Shared planning and control for mobile robots with integral haptic feedback

The International Journal of Robotics Research
XX(X) 1–27

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

Shared Planning and Control for Mobile Robots with Integral Haptic Feedback

Carlo Masone[1], Mostafa Mohammadi[2], Paolo Robuffo Giordano[3] and Antonio Franchi[4]

Abstract

This paper presents a novel bilateral shared framework for online trajectory generation for mobile robots. The robot navigates along a dynamic path, represented as a B-spline, whose parameters are jointly controlled by a human supervisor and by an autonomous algorithm. The human steers the reference (ideal) path by acting on the path parameters which are also affected, at the same time, by the autonomous algorithm in order to ensure: i) collision avoidance, ii) path regularity and iii) proximity to some points of interest. These goals are achieved by combining a gradient descent-like control action with an automatic algorithm that re-initializes the traveled path (replanning) in cluttered environments in order to mitigate the effects of local minima. The control actions of both the human and the autonomous algorithm are fused via a filter that preserves a set of local geometrical properties of the path in order to ease the tracking task of the mobile robot. The bilateral component of the interaction is implemented via a force feedback that accounts for both human and autonomous control actions along the *whole* path, thus providing information about the mismatch between the reference and traveled path in an *integral* sense. The proposed framework is validated by means of realistic simulations and actual experiments deploying a quadrotor UAV supervised by a human operator acting via a force-feedback haptic interface. Finally, a user study is presented in order to validate the effectiveness of the proposed framework and the usefulness of the provided force cues.

Keywords

Shared Control, Motion Planning, Mobile Robot, Haptics

1 Introduction

Despite the significant progress of intelligent systems for robotics, many of the real world robotic missions cannot be reliably assigned to fully autonomous mobile robots. This is mainly due to the fact that robots are still limited in their world awareness and cognitive capabilities, and are thus unable to cope with the level of complexity encountered in field applications.

The presence of a human operator who possesses superior cognitive capabilities and skills, *e.g.*, reasoning and decision making, is a tenable way to overcome this limitation while still retaining the advantages of an autonomous robot. Therefore, most real-world applications include a *human-in-the-loop* whose role is to directly control or supervise the robot operation. This kind of human centered robotic system, which is defined based on shared autonomy between an autonomous robot and a human agent, to attain a collective behavior requires a comprehensive system to fuse the autonomous robot action plan with the human corrective actions.

Moreover, the presence of a human operator as supervisor of a robotic system is in many cases required by laws and regulations due to safety concerns; Unmanned Aerial Vehicles (UAV) are a representative example of this situation as they are generally not allowed to fly unsupervised in urban and populated environments.

When the robot is operated in a remote environment, providing the human operators with the haptic feedback besides visual feedback plays an essential role in improving the operator performance by increasing their situational awareness (Hokayem and Spong 2006; Lam et al. 2009; Farkhatdinov et al. 2009; Abbink et al. 2011). Furthermore,

[1]Max-Planck-Institut fur Biologische Kybernetik, Germany
[2]Univ. degli studi di Siena and Italian Institute of Technology, Italy
[3]CNRS, Univ Rennes, Inria, IRISA, Rennes, France
[4]LAAS-CNRS, France

Corresponding author:
Mostafa Mohammadi, Freie Universität Berlin Dahlem Center for Machine Learning and Robotics, Arnimallee 7, 14195, Berlin, Germany.
Email: mostafa.mohammadi@fu-berlin.de

haptic feedback has the advantage of requiring little bandwidth in comparison to video streaming, thus making it suitable for applications on which the telecommunication bandwidth is crucial, *e.g.*, remote control of underwater vehicles (Murphy et al. 2011) or intercontinental control of mobile robots over the Internet (Riedel et al. 2013).

The *bilateral teleoperation* of mobile robots is a paradigm that allows the operator to command the *current* desired state of the robot (*e.g.*, the desired position) while providing the operator with a feedback proportional to the 'mismatch' between the commanded action and the measured one. The downside of bilateral teleoperation is that the human operator is always in charge of directly steering the robot during the task—a commitment that can be overly demanding. Furthermore, such an *instantaneous* interaction between the human operator and the controlled robot is *unnecessary* in all those applications where the robot has to follow a predefined (*i.e.*, computed offline) path, and the human operator is just supposed to provide, if necessary, on-the-fly modifications of this plan. This is a common characteristic of many applications, such as environmental monitoring missions (Dunbabin and Marques 2012), airborne traffic surveillance systems (Srinivasan et al. 2004), robotic inspection and mapping systems for maintenance of construction projects (Lim et al. 2014), photogrammetry and remote sensing using unmanned aerial systems(Colomina and Molina 2014), the coverage tasks that are intended to create 2D or 3D map of the environment (Choset 2001; Montemerlo and Thrun 2006), and also the coverage tasks in 'urban search and rescue' applications (Batalin and Sukhatme 2004).

Inspired by these considerations, the first goal of this paper is to propose an extension of the classic bilateral shared control for mobile robots: the aim is to shift the interaction between the human operator and the mobile robot directly at the *planning level*, by letting the human controlling the planned path over a given future (and non-negligible) *time window* rather than the robot itself. The proposed shared control architecture will consist of several of these components:

1. a user in charge of modifying *online* the shape of a planned path;
2. a robot following the path modified by the human user, but also able to autonomously correct it in order to meet additional local requirements (for example collision avoidance, actuation feasibility, and so on);
3. a bilateral interface between robot and human able to feed back to the operator force cues based on the mismatch between the *commanded/ideal* path and the one actually followed by the robot after the autonomous corrections have taken place.

The second goal of this paper is to assess the effectiveness of the proposed shared planning architecture in terms of performance in task execution and usability by a human operator. For this purpose, we report the results of an extensive user study in a plausible application scenario.

The rest of the paper is organized as follows: Sec. 1.1 reviews the existing literature and illustrates the main ideas and contributions of this paper. Section 2 introduces the model of the path and of the environment. The structure of the proposed framework is then described in Sec. 3 and its components are detailed in Secs. 4 to 8. Simulations and real experiments that demonstrate the workings of the proposed framework works are presented in Secs. 9 and 10, while the results of the user study are discussed in Sec. 11. Finally, Appendix B presents some remarks to generalize the proposed approach to 3D, Appendix C contains some proofs, and Appendix D provides a short overview on the notation adopted for B-splines.

1.1 Related Work and Contributions

Shared control is a concept that has first emerged in telerobotics as a paradigm to distribute the control over the execution of a task between operator and teleoperator (Niemeyer et al. 2008). The fundamental idea of this paradigm is to endow the robot with a moderate intelligence that gives it the ability to alter the human's commands in two ways: 1) by assigning different subtasks to the human and to the autonomous system, and then *blending* their individual control actions; 2) by using a haptic (force) feedback that modifies the interaction of the human with the control interface.

More in general, shared control has been adopted in a broad variety of robotic assignments both to simplify the user's mental commitment and to increase performance diminishing his/her authority. For example, Hirzinger et al. (1994) exploited the autonomy of the robot to cope with the large time delays encountered in space applications. Ortmaier et al. (2005) required a surgical robot to autonomously compensate for the patients movement in order to facilitate the surgeon's task. Bukusoglu et al. (2008) proposed shared control as a mechanism to assist a human operator in the manipulation of microspheres with optical tweezers. Shared control was also used by Mulder et al. (2012) and Profumo et al. (2013) as a driver support system for car pilots and by Glassmire et al. (2004) for cooperative manipulation of objects between human workers and a remotely operated humanoid robot.

In recent years shared control has been successfully extended to mobile robots as a solution to increase the situation awareness of a human operator who is teleoperating ground robots or UAVs in cluttered environments (Diolaiti and Melchiorri 2002; Lam et al. 2009; Farkhatdinov and Ryu 2010; Rodríguez-Seda et al.

2010; Franchi et al. 2012b; Jiang et al. 2016a). Along these lines, Lee et al. (2013) presented a UAV bilateral teleoperation scheme in which the velocity of the robots formation is controlled by the position of a haptic device with a guaranteed passivity of the teleoperation system despite communication delays and other non-idealities (Lee and Huang 2010; Van Quang et al. 2012), and an admittance based bilateral teleoperation system architecture for environmental force reflection proposed for mobile robots in Hou et al. (2017). Another scenario is studied by Franchi et al. (2012a), where the formation is defined in terms of relative bearings among the robots and up to a scale factor, with the operator in charge of commanding the overall translations, rotations along a vertical axis and contractions/expansions of the whole group.

Recently another paradigm of remote control, called *assistive teleoperation*, has also been proposed. Similarly to shared control, this paradigm stems from the idea of embedding higher intelligence in the control interface of the user in order to augment his/her commands. The distinctive characteristic of assistive teleoperation, formalized by Dragan and Srinivasa (2012), is that the robot attempts to predict the user's intent and then uses this prediction for producing a motion plan which is finally blended with the operator's input via a weighted sum driven by an arbitration function. The method proposed by Dragan and Srinivasa (2012) lets the operator setting waypoints or giving velocity commands to the robot, and it is goal-oriented since the planning assumes that each new scene has a set of reachable goals. This goal-oriented formulation is well suited for tasks such as reaching and picking up an object, and relieves the user from a direct and continuous control of the robot. However, this formulation does not provide the human operator with the possibility of modifying the path to be taken, and it is not immediately applicable to goal-less applications that focus primarily on the route/path to be followed (as, *e.g.*, in monitoring or surveillance). The assistive framework has also been adapted by Hauser (2013) to perform tracking tasks; however, this adaptation relies on the fact that the user acts as a pilot who continuously steers the tracked point.

The application of shared control is not limited to mobile robots freely moving in the environment; shared control for physical human–robot interaction is also a popular approach; a collection of recent works about shared control for physical human–robot interaction with focusing on intent detection, arbitration, and communication aspects are presented in Losey et al. (2018). In particular, trajectory deformation in shared control of robots with physical interaction is considered in Losey and O'Malley (2018), a game-theoretic based adaption law presented in Li et al. (2015) allows the robot to adjust its own role according to human intention which is inferred through the measured interaction force, and a visual-based shared control for telemanipulation is presented in Pedemonte et al. (2017). Finally, shared control is also widely used in driver assisting systems for road vehicles Erlien et al. (2016); Nguyen et al. (2017); Benloucif et al. (2017).

A common denominator of all these prior works is that the human operator is forced to control directly the motion of the robot(s) in order to follow a certain trajectory. The controlled robot does possess some autonomy but it always needs the user's inputs for steering a target point (*e.g.* using velocity commands) in order to navigate through the environment. On the other hand, our proposed framework aims at changing the role of the human teleoperator in a shared architecture by putting him/her in charge of directly modifying the *path* followed by the robot. Note that this is also quite different from the assistive approach in which the operator sets waypoints and a planner uses them for computing a path. In fact, in that case the operator does not have direct control over the final path (which is ultimately chosen by the planner). Furthermore, adding new waypoints does not allow the operator to modify the part of the path that is already planned.

Summarizing, the framework presented in this paper proposes a novel perspective to the problem of shared and assistive control of mobile robots. The main features of the proposed solution are:

1. The human operator is in charge of correcting online the desired *path* followed by robot, rather than steering a point to be tracked. This implies that the user is not required to provide commands as long as the planned path does not need any modification (and the robot can just travel along it). Secondly, by modifying the planned path the operator can affect how the robot will behave over a future time window, and not just only around its current pose.

2. The mobile robot autonomously follows the path selected online by the user, but with the possibility of adjusting it to meet additional requirements, *e.g.*, feasibility.

3. In previous works, the blending of human and autonomous control actions is often fairly simple, typically a weighted sum of the two terms or slight variations of this technique (*e.g.* Jiang et al. 2016b, use a hysteresis switch to combine the control actions of the human and of the feedback loop). In this work, we instead introduce a new blending technique that uses the concept of null-space projection to ensure important geometrical properties of local continuity of the path.

4. The force feedback provided to the operator is proportional to the mismatch between the planned

and actual *path* travelled by the robot (*i.e.*, exploiting an error signal evaluated, in an *integral* sense, along the whole future trajectory). Therefore, the force feedback do not provide an information about the current tracking error but, rather, about the overall planned motion — from which the proposed terminology "integral haptic feedback".

The framework in the paper proposed to share the planning and control between a human operator a robot. For the practical applications, such as the presented USAR scenario of the Sec.11, the proposed framework requires localization in the environment map. This map could be a global map of the environment, but not necessarily a perfect one; the map could be updated while performing the task, by detecting objects in the environment using the on-board perception system (that is also a necessity for any collision avoidance system). It is important for the the proposed algorithm to have the pose of the robot, obstacles, and points of interest. In other words, planning and correction could be performed locally to the extend of known map in which the robot is localized. If the localization fails, and that could be recognized by the human operator, who benefits from a superior intelligence and cognitive capabilities, or even autonomously by checking measures such as pose covariance in case of deploying filtering methods, e.g., EKF localization or by checking the particles distribution of in case of using particle filter method, the human operator takes takes over the control and will safely guide the robot by switching back to teleoperation control mode, which could be implemented based on visual servoing techniques in which the robot receives velocity commands.

Parts of this work have been preliminarily presented (Masone et al. 2012, 2014). With respect to these previous publications, in this paper we provide: (i) a more thorough and formal explanation of all the components of the framework, (ii) additional theoretical proofs about the regularity and collision avoidance of the planned path, (iii) novel simulations and experiments, and (iv) a user study that investigates the benefits of the framework in terms of performance and ease of use. We want to remark that the simulations and experiments presented in this paper are original and expand our previous results shown. On one hand, w.r.t. our previous simulations (Masone et al. 2012) in this manuscript we offer a more detailed view of the framework. On the other hand, while our previous experiments (Masone et al. 2014) focused on demonstrating the real world effectiveness of the autonomous components of the framework (obstacle avoidance), the experiments in this paper are meant to show how the user can command different paths, both open and closed, using a variety of commands and give more insight on the effect of these commands on the path itself.

2 Preliminaries

Path We consider paths represented by B-splines. Without loss of generality, and for the sake of clarity, we present the proposed framework in $\mathbb{R}^2$. This approach is independent from the dimension of the path, and deploying it in $\mathbb{R}^3$ is straightforward as presented in Appendix B .

B-splines are a simple yet powerful tool for path planning which can be used to exactly represent or approximate arbitrary shapes with desired smoothness. The family of planar B-spline curves here considered is described by the function

$$\boldsymbol{\gamma} : \mathbb{R}^{2n} \times S \to \mathbb{R}^2, \tag{1}$$

where $S \subset \mathbb{R}$ is a compact set. A B-spline curve of this family is a function

$$\boldsymbol{\gamma}(\boldsymbol{x}, \cdot) : S \to \mathbb{R}^2, \quad s \mapsto \boldsymbol{\gamma}(\boldsymbol{x}, s) \tag{2}$$

that is parameterized by the vector of control points $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1^T & \cdots & \boldsymbol{x}_n^T \end{pmatrix}^T \in \mathbb{R}^{2n}$. According to this notation $\boldsymbol{\gamma}(\boldsymbol{x}, s) \in \mathbb{R}^2$ is a single point of the B-spline curve, *i.e.*, the point obtained by evaluating the function $\boldsymbol{\gamma}(\boldsymbol{x}, \cdot)$ in $s \in S$. Finally, the path corresponding to the B-spline curve $\boldsymbol{\gamma}(\boldsymbol{x}, \cdot)$ is

$$\boldsymbol{\gamma}_S(\boldsymbol{x}) = \{\boldsymbol{\gamma}(\boldsymbol{x}, s) \in \mathbb{R}^2 \,|\, s \in S\}, \tag{3}$$

i.e., the set of points obtained by varying the coordinate s within S. The control points $\boldsymbol{x}$ parameterizing the B-spline determine the shape of the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$.

B-splines depend on other parameters beside the ones introduced so far, namely 1) the degree λ, which determines how many times the curve is continuously differentiable w.r.t. s, and 2) the knots $s_1, s_2, \ldots, s_l$ that determine the set $S = [s_1,\ s_l]$ and are used in the evaluation of the B-spline for a certain s. For the purpose of this framework these additional parameters are considered constant and have been omitted from (1)-(3) in order to ease the notation. Further details on these parameters and, in general, on the B-spline structure can be found in Appendix D .

Robot The mobile robot considered for the task is assumed to possess a characteristic point capable of traveling with non-zero speed along sufficiently smooth B-spline paths (3). Such a path-followability property is met by differentially flat systems under mild conditions (Faulwasser et al. 2011, Van Loock et al. 2014), *i.e.*, i) the initial state of the system is consistent with the path, and ii) compatibly with the constraints on input and state, the system can visit in steady state each point of the path, *e.g.*, with a low constant speed. These conditions are reasonable for mobile robots because they are in large part (differentially) flat with a characteristic point as part of the flat output (see *e.g.* Murray et al. 1995; Hauser and Hindman 1997, Mistler et al. 2001), or equivalently,

feedback linearizable with the characteristic point taken as linearizing output (Isidori 2013), and most real world applications require slow motions, *e.g.*, to collect data from the environment. As a concrete example of such mobile robot, the robot used in our simulations and experiments (Secs. 9 to 11) is a quadrotor UAV flying at a constant altitude.

Constraints The quadrotor UAV used in our simulations can freely translate in space, up to its actuation limits. However, other robots may have other constraints on the possible paths they can follow. Indeed, while for an omnidirectional robot any path is feasible, non-holonomic robots in general require a smooth-enough path with, often, bounded geometric derivatives (Laumond et al. 1998). For instance, the bounds on velocity and bank angle of a fixed-wing aircraft can be formulated as a minimum curvature requirement (Bicchi and Pallottino 2000), whereas the motion of a car-like vehicle with n trailers poses constraints on the first n derivatives of the curvature (Laumond et al. 1998). Also relevant in this regard is the work Majewicz and Okamura (2013), which achieves the teleoperation of a non-holonomic steerable needle by imposing a curvature constraint on the Cartesian path.

In order to keep the formulation general and not overly complicated we do not consider in this paper such constraints but we will address their inclusion in future studies. A few comments in this sense are given in Sec. 12. We do, however, consider one important constraint that is generally required by any mobile robot for producing a smooth motion without any stops: the path must be free of singularities. We recall that a singularity for a parametric path is defined as follows:

Definition 1. *A point $\boldsymbol{\gamma}(\boldsymbol{x}, s)$ with $\boldsymbol{x} \in \mathbb{R}^{2n}$, $s \in S$ and such that $\frac{\partial \boldsymbol{\gamma}}{\partial s}\Big|_{(\boldsymbol{x},s)} = (0\ 0)^T \in \mathbb{R}^2$ is called a* singularity *of $\boldsymbol{\gamma}_S(\boldsymbol{x})$. A path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ without singularities is called "regular".*

Regularity of the path is an important requirement since at a singularity the direction of motion, *i.e.*, the tangent vector, vanishes. Geometrically, this situation could correspond to a cusp or a backtracking in the path, as illustrated in Fig. 1. In order to prevent the occurrence of singularities in $\boldsymbol{\gamma}_S(\boldsymbol{x})$, the control points $\boldsymbol{x}$ must be chosen accurately. This is explained by the following definition.

