nominal path $\mathcal{C}_p$ is the same as Figure 6ab, but the obstacle are different and are avoided in $\mathcal{R}^-(f_1)$. A closed curve is produced in the area surrounded by obstacles, i.e., $L_{1,1}$ leads to $H_{1,2}$ along $\mathcal{C}'(L_{1,1}, H_{1,2})$ (which is unaffected by obstacles), and $H_{1,2}$ leads back to $L_{1,1}$ along a segment that entirely lies in $\mathcal{L}_1$. However,

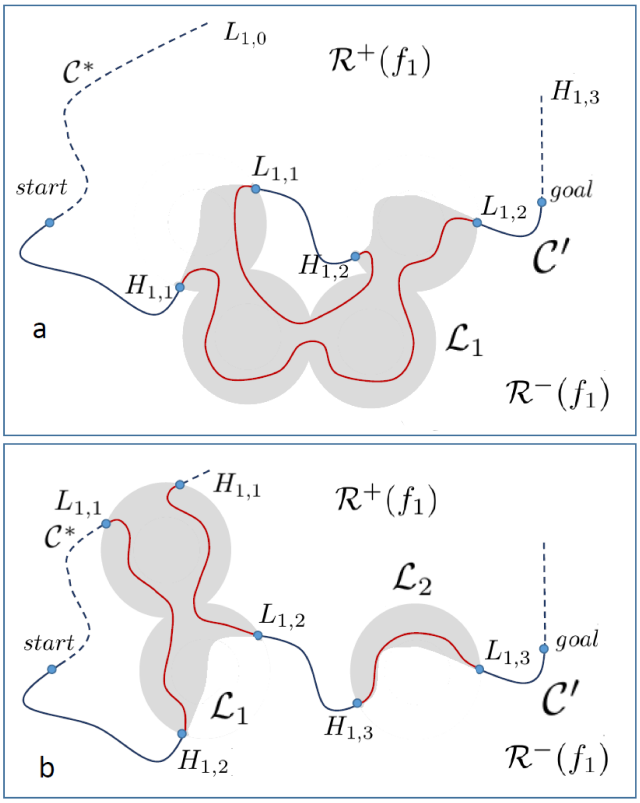

Figure 7. a) The curve $\mathcal{C}'$ (after deformation) lying in $\mathcal{R}^-(f_1)$: a single island $\mathcal{L}_1$ in $\mathcal{R}^-(f_1)$ is visible; b) the curve $\mathcal{C}'$ lying in $\mathcal{R}^+(f_1)$: two islands $\mathcal{L}_1$ and $\mathcal{L}_2$ in $\mathcal{R}^+(f_1)$ are visible.

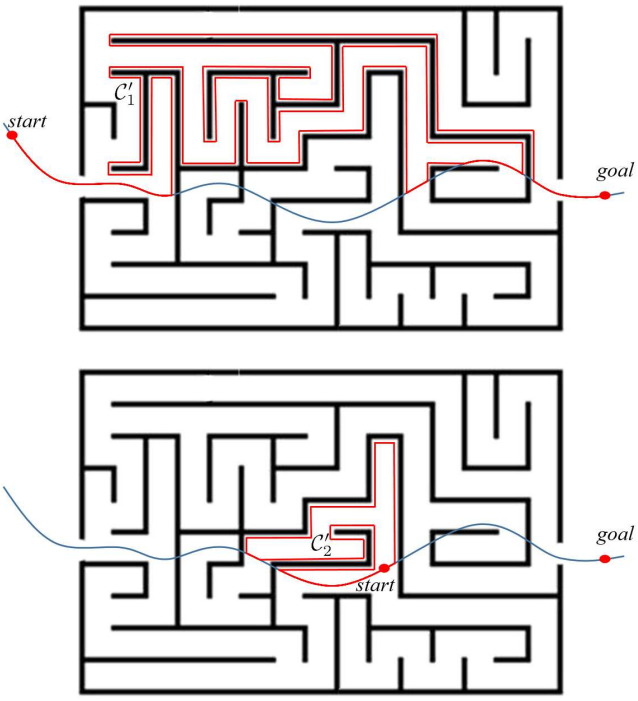

Figure 8. a) The conditions of Theorem 2 hold, and hence a path from *start* to *goal* is found. b) The conditions of Theorem 2 do not hold.

by ideally removing the loop (which can be never reached from $start$), the number of remaining segments is finite, and therefore a path leading from $start$ to $goal$ is finally found (i.e., from $start$ to $H_{1,1}$ along $\mathcal{C}'(start, H_{1,1})$, from $H_{1,1}$ to $L_{1,2}$ along a segment of unknown shape, from $L_{1,2}$ to $goal$ along $C'(L_{1,2}, goal)$.

The second costraint in Theorem 2 is worth an additional explanation. It requires that, if an island $\mathcal{L}_l$ intersects the nominal path $\mathcal{C}^p$, then the same island cannot intersect $\mathcal{C}$ outside $\mathcal{C}^p$: given that $A_j > 0$ has been chosen such that the robot avoids obstacles in $\mathcal{R}^-(f_1)$, this constraint is necessary to prevent that $start$ or $goal$ are surrounded by regions in $\mathcal{R}^-(f_1)$ that may be untraversable, Remark 20 (the same rationale holds if $A_j < 0$). Figure 7b shows a different situations in which the second costraint in Theorem 2 does not hold: in this case it exists an island $\mathcal{L}_1$ such that $\mathcal{L}_1 \cap \mathcal{C}^p \neq \emptyset$ and $\mathcal{L}_1 \cap \mathcal{C}^* \neq \emptyset$, i.e., it intersect $\mathcal{C}$ both in the nominal path and outside it. The vehicle avoids obstacles in $\mathcal{R}^+(f_1)$: since the continuity of $\mathcal{C}'$ is preserved, the deformed path will be constituted by the segment $C'(start, H_{1,2})$ from $start$ to $H_{1,2}$, followed by a segment $C'(H_{1,2}, L_{1,1})$ of unknown shape that lies entirely in $\mathcal{L}_1$, followed by another segment that belongs to C^* and finally leads back to $start$. That is, as long as $\mathcal{L}_1$ intersects both $\mathcal{C}^*$ and $\mathcal{C}^p$, a loop can be produced which never reaches $goal$. Notice that, in this case, a path from $start$ to $goal$ exists if the robot is allowed to move in $\mathcal{R}^-(f_1)$ instead of $\mathcal{R}^+(f_1)$, but such path cannot be found as the sign of A_j is a priori determined: the next Section proposes a solution allowing the robot to deal with such case. Finally, the second constraint in Theorem 2 is a sufficient but not necessary condition for $goal$ reachability: according to Remark 20, it may happen that the deformed path in Figure 7b leads from $H_{1,2}$ to $L_{1,2}$ (instead of $L_{1,1}$) by avoiding the first obstacle on the left and then traversing $\mathcal{L}_1$ through a passage between the two obstacles.

Summarizing, Theorem 2 allows us to make a strong claim in line with (Lumelsky and Stepanov 1987): under proper some assumptions, a path leading to destination is guaranteed to be found during path following *without the need of an additional planning component*, however "maze-like" is the configuration of obstacles, given that $start$ and $goal$ are located "outside the maze". Figure 8a and b illustrate this concept.

Advanced maze-solving capabilities

The obstacle avoidance and maze-solving capabilities discussed in the previous Section turn out to be sufficient in many real-world cases. A more complex strategy can be implemented when the conditions of Theorem 2 do not hold,

i.e., either $start$ or $goal$ are located "inside the maze" created by obstacles. The following definitions are in order.