Definition 2. *Consider a regular path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ of degree $\lambda > 0$, with $S \subset \mathbb{R}$ and $\boldsymbol{x} = (\boldsymbol{x}_1^T\ \boldsymbol{x}_2^T\ \dots\ \boldsymbol{x}_n^T)^T \in \mathbb{R}^{2n}$. Let $B_i^\lambda(\boldsymbol{s}, s)$ be the basis function associated to $\boldsymbol{x}_i$ and let $\boldsymbol{x}_i^\star(\boldsymbol{x}, s) \in \mathbb{R}^2$, with $s \in S_i = \{s \in [s_{i-\lambda},\ s_i) : \frac{d\, B_i^\lambda(\boldsymbol{s},s)}{d\, s} \neq 0\}$, indicate a point such that $\boldsymbol{\gamma}\left((\boldsymbol{x}_1^T\ \dots\ \boldsymbol{x}_{i-1}^T\ \boldsymbol{x}_i^\star(\boldsymbol{x}, s)^T\ \boldsymbol{x}_{i+1}^T\ \dots\ \boldsymbol{x}_n^T)^T, s\right)$ is a singularity. The 'singular curve' of the control point $\boldsymbol{x}_i \in \mathbb{R}^2$ is the collection of points $\Omega_i(\boldsymbol{x}) = \{\boldsymbol{x}_i^\star(\boldsymbol{x}, s) \,|\, s \in S_i\}$.*

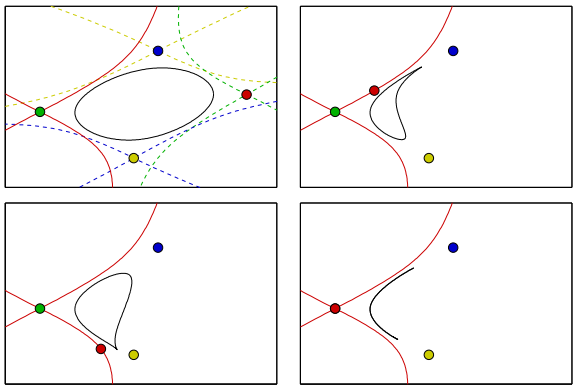


Figure 1. Example of a B-spline (black line) of degree $\lambda = 3$, with 4 control points (colored points). By moving one control point (the red one), the B-spline is made non-regular. Top-Left: initial regular B-spline and singular curves (colored lines) of the control points (with the same color pattern). The dashed lines are the singular curves of the fixed control points. Other boxes: the B-spline becomes non-regular when one control point (red one) is moved onto its singular curve.

Remark 1. *In definition 2, if $i - \lambda \leq 0$, use s_1 instead of $s_{i-\lambda}$.*

Details on the computation of $\boldsymbol{x}_i^\star(\boldsymbol{x}, s)$ are given in Appendix D where it is also shown that for each $s \in S_i$ the point $\boldsymbol{x}_i^\star(\boldsymbol{x}, s)$ is unique and independent from $\boldsymbol{x}_i$. The interpretation of singular curves is twofold. Firstly, a path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is regular if none of its control points lies on the corresponding singular curve. Secondly, if $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is regular, the singular curve Ω_i describes how the control point $\boldsymbol{x}_i$ can be modified without creating singularities. These concepts are illustrated with an example in Fig. 1.

Environment The environment where the task takes place is populated by static *obstacles* to be avoided, and by *points of interest* to be reached.

In this venue, obstacles are simple primitive shapes that can approximate objects to be avoided with a sufficient level of accuracy. Indeed, since the focus of this paper is not on obstacles detection and mapping, we keep a simple formulation and 1. approximate the any object to be avoided by only using circles with a fixed radius $R_{\mathcal{O}}$, and 2. assume presence of an "obstacle provider" in charge of generating the position of the centers $\mathcal{O} \in \mathbb{R}^{2 \times n_{\mathcal{O}}}$ of the obstacle circles. With this setting, the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is considered to be collision free if it lies outside the obstacle circles.

The extension of our framework to use other approximating shapes (*e.g.* rectangles, ellipses) is straightforward. In fact, the actual information needed by the reactive controller presented in Sec. 5 is the relative distance and

direction between the obstacles (simple primitive shapes) and the path (numerically discretized). The points of interest (PoI) represent important locations for the task, *e.g.*, meeting points, or fixed stations for data transfer, and are represented as a finite set of points whose positions are collected in the vector $\mathcal{I} \in \mathbb{R}^{2\times n_{\mathcal{R}}}$.

3 Overview of the proposed framework

The idea behind our approach is to let a human operator modify in real-time the shape of the path to be followed by the mobile robot with the assistance of an autonomous algorithm in charge of correcting, when necessary, the operator's commands for ensuring path regularity and satisfaction of any requirement regarding obstacles and PoIs. Note that other goals of interest, such as, *e.g.*, minimum path curvature or length, could be also be included in our framework.

Path modifications are realized by introducing a time dependency in $\boldsymbol{x}$, so that $\boldsymbol{\gamma}(\boldsymbol{x}(t), s)$ in (2) becomes a time-varying point and $\boldsymbol{\gamma}_S(\boldsymbol{x}(t))$ in (3) a time-varying path. Note that, by introducing also a signal $s(t)$ in (3), then $\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$ provides the reference trajectory for the robot according to the well-known decoupled design in *path* and *timing law* (Kant and Zucker 1986, Peng and Akella 2005). We do not consider the design of a timing-law $s(t)$ to be the focus of this paper and, therefore, in our simulations and experiments (Secs. 9 to 11) we simply adopted a signal $s(t)$ able to keep a low travelling speed by modulating it with the curvature of the path. The interested reader can find in the literature more sophisticated algorithms for the generation of $s(t)$, see for example the works of Faulwasser et al. (2011) and Smith et al. (2012).

The core of our framework is the dynamic system that generates online the signal $\boldsymbol{x}(t)$. Formally,

$$\dot{\boldsymbol{x}} = \boldsymbol{N}\left(\boldsymbol{u}_h + \boldsymbol{u}_a\right), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \boldsymbol{N} \in \mathbb{R}^{2n\times 2n} \tag{4}$$

where $\boldsymbol{u}_h \in \mathbb{R}^{2n}$ is an input provided by the human operator (described in Sec. 4), $\boldsymbol{u}_a \in \mathbb{R}^{2n}$ is the automatic action provided by system (described in Sec. 5), and $\boldsymbol{N} \in \mathbb{R}^{2n\times 2n}$ is a filtering term used to blend the two control signals (described in Sec. 6). We assume that the initial condition $\boldsymbol{x}_0$ corresponds to a regular and collision-free path. For example, the initial path can be the output of an exploration algorithm planning the next move based on the current partial map, or of a coverage method that selects one specific path among a predefined family of curve patterns. In general, the trajectory initialization could also be repeated over time, for instance when the robot has almost completed the current trajectory. In Sec. 7 we also show how $\boldsymbol{x}$ can be automatically reinitialized in order to better span the environment in presence of obstacles.

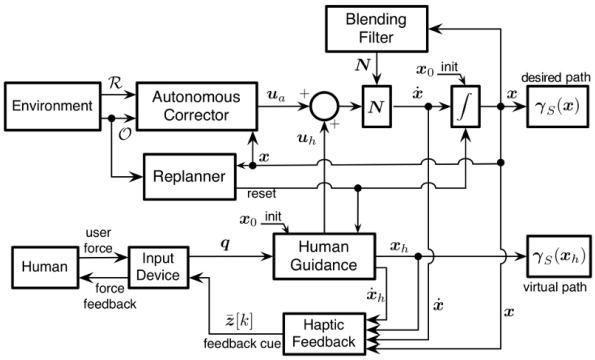


Figure 2. Overview of the framework. The signals $\boldsymbol{x}_h$ and $\dot{\boldsymbol{x}}_h$ indicate the desired corrections given by the human (see Sec. 4).

The proposed framework, illustrated in Fig. 2, consists of the following elements:

Human guidance it provides the signal $\boldsymbol{u}_h$ in (4) for steering the *travelled path* $\boldsymbol{\gamma}_S(\boldsymbol{x}(t))$ towards a *desired path* $\boldsymbol{\gamma}_S(\boldsymbol{x}_h(t))$, which is modified by the human operator via an actuated multi-DoFs input device.

Autonomous corrector it provides the signal $\boldsymbol{u}_a$ in (4) for correcting, when necessary, the human's commands so as to keep the travelled path $\boldsymbol{\gamma}_S(\boldsymbol{x}(t))$ free of singularities and compliant with the presence of obstacles and PoI in the environment.

Blending filter it provides the projection term $\boldsymbol{N}$ in (4) which blends the actions of the human guidance and autonomous corrector by locally filtering path modifications at the point travelled by the robot.

Replanner it reinitializes the vector $\boldsymbol{x}$ for replanning the actual (travelled) path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ in proximity of obstacles (when necessary) in order to escape from local minima.

Haptic feedback it closes the interaction-loop with the human operator by providing force cues (via an actuated input device) meant to physically inform the operator about any change generated by the autonomous correction or by the correction filter to his/her suggested path modifications.

The five parts of the framework are thoroughly described in Secs. 4 to 8.

4 Human guidance

The human guidance is implemented via an input device having m fully-actuated DoFs, with $1 \leq m \leq 2n$. An example of actuated input device is illustrated in Fig. 5. The device is modelled as a generic (gravity pre-compensated) mechanical system

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{\tau} + \boldsymbol{\tau}_h \tag{5}$$

where $\boldsymbol{q} \in \mathbb{R}^m$ is the configuration vector of the device, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{m \times m}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^m$ are the Coriolis and centrifugal terms, and $\boldsymbol{\tau}$, $\boldsymbol{\tau}_h \in \mathbb{R}^m$ are the control and human forces, respectively. The computation of $\boldsymbol{\tau}$ is done automatically by the haptic feedback algorithm and is described in Sec. 8.

The idea of the human guidance is to use the configuration vector $\boldsymbol{q}$ to generate $\boldsymbol{u}_h$ in (4) and thus modify the reference path. When designing this action we must take into account the dissimilarity between the workspace of the input device and the generally much larger workspace where the path is specified. A common approach to overcome this limitation, in particular in the case of mobile robots, is to map the configuration $\boldsymbol{q}$ to a velocity of the controlled system (here the control points $\boldsymbol{x}$) through a memoryless mapping (*e.g.* Lee et al. 2013; Franchi et al. 2012c a). We depart from this solution by introducing a memory of the operator's directives so that, in case of vanishing autonomous perturbations (*e.g.*, when the path is steered away from obstacles), $\boldsymbol{\gamma}_S(\boldsymbol{x})$ can be restored to the ideal path intended by the human. We define the mapping from $\boldsymbol{q}$ to $\boldsymbol{u}_h$ as the dynamic system

$$\dot{\boldsymbol{x}}_h = \boldsymbol{Q}\left(\boldsymbol{x}_h\right)\boldsymbol{K}\boldsymbol{q}\,, \qquad \boldsymbol{x}_h(0) = \boldsymbol{x}_0 \tag{6}$$

$$\boldsymbol{u}_h = \dot{\boldsymbol{x}}_h + k_h(\boldsymbol{x}_h - \boldsymbol{x}) \tag{7}$$

where $\boldsymbol{x}_h = \begin{pmatrix}\boldsymbol{x}_{h,1}^\top & \cdots & \boldsymbol{x}_{h,n}^\top\end{pmatrix}^\top \in \mathbb{R}^{2n}$, $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ is a diagonal matrix of positive gains, $\boldsymbol{Q} : \mathbb{R}^{2n} \to \mathbb{R}^{2n \times m}$ is a nonlinear mapping and $k_h > 0$. Vector $\boldsymbol{x}_h$ defines the desired path $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ that is *only* modified by the operator through mapping (6) and thus constitutes the memory of the user's commands. Equation (7) then implements $\boldsymbol{u}_h$ by combining the velocity commanded by the human (*i.e.*, $\dot{\boldsymbol{x}}_h$) with an action proportional to the discrepancy between the 'travelled' $\boldsymbol{x}$ and 'desired' $\boldsymbol{x}_h$.

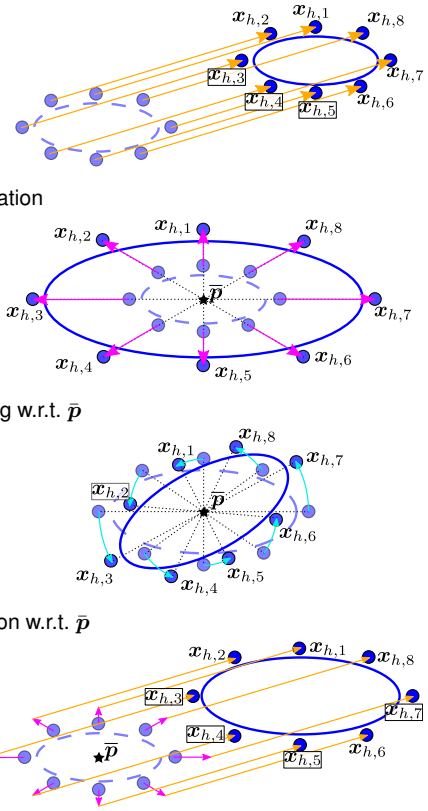
Matrix $\boldsymbol{Q}$ in (6) determines how the operator can influence the path by manipulating the input device, therefore it should be chosen carefully. Clearly there is not a unique "correct" choice for $\boldsymbol{Q}$. However, a reasonable guideline for an intuitive interface is to map $\boldsymbol{q}$ to a limited number of 'canonical' transformations of the path that can be easily managed by the operator, such as translations or changes of scale. Following this suggestion, $\boldsymbol{Q}$ is taken as the juxtaposition of l elementary matrices $\boldsymbol{Q}_i(\boldsymbol{x}_h) \in \mathbb{R}^{2n \times \nu_i}$ with $i = 1 \dots l$

$$\boldsymbol{Q}(\boldsymbol{x}_h) = \begin{pmatrix}\boldsymbol{Q}_1(\boldsymbol{x}_h) \mid & \dots & \mid \boldsymbol{Q}_l(\boldsymbol{x}_h)\end{pmatrix}, \tag{8}$$

where $1 \leq l \leq m$, $1 \leq \nu_i \leq m$, and $\sum_{i=1}^{l} \nu_i = m$. Partition (8) induces a corresponding partition of $\boldsymbol{q}$

$$\boldsymbol{q} = \begin{pmatrix}\boldsymbol{q}_1^T \mid & \dots & \mid \boldsymbol{q}_l^T\end{pmatrix}^T$$

with $\boldsymbol{q}_i \in \mathbb{R}^{\nu_i}$ for $i = 1 \dots l$. Each $\boldsymbol{q}_i$ is thus mapped through the corresponding elementary matrix $\boldsymbol{Q}_i(\boldsymbol{x}_h)$ to a different canonical transformation of the desired path.



(a) Translation

(b) Scaling w.r.t. $\bar{\boldsymbol{p}}$

(c) Rotation w.r.t. $\bar{\boldsymbol{p}}$

(d) Simultaneous translation and scaling

Figure 3. Example of canonical path transformations. Orange arrows refer to translations, magenta arrows refer to changes of scale, cyan arrows refer to rotations. The initial path is represented by the dashed transparent curve, the final path is represented by the solid blue curve. The control points are represented by the circles. (a) Translation: all the control points move in the same way. (b) Scaling: the control points move in a coordinated way from the center $\bar{\boldsymbol{p}}$. (c) Rotation: each control point moves on a circle around $\bar{\boldsymbol{p}}$. (d) Simultaneous translation and scaling: the two commands are added together.

Clearly, the expression of matrix $\boldsymbol{Q}_i(\boldsymbol{x}_h)$ depends on the chosen canonical path transformation and cannot be described in a unique, general way. Here we present three examples of canonical path transformations that can result in an intuitive interface for the human operator: translations, changes of scale and rotations.

Translation As a first canonical path transformation we want to map the sub-configuration $\boldsymbol{q}_1 \in \mathbb{R}^2$ of the input device to a translation of the whole path. This can be easily achieved by moving all the control points $\boldsymbol{x}_{h,1}, \dots, \boldsymbol{x}_{h,n}$ with the same velocity, namely choosing $\boldsymbol{Q}_1(\boldsymbol{x}_h)$ as

$$\boldsymbol{Q}_1(\boldsymbol{x}_h) = \boldsymbol{I}_{2n} \tag{9}$$

where $\boldsymbol{I}_{2n} \in \mathbb{R}^{2n\times 2n}$ is the identity matrix. Notice that $\boldsymbol{Q}_1$ does not depend on $\boldsymbol{x}_h$, since the velocity applied to each control point is simply a scaled version of vector $\boldsymbol{q}_1$, without further transformations. An illustration of this transformation is shown in Fig. 3a.

Scaling As a second canonical path transformation we want to find a map that transforms the sub-configuration $\boldsymbol{q}_2 \in \mathbb{R}$ to a change of the scale of the whole path w.r.t. a point $\bar{\boldsymbol{p}} \in \mathbb{R}^2$. This behaviour can be achieved by a coordinated motion of the control points towards/from $\bar{\boldsymbol{p}}$. We then choose $\boldsymbol{Q}_2(\boldsymbol{x}_h)$ as

$$\boldsymbol{Q}_2(\boldsymbol{x}_h) = \boldsymbol{x}_h - \mathbb{1}_n \otimes \bar{\boldsymbol{p}} \tag{10}$$

where $\mathbb{1}_n$ is an n-dimensional column vector of ones and $\otimes$ is the Kronecker product. An illustration of this transformation is shown in Fig. 3b.

Rotation As a third canonical path transformation we want to find a map that transforms the sub-configuration $\boldsymbol{q}_3 \in \mathbb{R}$ to a rotation of the whole path w.r.t. a point $\bar{\boldsymbol{p}} \in \mathbb{R}^2$. This behaviour can be achieved by a coordinated motion of the control points on circles centered in $\bar{\boldsymbol{p}}$. We then choose $\boldsymbol{Q}_3(\boldsymbol{x}_h)$ as

$$\boldsymbol{Q}_3(\boldsymbol{x}_h) = \mathrm{diag}(\underbrace{\bar{\boldsymbol{I}}_2, \cdots, \bar{\boldsymbol{I}}_2}_{n \text{ times}})\,(\boldsymbol{x}_h - \mathbb{1}_n \otimes \bar{\boldsymbol{p}})\,, \tag{11}$$

where $\bar{\boldsymbol{I}}_2 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ and $\mathrm{diag}(\cdot)$ denotes a diagonal (or block diagonal) matrix with its argument as the diagonal entries. See Fig. 3c for an example.

Remark 2. *As explained, the three canonical path transformations here presented are meant to demonstrate the flexibility of the command interface given by (6) to (8). It would also be possible to implement a command interface that enables local modifications for finer control of the planned motion, as shown by Masone et al. (2012). Note also that the chosen individual canonical path transformations are not limited to be used separately, but they can also be exploited simultaneously as shown by the example in Fig. 3d.*

5 Autonomous correction

The purpose of this control action is to guarantee feasibility of the path by realizing the following two objectives ("hard constraints"):

Objective 1. *Maintain the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ collision free.*

Objective 2. *Maintain the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ regular.*

Other requirements not relevant for path feasibility, but still of interest for the application at hand, can be included as lower priority objectives ("soft constraints").

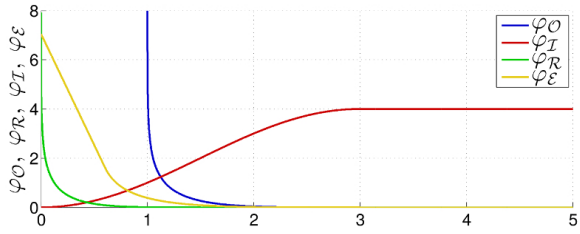


Figure 4. Example of the artificial potentials $\varphi_{\mathcal{O}}$, $\varphi_{\mathcal{R}}$ and $\varphi_{\mathcal{I}}$ used to compute $\boldsymbol{u}_a$, and of the potential $\varphi_{\mathcal{E}}$ that is used in Sec. 7.

As previously mentioned, here we consider the following qualitative requirement:

Secondary objective *Attract the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ towards those PoIs within a distance $R_{\mathcal{I}}$ from the path itself.*

In order to satisfy all the objectives we design $\boldsymbol{u}_a$ as the sum of three reactive terms

$$\boldsymbol{u}_a = \boldsymbol{u}_{a,\mathcal{O}}(\boldsymbol{x}, \mathcal{O}) + \boldsymbol{u}_{a,\mathcal{R}}(\boldsymbol{x}) + \boldsymbol{u}_{a,\mathcal{I}}(\boldsymbol{x}, \mathcal{R}). \tag{12}$$

The three terms in (12) are detailed in the rest of this section.

Objective 1: implementation We choose $\boldsymbol{u}_{a,\mathcal{O}}$ as

$$\boldsymbol{u}_{a,\mathcal{O}} = -\sum_{o\in\mathcal{O}} \int_S \left(\frac{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial \boldsymbol{x}}^{\dagger} \underbrace{\frac{\partial \varphi_{\mathcal{O}}(\|\boldsymbol{\gamma}(\boldsymbol{x},s)-\boldsymbol{o}\|)}{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}^{T}}_{\dot{\boldsymbol{p}}_{\boldsymbol{o}}(s)} \right)\Bigg|_{\boldsymbol{x}(t)} ds \tag{13}$$

where $\varphi_{\mathcal{O}} : \mathbb{R}_{\geq R_{\mathcal{O}}} \to \mathbb{R}_{\geq 0}$ is a smooth distance-based artificial potential function such that

$$\begin{aligned} \varphi_{\mathcal{O}} &= 0 \quad \text{if } \|\boldsymbol{\gamma}(\boldsymbol{x},s) - \boldsymbol{o}\| \geq \bar{R}_{\mathcal{O}} \\ \varphi_{\mathcal{O}} &\to \infty \quad \text{if } \|\boldsymbol{\gamma}(\boldsymbol{x},s) - \boldsymbol{o}\| \to R_{\mathcal{O}}^{+} \end{aligned}$$

where $\bar{R}_{\mathcal{O}} > R_{\mathcal{O}}$ and $R_{\mathcal{O}}^{+}$ indicates that the threshold $R_{\mathcal{O}}$ is approached from the right. An example of $\varphi_{\mathcal{O}}$ is depicted in Fig. 4.

The intuitive interpretation of (13) is that, for every obstacle $\boldsymbol{o} \in \mathcal{O}$, the artificial potential $\varphi_{\mathcal{O}}$ exerts on every point $\boldsymbol{\gamma}(\boldsymbol{x}(t), s)$ a repulsive velocity $-\dot{\boldsymbol{p}}_{\boldsymbol{o}}(s)$ directed away from the obstacle. The intensity of the repulsive velocity grows as $\boldsymbol{\gamma}(\boldsymbol{x}(t), s)$ approaches the boundary $R_{\mathcal{O}}$ of the obstacle sphere. This repulsive velocity is mapped onto the $\mathbb{R}^{2n}$ space of control points by the pseudo-inverse $\frac{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial \boldsymbol{x}}^{\dagger}$ which inverts the relation

$$-\dot{\boldsymbol{p}}_{\boldsymbol{o}}(s) = \frac{d\boldsymbol{\gamma}(\boldsymbol{x},s)}{dt} = \frac{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}},$$

where $\frac{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial s}\dot{s}$ does not appear because s in (13) is not a function of time. Finally, the line integral in (13) evaluates

the effect of the artificial potential over all the points of the path. From a practical standpoint, the analytical expression of (13) can be hard to determine, so that an efficient numerical evaluation of the integral may be used.

Objective 2: implementation We choose $\boldsymbol{u}_{a,\mathcal{R}}$ as

$$\boldsymbol{u}_{a,\mathcal{R}} = -\sum_{i=1}^{n} \int_{S_i} \frac{\partial \varphi_{\mathcal{R}}(\|\boldsymbol{x}_i - \boldsymbol{x}_i^\star(\boldsymbol{x},s)\|)}{\partial \boldsymbol{x}}^T \Bigg|_{\boldsymbol{x}(t)} ds \quad (14)$$

where S_i was introduced in Definition 2 and $\varphi_{\mathcal{R}} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a smooth distance-based artificial potential function such that

$$\begin{array}{ll} \varphi_{\mathcal{R}} = 0 & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_i^\star\| \geq R_{\mathcal{R}} \\ \varphi_{\mathcal{R}} \to \infty & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_i^\star\| \to 0^+. \end{array}$$

Furthermore, $\varphi_{\mathcal{R}}$ is strictly monotonic in $[0, R_{\mathcal{R}}]$. An example of $\varphi_{\mathcal{R}}$ is depicted in Fig. 4.

The action of potential $\varphi_{\mathcal{R}}$ in (14) is twofold. On one hand, it steers $\boldsymbol{x}_i$ away from the points $\boldsymbol{x}_i^\star$ forming the singular curve $\Omega_i(\boldsymbol{x})$ (see Definition 2). On the other hand, it steers the control points which determine the shape of $\Omega_i(\boldsymbol{x})$, so that $\Omega_i(\boldsymbol{x})$ moves away from $\boldsymbol{x}_i$. As in the previous case, for the implementation of (14), a numerical evaluation of the integral may be needed in practice.

Secondary objective: implementation We choose $\boldsymbol{u}_{a,\mathcal{I}}$ as

$$\boldsymbol{u}_{a,\mathcal{I}} = -\sum_{\boldsymbol{r}\in\mathcal{I}} \left(\frac{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial \boldsymbol{x}}^\dagger \underbrace{\frac{\partial \varphi_{\mathcal{I}}(\|\boldsymbol{\gamma}(\boldsymbol{x},s) - \boldsymbol{r}\|)}{\partial \boldsymbol{\gamma}(\boldsymbol{x},s)}^T}_{\dot{\boldsymbol{p}}_{\boldsymbol{r}}(\bar{s}_{\boldsymbol{r}})} \right) \Bigg|_{(\boldsymbol{x},\bar{s}_{\boldsymbol{r}})} \quad (15)$$

where $\bar{s}_{\boldsymbol{r}}$ indicates the coordinate of the point of $\boldsymbol{\gamma}_S(\boldsymbol{x})$ closest to $\boldsymbol{r}$, *i.e.*, $\bar{s}_{\boldsymbol{r}} = \operatorname{argmin}_{s\in S} \|\boldsymbol{\gamma}(\boldsymbol{x},s) - \boldsymbol{r}\|$, and $\varphi_{\mathcal{I}} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a smooth distance-based artificial potential function designed such that

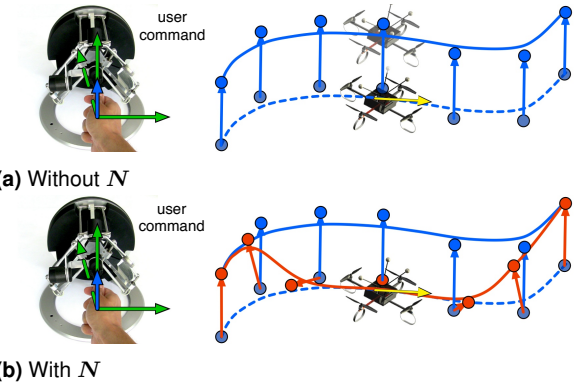
$$\begin{array}{ll} \varphi_{\mathcal{I}} = 0 & \text{if } \|\boldsymbol{\gamma}(\boldsymbol{x},s) - \boldsymbol{r}\| = 0 \\ \varphi_{\mathcal{I}} = U_{\mathcal{I}} > 0 & \text{if } \|\boldsymbol{\gamma}(\boldsymbol{x},s) - \boldsymbol{r}\| \geq R_{\mathcal{I}} \end{array}.$$

Furthermore, $\varphi_{\mathcal{I}}$ is strictly monotonic in $[0, R_{\mathcal{I}}]$ and it has zero slope at 0 and $R_{\mathcal{I}}$. An example of $\varphi_{\mathcal{I}}$ is depicted in Fig. 4. Unlike the potential functions $\varphi_{\mathcal{O}}$ and $\varphi_{\mathcal{R}}$, function $\varphi_{\mathcal{I}}$ has bounded slope because the secondary objective 5 is at lower priority w.r.t. Objectives 1–2.

The interpretation of (15) is similar to the one of $\boldsymbol{u}_{a,\mathcal{O}}$ (cfr. (13)), but in this case the potential $\varphi_{\mathcal{I}}$ exerts an attractive velocity with bounded intensity on the single point $\boldsymbol{\gamma}(\boldsymbol{x}, \bar{s}_{\boldsymbol{r}})$. From a practical point of view, $\bar{s}_{\boldsymbol{r}}$ in (15) can again be computed numerically.

6 Blending filter and objective fulfillment

The previous sections illustrated how the human's and autonomous control actions, *i.e.*, $\boldsymbol{u}_h$ and $\boldsymbol{u}_a$, are designed.



(a) Without $\boldsymbol{N}$

(b) With $\boldsymbol{N}$

Figure 5. User commanding a translation (blue arrow) to the path $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ (blue line). while the robot is traveling it with nonzero speed (yellow arrows). a) Without the blending filter, the travelled path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ follows exactly the command, but the resulting motion is unfeasible for the robot. b) When using the blending filter (16) and (17), the local geometric properties of $\boldsymbol{\gamma}_S(\boldsymbol{x})$ are preserved and the path translation does not incorrectly affect the instantaneous motion of the robot.

The next step is to blend these two actions and finally provide the evolution $\dot{\boldsymbol{x}}$ of the control points in (4). The most common approach to blend human's and autonomous controls is to sum the two signals (*e.g.* Franchi et al. 2012b). In this work we adopt a novel approach in order to satisfy a further objective described in the following. To this end, let $\bullet^{(k)}$ indicate the k-th derivative of a function w.r.t. time, *e.g.*, $\boldsymbol{x}^{(k)}(t) = \frac{d^k \boldsymbol{x}(t)}{d\, t^k}$.

Objective 3. *Suppose that an external algorithm provides a timing law $s(t) \in \mathcal{C}^k$ together with its first k derivatives, and denote with $\boldsymbol{p}(t) = \boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$ the trajectory tracked by the robot. The trajectory time derivatives $\dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t), \ldots, \boldsymbol{p}^{(k)}(t)$ must not be affected by the time derivatives of the curve parameters $\boldsymbol{x}(t)$ at the* current *$s(t)$.*

Objective 3 is particularly important for preventing that path modifications caused by $\boldsymbol{u}_h$ and $\boldsymbol{u}_a$ result in an unfeasible reference trajectory for the robot *at its current location on the curve*. This could happen, for example, if the operator abruptly steers the path sideways with respect to the current velocity of the robot (see Fig. 5a).

Furthermore, in order to exploit the differential flatness for the computation of the robot input commands, the signals $\dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t), \ldots, \boldsymbol{p}^{(k)}(t)$ must be available. Objective 3 allows to compute these signals without knowledge of the derivatives of $\boldsymbol{u}_h$ and $\boldsymbol{u}_a$ (which are not available) that would be required for evaluating $\ddot{\boldsymbol{x}}(t), \ldots, \boldsymbol{x}^{(k)}(t)$.

Another useful property resulting from Objective 3 is that the trajectory derivatives at a given time t only depends on the derivatives of $s(t)$ and not on the derivatives of

$\boldsymbol{x}(t)$. This allows to control the instantaneous reference motion of the robot by just choosing $s(t)$, regardless of any underlying path modification. For instance, by keeping $s(t)$ constant the robot would be commanded to remain still despite possible changes in $\boldsymbol{x}(t)$ given by the human operator or by autonomous corrections. Additionally, $s(t)$ can be modulated so as to travel the path with a desired cruise speed.

In order to achieve Objective 3 we design $\boldsymbol{N}$ in (4) as

$$\boldsymbol{N} = \boldsymbol{I}_{2n} - \boldsymbol{J}^{\dagger}\boldsymbol{J} \tag{16}$$

where $\boldsymbol{I}_{2n} \in \mathbb{R}^{2n\times 2n}$ is the identity matrix, $\boldsymbol{J}^{\dagger}$ indicates the Moore-Penrose pseudoinverse of $\boldsymbol{J} \in \mathbb{R}^{2k\times 2n}$, with $k < n$, and $\boldsymbol{J}$ is defined as

$$\boldsymbol{J}(\boldsymbol{x}(t), s(t)) = \left(\frac{\partial \boldsymbol{\gamma}}{\partial \boldsymbol{x}}^T \; \frac{\partial}{\partial \boldsymbol{x}}\left(\frac{\partial \boldsymbol{\gamma}}{\partial s}\right)^T \; \cdots \; \frac{\partial}{\partial \boldsymbol{x}}\left(\frac{\partial^k \boldsymbol{\gamma}}{\partial s^k}\right)^T \right)\Bigg|^T_{(\boldsymbol{x}(t),s(t))} \tag{17}$$

The Jacobian $\boldsymbol{J}$ relates variations of $\boldsymbol{x}$ to changes of local geometric properties of the path in $s(t)$, such as the position of the point $\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$, the tangent vector $\frac{\partial}{\partial s}\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$, the curvature vector $\frac{\partial^2}{\partial s^2}\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$, and so on.

Matrix $\boldsymbol{N}$ in (16) is the orthogonal projection matrix in the null-space of $\boldsymbol{J}$ (Chiaverini et al. 2008), *i.e.*, it is such that $\boldsymbol{J}\boldsymbol{N} = \mathbb{0}_{2k\times 2n}$. This property gives an intuitive interpretation to our choice of (16) and (17) and their effect on (4). Namely, it imposes the invariance of the *local* geometric properties of the path at the current location of the robot regardless of the *global* changes brought by $\boldsymbol{u}_h$ and $\boldsymbol{u}_a$ in (4), as illustrated in the example of Fig. 5b. Moreover, the following property holds:

Property 1. *Let $\boldsymbol{\gamma}(\boldsymbol{x}, \cdot)$ be a B-spline of order λ and suppose that the current value of the path coordinate is $s(t) \in [s_i, s_{i+1})$ with $i \in \{1, \dots, n-1\}$. The range space $\Im(\boldsymbol{J})$ of $\boldsymbol{J}$ has dimension $\dim(\Im(\boldsymbol{J})) \leq 2(\lambda+1)$ and the projection of $\boldsymbol{u}_h + \boldsymbol{u}_a$ through $\boldsymbol{N}$ is*

$$\boldsymbol{N}\left(\boldsymbol{u}_h + \boldsymbol{u}_a\right) = \begin{pmatrix} \boldsymbol{I}_{2(n-i-\lambda)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N}_\lambda & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{2(n-i)} \end{pmatrix} \left(\boldsymbol{u}_h + \boldsymbol{u}_a\right), \tag{18}$$

where $\boldsymbol{N}_\lambda \in \mathbb{R}^{2(\lambda+1)\times 2(\lambda+1)}$ is a projection matrix and $\boldsymbol{0}$ indicate matrices of zeros with suitable dimensions.

Proof. Provided in Appendix C .

Property 1 implies that the projection matrix $\boldsymbol{N}$ modifies only the velocity of the points $\boldsymbol{x}_{i-\lambda}, \dots, \boldsymbol{x}_i$ that (locally) control the shape of the path around $\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$, while the rest of the path follows exactly the corrections specified by $\boldsymbol{u}_h$ and $\boldsymbol{u}_a$ (as desired and expected).

We are now able to present our first important result which states that the proposed controller fulfills Objective 3.