- $s^A = \pm 1$ denotes the a priori choice about the sign of A_j, made for all obstacles in order to satisfy (12).
- $\mathcal{C}'^- \subseteq \mathcal{R}^-(f_1)$ is the intersection that results when $s^A = 1$ ($A_j > 0$), and $\mathcal{C}'^+ \subseteq \mathcal{R}^+(f_1)$ the intersection that results when $s^A = -1$ ($A_j < 0$) (Remark 6).
- $s^f = \pm 1$ is a multiplying factor for $f_1'(x, y, z)$ in (6) that allows for determining the direction of motion along the deformed intersection $\mathcal{C}'^-$ or $\mathcal{C}'^+$ (Remarks 16, 17, and 19).

In the following, we assume that the product $s^f s^A$ is constant during navigation: that is, if the sign of s^f is inverted, the a priori assumption on the sign of A_j in (12) is inverted as well, thus switching from $\mathcal{C}'^-$ to $\mathcal{C}'^+$ and vice versa. The effect is visible in Figure 9a): suppose that we initially chose $s^A = -1$ and $s^f = 1$. By departing from $start$, the robot moves along $\mathcal{C}'^+(start, H_{1,2})$ by keeping $R^-(f_1)$ on its right until $H_{1,2}$ is reached, then along $\mathcal{C}'^+(H_{1,2}, L_{1,1})$ following the obstacles boundary until $L_{1,1}$ is reached, followed by another segment in the free space, and so on (continuous arrows). On the opposite, by departing from $goal$ with $s^A = 1$ and $s^f = -1$ (and by keeping the denomination of hit / leave points unaltered), the robot moves backward from $goal$ to $L_{1,3}$ along $\mathcal{C}'^-(L_{1,3}, goal)$ by keeping $R^+(f_1)$ on its right until the leave point $L_{1,3}$ is reached, then along $\mathcal{C}'^-(H_{1,3}, L_{1,3})$ following the obstacles boundary until the hit point $H_{1,3}$ is reached, and so on (dashed arrows). Segments in the free space $C'^+(L_{j,i}, H_{j,i+1})$ and $C'^-(L_{j,i}, H_{j,i+1})$ overlap, and then can be followed in both directions depending on the value of s^f; segments along the obstacle boundaries either lie in C'^+ or C'^- (and, in general, connect different hit and leave points), and therefore can be followed only in one direction.

Now let:

- $\mathcal{C}^{bp} \subseteq \mathcal{C}$ be an open curve that includes $start$ and $goal$ (not necessarily as its endpoints), henceforth referred to as the "nominal bug path"*;
- $\mathcal{C}^{b*} = \mathcal{C} \setminus \mathcal{C}^{bp}$ be the set of points that belong to $\mathcal{C}$ but are not part of $\mathcal{C}^{bp}$;
- $\mathcal{C}'^{bp} = \mathcal{C}'^+ \cup \mathcal{C}'^- \setminus \mathcal{C}^{b*}$ be the "deformed bug path", i.e., the union of the deformed intersections $\mathcal{C}'^+$ and $\mathcal{C}'^-$ deprived of the set of points that belong to $\mathcal{C}$ but are not part of $\mathcal{C}^{bp}$,

with the following constraints:

C.4) for all couples of points $\{(x_1, y_1, z_1), (x_2, y_2, z_2)\} \subset \mathcal{C}^{bp}$, given $d^{\mathcal{C}}(x_i, y_i, z_i)$ the shortest distance from (x_i, y_i, z_i) to $goal$ measured along t, given $d^{\mathcal{E}}(x_i, y_i, z_i)$ the Euclidean distance to the goal, it holds $d^{\mathcal{C}}(x_1, y_1, z_1) < d^{\mathcal{C}}(x_2, y_2, z_2) \iff d^{\mathcal{E}}(x_1, y_1, z_1) < d^{\mathcal{E}}(x_2, y_2, z_2)$.

C.5) when the robot is moving along $\mathcal{C}'^- \subset \mathcal{R}^-(f_1)$ (respectively, along $\mathcal{C}'^+ \subset \mathcal{R}^+(f_1)$), for every region $\mathcal{L}_l$ defined as the union of $\mathcal{S}_j^-$ (respectively $\mathcal{S}_j^+$), it holds $\mathcal{L}_l \cap \mathcal{C}^{bp} \neq \emptyset \implies \mathcal{L}_l \cap \mathcal{C}^{b*} = \emptyset$.

An explanation is required. First, it can be easily verified that $\mathcal{C}'^{bp}$ (the union of $\mathcal{C}'^+$ and $\mathcal{C}'^-$ in Figure 9a) is not a simple curve, since it has nodes corresponding to hit and leave points. Nodes in $\mathcal{C}'^{bp}$ corresponds to decision points: by choosing the sign of s^f (and consequently s^A to keep $s^f s^A$ constant), the robot can choose to continue along $\mathcal{C}'^+$ or $\mathcal{C}'^-$. In the Figure, if the robot is located in a hit point (e.g., $H_{1,3}$), it can choose to follow the boundary by setting $s^f = 1$ (along $\mathcal{C}^+$) or move in the free space in the opposite direction by setting $s^f = -1$ (where $\mathcal{C}'^-$ and $\mathcal{C}^+$ overlap). If the robot is located in a leave point (e.g., $L_{1,4}$), it can choose to move in the free space by setting $s^f = 1$ or follow the boundary in the opposite direction by setting $s^f = -1$ (along $\mathcal{C}'^-$).

Second, C.4 adds a constraint on the nominal path $\mathcal{C}^{bp}$, as it requires the existance of $\mathcal{C}^{bp} \subseteq \mathcal{C}$ such that the ordering of points $(x_i, y_i, z_i) \in \mathcal{C}^{bp}$ is preserved either when considering their distance to $goal$ computed along the curve or the Euclidean distance. The ordering is not preserved, for instance, if $\mathcal{C}^{bp}$ lies on a circle where $start$ and $goal$ are close to each other, but t determine the vehicle to follow the longest way around. Given that C.4 is satisfied for $\mathcal{C}^{bp}$, C.5 additionally requires that, if an island $\mathcal{L}_l$ intersects $\mathcal{C}^{bp}$, then the same island cannot intersect $\mathcal{C}$ outside C^{bp}: that is, the vehicle cannot reach a segment of $\mathcal{C}$ where the constraint C.4 does not hold. Indeed, if the constraint C.4 always holds in $\mathcal{C}$ (e.g., $\mathcal{C}$ is a straight line), the constraint C.5 is no more required.

The elements above are sufficient to implement a version of the BUG2 algorithm adapted to our framework for obstacle avoidance and path following. Please recall that, in order to search for a path out of a maze, the original BUG2 provides that the nominal path is a straight line. The vehicle starts moving along the nominal path towards $goal$, and it

*Remember that we have assumed that t (which depends on the sign of s_1) has been chosen in such a way that it is possible to move on $\mathcal{C}$ from $start$ to $goal$ along t, and then from $goal$ to $start$ along $-t$.