Propositon 1. *If $\boldsymbol{N}$ in (4) is chosen as in (16) and (17), then the trajectory derivatives $\dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t), \dots, \boldsymbol{p}^{(k)}(t)$ are not functions of the time derivatives of $\boldsymbol{x}(t)$.*

Proof. The proof proceeds by expressing the trajectory derivatives $\dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}, \dots, \boldsymbol{p}^{(k)}$ under the assumption that condition $\boldsymbol{J}\dot{\boldsymbol{x}} = \mathbb{0}_{2n}$, with $\boldsymbol{J}$ defined in (17), is verified. Starting from $\boldsymbol{p}(t) = \boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$, the first derivative of the trajectory is

$$\begin{aligned} \tfrac{d\,\boldsymbol{\gamma}(\boldsymbol{x}(t),s(t))}{d\,t} &= \tfrac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,\boldsymbol{x}}\dot{\boldsymbol{x}} + \tfrac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,s}\dot{s} = 0 + \tfrac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,s}\dot{s} \\ &= \dot{\boldsymbol{p}}(\boldsymbol{x}(t), s(t), \dot{s}(t)) \end{aligned}$$

where $\frac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,\boldsymbol{x}}\dot{\boldsymbol{x}} = 0$ results from the initial assumption. Applying the chain rule, the second derivative is

$$\begin{aligned} \tfrac{d^2\,\boldsymbol{\gamma}(\boldsymbol{x}(t),s(t))}{d\,t^2} &= \underbrace{\tfrac{\partial}{\partial\,\boldsymbol{x}}\left(\tfrac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,s}\dot{s}\right)\dot{\boldsymbol{x}}}_{=(0\;0)^T} + \sum_{j=1}^{2} \tfrac{\partial}{\partial s^{(j-1)}}\left(\tfrac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,s}\dot{s}\right) s^{(j)} \\ &= \ddot{\boldsymbol{p}}(\boldsymbol{x}(t), s(t), \dot{s}(t), \ddot{s}(t)) \end{aligned}$$

where it was imposed $\frac{\partial}{\partial\,\boldsymbol{x}}\left(\frac{\partial\,\boldsymbol{\gamma}(\boldsymbol{x},s)}{\partial\,s}\right)\dot{\boldsymbol{x}} = 0$ from the initial assumption. When iterating the chain rule for computing higher order trajectory derivatives, at each step the initial assumption $\boldsymbol{J}\dot{\boldsymbol{x}} = \mathbb{0}_{2n}$ annihilates the terms containing the partial derivatives w.r.t. $\boldsymbol{x}$, and the i-th derivative, with $1 \leq i \leq k$, becomes

$$\begin{aligned} \tfrac{d^i\,\boldsymbol{\gamma}(\boldsymbol{x}(t),s(t))}{d\,t^i} &= \underbrace{\tfrac{\partial}{\partial\boldsymbol{x}}\left(\boldsymbol{p}^{(i-1)}(\boldsymbol{x}, s, \dot{s}, \dots, s^{(i-1)})\right)\dot{\boldsymbol{x}}}_{=(0\;0)^T} \\ &\quad + \sum_{j=1}^{i} \tfrac{\partial}{\partial s^{(j-1)}}\left(\boldsymbol{p}^{(i-1)}(\boldsymbol{x}, s, \dot{s}, \dots, s^{(i-1)})\right) s^{(j)} \\ &= \boldsymbol{p}^{(i)}(\boldsymbol{x}(t), s(t), \dot{s}(t), \dots, s^{(i)}(t)) \end{aligned} \tag{19}$$

Having proven that the blending filter $\boldsymbol{N}$ satisfies Objective 3, we can now show that the dynamic system (4) with $\boldsymbol{N}$ chosen as in (16) accomplishes also Objectives 1 and 2.

Propositon 2. *Suppose that $\boldsymbol{u}_h$ is bounded and that $\|\boldsymbol{u}_h + \boldsymbol{u}_{a,\mathcal{I}}\| \leq \bar{u}$. Then, $\boldsymbol{\gamma}_S(\boldsymbol{x})$ remains collision free and regular.*

Proof. The proof relies on the structure of B-splines that is described in Appendix D and it uses an argument similar to the one adopted by Lee et al. (2011, 2013) in the proof of their Proposition 1. Consider the energy function

$$\begin{aligned} V(t) := &\sum_{\boldsymbol{o}\in\mathcal{O}} \int_S \frac{1}{k(s)} \varphi_{\mathcal{O}}(\|\boldsymbol{\gamma}(\boldsymbol{x}, s) - \boldsymbol{o}\|) ds + \\ &\sum_{i=1}^{n} \int_{s_{i-\lambda}}^{s_i} \varphi_{\mathcal{R}}(\|\boldsymbol{x}_i - \boldsymbol{x}_i^{\star}(\boldsymbol{x}, s)\|) ds \end{aligned} \tag{20}$$

where the dependence on time is in $\boldsymbol{x}(t)$, and $k(s) = \sum_{i=1}^{n} B_i^2(\boldsymbol{s}, s) > 0$ since the basis functions B_i are positive. Therefore $V(t) \geq 0$ and, since i) $\varphi_{\mathcal{O}} \to \infty$ iff the path is approaching a collision, and ii) $\varphi_{\mathcal{R}} \to \infty$ iff the path is becoming non regular, proving the boundedness of $V(t)$ implies that the path remains regular and collision free. As a starting condition, by hypothesis, we have that the path is initially regular and collision free, *i.e.*, $V(0)$ is finite.

The time derivative of $V(t)$ is

$$\dot{V}(t) = \underbrace{\left(\sum_{\boldsymbol{o} \in \mathcal{O}} \int_{S} \frac{\partial \varphi_{\mathcal{O}}}{\partial \boldsymbol{\gamma}} \frac{\boldsymbol{B_s}}{k(s)} ds + \sum_{i=1}^{n} \int_{s_{i-\lambda}}^{s_i} \frac{\partial \varphi_{\mathcal{R}}}{\partial \boldsymbol{x}}^T ds \right)}_{\boldsymbol{w}^T = \frac{\partial V}{\partial \boldsymbol{x}}} \dot{\boldsymbol{x}} \tag{21}$$

where we used the facts that $\frac{\partial \boldsymbol{\gamma}(\boldsymbol{x}, s)}{\partial \boldsymbol{x}} = \boldsymbol{B_s}(s)$, $\dot{\boldsymbol{o}} = 0$ (*i.e.*, static obstacles), and that $k(s)$ is not a function of time because the knots $\boldsymbol{s}$ are fixed (see Sec. 2).

By injecting (4) and (12) in (21) we have

$$\dot{V}(t) = \boldsymbol{w}^T \boldsymbol{N} \Big(\underbrace{\boldsymbol{u}_h + \boldsymbol{u}_{a,\mathcal{I}}}_{\boldsymbol{v}} - \underbrace{(-\boldsymbol{u}_{a,\mathcal{O}} - \boldsymbol{u}_{a,\mathcal{R}})}_{\tilde{\boldsymbol{w}}} \Big). \tag{22}$$

By comparing (21) to (13) and (14) it is clear that $\tilde{\boldsymbol{w}}$ differs from $\boldsymbol{w}$ only because in $\boldsymbol{u}_{a,\mathcal{O}}$ the pseudoinverse $\boldsymbol{B_s}(s)^\dagger$ replaces the term $\frac{\boldsymbol{B_s}(s)^T}{k(s)}$. However, from (45) and from the definition of $k(s)$ one has

$$\boldsymbol{B_s}(s)^\dagger = (\boldsymbol{B_s}(s)^T \boldsymbol{B_s}(s))^{-1} \boldsymbol{B_s}(s)^T = \frac{\boldsymbol{B_s}(s)^T}{k(s)}, \tag{23}$$

hence $\tilde{\boldsymbol{w}} = \boldsymbol{w}$ and (22) becomes

$$\dot{V}(t) = -\boldsymbol{w}^T \boldsymbol{N} \boldsymbol{w} + \boldsymbol{w}^T \boldsymbol{N} \boldsymbol{v} \tag{24}$$

We also proved in (42) (Appendix C) that $\boldsymbol{N}$ has a particular structure, and this can be exploited (after a possible row rearrangement) for rewriting (24) as

$$\dot{V}(t) = -(\boldsymbol{w}_1^T \, \boldsymbol{w}_2^T) \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{N}_\lambda \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{pmatrix} + (\boldsymbol{w}_1^T \, \boldsymbol{w}_2^T) \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{N}_\lambda \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} \tag{25}$$

where $\boldsymbol{w} = (\boldsymbol{w}_1^T \, \boldsymbol{w}_2^T)^T$ and $\boldsymbol{v} = (\boldsymbol{v}_1^T \, \boldsymbol{v}_2^T)^T$ are the partitions corresponding to the block diagonal structure of $\boldsymbol{N}$. Matrix $\boldsymbol{N}_\lambda \in \mathbb{R}^{2(\lambda+1) \times 2(\lambda+1)}$ is a projection matrix and, thus, it only has 0 and 1 as eigenvalues and can be diagonalized as $\boldsymbol{N}_\lambda = \boldsymbol{H} \Lambda \boldsymbol{H}^T$, where $\boldsymbol{H}$ and $\boldsymbol{H}^T$ are the right and left eigenvectors[1] of $\boldsymbol{N}_\lambda$ and Λ is the diagonal matrix of the eigenvalues. Without loss of generality, we assume that the first elements of the diagonal of Λ are the 1 eigenvalues and we indicate $\Lambda = \mathrm{diag}(\Lambda_1 \, \Lambda_0)$. With the change of coordinates $\boldsymbol{H}^T \boldsymbol{w}_2 = (\boldsymbol{w}_{\Lambda_1}^T \, \boldsymbol{w}_{\Lambda_0}^T)$ and $\boldsymbol{H}^T \boldsymbol{v}_2 = (\boldsymbol{v}_{\Lambda_1}^T \, \boldsymbol{v}_{\Lambda_0}^T)$, we can then write (25) as

$$\begin{aligned} \dot{V}(t) = & -(\boldsymbol{w}_1^T \, \boldsymbol{w}_{\Lambda_1}^T \, \boldsymbol{w}_{\Lambda_0}^T) \begin{pmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \Lambda_1 & 0 \\ 0 & 0 & \Lambda_0 \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_{\Lambda_1} \\ \boldsymbol{w}_{\Lambda_0} \end{pmatrix} + \\ & (\boldsymbol{w}_1^T \, \boldsymbol{w}_{\Lambda_1}^T \, \boldsymbol{w}_{\Lambda_0}^T) \begin{pmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \Lambda_1 & 0 \\ 0 & 0 & \Lambda_0 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_{\Lambda_1} \\ \boldsymbol{v}_{\Lambda_0} \end{pmatrix} \\ = & -\boldsymbol{w}_{\mathfrak{F}}^T \boldsymbol{w}_{\mathfrak{F}} + \boldsymbol{w}_{\mathfrak{F}}^T \boldsymbol{v}_{\mathfrak{F}} \leq -\|\boldsymbol{w}_{\mathfrak{F}}\|^2 + \|\boldsymbol{w}_{\mathfrak{F}}\| \bar{u} \end{aligned} \tag{26}$$

where $\boldsymbol{w}_{\mathfrak{F}} = (\boldsymbol{w}_1^T \, \boldsymbol{w}_{\Lambda_1}^T)^T$ and $\boldsymbol{v}_{\mathfrak{F}} = (\boldsymbol{v}_1^T \, \boldsymbol{v}_{\Lambda_1}^T)^T$ are the components of $\boldsymbol{w}$ and $\boldsymbol{v}$ that are not annihilated by the null space of $\boldsymbol{N}$ and where we used the assumption $\|\boldsymbol{v}_{\mathfrak{F}}\| \leq \|\boldsymbol{v}\| \leq \bar{u}$ from the statement of Proposition 2.

Since $\varphi_{\mathcal{O}}$ and $\varphi_{\mathcal{R}}$ are unbounded with unbounded gradients, we can always find a finite value $M > V(0)$ such that, when $V(t) \geq M$ and if $\|\boldsymbol{w}_{\mathfrak{F}}\| \neq 0$ then $\|\boldsymbol{w}_{\mathfrak{F}}\| \geq \bar{u}$. Suppose now that $V(t) = M$, then from the previous consideration $\dot{V}(t) \leq 0$, thus showing that $V(t) \leq M \; \forall t \geq 0$ and therefore proving that the path remains collision free and regular.

We remark that the proof of Proposition 2 exploits the assumption of static obstacles in the derivation of (21): this is due to the fact that a reactive path correction cannot easily handle moving obstacles in a proper way, as noted also by Brock and Khatib (2002). Indeed, one can think of the situation in which two obstacles are advancing towards the same point of the path but from opposite directions: their repulsive forces would then cancel each other out, thus failing the collision avoidance objective. The null-space projection blending mechanism is another factor that limits the applicability to moving obstacles because, by forbidding path transformations at the current location of the robot, it becomes impossible to react to an obstacle that is approaching towards the exact location of the robot. In general both these problems could be solved by extending the present framework with 1. a replanning strategy that initializes a new path when obstacles get too close to a 'dangerous situation', and 2. a timing law planner that modulates the robots traveling speed along the path, *e.g.*, by accelerating/decelerating in order to avoid an incoming object.

Because of its non-triviality, the explicit inclusion of moving obstacles in our framework will be considered in future extensions of this work. In this sense the replanning strategy presented in the next section, even though developed and tested only for the case of static obstacles, can be considered as a first step in this direction.

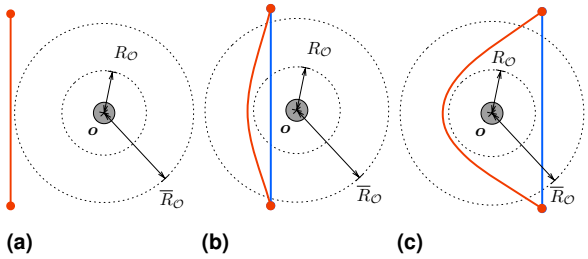


Figure 6. a) to c): Sequence showing the deformation of a path $\gamma_S(\boldsymbol{x})$ (red curve) with respect to the desired path $\gamma_S(\boldsymbol{x}_h)$ (blue line) that is moved through an obstacle (from left to right). In the end, the deformed path becomes a suboptimal w.r.t., *e.g.*, a straight line.

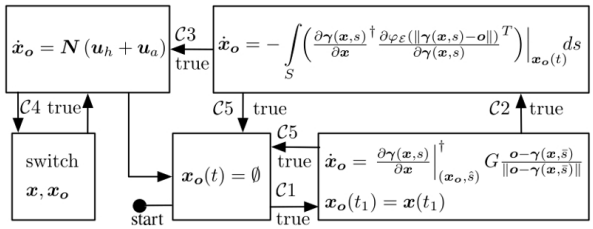


Figure 7. Block representation of the replanning algorithm. The algorithm starts from the central block and proceeds as conditions $\mathcal{C}1$ to $\mathcal{C}5$ are met.

7 Path reinitialization in presence of obstacles

The autonomous correction described in Sec. 5 ensures that the travelled path is collision-free but it does not allow it to 'pass through an obstacle' in case of particularly cluttered environments. This limitation, which can lead to suboptimal paths w.r.t. the one specified by the human (see Fig. 6 for an example), is inherently due to the reactive (local) nature of the proposed planning strategy. In order to overcome this problem we introduce a strategy for generating new *alternative* paths in presence of obstacles. Our strategy is based on the well-known concept of homotopy: given an obstacle $\boldsymbol{o}$ and a collision-free path $\gamma_S(\boldsymbol{x})$ between two points (or portion of a path), we can find another vector of control points $\boldsymbol{x_o} \in \mathbb{R}^{2n}$ such that $\gamma_S(\boldsymbol{x_o})$ is also collision-free, it has the same endpoints of $\gamma_S(\boldsymbol{x})$, and it is *non-homotopic* (LaValle 2006) to $\gamma_S(\boldsymbol{x})$. Namely, $\gamma_S(\boldsymbol{x_o})$ cannot be continuously morphed into $\gamma_S(\boldsymbol{x})$ without intersecting $\boldsymbol{o}$. Following this insight, our strategy to avoid suboptimal paths in presence of obstacles is to continuously morph the current path $\gamma_S(\boldsymbol{x})$ so as to produce a new non-homotopic path $\gamma_S(\boldsymbol{x_o})$.

For each obstacle $\boldsymbol{o} \in \mathcal{O}$, the generation of $\boldsymbol{x_o}$ is articulated in four steps, named *Crossing*, *Expansion*, *Activation* and *Stop*. The overall algorithm, depicted in Fig. 7, is explained hereinafter. Before proceeding with the details, we introduce one function that simplifies the following explanation:

$$\begin{aligned} \boldsymbol{d} : \mathbb{R}^2 \times \mathbb{R}^{2n} \times S &\to \mathbb{R}^2 \\ \boldsymbol{o}, \boldsymbol{x}, s &\mapsto \boldsymbol{d}(\boldsymbol{o}, \boldsymbol{x}, s) = \boldsymbol{o} - \gamma_S(\boldsymbol{x}, s) \end{aligned} \tag{27}$$

Crossing The algorithm starts when the maximum repulsion applied by $\boldsymbol{o}$ to $\gamma_S(\boldsymbol{x})$ becomes greater than a predefined threshold $\overline{F} > 0$. Since $\varphi_\mathcal{O}$ is a distance based potential this condition is expressed as

$$(\textit{Cond. } \mathcal{C}1) \quad \begin{cases} \left\| \frac{\partial \varphi_\mathcal{O}(\|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},\bar{s})\|)}{\partial \gamma(\boldsymbol{x},\bar{s})} \right\| \geq \overline{F} \\ \text{s.t. } \bar{s} = \operatorname{argmin}_{s \in S} \|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},s)\| \end{cases} \tag{28}$$

When condition $\mathcal{C}1$ becomes true at some time $t = t_1$, a copy $\gamma_S(\boldsymbol{x_o})$ of $\gamma_S(\boldsymbol{x})$ is created and the morphing starts. The morphing is performed in an intuitive way by 'pulling' the new curve from a single point to bring it on the other side of the obstacle (see Fig. 8a). The point to be pulled is denoted by $\gamma(\boldsymbol{x_o}, \hat{s})$ and it is taken as the intersection between $\gamma_S(\boldsymbol{x_o})$ and $\boldsymbol{d}$. As for the pulling action, this is implemented by the system

$$\dot{\boldsymbol{x}}_o = \frac{\partial \gamma(\boldsymbol{x},s)}{\partial \boldsymbol{x}}\Big|^\dagger_{(\boldsymbol{x}_o,\hat{s})} G \frac{\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},\bar{s})}{\|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},\bar{s})\|}, \quad \boldsymbol{x_o}(t_1) = \boldsymbol{x}(t_1) \tag{29}$$

where $G > 0$ determines the strength of the pulling force and $\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},s)/\|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},s)\|$ is the direction of the pulling force.