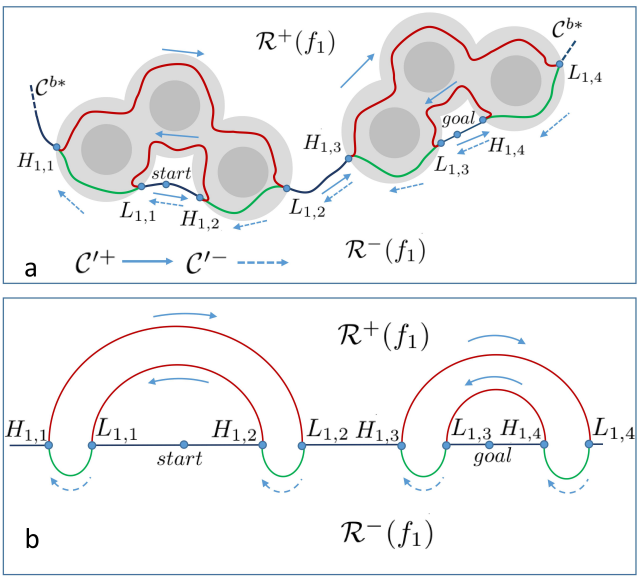

Figure 9. a) The curves $\mathcal{C}'^+$ (blue and red, $s^A = -1$) and $\mathcal{C}'^-$ (blue and green, $s^A = +1$) are shown. b) BUG2 algorithm: the nominal path is a straight line, but the order in which hit and leave points are connected to each other is the same as above.

departs from the path when required to avoid an obstacle by turning left or right (the decision among the two is made a priori). BUG2 allows the robot to leave the boundary of an obstacle if and only if all the conditions below are met, which guarantee completeness (Lumelsky and Stepanov 1987):

B.1) the robot's position (x, y, z) has come back to the nominal path;

B.2) starting from (x, y, z), it is possible to reduce the distance of the robot to the goal $d^{\mathcal{E}}(x, y, z)$ by moving along the nominal path;

B.3) the current distance of the robot to the goal $d^{\mathcal{E}}(x, y, z)$ is lower than the distance of the last point in which the robot abandoned the nominal path.

As an example, consider Figure 9b: according to BUG2, the vehicle moves along the nominal path from $start$ to $H_{1,2}$ and then it follows the boundary from $H_{1,2}$ to $L_{1,1}$ by turning left, i.e., by entering the region $\mathcal{R}^+(f_1)$. Since conditions B.1 and B.2 are verified in $L_{1,1}$ but B.3 is not, the vehicle continues to follow the obstacles boundary by entering $\mathcal{R}^-(f_1)$ until it reaches $H_{1,1}$. Here, conditions B.1 and B.3 are satisfied but B.2 is not: the vehicle continues along the obstacles boundary by entering again $\mathcal{R}^+(f_1)$ until it reaches $L_{1,2}$, where all conditions are satisfied and it can proceed along the nominal path by meeting in sequence $H_{1,3}$, $L_{1,4}$, $H_{1,4}$, $goal$. In our framework, provided that the constraints C.1 to C.5 are met, the same principles can be implemented even if the nominal path $\mathcal{C}^{bp}$ is not a straight line but a curve on a 3D surface, Figure 9a. If we set $s^A = -1$ and $s^f = 1$, the vehicle initially moves along $\mathcal{C}'^+$, which allows it to avoid obstacles in $\mathcal{R}^+(f_1)$. Whenever the vehicle comes back to the nominal path, the robot is allowed to proceed along $\mathcal{C}^{bp}$ if and only if the BUG2 conditions are met. Specifically, as the robot reaches $L_{1,1}$, boundary following* in $R^-(f_1)$ is achieved by simply inverting the sign of s^f (and then S^A), and then iteratively switching between $\mathcal{C}'^+ \subseteq R^+(f_1)$ and $\mathcal{C}'^- \subseteq R^-(f_1)$ until $goal$ is finally reached.

Remark 21. *(Lumelsky and Stepanov 1987) states that BUG2 can be used on any surface homeomorphic to a plane. Since the resulting algorithm searches only for solutions on $f_2(x, y, z) = 0$ by deforming $f_1(x, y, z) = 0$, it is possible that a path to the goal is not found even if a solution in the 3D space exists (Kutulakos et al. 1993).* □

Remark 22. *If C.4 and C.5 do not hold, one should substitute the Euclidean distance $d^{\mathcal{E}}(x, y, z)$ with $d^{\mathcal{C}}(x, y, z)$ in B.2 and B.3.* □

Algorithm 1 Maze-solving

Require: f_1, f_2, $start$, $goal$, s_0^f, s_0^A

1: $(x, y, z) = start$, $state = free$, $s^f = s_0^f$, $s^A = s_0^A$, $ld = \infty$
2: **while** $(x, y, z) \neq goal \wedge reachable(goal)$ **do**
3: $\{O_j\} = sense\_obstacles(\sigma, s^A, f_1)$
4: $(x, y, z) = move(s^f, f_1, f_2, \{O_j\}, (x, y, z))$
5: **if** $state = free \wedge near\_obstacles(\{O_j\}, (x, y, z))$ **then**
6: $state = follow$
7: $ld = d^{\mathcal{E}}(x, y, z)$
8: **end if**
9: **if** $state = follow \wedge B.1(f_1, (x, y, z))$ **then**
10: **if** $B.2(s^f, f_1, \{O_j\}, (x, y, z)) \wedge B.3((x, y, z), ld)$ **then**
11: $state = free$
12: **else**
13: $s^f = -s^f$,$s^A = -s^A$
14: **end if**
15: **end if**
16: **end while**

Algorithm 1 reports the corresponding pseudocode. Line 1 initializes the variables: the vehicle's position (x, y, z) is set to $start$, the vehicle' $state$ is set as "free to move along the nominal path", the initial values of s^f and s^A are chosen in order to initially move along $\mathcal{C}^{bp}$ in the direction t from $start$ to $goal$, and finally the "last measured distance" ld from (x, y, z) to $goal$ is initialized to "infinity". Then, lines

*Strictly speaking, the term "boundary following" is not appropriate, since the robot is moving in the interior of an island from $L_{1,1}$ to $H_{1,1}$, not on its boundary: we will continue to use this term in analogy with BUG2.

2 to 16 are repeated until the vehicle reaches $goal$ or a dedicated subroutines detects that the $goal$ is not reachable. Specifically, Lines 3 and 4 are executed at every iteration. Line 3 computes the parameters of the obstacle functions $\{O_j\}$ (i.e., x_j, y_j, z_j, A_j) for those obstacles within sensing range, which requires σ, s^A, and $f_1(x, y, z)$ (7)(26). Line 4 updates the vehicle state depending on s^f, $f_1(x, y, z)$, $f_2(x, y, z)$, obstacles $\{O_j\}$ as well as the previous vehicle state (the vehicle orientation as well as derivative terms are not shown for sake of brevity).

Lines 5 to 8 are executed only when the vehicle is moving along the nominal path, to check if it is necessary to start following the boundary of obstacles depending on the value of $\{O_j\}$ in (x, y, z). To this end, $near\_obstacles(...)$ evaluates the obstacle functions $\{O_j\}$ in (x, y, z) (i.e., only for those obstacles within sensing range) to check if the vehicle is entered into the sphere of influence of any obstacles: if this happens, $state$ switches to $follow$ and the value of ld is updated with the current distance to the goal.

Symmetrically, Lines 9 to 15 are executed only when the vehicle is following the obstacles' boundaries, to check if it is possible to head towards the goal along the nominal path. To this end, $B.1(...)$ checks if the vehicle in (x, y, z) has come back to the nominal path determined by f_1. $B.2(...)$ checks if in (x, y, z) it is possible to proceed towards the goal along the nominal path in the direction given by s^f; $B.3(...)$ checks if the distance from (x, y, z) to the goal is less than the last measured distance ld. If all conditions are verified, $state$ switches to $free$; if $B.1(...)$ is verified but $B.2(...)$ and $B.3(...)$ are not, s^f and s^A are inverted, allowing the vehicle to continue to follow the boundary (by switching from $\mathcal{C}'^+$ to $\mathcal{C}'^-$ and *vice versa*).

Remark 23. *In order for a vehicle to reach the goal, it is not sufficient that a path exists, but it is additionally required that the vehicle moves on such path with arbitrarily low control errors when executing $move(...)$ in Line 4. This is obviously unrealistic: then, in real cases, the maze-solving capabilities of Algorithm 1 depend also on the performance of the control algorithm that – on its turn – depends on velocity, perceptual errors, unmodelled dynamics, etc. All these aspects will be considered through simulated experiments in Section 6.* □

Theorem 3. *Given the nominal bug path $\mathcal{C}^{bp}$ and the deformed bug path $\mathcal{C}'^{bp}$. Given a finite set of obstacles $\mathcal{O}_j$ whose positions (x_j, y_j, z_j) can assume arbitrary values in $\Re^3$. Given that $start$ and $goal$ belong to a connected subregion $\mathcal{R}^{sg}$ of $\mathcal{R} \setminus Int(\mathcal{S})$.*

Then, Algorithm 1 is guaranteed to drive the vehicle from $start$ to $goal$.

Proof. The existance of $\mathcal{R}^{sg}$ guarantees that a path can be found in $\mathcal{R}$ not traversing the interior of the area of influence of obstacles $\mathcal{S}$. This excludes ambiguous situations in which the robot is surrounded by obstacles and the only way out passes through a couple of neighbouring obstacles $\mathcal{O}_j$ and $\mathcal{O}_l$ whose areas of influence $\mathcal{S}_j$ and $\mathcal{S}_l$ overlap, thus belonging to the same island $\mathcal{L}_i$, Remark 20. In this situation, even if a path to the goal may exists in principle, determining in advance if Algorithm 1 may lead the robot through a passage between the two obstacles is not straighforward*.

As a first step notice that, from simple topological considerations (Figure 9a), all hit / leave points that lie on the boundary of the same island $\mathcal{L}_i$ and belong to $\mathcal{R}^{sg}$ are mutually reachable by moving along $\mathcal{C}'^{bp} \cap \mathcal{L}_i$ (informally, along the boundary of $\mathcal{L}_i$), by properly switching s^f and s^A whenever the next hit / leave point is encountered.

Second, when moving along C^{bp} in the direction t from $start$ to $goal$, a point $H_{j,i}$ on the boundary of $\mathcal{L}_i$ is followed by a point $L_{j,i}$ lying on the same boundary. If both $H_{j,i}$ and $L_{j,i}$ are located before $goal$ (e.g., $H_{1,2}$ and $L_{1,2}$ in Figure 9a), $L_{j,i}$ is necessarily closer to $goal$ than $H_{j,i}$ as a consequence of C.4, C.5. If they are located after $goal$, the opposite is true (e.g., $H_{1,4}$ and $L_{1,4}$ in Figure 9a). $L_{j,i}$ (respectively, $H_{j,i}$) is a possible candidate for leaving $\mathcal{L}_i$ in the direction t (respectively, $-t$) as it satisfies B.1, B.2, B.3.

Third, if the candidate point $L_{j,i}$ belongs to $\mathcal{R}^{sg}$, it is reachable from $H_{j,i}$ as discussed in the first step. If $L_{j,i}$ does not belong to $\mathcal{R}^{sg}$, it is necessarily followed (moving along C^{bp} in the direction t) by $H_{j,i+1}$ closer to $goal$ than $L_{j,i}$, which is followed by $L_{j,i+1}$ closer to $goal$ than $H_{j,i+1}$ and so on. Sooner or later, a point $L_{j,l}$ before $goal$ is encountered that belongs to $\mathcal{L}_i \cap \mathcal{R}^{sg}$ and then is reachable from $H_{j,i}$, and satisfies B.1, B.2, B.3 (the same rationale holds with $H_{j,i}$ after $goal$ moving along C^{bp} in the direction $-t$).

To conclude, every time a robot leaving from $start$ meets the boundary of an island $\mathcal{L}_i$, it always exists at least one point on the same boundary (before or after $goal$) such that (i) it is reachable by properly switching s^f and s^A according to Algorithm 1; (ii) B.1, B.2, B.3 are satisfied in that point; (iii) the distance to $goal$ monotonically decreases after departing from it. This is sufficient to guarantee that Algorithm 1 reaches $goal$ in a finite time†. □

*In a sense, we are considering islands $\mathcal{L}_i$ as if they were obstacles in the BUG2 terminology, which boundary shall necessarily be followed when encountered along the path.

†By making the obvious assumptions that the velocity is not null and each segment of $\mathcal{C}'^{bp}$ has a finite length.

As in the case of Theorem 1, the constraint that a path cannot be found through overlapping areas of influence is only a sufficient condition: indeed, it may happen that the deformed bug path $\mathcal{C}'^{bp}$ passes between two obstacles with $S_j \cap S_l \neq 0$, Remark 20. Should this happen, $\mathcal{C}'^{bp}$ has a different shape than expected and hit / leave points may be visited in a different order, without invalidating Theorem 3.

6. Simulated Experiments

All the experiments presented in this article have been performed in simulation*. This choice has been dictated by different reasons, among which the difficulty and the cost to perform and present in a single article real experiments with three different robot typologies (wheeled robots, underwater robots and multicopters) in complex, maze-like environments. This is obviously a limitation: however, the aim of the article is to present a general integrated approach for planning, obstacle avoidance, path following and control in 2D and 3D, rather than focusing on implementation issues. Then, experiments in a realistic simulation environment seem adequate to validate the theoretical results presented throughout the article.

Materials and Methods

Simulink models have been created, with the function $move(\ldots)$ in Line 4 of Algorithm 1 properly re-defined for each robot†. Specifically, to simulate a wheeled robot a purely kinematics model has been created; for the underwater robot, we have used a set of Simulink blocks simulating the dynamics of the Fòlaga autonomous underwater vehicle with glider capability; for the multicopter, Peter Corke's Robotic Toolbox for MATLAB has been used‡.

To perform experiments:

- The two surfaces $f_1(x, y, z) = 0$ and $f_2(x, y, z) = 0$ are chosen to meet constraints C.1 to C5. For the wheeled robot, the undeformed surface $f_2(x, y, z) = 0$ defines a horizontal plane; for the underwater robot and the multicopter, $f_2(x, y, z) = 0$ defines a sinusoidal plane wave, thus operating on the vehicle's altitude. The surface $f_1(x, y, z)$ defines either a paraboloid or a plane (depending on experiments).

- 200 "simulated environments" are randomly generated, i.e., obstacle regions $\mathcal{O}$ with a random distribution of obstacles $\mathcal{O}_j$ (10). The environments are generated using different algorithms and are classified into four categories: (i) 50 environments made of $20m \times 20m$ square halls which walls can either have an opening or not – Figure 10; (ii) 50 open environments with elongated obstacles of random length – Figure 11; (iii) 50 environments made by merging the characteristics of (i) and (ii) – Figure 12; (iv) 50 environments made of $5m \times 5m$ square rooms which walls can either have an opening or not, cluttered with random obstacles – Figure 13. Maze-like worlds (i), (iii), and especially (iv) require robots to exploit advanced path planning capabilities; (ii), (iii), and especially (iv) are more likely to create narrow passages and high-curvature paths, that can be challenging for path following.