Expansion System (29) remains active until $\gamma_S(\boldsymbol{x_o})$ becomes non-homotopic to $\gamma_S(\boldsymbol{x})$ w.r.t. $\boldsymbol{o}$, *i.e.*,

$$(\textit{Cond. } \mathcal{C}2) \quad \frac{\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},\bar{s})^T(\gamma(\boldsymbol{x_o},\hat{s}) - \gamma(\boldsymbol{x},\bar{s}))}{\|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},\bar{s})\|^2} \geq 1 + F_c \tag{30}$$

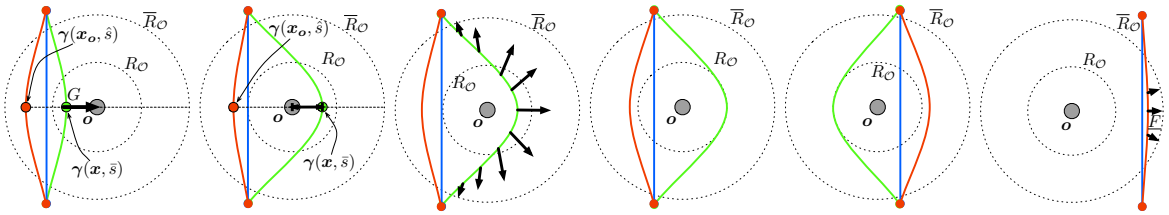
where $F_c > 0$ is a user defined threshold (see Fig. 8b). Once (30) becomes true, a repulsive action 'pushes' $\gamma_S(\boldsymbol{x_o})$ outside the obstacle ball centered in $\boldsymbol{o}$ (see Fig. 8c), *i.e.*,

$$\dot{\boldsymbol{x}}_o = -\int_S \left(\frac{\partial \gamma(\boldsymbol{x},s)}{\partial \boldsymbol{x}}^\dagger \frac{\partial \varphi_\mathcal{E}(\|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x},s)\|)}{\partial \gamma(\boldsymbol{x},s)}^T \right)\Big|_{(\boldsymbol{x}_o,s)} ds \tag{31}$$

where $\varphi_\mathcal{E} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a smooth distance-based artificial potential function, that is strictly monotonic in $[0, \bar{R}_\mathcal{O}]$ and such that

$$\begin{aligned} \varphi_\mathcal{E} &= 0 && \text{if } \|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x_o},s)\| \geq \bar{R}_\mathcal{O} \\ \varphi_\mathcal{E} &\to U > 0 && \text{if } \|\boldsymbol{d}(\boldsymbol{o},\boldsymbol{x_o},s)\| \to 0^+ \end{aligned}$$

An example of $\varphi_\mathcal{E}$ is depicted in Fig. 4.



(a) Crossing. **(b)** $\mathcal{C}2 =$ true. **(c)** Expansion. **(d)** Activation. **(e)** Switch ($\mathcal{C}4$). **(f)** $\mathcal{C}5 =$ true.

Figure 8. Generation of an alternative path: $\boldsymbol{\gamma}_S(\boldsymbol{x_o})$ (green line), $\boldsymbol{\gamma}_S(\boldsymbol{x})$ (red line), $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ (blue line), obstacle $\boldsymbol{o}$ (gray disc). From a) to f), $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is moving from left to right, passing over the obstacle.

Activation Once $\boldsymbol{\gamma}_S(\boldsymbol{x_o})$ is collision free, *i.e.*,

$$(\textit{Cond. } \mathcal{C}3) \quad \min_{s\in S} \|\boldsymbol{d}(\boldsymbol{o}, \boldsymbol{x_o}, s)\| > R_{\mathcal{O}} \tag{32}$$

the evolution of $\boldsymbol{x_o}$ changes to (4)[2] and $\boldsymbol{\gamma}_S(\boldsymbol{x_o})$ is available as an alternative route (see Fig. 8d).

Path $\boldsymbol{\gamma}_S(\boldsymbol{x_o})$ becomes active (*i.e.*, $\boldsymbol{x}$ and $\boldsymbol{x_o}$ are switched) only if

$$(\textit{Cond. } \mathcal{C}4) \quad \begin{cases} \|\boldsymbol{x_o} - \boldsymbol{x}_h\| < \|\boldsymbol{x} - \boldsymbol{x}_h\| \\ \|\boldsymbol{\gamma}(\boldsymbol{x}, s(t)) - \boldsymbol{\gamma}(\boldsymbol{x_o}, s(t))\| \simeq 0 \\ \vdots \\ \left\| \dfrac{d^k\, \boldsymbol{\gamma}(\boldsymbol{x}, s(t))}{d\, t^k} - \dfrac{d^k\, \boldsymbol{\gamma}(\boldsymbol{x_o}, s(t))}{d\, t^k} \right\| \simeq 0 \end{cases} . \tag{33}$$

Condition $\mathcal{C}4$ requires that i) $\boldsymbol{x_o}$ is closer to $\boldsymbol{x}_h$ than $\boldsymbol{x}$, and ii) the change from $\boldsymbol{x}$ to $\boldsymbol{x_o}$ causes a negligible discontinuity in the trajectory tracked by the robot. The switch to the new path is depicted in Fig. 8e.

Stop At any point during its execution the algorithm is stopped and $\boldsymbol{\gamma}_S(\boldsymbol{x_o})$ is deleted if the current path is steered away from the obstacle, *i.e.*, if

$$(\textit{Cond. } \mathcal{C}5) \quad \max_{s\in S} \left\| \frac{\partial \varphi_{\mathcal{O}}(\|\boldsymbol{d}(\boldsymbol{o}, \boldsymbol{x_o}, s)\|)}{\partial \boldsymbol{\gamma}(\boldsymbol{x}, s)} \right\| \leq \underline{F} \tag{34}$$

where $0 < \underline{F} < \overline{F}$ (see Fig. 8f).

The algorithm has been discussed for a single obstacle but it generalizes to multiple obstacles. By taking into account all the $n_{\mathcal{O}}$ obstacles in $\mathcal{O}$, there could be up to $2^{n_{\mathcal{O}}} - 1$ new paths non-homotopic to $\boldsymbol{\gamma}_S(\boldsymbol{x})$ and to each other. In practice, we limited the number of alternative paths considered at once to one per obstacle. Although not complete, this solution resulted very effective in practice. Simulation results are shown in Sec. 9.

8 Haptic feedback

The haptic feedback algorithm computes the force $\boldsymbol{\tau}$ rendered by the input device (cfr. (5)) in order to inform the operator about the discrepancies between the desired path $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ and the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ travelled by the robot. Recalling that the human guidance has been designed as a system with memory (see Sec. 4), we choose $\boldsymbol{\tau}$ as the combination of two haptic cues, $\boldsymbol{e}_{\dot{\boldsymbol{x}}}$ and $\boldsymbol{e}_{\boldsymbol{x}}$: the first provides a feedback indicating how well the control points velocity $\dot{\boldsymbol{x}}$ tracks the instantaneous command $\dot{\boldsymbol{x}}_h$ specified by (6), and the second provides an information of the mismatch between $\boldsymbol{x}$ and $\boldsymbol{x}_h$. The reason for introducing the second feedback term is that the simple memoryless feedback on the velocity (*i.e.*, $\boldsymbol{e}_{\dot{\boldsymbol{x}}}$) does not provide any information about the difference between the travelled path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ and the commanded path $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ which can, instead, be of relevance for the operator's awareness. The two terms are detailed hereinafter.

$\boldsymbol{e}_{\dot{\boldsymbol{x}}}$ The haptic cue $\boldsymbol{e}_{\dot{\boldsymbol{x}}}$ represents the mismatch between $\dot{\boldsymbol{x}}_h$ itself and the actual velocity $\dot{\boldsymbol{x}}$. This mismatch is obtained by first mapping $\dot{\boldsymbol{x}}_h$ and $\dot{\boldsymbol{x}}$ back onto the space of input device configurations and then by taking their difference, *i.e.*,

$$\begin{aligned} \boldsymbol{e}_{\dot{\boldsymbol{x}}} &= \boldsymbol{Q}(\boldsymbol{x}_h)^{\dagger}\dot{\boldsymbol{x}}_h - \boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}} \\ &= \boldsymbol{K}\boldsymbol{q} - \boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}} \end{aligned} \tag{35}$$

where $\boldsymbol{Q}(\cdot)^{\dagger} = (\boldsymbol{Q}(\cdot)^T\, \boldsymbol{Q}(\cdot))^{-1}\boldsymbol{Q}(\cdot)^T$. Observe that:

Remark 3. *$\boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}}$ is the mapping onto the space of input device configurations of the orthogonal projection of $\dot{\boldsymbol{x}}$ on the range space of $\boldsymbol{Q}(\boldsymbol{x})$.*

Proof. The statement is a simple result of linear algebra (Meyer, 2000). The well known orthogonal projection operator onto the range space of $\boldsymbol{Q}(\boldsymbol{x})$ is the matrix $\boldsymbol{Q}(\boldsymbol{x})(\boldsymbol{Q}(\boldsymbol{x})^T\, \boldsymbol{Q}(\boldsymbol{x}))^{-1}\boldsymbol{Q}(\boldsymbol{x})^T \equiv \boldsymbol{Q}(\boldsymbol{x})\boldsymbol{Q}(\boldsymbol{x})^{\dagger}$, therefore the orthogonal projection of $\dot{\boldsymbol{x}}$ on the range space of $\boldsymbol{Q}(\boldsymbol{x})$ is $\dot{\boldsymbol{x}}_{proj} = \boldsymbol{Q}(\boldsymbol{x})\boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}}$. The mapping of $\dot{\boldsymbol{x}}_{proj}$ on the space of input device configurations is $\boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}}_{proj} = \boldsymbol{Q}(\boldsymbol{x})^{\dagger}\boldsymbol{Q}(\boldsymbol{x})\boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}} = \boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}}$.

The meaning of Remark 1 is that, when $\dot{\boldsymbol{x}}$ cannot be exactly traced back to a human input, then the projection

$\boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}}$ yields the command whose outcome is the closest possible to $\dot{\boldsymbol{x}}$.

$\boldsymbol{e_x}$ The haptic cue $\boldsymbol{e_x}$ represents the mismatch between $\boldsymbol{x}_h$ itself and $\boldsymbol{x}$. In particular, we want the force originated from $\boldsymbol{e_x}$ to guide the operator in steering $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ towards $\boldsymbol{\gamma}_S(\boldsymbol{x})$. This result can be achieved by implementing $\boldsymbol{e_x}$ as

$$\boldsymbol{e_x} = k\boldsymbol{Q}(\boldsymbol{x}_h)^{\dagger}(\boldsymbol{x}_h - \boldsymbol{x}). \tag{36}$$

with $k > 0$. As in (35), $\boldsymbol{Q}(\boldsymbol{x}_h)^{\dagger}$ maps $k(\boldsymbol{x}_h - \boldsymbol{x})$ onto the space of input device configurations.

Finally, the rendered force $\boldsymbol{\tau}$ corresponding to the two haptic cues $\boldsymbol{e}_{\dot{\boldsymbol{x}}}$ and $\boldsymbol{e_x}$ is

$$\boldsymbol{\tau} = -\boldsymbol{B}\dot{\boldsymbol{q}} - \boldsymbol{K}_M\boldsymbol{q} - \boldsymbol{K}^*(\boldsymbol{e}_{\dot{\boldsymbol{x}}} + \boldsymbol{e_x}) \tag{37}$$

where $\boldsymbol{B}$ is a positive definite damping matrix used to stabilize the device, $\boldsymbol{K}^*$ is a diagonal positive definite matrix of gains and $\boldsymbol{K}_M$ is a diagonal non-negative matrix used to provide a perception of the distance from the zero-commanded velocity. Note that the effect of $\boldsymbol{K}_M$, when not desired, can be disabled by taking $\boldsymbol{K}_M = 0$. As in all bilateral teleoperation applications, the presence of the force feedback $\boldsymbol{\tau}$ may cause unstable behaviors of the haptic interface because of non-modeled dynamics, communication delays and packet losses. In order to guarantee stability despite all these shortcomings we make use of the *passive set-position modulation* (PSPM) approach from Lee and Huang (2010), a very general and flexible framework for guaranteeing stability (passivity) of the master side and of the closed-loop teleoperation system. Let $\bar{\boldsymbol{z}}[k]$ be the PSPM version of the following signal

$$\boldsymbol{z} = \boldsymbol{Q}(\boldsymbol{x})^{\dagger}\dot{\boldsymbol{x}} - k\boldsymbol{Q}(\boldsymbol{x}_h)^{\dagger}(\boldsymbol{x}_h - \boldsymbol{x}), \tag{38}$$

that is sampled and sent from the mobile robot to the haptic interface through the (possibly non-ideal) communication channel. Exploiting the PSPM action, the final *passive* implementation of $\boldsymbol{\tau}$ in (37) then becomes

$$\boldsymbol{\tau} = -\boldsymbol{B}\dot{\boldsymbol{q}} - \boldsymbol{K}_M\boldsymbol{q} - \boldsymbol{K}^*\left(\boldsymbol{K}\boldsymbol{q} - \bar{\boldsymbol{z}}[k]\right). \tag{39}$$

This is sufficient for guaranteeing stability (passivity) of the bilateral system assuming that the human operator behaves as a passive system (see Lee and Huang 2010, for more details).

9 Human/hardware-in-the-loop simulations

The working principles of the proposed framework have been demonstrated in several simulations human/hardware-in-the-loop simulation framework. The simulation setup consists of three software components: i) TeleKyb (Grabe et al. 2013), an open source software framework for the development of (bilateral) teleoperation systems between human interfaces (*e.g.*, haptic force feedback devices or gamepads) and groups of quadrotor Unmanned Aerial Vehicles (UAVs); ii) SwarmSimX (Lächele et al. 2012), a graphical and physical real-time simulation environment; iii) Simulink™. The simulations feature a physically simulated quadrotor UAV and B-spline paths of degree $\lambda = 5$. Two haptic devices, an Omega.6 and a Phantom Omni have been used as input interfaces for the human operator. For a better appreciation of these simulation results, we encourage the reader to see Extension 1.

Simulation 1 The purpose of this simulation is to demonstrate the effect of the filter $\boldsymbol{N}$ in (4) by letting the quadrotor travel a path whose reference $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ is a straight line that is (purposely) moved sideways by the operator (translational command (9)). The simulation is repeated twice, once implementing exactly (4) (snapshot in Fig. 9a) and once removing the term $\boldsymbol{N}$ from (4) (snapshot in Fig. 9b). The same operator's commands are used in both cases. Visual inspection of Figs. 9a and 9b shows that:

1. Without $\boldsymbol{N}$, the path modification commanded by the user is implemented exactly, *i.e.*, the reference path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is identical to $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$, but the robot cannot follow it.

2. With $\boldsymbol{N}$, the path modification commanded by the user is altered, *i.e.*, $\boldsymbol{\gamma}_S(\boldsymbol{x})$ differs from $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$, and thanks to this change the robot can follow the reference quite precisely.

The different behaviour in the two cases is also clear in Figs. 9c and 9d, where the trajectory $\boldsymbol{\gamma}_S(\boldsymbol{x})$ in four different moments ($t = 0,\ 2,\ 2.5,\ 3$s) is depicted, together with the corresponding reference trajectory of the robots. We see in Fig. 9d that without $\boldsymbol{N}$ the reference path remains a straight line but the robot has to perform a very large lateral motion in order to follow the commanded path. On the other hand, in Fig. 9c we observe that the term $\boldsymbol{N}$ modifies the reference path near to the robot current location, thus limiting considerably the lateral excursion required to travel along the commanded path. This visual analysis is also confirmed by the plots of the lateral velocity and tilt angle shown in Figs. 9e and 9f for the two conditions, respectively. Lastly, the more demanding and abruptly varying trajectory in the absence of the term $\boldsymbol{N}$ is also reflected by the larger tracking error of the reference $\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$, as shown by Figs. 9g and 9h. The results of this simulation show that the filter $\boldsymbol{N}$ is able to generate a reference path trajectory compliant with the robot actuation capabilities, by in particular eliminating the possibly large overshoots caused by abrupt user's commands.

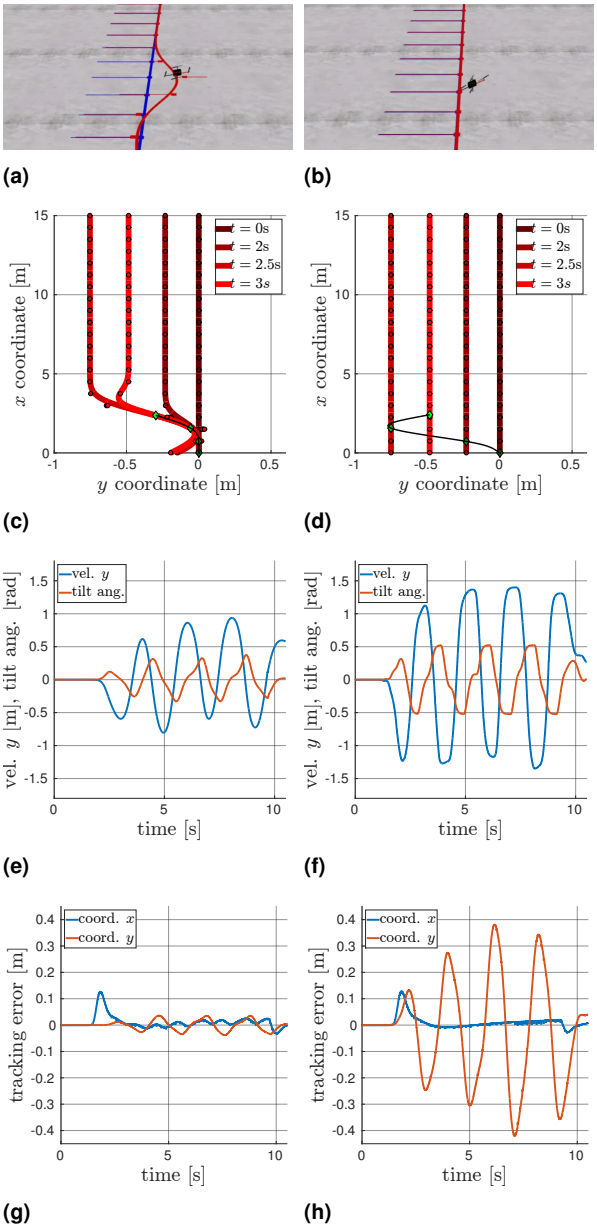
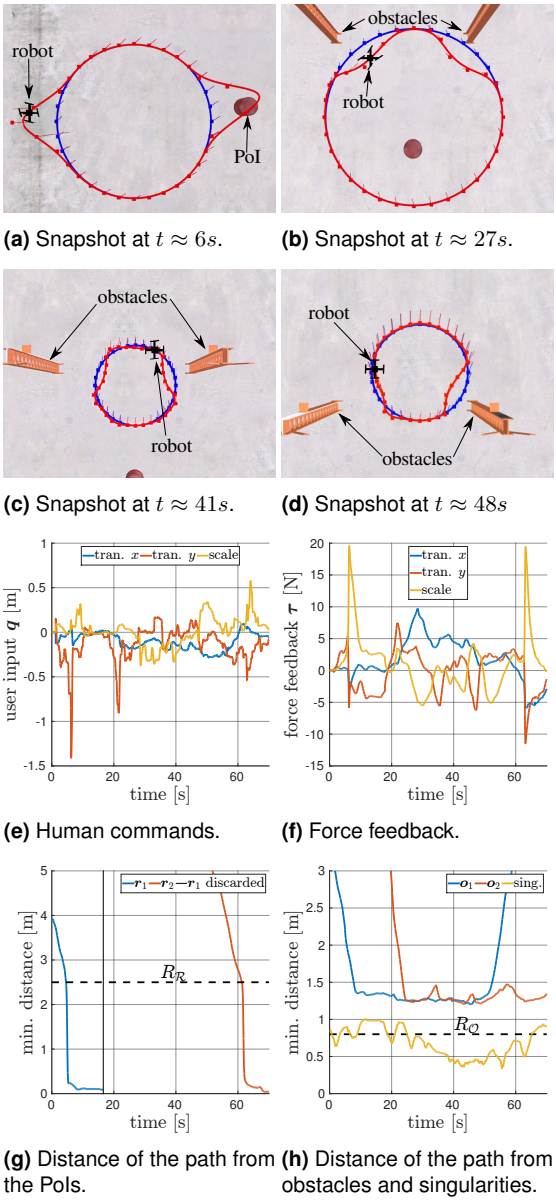


Figure 9. Simulation 1. (a), (c), (e), (g) refer to the case with $\boldsymbol{N}$. (b), (d), (f), (h) refer to the case without $\boldsymbol{N}$. (a) and (b): snapshots of the simulation showing $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ (blue thick line), $\boldsymbol{\gamma}_S(\boldsymbol{x})$ (red thick line), $\boldsymbol{x}_h$ (blue squares), $\boldsymbol{x}$ (red squares), $\dot{\boldsymbol{x}}_h$ (thin blue lines) and $\dot{\boldsymbol{x}}$ (thin red lines). (c) and (d): illustration of $\boldsymbol{\gamma}_S(\boldsymbol{x})$ in four moments. The reference position $\boldsymbol{\gamma}(\boldsymbol{x}, s)$ in these moments is denoted by the green markers. The black line shows the reference trajectory followed by the robot. (e) and (f): lateral velocity and tilt angle of the robot. (g) and (h): tracking error (robot's position - reference position).