- Obstacles $\mathcal{O}_j$ are modelled as spheres with radius $r_j = 1m$ in (i)(ii) and (iii) and $r_j = 0.3m$ in (iv),to take into account both the robot and the obstacles dimensions, as well as a safety margin. $\mathcal{O}_j$ is sensed (Line 3 of Algorithm 1) if and only if its center (x_j, y_j, z_j) is within an influence range σ from the robot (x, y, z) (σ varies in different experiments).

- In all experiments, $start$ and $goal$ are given: they are connected by a parabolic path (i), (ii), (iii) and by a straight line in (iv). The random distribution of obstacles around them may produce navigation problems for which a solution does not exist by moving on $f_2(x, y, z) = 0$.

- The same navigation problem is assigned to the wheeled robot, the underwater robot, and the multicopter. Finally, an ideal "vector robot" is implemented: at each iteration, the vector robot computes a reference heading vector as in (43), and then makes a step forward along such vector. The vector robot - even if physically unrealistic - allows us to decouple path planning and obstacle avoidance from path following: if the vector robot cannot find a path from $start$ to $goal$, it either means that such path does not exist *or* the method presented in this article is not able to find it. Then, it constitutes a reference for the other robots, which capability to reach the $goal$ may also depend on low level control, Remark 23.

*Older experiments with weheeled robots are described by (Sgorbissa and Zaccaria 2013) and, more recently, by (Tanveer et al. 2017). Experiments with flying robots are described by (Nguyen et al. 2017). Videos of preliminary experiments performed with an AscTech hexacopter can be seen here: `https://www.youtube.com/watch?v=O9yh7PoqYxQ`

†All Simulink models for replicating experiments are available upon request.

‡http://petercorke.com/Robotics_Toolbox.html. Block "Quadcopter Dynamics".

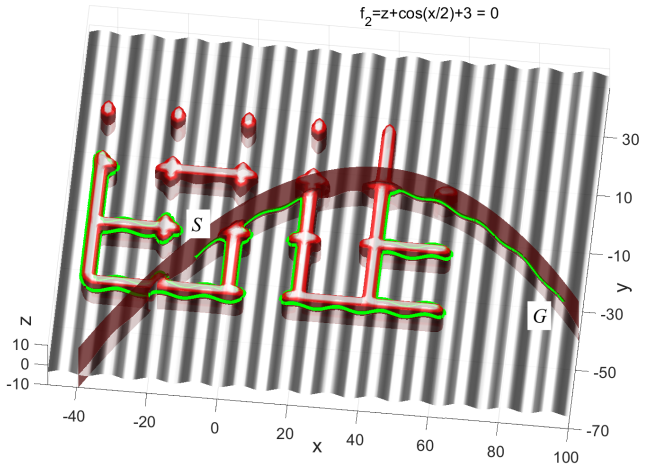

Figure 10. Environment of type (i): intersecting a parabola with a sinusoidal plane wave (path found shown in green).

- The linear velocity of the robots u has a constant value within the same experiment (u varies in different experiments);
- $\mathcal{O}_j$'s position relative to the robot is estimated by possibly adding an Additive White Gaussian Noise to range measurements, with zero mean and a standard deviation μ (μ varies in different experiments);

At each iteration step we record, among the others:

- the capability of the robot to reach $goal$;
- the robot's position (x, y, z), together with hit and leave points;
- the distance d^* from the robot's position to the center (x_j, y_j, z_j) of the closest obstacle.

Please remember that every obstacle is individually considered, with no explicit attempt to cluster neighbouring obstacles. In spite of this, when a group of obstacles are close to each other within sensing range, the two deformed surfaces in Figures 10, 11, 12, 13 (produced by setting $A_j > 0$ and $A_j < 0$ – both shown in the Figures) may produce the "red envelopes" enclosing all of them. The path from $start$ to $goal$ is shown in green in the Figures.

Results and Discussion

A navigation problem is considered as solved if the robot manages to reach $goal$ by avoiding any collision with obstacles. All failures are manually inspected to check the reason why the robot failed, i.e., either a path from $start$ to $goal$ does not exist *or* the robot was not able to find it.

Tables 1, 2, 3, 4 reports success rates of robots in solving navigation problems, each Table referring to a set of 50

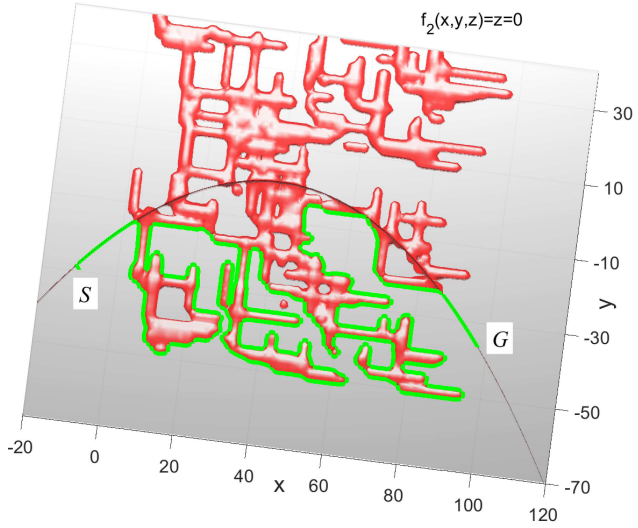

Figure 11. Environment of type (ii): intersecting a parabola with a plane (path found shown in green).

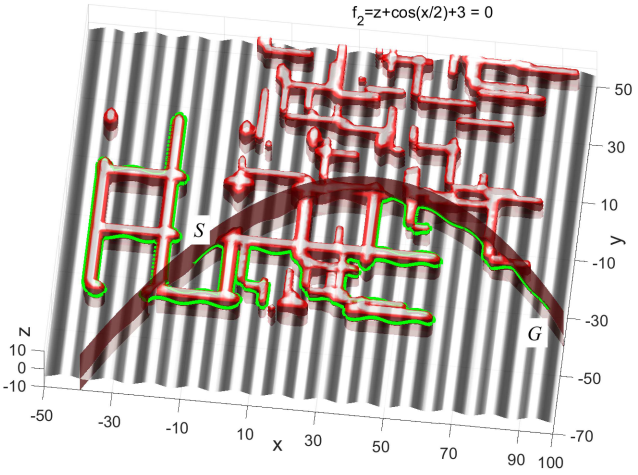

Figure 12. Environment of type (iii): intersecting a parabola with a sinusoidal plane wave (path found shown in green).

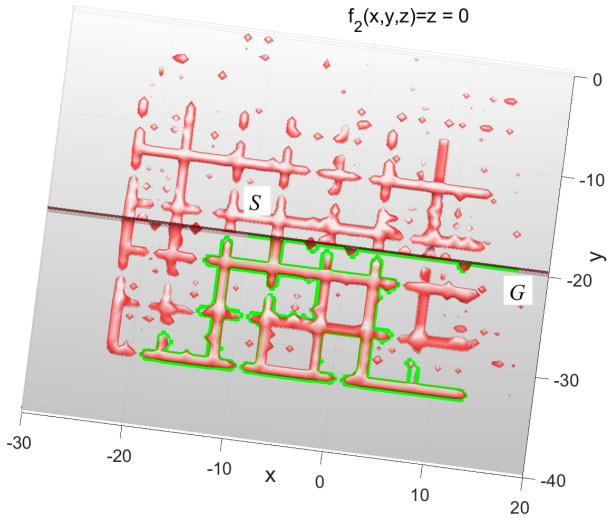

Figure 13. Environment of type (iv): intersecting a straight line with a plane (path found shown in green).