(a) Snapshot at $t \approx 6s$. **(b)** Snapshot at $t \approx 27s$.

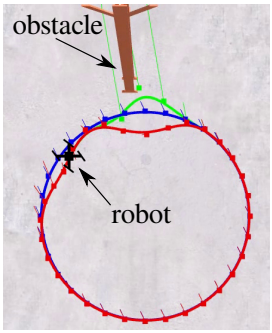
(c) Snapshot at $t \approx 41s$. **(d)** Snapshot at $t \approx 48s$

(e) Human commands. **(f)** Force feedback.

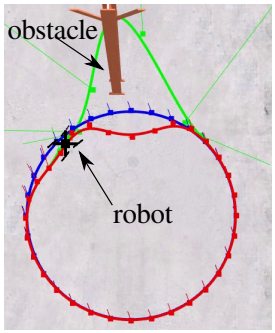
(g) Distance of the path from the PoIs. **(h)** Distance of the path from obstacles and singularities.

Figure 10. Simulation 2: obstacles and PoIs (plots).

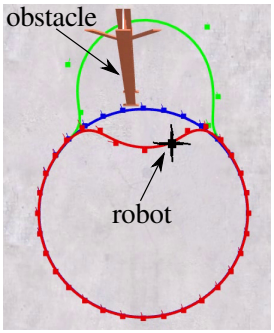
Simulation 2 The purpose of this simulation is to demonstrate the various features of the proposed shared planning framework in an environment populated by obstacles and PoIs. In this scenario the user is allowed to steer the desired path (a circle) by commanding translations (map (9)) and changes of scale (map (10)). Rotational commands are not used because rotating a circular path would be pointless. Additionally, once a PoI is reached the user can discard it by pressing a button, thus removing the



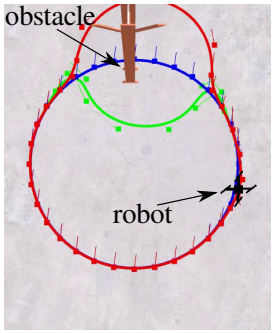
(a) Crossing, $t = 7.52s$.



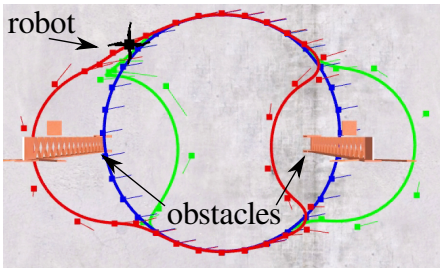
(b) Expansion, $t = 7.80s$.



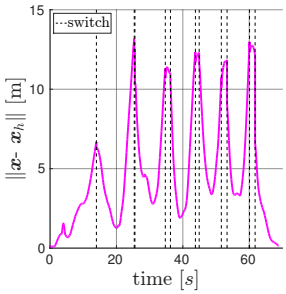
(c) Activation, $t = 9.96s$.



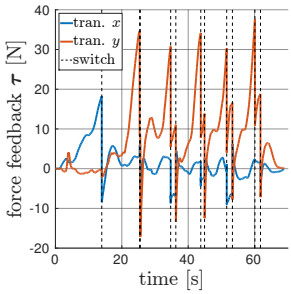
(d) Switch, $t = 13.00s$.



(e) Path reinitialization with multiple obstacles.



(f) Mismatch $\|\boldsymbol{x} - \boldsymbol{x}_h\|$.



(g) Force feedback.

Figure 11. Simulation 4: generation of alternative paths. (a)-(d) Snapshots of the proposed algorithm phases. In the snapshots the desired path $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ is drawn a thick blue line, the traveled path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is drawn as thick red line and the replanned path $\boldsymbol{\gamma}_S(\boldsymbol{x}_o)$ is drawn as a thick green line.(e) Alternative paths in presence of multiple obstacles. (f) Mismatch $\|\boldsymbol{x} - \boldsymbol{x}_h\|$. The vertical dashed lines indicate the moments when the current path is switched to an alternative one. (g) Force feedback.

corresponding autonomous attractive force. The obstacles in the environment are represented by columns, whereas the two PoIs present are shown as barrels. All these elements of the environment are visible in Figs. 10a to 10d, which are snapshots captured at various meaningful moments during the simulation:

a) In Fig. 10a $\boldsymbol{\gamma}_S(\boldsymbol{x})$ (red path) is attracted by a nearby PoI.

b) In Fig. 10b $\boldsymbol{\gamma}_S(\boldsymbol{x})$ (red path) is steered upwards and it is modified by two obstacles. The PoI visible in the snapshot has already been visited and it is therefore discarded.

c) In Fig. 10c the operator shrinks the path to make it pass upwards between the two obstacles.

d) In Fig. 10d $\boldsymbol{\gamma}_S(\boldsymbol{x})$ (red path) and the robot managed to travel between the two obstacles.

The commands given by the operator and the forces rendered on the input devices are depicted in Figs. 10e and 10f, respectively. The automatic reaction to nearby PoIs (at around 6 s and 63 s) produces two clear peaks in the force feedback. What is interesting to observe is that the human rapidly reacts to these force cues by promptly steering the path towards the PoIs, as shown by the peaks at around 6 s and 63 s in Fig. 10e. This behaviour is confirmed by the plot of the minimum distance between $\boldsymbol{\gamma}_S(\boldsymbol{x})$ and the PoIs is plotted in Fig. 10g. Indeed, the distance from the PoIs rapidly decreases when they are within the range $R_{\mathcal{I}}$ (range of action of the potential $\varphi_{\mathcal{I}}$). During the whole task, while the human is trying to reach the PoIs, the autonomous corrector also ensures that $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is collision-free and regular. Figure 10h shows that the distance from the obstacles always stays above the threshold $R_{\mathcal{O}}$ and the distance $\min_{i=1,\dots,n} \|\Omega_i(\boldsymbol{x}) - \boldsymbol{x}_i\|$ between the control points and their singular curve is always greater than zero.

Simulation 3 The purpose of this simulation is to demonstrate the generation of alternative paths in a cluttered environment. Once again the reference path used in this simulation is a circle, however, unlike the previous simulation, the user can only command translations because we want to focus on the autonomous corrections rather than on the user's commands. Figures 11a to 11d illustrate the different phases described in Sec. 7.

a) Figure 11a shows the *crossing* step. The velocity vectors forming $\dot{\boldsymbol{x}}_{\boldsymbol{o}}$ (thin green lines) are pointing in the

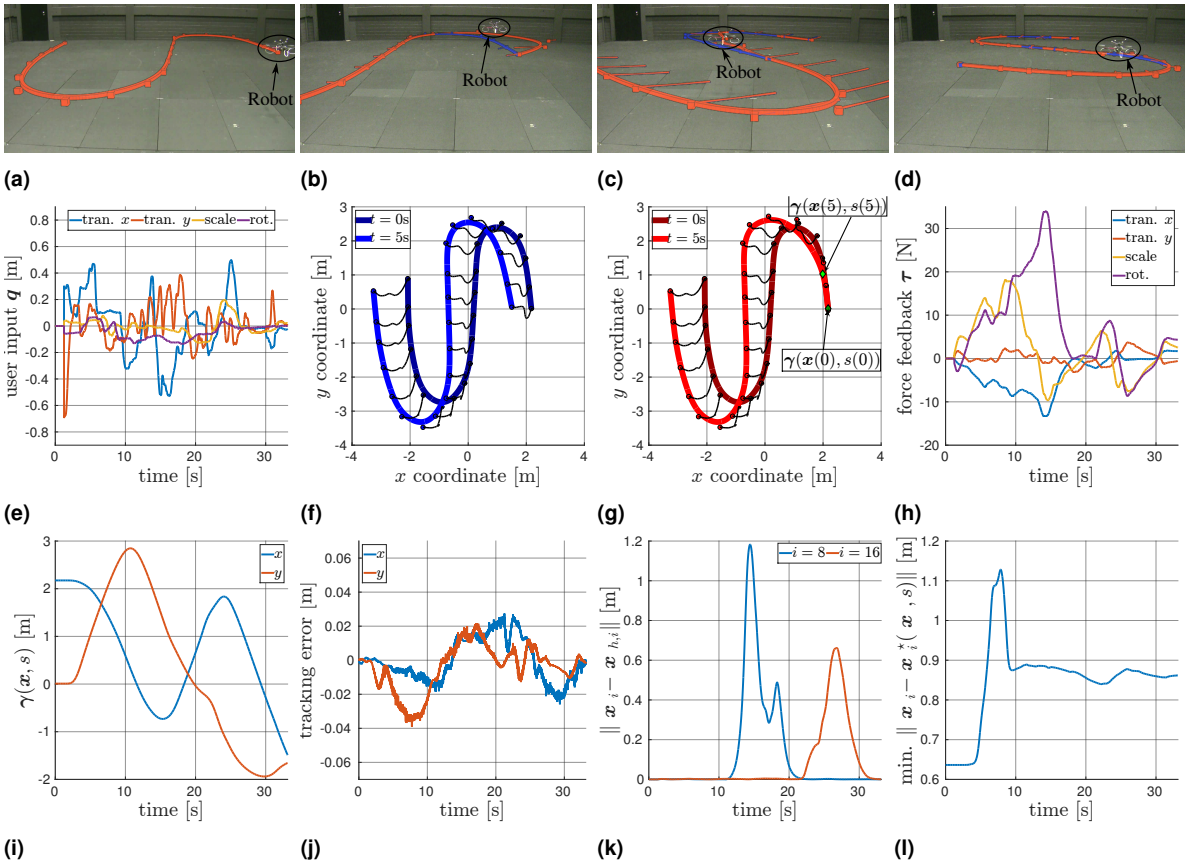


Figure 12. Experiment 1: open path. (a)-(d) Snapshots from the experiment with an overlay of the paths $\gamma_S(\boldsymbol{x})$ (red) and $\gamma_S(\boldsymbol{x}_h)$ (blue). (a) Initial condition. (b) The path is being translated. (c) The path is being rotated. (d) The path is scaled down. (e) Input device's configuration q controlled by the user to give the commands. (f) Evolution of the desired path $\gamma_S(\boldsymbol{x}_h)$ in the first five seconds. The circles are the control points and the black lines show their trajectories. (g) Evolution of the desired path $\gamma_S(\boldsymbol{x})$ in the first five seconds. The circles are the control points, the black lines show their trajectories and the green marker indicates the point $\gamma(\boldsymbol{x}(t), s(t))$ tracked by the robot. (h) Force feedback. (i) Reference trajectory for the robot. (j) Robot's trajectory tracking error. (k) Mismatch $\|\boldsymbol{x}_i - \boldsymbol{x}_{h,i}\|$ for individual control points. (l) Minimum distance from the singular curves.

direction of the obstacle and $\gamma_S(\boldsymbol{x_o})$ (thicker green line) is 'pulled' to the other side of the obstacle.

b) Figure 11b shows the *expansion* step. The velocity vectors forming $\dot{\boldsymbol{x}}_{\boldsymbol{o}}$ (thin green lines) are pointing away from the obstacle and $\gamma_S(\boldsymbol{x_o})$ (thicker green line) is 'expanded' around the obstacle.

c) Figure 11c shows the *activation* step. The alternative path is fully generated.

d) Figure 11d shows the moment when condition $\mathcal{C}4$ is verified. The reference path (red line) and the alternative one (green line) are switched.

During the simulation the automatic mechanism for generating alternative paths is also used in presence of multiple obstacles, as shown in Fig. 11e. In this case, one alternative portion of the path (green lines) is created independently for each obstacle. For giving a concrete evaluation of the effect of the path reinitialization method, one can focus on the evolution of the mismatch $\|\boldsymbol{x} - \boldsymbol{x}_h\|$ in Fig. 11e. This graph can be understood by recalling that the problem of the purely reactive path modifications is that the path $\gamma_S(\boldsymbol{x})$ (red line) cannot pass through obstacles. Indeed, by looking at Fig. 11e, one can observe that the rapid increases in the mismatch $\|\boldsymbol{x} - \boldsymbol{x}_h\|$ correspond to the moments when the user steers the desired path $\gamma_S(\boldsymbol{x}_h)$ over an obstacle. We can also note that after the switches to the newly generated alternatives (moments identified by the vertical dashed lines) the mismatch decreases. This shows that this reinitialization strategy is a viable solution for escaping from local minima in presence of obstacles. Note also that in some cases the reduction of the mismatch

$\|\boldsymbol{x} - \boldsymbol{x}_h\|$ only happens after two consecutive switches. This is because in these moments the user is trying to cross over two obstacles at the same time. Lastly, we can also observe that the path switches also reflect on the force feedback given to the user, shown in Fig. 11f, thus giving a clear cue that the path was steered past an obstacle.

10 Experiments

The proposed framework has also been tested with a real UAV (a quadrotor from Mikrokopter) interfaced with a human operator. In order to allow for a fair comparison with the results obtained in simulation, we used the same software setup for the experiments. Since the planned path is not visible in the real environment, during the experiments we provided the user with a crude 3D visualization similar to what used in the simulations (see for example Fig. 9a). This visualization only showed an empty room (environment), the robot, and the paths $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ and $\boldsymbol{\gamma}_S(\boldsymbol{x})$.

The experiments hereinafter are meant to show how the user can operate the desired path by combining several canonical path transformations. Specifically, we used the three canonical transformations described in Sec. 4, *i.e.*, whole path translations, rotations and changes of scale. Additionally, to show that the framework can be used identically with open or closed paths, in the first experiment the user controls an 'S-shape' path with 21 control points, in the second one an 'eight-shape' path with 32 control points. Finally, we remark that in these experiments the environment is free of PoIs and obstacles because i) the focus is on the user interface, and ii) previous experiments (Masone et al. 2014) already demonstrated the replanning algorithm and the autonomous corrections in presence of PoIs and obstacles. For a better appreciation of these results, we encourage the reader to see Extension 2.

Results The results of the experiments with the open and closed paths are shown in Figs. 12 and 13, respectively. Since Figs. 12 and 13 are organized in the same way, we will discuss the results for the two cases at once and highlight the main differences whenever necessary.

Figures 12a to 12d and Figs. 13a to 13d are screenshots from the execution of the task. These screenshots were taken from a camera recording the real robot and afterwards have been post-processed by adding an overlay of the paths $\boldsymbol{\gamma}_S(\boldsymbol{x})$ and $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$. These screenshots capture different moments of the experiments for giving an idea of the effect of the different canonical path transformations. The actual commands given by the user are shown in Figs. 12e and 13e in the form of the input device configuration $\boldsymbol{q}$ for the sake of having a uniform scale. Note that the user can command translations, rotations and changes of scale at the same time. In order to appreciate the effect of the user commands we need to look at the evolution of the path $\boldsymbol{\gamma}_s(\boldsymbol{x}_h)$. For this purpose we show in Figs. 12f and 13f how $\boldsymbol{\gamma}_s(\boldsymbol{x}_h,)$ changed in the first 5s, highlighting the trajectories of the control points with black lines. Observe that in both these figures there is no indication of the robot since the user commanded path $\boldsymbol{\gamma}_s(\boldsymbol{x}_h)$ is not the actual path tracked by the robot. The actual path tracked by the robot, $\boldsymbol{\gamma}_S(\boldsymbol{x})$, tries to follow $\boldsymbol{\gamma}_s(\boldsymbol{x}_h)$ but also needs to account for the presence of the robot (blending filter $\boldsymbol{N}$) and for the other objectives/requirements of the autonomous corrector. For better illustrating this difference, in Figs. 12g and 13g one can look at the evolution of the actual followed path $\boldsymbol{\gamma}_s(\boldsymbol{x})$ during the first 5 seconds. In both figures we indicate the point tracked by the robot with a green marker. By comparing Figs. 12g and 13g with Figs. 12f and 13f, one can clearly see how the blending filter $\boldsymbol{N}$ is able to prevent local path transformations at the point tracked by the robot. This mismatch between the evolution of $\boldsymbol{x}_h$ and $\boldsymbol{x}$ is captured by the force feedback rendered on the input device and depicted in Figs. 12h and 13h.