Table 1. Results of tests in environments (i)

	$u = 0.3\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0$	$u = 0.3\frac{m}{s}$ $\sigma = 2.8m$ $\mu = 0$	$u = 0.5\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0$	$u = 0.3\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0.1m$
VR	1	1	1	1
WK	1	1	1	.95
V3	1	1	1	1
UW	1	1	1/.76	.97
FR	1	1	1	1

Table 2. Results of tests in environments (ii)

	$u = 0.3\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0$	$u = 0.3\frac{m}{s}$ $\sigma = 2.8m$ $\mu = 0$	$u = 0.5\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0$	$u = 0.3\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0.1m$
VR	1	1	1	1
WK	.98	1	.98	.98
V3	1	1	1	.98
UW	.90/.78	.89/.83	.56/.04	.76/.62
FR	1	1	.96	.94

Table 3. Results of tests in environments (iii)

	$u = 0.3\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0$	$u = 0.3\frac{m}{s}$ $\sigma = 2.8m$ $\mu = 0$	$u = 0.5\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0$	$u = 0.3\frac{m}{s}$ $\sigma = 3.1m$ $\mu = 0.1m$
VR	1	1	1	1
WK	1	1	1	.97
V3	1	1	1	1
UW	.84/.68	.80/.64	.40/0	.64
FR	1	1	.97	1

Table 4. Results of tests in environments (iv)

	$u = 0.2\frac{m}{s}$ $\sigma = 0.5m$ $\mu = 0$	$u = 0.2\frac{m}{s}$ $\sigma = 0.6m$ $\mu = 0$	$u = 0.2\frac{m}{s}$ $\sigma = 0.7m$ $\mu = 0$	$u = 0.2\frac{m}{s}$ $\sigma = 0.6m$ $\mu = 0m$
VR	1	1	1/.95	1/.95
WK	.90/.80	.94/.79	.51/.48	.84/.8
V3	.95	.94	1/.95	.95
FR	.87/.19	.94	.92/.87	1/.95

"random environments" belonging to the same category (i), (ii), (iii), (iv). Different rows correspond to different robots: specifically, the success rate of the vector robot moving on a plane (*VR*) is taken as a reference for the wheeled robot with kinematics (*WK*). The vector robot moving on a sinusoidal plane wave (*V3*) is taken as a reference for the underwater robot (*UW*) and the multicopter (*FR*). In case of Table 4, due to the reduced dimensions of rooms and higher complexity of the environment which poses severe challenges to the Fólaga underwater vehicle, *UR* is not considered.

The cells report the ratio of navigation problems that have been successfully solved by the corresponding robot, by first removing the environments for which a path from $start$ to $goal$ does not exist while moving on $f_2(x, y, z) = 0$. If only one value is present in the cell, it means that the distance d^* from obstacles $\mathcal{O}_j$ is never lower than r_j during navigation. If two values are present, the first value corresponds to the ratio of problems for which a path has been found but $d^* < 0.9r_j$ at least in one point, the second (lower) value reports the ratio of problems for which $d^* \geq 0.9r_j$ along the whole path. Notice that, according to (12), $d^* < r_j$ should never happen: however, in a practical implementation this depends on low level control (Remark 23), which explains why r_j includes a safety margin.

Different columns show how the success rate varies by increasing the linear velocity u (second column), by decreasing the sensing range σ (third column), and finally by adding a Gaussian noise with zero mean and standard deviation μ to range measurements (fourth column).

1/.97 .9/.87 .8/.78 .75/.73 By analysing results, it can be observed that, for *VR* and *V3*, the ratio of successfully solved problems is always 1 in all Tables (the only exception, *V3* is affected by noise in the fourth column of Table 2), which validates the path planning and obstacle avoidance capabilities presented throughout the article. However, path following capabilities may vary for different robots depending on the complexity of the environment, the velocity*, the sensing range and the noise, thus yielding a success rate lower than 1 for *WK*, *UW*, and *FR*.

As expected, environments (ii) and (iii) are more challenging than (i), because they are more likely to create narrow passages and high-curvature paths, which are problematic for *WK* and *FR* and can be dramatic *UW*. For these robots, the success rate decreases when increasing the linear velocity u or adding noise. By inspecting failed attempts, it can be observed that robots tend to have problems in presence of "tight bends" in the path: when steering to face a tight curve, the robot may sometimes escape from the influence range of obstacles and converge again to the path but in a position preceeding the curve (thus looping forever), or it can be attracted by another curve C'_j that does not lead to the $goal$.

Another behaviour is clearly observable for *WK* and *UW*: having properly tuned gains in (37) and (39) in order to achieve a satisfactory path following when the robot is close to the nominal path, oscillations may appear when the robot moves far away from the nominal path to avoid obstacles. This can be explained by noticing that not only the values of $f'_1(x, y, z)$ and $f_2(x, y, z)$ (i.e., the errors to be regulated to

*Please notice that the velocity u is constant along the whole path, and therefore set to a safety value allowing for path following even in presence of tight curves: in a real-world implementation, it may be decreased or increased in run-time depending on the actual path curvature.

zero) contribute to determining the steering velocities r and q, but also the norms of their gradients $\|\nabla f_1'\|$ and $\|\nabla f_2\|$, which can be interpreted as variable multiplying factors for the gains that depend on the robot's position (x, y, z)*. Currently, this problem is avoided by limiting the extension of the workspace: thanks to this, the robot can never get too far from the nominal path during obstacle avoidance. Integration with methods for choosing the nominal path if a priori knowledge is available (Remark 4) or strategies to adaptively tune gains in different regions of the space shall be explored, and may be subject of future work.

Environments (iv) deserve a special attention due to the smaller dimensions of rooms and passages: experiments confirm that robots may be able to pass through narrow openings between two obstacles ($< 1m$) even when their areas of influences overlap ($1m < 2\sigma$), Remark 20.

In most cases, *FR* has a higher success rate than *WK* and *UW* since it is less affected by path following problems. This is a consequence of its dynamics properties and then the control approach adopted (43)(44), which is based on the idea of computing a velocity vector heading towards the path (similarly to *V3*) and then control the thrust and torques to converge to such vector.

Finally, it shall be noticed that the focus of the article is not on quantitative performance in terms, for instance, of path length: even if the path found may be unefficient, this is a limitation that cannot be avoided when the robot has no a priori knowledge about obstacles and is only able to acquire local information in run-time, Remark 4. Under limited sensing range, any choice to avoid obstacles to the left or to the right which initially looks promising may turn out to be suboptimal, as soon as new information is acquired. As a basis for a quantitative comparison with other algorithms when advanced maze-solving capabilities are required, the reader may refer to (Lumelsky and Stepanov 1987), analysing path length in the worst case and in prototypical situations.

Computational Complexity

Computational complexity is $O(N)$, where N is the number of obstacles: at each step, given the robot's position (x, y, z) as well as N obstacles $\mathcal{O}_j$ located in (x_j, y_j, z_j), Algorithm 1 requires to evaluate $f_1'(x, y, z)$ in the current robot's position (x, y, z) (6), which requires to evaluate $O_j(x, y, z)$ in (x, y, z) for each obstacle (7). This, on its turn, requires to compute A_j for each obstacle based on the minimum of $f_1(x, y, z)$ on $\partial\mathcal{O}_j$ (24)(32) – possibly after linearization (31). Finally, in order to control the robots (37)(39)(43), it is necessary to evaluate the the gradient $\nabla f_1'$ in (x, y, z),

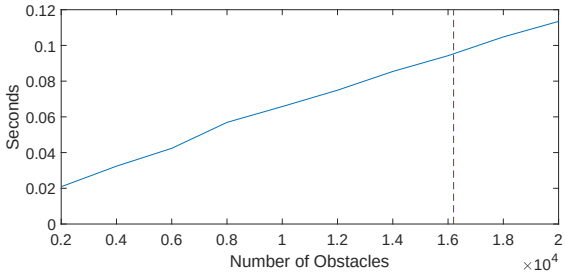

Figure 14. Computational time vs. numbers of obstacles.

which requires to evaluate the derivative of each $O_j(x, y, z)$ in (x, y, z) and, for the wheeled and the underwater robots, the second derivative, which is required for the path's curvature (Morro et al. 2011; Sgorbissa and Zaccaria 2012). It shall be noticed that, for all these computations, analytic expressions are given in this article or can be straighforwardly derived, and therefore all computations require only to substitute (x, y, z) or (x_j, y_j, z_j) in the corresponding analytic expressions.

In simulated experiments, the total number of obstacles in the environment is tens of thousands, but the number of obstacles N within sensing range σ at each iteration of Algorithm 1 is a few tens. This is due to the fact that "random walls" are modelled as "lattices of spherical obstacles" which are horizontally and vertically spaced apart – $1m$ in (i), (ii), and (iii); $0.25m$ in (iv): an extremely low resolution which however proves to be sufficient for safe navigation†.

To measure computational performance in presence of a higher number of sensed obstacles, tests have been performed in MATLAB on an Intel Core i7 with a 4.20GHz CPU (without any optimization), by evaluating all quantities for an increasing number of obstacles (Figure 14). Imagine now, without making assumptions on the sensor used, to scan a spherical region around the robot, by taking a range measurement every two degrees on the horizontal plane (totalling 180 measurements), and then assuming a 180 degrees rotation of the scanning plane (by taking a full scan every two degrees, totalling $180 \times 90 = 16200$ measurements). In the worst case that each range measurement returns a value within the influence range σ at each iteration, the computational time turns out to be adequate for path following with a $0.1sec$ sample time.

*$\|\nabla f_1\|$ and $\|\nabla f_2\|$ (i.e., in absence of obstacles) are constant in $\Re^3$ only if $f_1 = 0$ and $f_2 = 0$ define two planes.

†Please, consider that sensor acquisition is not aimed at building a map, which typically requires a higher resolution for map matching.

7. Conclusions

In the article, we have proposed a novel integrated solution to path planning, path following and obstacle avoidance that is suitable both for 2D and 3D navigation. The proposed method takes in input a generic curve connecting a *start* and a *goal* position, and then is able – under proper conditions – to find a corresponding path from *start* to *goal* in a maze-like environment even in absence of global information, it guarantees convergence to the path, and finally avoids locally-sensed obstacles without being trapped in deadlocks.

It may be argued that the approach proposed looks like another variant of Artificial Potential Fields. This requires an explanation. Indeed, similarly to APFs, the approach considers all obstacles as different entities without clustering them: all obstacles provide distinct contributions to determine the path. But the similarities stop here.

First, it shall be remembered that APFs or other force field approaches (in their original formulation) do not explicitly represent a path to be followed: when the robot is in the free-space and far from obstacles, it typically heads towards the goal along a straight line. There is no possibility, in the original force field formulation, to consider a path described as a generic curve in the 3D or 2D space.

Second, force field methods have been extended to drive a robot along a prescribed path with obstacle avoidance capabilities. For instance, (Lapierre et al. 2007) achieve obstacle avoidance by introducing the Deformable Virtual Zone (DVZ) principle, that induces a local deformation in the path. Similarly, (Quinlan and Khatib 1993; Brock and Khatib 2002; Khatib et al. 1997) rely on some sort of force field to produce a continuous deformation of the path to safely avoid obstacles. However, the former admits that the method suffers of the well-known problem of local-minima in the potential field, whereas the latter declare that they can only produce paths that are homotopic to the nominal one (i.e., produced through continuous deformations).

On the opposite, using the approach proposed in this article, the path that is produced in presence of obstacles is a simple closed or infinite curve, i.e., the robot cannot be trapped in anything similar to a "local minimum": this is a straightforward consequence of how the path is defined by two surfaces that intersect transversally, see Remarks 1, 2, and 8. Moreover, the approach has been proven to found a path to the goal under more general conditions than Elastic Bands, Elastic Strips or similar approaches: up to the case that - if a path from *start* to *goal* exists on a surface homeomorphic to a plane - the approach is always able to find it however "maze-like" is the configuration of obstacles, under the condition that $start$ and $goal$ are located "outside the maze". If the condition does not hold, a more complex strategy behaving similarly to the BUG2 algorithm is implemented.

Finally, the reader may observe that the article mostly deals with kinematic control, which may not look sufficient especially when controlling flying and underwater vehicles. Indeed, the focus of the present work is on providing an integrated solution to navigation in 2D and 3D, rather than exploring new control algorithms. However, it can be also observed that the solutions proposed for path representation and path deformation do not depend on the path following algorithm adopted. Then, such solutions may constitute a theoretical framework to design a family of control algorithms for dynamic path following, a challenge that - we hope - the interested reader may wish to take up.

References

Aicardi M, Casalino G, Bicchi A and Balestrino A (1995) Closed loop steering of unicycle like vehicles via lyapunov techniques. *IEEE Robotics Automation Magazine* 2(1): 27 –35. DOI: 10.1109/100.388294.

Alvarez A, Caffaz A, Caiti A, Casalino G, Gualdesi L, Turetta A and Viviani R (2009) Fòlaga: A low-cost autonomous underwater vehicle combining glider and AUV capabilities. *Ocean Engineering* 36(1): 24–38. DOI:10.1016/j.oceaneng.2008.08.014.

Borenstein J and Koren Y (1991) The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation* 7(3): 278 –288. DOI:10.1109/70.88137.

Braginsky B and Guterman H (2016) Obstacle avoidance approaches for autonomous underwater vehicle: Simulation and experimental results. *IEEE Journal of Oceanic Engineering* 41(4): 882–892. DOI:10.1109/JOE.2015.2506204.

Brock O and Khatib O (2002) Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research* 21(12): 1031–1052. DOI:10.1177/027836490202101202.

Consolini L, Maggiore M, Nielsen C and Tosques M (2010) Path following for the PVTOL aircraft. *Automatica* 46(8): 1284–1296. DOI:10.1016/j.automatica.2010.05.014.

Dafflon B, Chen B, Gechter F and Gruer P (2014) A self-adaptive agent-based path following control lateral regulation and obstacles avoidance. In: *Proc. 2014 International Conference on High Performance Computing Simulation*

(HPCS). Bologna, Italy, pp. 452–459. DOI:10.1109/HPCSim.2014.6903720.

Do KD, Jiang ZP and Pan J (2003) On global tracking control of a VTOL aircraft without velocity measurements. *IEEE Transactions on Automatic Control* 48(12): 2212–2217. DOI:10.1109/TAC.2003.820148.

Foo J, Knutzon J, Kalivarapu V, Oliver J and Winer E (2009) Path planning of unmanned aerial vehicles using B-splines and particle swarm optimization. *Journal of Aerospace Computing, Information and Communication* 6(4): 271–290. DOI:10.2514/1.36917.

Fox D, Burgard W and Thrun S (1997) The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine* 4(1): 23 –33. DOI:10.1109/100.580977.

Franchi A, Secchi C, Ryll M, Blthoff H and Giordano P (2012) Shared control: Balancing autonomy and human assistance with a group of quadrotor UAVs. *IEEE Robotics and Automation Magazine* 19(3): 57–68. DOI:10.1109/MRA.2012.2205625.