Focusing on the robot, we show the actual tracked trajectory $\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$ and the associated tracking error in Figs. 12i and 13i and Figs. 12j and 13j, respectively. We can observe that, in open path case, the tracking error is always below 0.036 m on both x and y coordinates, whereas with the closed path the tracking error is always below 0.062 m. This difference is due to the fact that, in the first case, the maximum travelling speed along the path was around 0.5 m/s, whereas in the second case it was around 0.7 m/s. Note also that these results are in agreement with the ones obtained in Simulation 1, see Fig. 9g, where we obtained a maximum tracking error of 0.13 m x and y but with a travelling speed of about 1.5 m/s.

In this experiment it is also interesting to look at the behaviour of the individual control points. For this purpose in Figs. 12k and 13k we show the evolution of the mismatch $\|\boldsymbol{x}_i - \boldsymbol{x}_{h,i}\|$ for two separate control points. One can see that for each of the control points the mismatch is always zero except for some spikes of limited duration, and that the spikes for the different control points are always shifted in time. These spikes are caused by the blending filter $\boldsymbol{N}$ which only affects the control points associated to the portion of path currently travelled by the robot. Therefore, the duration of these spikes depends on how fast the robot is moving along the path, *i.e.*, how brief is the duration of the effect of $\boldsymbol{N}$ on the individual control points. In the case of the open path with a maximum travelling speed of around 0.5 m/s, the spike are about 10 s long, whereas in the case of the closed path with a maximum travelling speed of 0.7 m/s, the individual spikes last about 7 s.

We can also observe that, in the closed path case, the spikes occur cyclically for each control point since the robot keeps traveling along the same (closed) path. Lastly, we

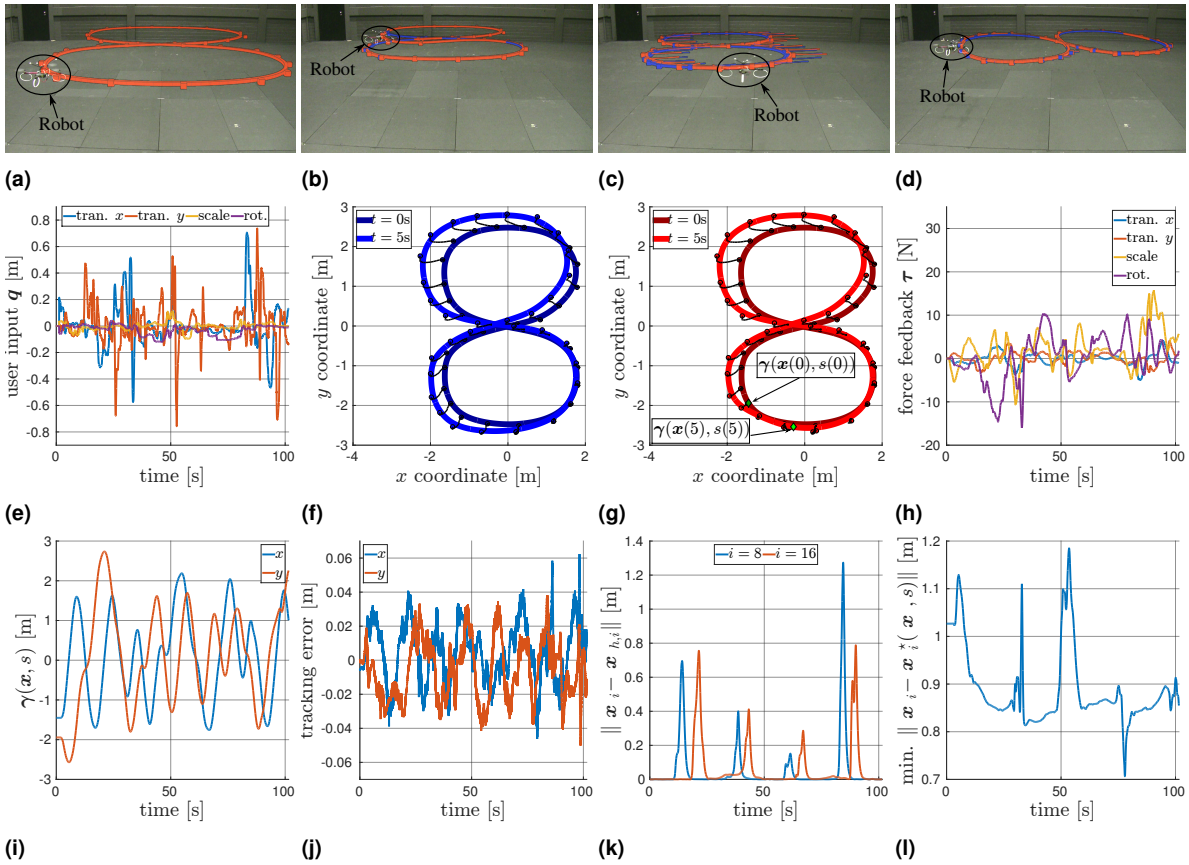


Figure 13. Experiment 1: closed path. (a)-(d) Snapshots from the experiment with an overlay of the paths $\boldsymbol{\gamma}_S(\boldsymbol{x})$ (red) and $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ (blue). (a) Initial condition. (b) The path is scaled down. (c) The path is being translated. (d) The path is being rotated. (e) Input device's configuration q controlled by the user to give the commands. (f) Evolution of the desired path $\boldsymbol{\gamma}_S(\boldsymbol{x}_h)$ in the first five seconds. The circles are the control points and the black lines show their trajectories. (g) Evolution of the desired path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ in the first five seconds. The circles are the control points, the black lines show their trajectories and the green marker indicates the point $\boldsymbol{\gamma}(\boldsymbol{x}(t), s(t))$ tracked by the robot. (h) Force feedback. (i) Reference trajectory for the robot. (j) Robot's trajectory tracking error. (k) Mismatch $\|\boldsymbol{x}_i - \boldsymbol{x}_{h,i}\|$ for individual control points. (l) Minimum distance from the singular curves.

note in Figs. 12l and 13l that during the experiments the distance between the control points and their corresponding singular curves is always positive, thus proving that the path never becomes singular.

11 Human subject study

In order to evaluate the efficacy of the proposed approach, we conducted a human subject study on performing a *coverage task* Wong et al. (2002) with the goal of assessing 1. the superiority of the proposed framework with respect to the simple teleoperation approach, and 2. the importance of haptic feedback in improving the human subjects' performance.

We selected an Urban Search And Rescue (USAR) scenario (search the victims in a disaster scene) for this study for two main reasons: 1. USAR is a field where human robot cooperation is of paramount importance because disaster scenes are generally too dangerous or inaccessible to humans but at the same time too complex for fully autonomous robots. Indeed, the usual approach in USAR is remote teleoperation, *i.e.*, a human operator remotely drives a robot in the disaster scene and inspects the environment from camera(s) installed on the robot (Murphy 2004). Therefore, USAR is a relevant case study for shared control approaches. 2. It has been shown by Kruijff et al. (2014) that driving a robot in a disaster scene is highly stressful and difficult for human operators and, as a consequence, a considerable portion of the operators' effort is simply directed towards maneuvering the robot and avoiding obstacles (basic low-level tasks). Therefore, the operators' performance in covering the search area

and success in finding the victims are degraded. These challenges make USAR a perfect test for assessing the performance impact of the proposed strategy.

11.1 Setup and Task Definition

The task considered in this study is finding victims in an orange level disaster scene (Wang et al. 2003) using a single VTOL aerial robot (a quadrotor UAV) capable of localizing itself in the environment. The UAV is equipped with a down-facing camera with a 60 deg wide field-of-view. The disaster scene simulates the concrete structure of a building after an moderate earthquake (Fig. 14 (a)). Victims and irrelevant objects, *e.g.*, construction equipments and material, are randomly located on the second floor of the building, that is a $20 \times 20\,\mathrm{m}^2$ area with 15 concrete columns forming a square grid graph (Fig. 14 (b)). The operator is provided with the view from the onboard camera and with a visualization showing the position of the robot in the bare map (Fig. 14 (b)). The planning and control algorithms are implemented in Simulink®, and the remote environment is implemented using the open source game engine Blender™.

The participants in this study were asked to navigate the robot in order to cover the area as much as possible, and locate the victims in the environment. Additionally, in order to simulate a realistic USAR mission, they were asked to mark the positions of the victims on a paper map as soon as they were discovered "during" the execution of the task (Fig. 14 (b)). In order to compare the performance of the proposed shared planning and control to simple teleoperation control, and to assess how important is the role of haptic feedback in improving the performance, each subject repeated the task for each of the following conditions in a three minutes given time for each trial.

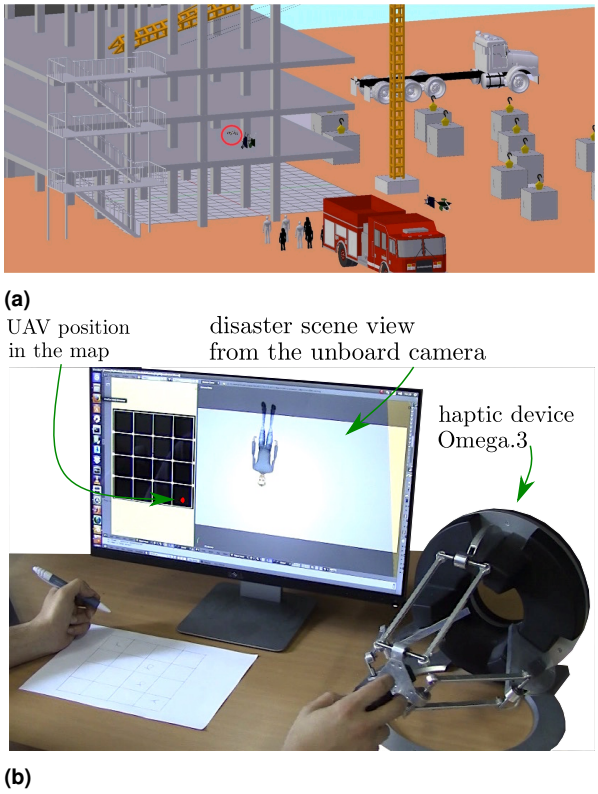


Figure 14. (a) A human robot team in an orange level disaster scene in the USAR scenario. The quadrotor UAV is shown inside the red circle. (b) Human/Hardware in the loop experimental setup. The human operator marks the discovered victims' positions.The operator sees the position of the robot in the map. Red dot shows the robot position and the white squares show the concrete columns. The operator inspects the scene from the down-facing UAV camera.

Condition T-Uni : Unilateral Teleoperation control, in which the subjects are provided only with the visual feedback, that is the view from the onboard camera, and the robot position in the map—without the global view of the robot in the map— as shown in Fig. 14 (b). The subjects guide the robot on simple teleoperation control mode, that is the desired position of the robot is obtained by integrating the motion of the haptic device.

Condition T-Bil : Bilateral Teleoperation control, in which the subjects navigate the robot in the teleoperation mode by commanding the desired position. The subjects, similar to the previous condition, are provided with the visual feedback, and beside that, in this condition they are also provided with the haptic feedback. The haptic feedback in this case is proportional to UAV position and velocity error; therefore, prevents the subjects from commanding the UAV faster than its capability.

Condition S-Uni : Unilateral Shared planning and control, in which the subjects are only provided with visual feedback, and the robot motion is based on the proposed shared planning and control approach, that is, the robot moving on a circular path automatically, and the subjects translate the circular path in order to cover the area. The desired path, using the proposed method in this paper, is generated considering the human command to the path, the robot velocity, and the obstacles in the environment. The haptic feedback is deactivated in this case.

Condition S-Bil : Bilateral Shared planning and control, in which the subjecte are provided with both visual feedback and haptic feedback, and the robot motion

is commanded by the proposed shared planning and control approach.

In order to avoid the learning effect, the conditions were randomly assigned to each participant in the experiment.

In all conditions the operator was assisted by an automatic obstacle avoidance: in condition T-Uni and T-Bil the obstacle avoidance acted on the robot itself (thus implementing a local action), while in conditions S-Uni and S-Bil the obstacle avoidance action acted on the whole planned path as per the proposed shared control framework. In both cases, the minimum and maximum obstacle distances determining the artificial potentials were fixed to 0.60 m and 1.5 m, respectively.

For what concerns the haptic feedback, in S-Bil it was implemented according to the approach described in Sec. 8, whereas in T-Bil the feedback cues were based on the (instantaneous) mismatch between the robot position and its target point as classically done (*e.g.* Diolaiti and Melchiorri 2002, Lam et al. 2009). For all four conditions the maximum velocity and acceleration of the robots were set to 1 m/s and 1 m/s^2, and the robot altitude was fixed (2 m above the floor). In order to have a rational performance comparison, the attitude and position controller coefficients are set the same for all conditions. The closed B-spline used in conditions S-Uni and S-Bil was defined as a circle with order 5 and 10 knots, and its initial scale was chosen by a well-trained user according to the map. This initial scale of the circle (approx. radius 2 m) makes the experiments easier for non-trained users.

An Omega.3 haptic device with 3 actuated DoFs was used by the subjects for driving the robot. In conditions T-Uni and T-Bil the subject could use two axes of the Omega.3 to command the desired translational velocity of the robot in the $x-y$ plane. In conditions S-Uni and S-Bil, the subject could use two axes of the Omega.3 to command the desired translational velocity of the path in the $x-y$ plane and the third axis of the Omega.3 to change the scale of the path. For a better appreciation of the setup used in the user study, we encourage the reader to see Extension 3.

11.2 Subjects and Experiment

The human subject study was a *within subject design*, that is all the participants in the experiment participate in all the conditions. Within subject design method compared to the *between subject design* method in which each experimental condition is experienced by a separate group of the subject, requires fewer number of participants. In order to avoid the learning effect, which is possible in within subject design experiments, the conditions were randomly assigned to each participant in the experiment.

Fifteen subjects (12 males, 3 females, age range 22-49 years old) took part in the experiments, voluntarily. The subjects signed an informed consent form. They had no eyesight problems, and no one reported any deficiencies in perceptual and motor abilities. Four of them had previous experience with haptic interfaces and five of them had previous experience with aerial robots, but none served in USAR teams before.

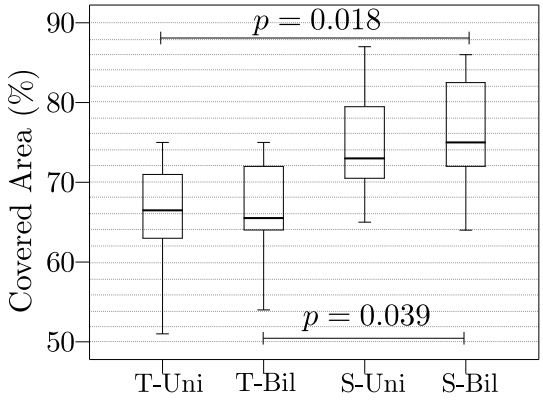


Figure 15. Covered area (CA) in the four conditions.

They were given thorough explanation of the experimental setup, robot, conditions, and the experimental procedure. Moreover, before the experiment the subjects had a test trial in each condition in order to get acquainted with the robot and with the different conditions. The test trial in each condition had the same time of the main experiments (three minutes).

The metric chosen to compare the subjects performance in different conditions is the Covered Area (CA), *i.e.*, the area seen via the onboard camera during the three minutes trial and normalized by the total area of interest. Namely, the CA is a value between 0 and 1, where 1 represents an ideal performance. Note that a variable number of victims (up to 4) were present in the environment and the subjects were not aware of this number. We believe that the CA, being proportional to the success in finding victims "independently" from their variable number, is a reasonable metric to evaluate the effectiveness of the different control approaches in the USAR scenario. Furthermore, CA is in a close conjunction with the time being spent on search, and time is of utmost importance in any USAR mission.

11.3 Results and statistical analysis

All the subjects successfully completed the task in all the conditions. Fig. 15 shows the results of the CA for the four different conditions as box plot. Mean and standard deviation of the CA in different conditions are reported in Table. 1. The S-Uni and S-Bil conditions yielded distinctively better results.

We used *repeated measures ANOVA* test to assess the compare the result of the experiment in the four different

Table 1. Mean and standard deviation of CA in between subject user study.

Condition	T-Uni	T-Bil	S-Uni	S-Bil
Mean	0.6575	0.6675	0.7467	0.7617
Std.	0.0680	0.0583	0.0720	0.0721

Table 2. Mean and standard deviation of subjects answer about the 'ease of use' question, by bipolar Likert-type seven-point scales.

Condition	T-Uni	T-Bil	S-Uni	S-Bil
Mean	5.29	6.14	3.64	4.57
Std.	1.07	0.66	1.87	1.55

condition together. This statistical analysis method is a standard approach to detect any overall differences between related means. The required assumptions to use this method are: 1. the dependent variable (CA in this experiment) must be continuous, 2. the independent variable (four conditions in this experiment) should consist of at least two categorical "related groups", 3. no significant outliers should exists in the related groups, 4. the dependent variable should have a approximately *normal distribution*, 5. the equality of the variances of the differences between all combinations of related groups, which is referred to as *sphericity* lae (2015).

The first two assumptions are correct for our experiment, and no significant outliers was found in the data. The collected data passed the Shapiro-Wilk normality test. Mauchly's Test of Sphericity showed that the sphericity assumption of the collected data is violated ($\chi^2(5) = 7.66$, $p = 0.178$). However, with a Greenhouse-Geisser correction, the repeated measures ANOVA revealed statistically significant different between the CA results of the four different conditions ($F(1.943, 21.377) = 8.255$, $p = 0.002$, $a = 0.05$). A pairwise comparison by Bonferroni adjustments determined that there are statistically significant differences between conditions S-Bil and T-Uni ($p = 0.018$), and S-Bil and T-Bil ($p = 0.039$). The differences between T-Uni and T-Bil were not statistically significant, also the difference between S-Uni with all the other three conditions were not statistically significant.

In conclusion, the user study revealed that the proposed shared control approach has a positive impact on the performance of the task, measured by metric CA, in comparison to classical teleoperation approaches (T-Uni and T-Bil), and this difference is statistically significant only when the subject benefits from haptic feedback. This outcome can be explained by the superior autonomy present in the shared architecture and by the transfer of the user's authority to the planning level, which allow the operator to ignore the behavior of the robot and rather focus on directing the task (*i.e.*, the search location). Additionally, the replanning strategy implemented in our framework prevents the search to get 'stuck' in an obstacle, conversely to T-Uni and T-Bil.