Gageik N, Benz P and Montenegro S (2015) Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors. *IEEE Access* 3: 599–609. DOI:10.1109/ACCESS.2015.2432455.

Hrabar S (2008) 3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs. In: *Proc. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nice, France, pp. 807–814. DOI:10.1109/IROS.2008.4650775.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30(7): 846–894. DOI:10.1177/0278364911406761.

Khatib M, Jaouni H, Chatila R and Laumond J (1997) Dynamic path modification for car-like nonholonomic mobile robots. In: *Proc. 1997 IEEE International Conference on Robotics and Automation (ICRA97)*, volume 4. Albuquerque, NM, USA, pp. 2920–2925. DOI:10.1109/ROBOT.1997.606730.

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research* 5(1): 90–98. DOI:10.1109/ROBOT.1985.1087247.

Kutulakos K, Lumelsky VJ and CR D (1993) Vision-guided exploration: A step toward general motion planning in three dimensions. In: *Proc. of 1999 IEEE International Conf. on Robotics and Automation (ICRA99)*, volume 1. Atlanta, GA, USA, pp. 289–296. DOI:10.1109/ROBOT.1993.291997.

Lapierre L and Soetanto D (2007) Nonlinear path-following control of an AUV. *Ocean Engineering* 34: 1734–1744. DOI:10.1016/j.oceaneng.2006.10.019.

Lapierre L, Zapata R and Lepinay P (2007) Combined path-following and obstacle avoidance control of a wheeled robot. *Int. J. of Robotics Research* 26(4): 361–376.

Latombe JC (1991) *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers. ISBN 079239206X.

LaValle SM (2006) *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. ISBN 9780521862059. Available at http://planning.cs.uiuc.edu/.

Lee H, Kim H and Kim HJ (2016) Planning and control for collision-free cooperative aerial transportation. *IEEE Transactions on Automation Science and Engineering* PP(99): 1–13. DOI:10.1109/TASE.2016.2605707.

Liu C, Fan S, Li B, Chen S, Xu Y and Xu W (2016) Path planning for autonomous underwater vehicle docking in stationary obstacle environment. In: *Proc. OCEANS 2016 - Shanghai*. Shanghai, China, pp. 1–5. DOI:10.1109/OCEANSAP.2016.7485467.

Liu Z, Ciarletta L, Yuan C, Zhang Y and Theilliol D (2017) Path following control of unmanned quadrotor helicopter with obstacle avoidance capability. In: *Proc. 2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. Miami, FL, USA, USA, pp. 304–309. DOI:10.1109/ICUAS.2017.7991316.

Lumelsky VJ and Stepanov AA (1987) Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica* 2: 403–430. DOI:10.1007/BF01840369.

Michalek M (2014) A highly scalable path-following controller for N-trailers with off-axle hitching. *Control Engineering Practice* 29: 61–73. DOI:10.1016/j.conengprac.2014.04.001.

Millar G (2014) An obstacle avoidance system for autonomous underwater vehicles: A reflexive vector field approach utilizing obstacle localization. In: *Proc. 2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Oxford, MS, USA, pp. 1–4. DOI:10.1109/AUV.2014.7054405.

Morro A, Sgorbissa A and Zaccaria R (2011) Path following for unicycle robots with an arbitrary path curvature. *IEEE Transactions on Robotics* 27(5): 1016–1023. DOI:10.1109/TRO.2011.2148250.

Nelson DR, Barber DB, McLain TW and Beard RW (2007) Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics* 23(3): 519–529. DOI:10.1109/TRO.2007.898976.

Nguyen DHP, Recchiuto CT and Sgorbissa A (2016) Real-time path generation for multicopters in environments with obstacles. In: *Proc. 2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Daejeon Convention Center, Daejeon, Korea, pp. 1582 – 1588.

Nguyen DHP, Recchiuto CT and Sgorbissa A (2017) Real-time path generation and obstacle avoidance for multirotors: A novel approach. *Journal of Intelligent and Robotic Systems: Theory and Applications* : 1–23DOI:10.1007/s10846-017-0478-9. Article in Press.

Ogren P and Leonard N (2005) A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics* 21(2): 188 – 195. DOI:10.1109/TRO.2004.838008.

Orsag M, Haus T, Palunko I and Bogdan S (2015) State estimation, robust control and obstacle avoidance for multicopter in cluttered environments: Euroc experience and results. In: *Proc. 2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. pp. 455–461. DOI:10.1109/ICUAS.2015.7152323.

Pathak K and Agrawal SK (2005) An integrated path-planning and control approach for nonholonomic unicycles using switched local potentials. *IEEE Transactions on Robotics* 21(6): 1201–1208. DOI:10.1109/TRO.2005.853484.

Quinlan S and Khatib O (1993) Elastic bands: connecting path planning and control. In: *Proc. 1993 IEEE International Conference on Robotics and Automation (ICRA93)*, volume 2. pp. 802–807. DOI:10.1109/ROBOT.1993.291936.

Regier P, Missura M and Bennewitz M (2017) Predicting travel time from path characteristics for wheeled robot navigation. In: *Proc. 2017 European Conference on Mobile Robots (ECMR)*. pp. 1–6. DOI:10.1109/ECMR.2017.8098664.

Sgorbissa A and Zaccaria R (2010) 3D path following with no bounds on the path curvature through surface intersection. In: *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2010)*. Taipei, Taiwan, pp. 4029 – 4035. DOI: 10.1109/IROS.2010.5653235.

Sgorbissa A and Zaccaria R (2012) Planning and obstacle avoidance in mobile robotics. *Robotics and Autonomous Systems* 60(4): 628 – 638. DOI:10.1016/j.robot.2011.12.009.

Sgorbissa A and Zaccaria R (2013) Integrated obstacle avoidance and path following through a feedback control law. *Journal of Intelligent and Robotic Systems: Theory and Applications* 72(3-4): 409–428. DOI:10.1007/s10846-012-9806-2.

Shiller Z, Sharma S, Stern I and Stern A (2013) Online obstacle avoidance at high speeds. *International Journal of Robotics Research* 32(9-10): 1030–1047. DOI:10.1177/0278364913489360.

Shim DH, Chung H and Sastry SS (2006) Conflict-free navigation in unknown urban environments. *IEEE Robotics Automation Magazine* 13(3): 27–33. DOI:10.1109/MRA.2006.1678136.

Tanveer M, Sgorbissa A and Recchiuto C (2017) Collision-free navigation of multiple unicycle mobile robots. In: *Proc. 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2017)*. Lisbon, Portugal, pp. 1–8.

Todoran G and Bader M (2016) Expressive navigation and local path-planning of independent steering autonomous systems. In: *Proc. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, pp. 4742–4749.

Tomić T, Ott C and Haddadin S (2017) External wrench estimation, collision detection, and reflex reaction for flying robots. *IEEE Transactions on Robotics* 33(6): 1467 – 1482. DOI:10.1109/TRO.2017.2750703.

Ulrich I and Borenstein J (2000) VFH*: local obstacle avoidance with look-ahead verification. In: *Proc. of the IEEE International Conference on Robotics and Automation, ICRA00*, volume 3. pp. 2505 –2511 vol.3. DOI:10.1109/ROBOT.2000.846405.

Zhang W, Wang X, Liang Z and Sun X (2016) Study of obstacle information processing for unmanned underwater vehicle in unknown ocean environment. In: *Proc. OCEANS 2016 - Shanghai*. Shanghai, China, pp. 1–5. DOI:10.1109/OCEANSAP.2016.7485720.