Another reason for the superior performance of the shared control approaches compared to teleoperation approached in the dynamics of the system allowing a higher speed when the robot follows a smooth path (S-Uni and S-Bil) while when following the human commands, constant accelerating/decelerating lowers down the speed. Using higher controller coefficients for the teleoperation approaches would reduce the human operator performance in controlling the robot beside increasing the instability chance. In fact, with the current coefficients set the robot velocity in the teleoperation conditions would much higher if the users would have pushed the haptic device to the very end of it in one direction and would have let the robot to stay in this direction and accelerate enough to reach its maximum speed, however, in this speed controlling the robot in order to find the location of the victims is difficult for the users, and in fact this is their choice to which extent command the robot to move fast (by pushing the haptic device) and they learned what speed range they are comfortable with. Thus, the more covered area—that is in a close conjunction with operation speed—in the shared controlled approached is due to the higher functionally effective attainable speed.

Remark Ease of use. *Each subject after the experiment was asked to report the ease of use for each condition using bipolar Likert-type seven-point scales. Mean and standard deviation of the users' answer to this question is presented in Table. 2. Statistical analysis using Friedman test, an alternative to ANOVA for ordinal dependent variables, showed a statistically significant difference between four conditions ($\chi^2(3) = 18.022$, $p < 0.0005$). A post hoc analysis with Wilcoxon signed-rank tests with a Bonferroni correction showed that the only statistically significant difference was between S-Uni and T-Bil ($Z = -3.221$, $p = 0.001$). This outcome can be explained as: for a person it is more natural to control a tangible robot rather than an abstract entity (the path). However, it is important to notice that the force feedback mitigates this problem and makes the shared control strategy sensibly easier to use (in comparison between S-Bil and S-Uni) and closer to the classical teleoperation approaches. This demonstrates that the proposed integral force feedback is important to increase the user's awareness and to make the controlled object, i.e., path, feel more natural to manipulate.*

12 Conclusions

In this paper we have presented a novel framework for shared planning and control of mobile robots that extends

in several aspects the classical paradigm of bilateral teleoperation. Our framework combines an intuitive human interface which allows the user to command a broad variety of path modifications with reactive corrections that are autonomously executed to keep the path collision-free and regular. A filtering action has been introduced to blend human's and autonomous controls while locally preserving geometric properties of the followed path. The proposed 'integral' haptic feedback algorithm is, to the best of our knowledge, a novel concept in the bilateral teleoperation of mobile robots because it provides an information about the 'future' of the planned trajectory. The method also includes a replanner for generating new paths in cluttered environments. The correctness of the proposed framework is formally proven and its feasibility is extensively demonstrated with human/hardware-in-the-loop simulations and experiments. Finally, a user study assesses the benefit of the proposed framework and haptic feedback.

The proposed framework for cooperative shared planning has been demonstrated using a quadrotor, as a meaningful example of a mobile robot. As mentioned in Sec. 2, the application of this framework to non-holonomic robots would require to impose additional constraints on some geometric derivatives of the path (*e.g.*, curvature), *i.e.*, these path derivatives must be continuous and limited. Even though these constraints have not been considered in the present paper they could be included as a next development of our framework. As a hint of a possible approach in this sense, consider the case in which the curvature of the path must be limited and continuous. The expression of the curvature, *i.e.*, $\kappa(s) = \|\partial^2 \boldsymbol{\gamma}(\boldsymbol{x}, s)/\partial s^2\|$ can be easily computed analytically thanks to the B-spline definition of the path. Then, the reactive-based paradigm proposed in this paper could be simply extended by including an additional control term that applies on each point of the path a force opposing the reduction of the curvature with a suitable potential function. This very same approach could also be used for managing other path derivatives if needed.

Given the flexibility provided by our novel command interface, further user studies should try to determine what is the best/easiest way for a user to modify the path in different scenarios, considering both whole and local path transformations. Finally, in the future we also plan to extend this framework to the scenario of dynamic obstacles, adding a replanning strategy and a model predictive algorithm able to generate online a suitable timing law $s(t)$

Acknowledgements

We thank Prof. H. H. Bülthoff, Max Planck Institute for Biological Cybernetics, Germany and Prof. D. Prattichizzo, University of Siena, Italy for contribution to the financial support of this work.

Declaration of conflicting interests

The Authors declare that there is no conflict of interest.

Funding

Partially funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 644271 AEROARMS.

Notes

1. Since $\boldsymbol{N}_\lambda$ is symmetric $\boldsymbol{H}$ can be chosen as an orthonormal basis of $\mathbb{R}^{2(\lambda+1)}$.
2. Using $\boldsymbol{x_o}$ instead of $\boldsymbol{x}$ for the computation of $\boldsymbol{N}$, $\boldsymbol{u}_h$ and $\boldsymbol{u}_a$.

References

(2015) Laerd Statistics, repeated measures anova using spss statistics. https://statistics.laerd.com.

Abbink DA, Mulder M, van der Helm FCT, Mulder M and Boer ER (2011) Measuring neuromuscular control dynamics during car following with continuous haptic feedback. *IEEE Transactions on Systems, Man, & Cybernetics. Part B: Cybernetics* 41(5): 1239–1249.

Batalin MA and Sukhatme GS (2004) Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems* 26(2-4): 181–196.

Benloucif MA, Nguyen AT, Sentouh C and Popieul JC (2017) A new scheme for haptic shared lateral control in highway driving using trajectory planning. *IFAC-PapersOnLine* 50(1): 13834–13840.

Biagiotti L and Melchiorri C (2008) *Trajectory planning for automatic machines and robots*. Springer Science & Business Media.

Bicchi A and Pallottino L (2000) On optimal cooperative conflict resolution for air traffic management systems. *IEEE Transactions on Intelligent Transportation Systems* 1(4): 221–231.

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

Bukusoglu I, Basdogan C, Kiraz A and Kurt A (2008) Haptic manipulation of microspheres using optical tweezers under the guidance of artificial force fields. *Presence: Teleoperation and Virtual Environments* 17(4): 344–364.

Chiaverini S, Oriolo G and Walker ID (2008) Kinematically redundant manipulators. In: Siciliano B and Khatib O (eds.) *Springer Handbook of Robotics*. Springer, pp. 245–285.

Choset H (2001) Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence* 31(1-4): 113–126.

Colomina I and Molina P (2014) Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing* 92: 79–97.

Diolaiti N and Melchiorri C (2002) Teleoperation of a mobile robot through haptic feedback. In: *IEEE Int.Work. on Haptic Virtual Environments and Their Applications*. pp. 67–72.

Dragan AD and Srinivasa SS (2012) *Formalizing assistive teleoperation*. MIT Press, July.

Dunbabin M and Marques L (2012) Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine* 19(1): 24–39.

Erlien SM, Fujita S and Gerdes JC (2016) Shared steering control using safe envelopes for obstacle avoidance and vehicle stability. *IEEE Transactions Intelligent Transportation Systems* 17(2): 441–451.

Farkhatdinov I, j H Ryu and Poduraev J (2009) A user study of command strategies for mobile robot teleoperation. *Intelligent Service Robotics* 2(2): 95–104.

Farkhatdinov I and Ryu JH (2010) Improving mobile robot bilateral teleoperation by introducing variable force feedback gain. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, pp. 5812–5817.

Faulwasser T, Hagenmeyer V and Findeisen R (2011) Optimal exact path-following for constrained differentially flat systems. In: *18th IFAC World Congress*. pp. 9875–9880.

Franchi A, Masone C, Grabe V, Ryll M, Bülthoff HH and Robuffo Giordano P (2012a) Modeling and control of UAV bearing-formations with bilateral high-level steering. *The International Journal of Robotics Research* 31(12): 1504–1525.

Franchi A, Secchi C, Ryll M, Bulthoff HH and Giordano PR (2012b) Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs. *IEEE Robotics & Automation Magazine* 19(3): 57–68.

Franchi A, Secchi C, Son HI, Bülthoff HH and Robuffo Giordano P (2012c) Bilateral teleoperation of groups of mobile robots with time-varying topology. *IEEE Transactions on Robotics* 28(5): 1019–1033.

Glassmire J, O'Malley M, Bluethmann W and Ambrose R (2004) Cooperative manipulation between humans and teleoperated agents. In: *2004 HAPTICS Int. Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. pp. 114–120.

Grabe V, Riedel M, Bülthoff HH, Giordano PR and Franchi A (2013) The telekyb framework for a modular and extendible ros-based quadrotor control. In: *European Conference on Mobile Robots, ECMR 2013*.

Hauser J and Hindman R (1997) Aggressive flight maneuvers. In: *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, volume 5. IEEE, pp. 4186–4191.

Hauser K (2013) Recognition, prediction, and planning for assisted teleoperation of freeform tasks. *Autonomous Robots* 35(4): 241–254.

Hirzinger G, Brunner B, Dietrich J and Heindl J (1994) ROTEX-the first remotely controlled robot in space. In: *Robotics and Automation (ICRA), 1994 IEEE International Conference on*, volume 3. pp. 2604–2611.

Hokayem PF and Spong MW (2006) Bilateral teleoperation: An historical survey. *Automatica* 42(12): 2035–2057.

Hou X, Lan H, Xing X, Qu Y, Yuan D, Yan J and Huang P (2017) Environmental force reflection in an admittance configured haptic interface for teleoperation of vtol aerial robots. *IFAC-PapersOnLine* 50(1): 10262–10267.

Isidori A (2013) *Nonlinear control systems*. Springer Science & Business Media.

Jiang J, Di Franco P and Astolfi A (2016a) Shared control for the kinematic and dynamic models of a mobile robot. *IEEE Transactions Control System Technology* 24(6): 2112–2124.

Jiang J, Franco PD and Astolfi A (2016b) Shared control for the kinematic and dynamic models of a mobile robot. *IEEE Transactions on Control Systems Technology* PP(99): 1–13.

Kant K and Zucker SW (1986) Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research* 5(3): 72–89.

Kruijff GJM, Janíček M, Keshavdas S, Larochelle B, Zender H, Smets NJ, Mioch T, Neerincx MA, Diggelen J, Colas F, Liu M, Pomerleau F, Siegwart R, Svoboda VHT, Petříček T, Reinstein M, Zimmermann K, Pirri F, Gianni M, Papadakis P, Sinha A, Balmer P, Tomatis N, Worst R, Linder T, Surmann H, Tretyakov V, Corrao S, Pratzler-Wanczura S and Sulk M (2014) Experience in system design for human-robot teaming in urban search and rescue. In: *Field and Service Robotics*. Springer, pp. 111–125.

Lächele J, Franchi A, Bülthoff HH and Giordano PR (2012) Swarmsimx: Real-time simulation environment for multi-robot systems. In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, pp. 375–387.

Lam TM, Boschloo HW, Mulder M and Paassen MMV (2009) Artificial force field for haptic feedback in UAV teleoperation. *IEEE Transactions on Systems, Man, & Cybernetics. Part A: Systems & Humans* 39(6): 1316–1330.

Laumond JP, Sekhavat S and Lamiraux F (1998) *Guidelines in nonholonomic motion planning for mobile robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–53.

LaValle SM (2006) *Planning Algorithms*. New York, NY, USA: Cambridge University Press. ISBN 0521862051. Available at `http://planning.cs.uiuc.edu`.

Lee D, Franchi A, Giordano PR, Son HI and Bülthoff HH (2011) Haptic teleoperation of multiple unmanned aerial vehicles over the internet. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. pp. 1341–1347.

Lee DJ, Franchi A, Son HI, Ha C, Bülthoff HH and Robuffo Giordano P (2013) Semi-autonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. *IEEE/ASME Transactions on Mechatronics* 18(4): 1334–1345.

Lee DJ and Huang K (2010) Passive-set-position-modulation framework for interactive robotic systems. *IEEE Transactions on Robotics* 26(2): 354–369.

Li Y, Tee KP, Chan WL, Yan R, Chua Y and Limbu DK (2015) Continuous role adaptation for human–robot shared control. *IEEE Transactions on Robotics* 31(3): 672–681.

Lim RS, La HM and Sheng W (2014) A robotic crack inspection and mapping system for bridge deck maintenance. *IEEE Transactions on Automation Science and Engineering* 11(2): 367–378.

Losey DP, McDonald CG, Battaglia E and O'Malley MK (2018) A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction. *Applied Mechanics Reviews* 70(1): 010804.

Losey DP and O'Malley MK (2018) Trajectory deformations from physical human–robot interaction. *IEEE Transactions on Robotics* 34(1): 126–138.

Majewicz A and Okamura AM (2013) Cartesian and joint space teleoperation for nonholonomic steerable needles. In: *World Haptics Conference (WHC), 2013*. pp. 395–400.

Masone C, Franchi A, Bülthoff HH and Giordano PR (2012) Interactive planning of persistent trajectories for human-assisted navigation of mobile robots. In: *IROS*. Citeseer, pp. 2641–2648.

Masone C, Robuffo Giordano P, Bülthoff HH and Franchi A (2014) Semi-autonomous trajectory generation for mobile robots with integral haptic shared control. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. pp. 6468–6475.

Meyer CD (2000) *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: Siam. ISBN 0-89871-454-0.

Mistler V, Benallegue A and M'sirdi N (2001) Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In: *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*. IEEE, pp. 586–593.

Montemerlo M and Thrun S (2006) Large-scale robotic 3-d mapping of urban structures. In: *Experimental robotics IX*. Springer, pp. 141–150.

Mulder M, Abbink DA and Boer ER (2012) Sharing control with haptics: Seamless driver support from manual to automatic control. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 54(52): 786–798.

Murphy RR (2004) Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 34(2): 138–153.

Murphy RR, Dreger KL, Newsome S, Rodocker J, Steimle E, Kimura T, Makabe K, Matsuno F, Tadokoro S and Kon K (2011) Use of remotely operated marine vehicles at minamisanriku and rikuzentakata japan for disaster recovery. In: *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. IEEE, pp. 19–25.

Murray RM, Rathinam M and Sluis W (1995) Differential flatness of mechanical control systems: A catalog of prototype systems. In: *Proceedings of the 1995 ASME International Congress and Exposition*.

Nguyen AT, Sentouh C and Popieul JC (2017) Driver-automation cooperative approach for shared steering control under multiple system constraints: Design and experiments. *IEEE Transactions on Industrial Electronics* 64(5): 3819–3830.

Niemeyer G, Preusche C and Hirzinger G (2008) Telerobotics. In: Siciliano B and Khatib O (eds.) *Springer Handbook of Robotics*. Springer, pp. 741–757.

Ortmaier T, Groger M, Boehm DH, Falk V and Hirzinger G (2005) Motion estimation in beating heart surgery. *IEEE Transactions on Biomedical Engineering* 52(10): 1729–1740.

Pan J, Chitta S and Manocha D (2012) FCL: A general purpose library for collision and proximity queries. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. pp. 3859–3866.

Pedemonte N, Abi-Farraj F and Giordano PR (2017) Visual-based shared control for remote telemanipulation with integral haptic feedback. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 5342–5349.

Peng J and Akella S (2005) Coordinating multiple robots with kinodynamic constraints along specified paths. *The International Journal of Robotics Research* 24(4): 295–310.

Profumo L, Pollini L and Abbink DA (2013) Direct and indirect haptic aiding for curve negotiation. In: *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, pp. 1846–1852.

Riedel M, Franchi A, Giordano PR, Bülthoff HH and Son HI (2013) Experiments on intercontinental haptic control of multiple uavs. In: *Intelligent Autonomous Systems 12*. Springer, pp. 227–238.

Rodríguez-Seda EJ, Troy JJ, Erignac CA, Murray P, Stipanović DM and Spong MW (2010) Bilateral teleoperation of multiple mobile agents: Coordinated motion and collision avoidance. *IEEE Transactions on Control Systems Technology* 18(4): 984–992.

Smith SL, Schwager M and Rus D (2012) Persistent robotic tasks: Monitoring and sweeping in changing environments. *IEEE Transactions on Robotics* 28(2): 410–426.

Srinivasan S, Latchman H, Shea J, Wong T and McNair J (2004) Airborne traffic surveillance systems: video surveillance of highway traffic. In: *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*. ACM, pp. 131–135.

Van Loock W, Pipeleers G, Diehl M, De Schutter J and Swevers J (2014) Optimal path following for differentially flat robotic systems through a geometric problem formulation. *IEEE Transactions on Robotics* 30(4): 980–985.

Van Quang H, Farkhatdinov I and Ryu JH (2012) Passivity of delayed bilateral teleoperation of mobile robots with

ambiguous causalities: Time domain passivity approach. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 2635–2640.

Wang J, Lewis M and Gennari JS (2003) Interactive simulation of the nist usar arenas. In: *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 2. IEEE, pp. 1327–1332.

Wong SC, Middleton L, MacDonald BA and Auckland N (2002) Performance metrics for robot coverage tasks. In: *Proceedings of Australasian Conference on Robotics and Automation*, volume 27. p. 29.

Appendix A: Multimedia extensions

The three videos show the simulations, experiments and user study reported in the manuscript.

Extension	Type	Description
1	Video	Simulations.
2	Video	Experiments.
3	Video	User study.

Appendix B: Generalization to $\mathbb{R}^3$

In order to extend the approach to $\mathbb{R}^3$ it is enough to consider a B-spline curve as $\boldsymbol{\gamma}(\boldsymbol{x},\cdot) : S \to \mathbb{R}^3, \quad s \mapsto \boldsymbol{\gamma}(\boldsymbol{x}, s)$, in which $\boldsymbol{x} = \begin{pmatrix}\boldsymbol{x}_1^\top & \cdots & \boldsymbol{x}_n^\top\end{pmatrix}^\top \in \mathbb{R}^{3n}$ is the vector of control points parameterizing the path. Thus, the path is $\boldsymbol{\gamma}_S(\boldsymbol{x}) = \{\boldsymbol{\gamma}(\boldsymbol{x}, s) \in \mathbb{R}^3 \,|\, s \in S\}$.

The obstacles are represented by $\mathcal{O} \in \mathbb{R}^{3\times n_\mathcal{O}}$ as vector of the obstacle spheres, the path $\boldsymbol{\gamma}_S(\boldsymbol{x})$ is considered to be collision free if it lies outside the obstacle spheres. In the 3D environment the required information needed by the reactive controller presented in Sec. 5 is the relative distance and direction between the obstacles and the path. For obstacles with more complicated shapes, the obstacles should be approximated by primitive 3D shapes (such as spheres, boxes, cylinders, ellipsoids), and the distances and relative directions could be computed using well-established theoretical and software solutions, such as the open source *flexible collision library* (FCL) that implements functions for proximity checking (Pan et al. 2012). The points of interest are also collected in the vector $\mathcal{I} \in \mathbb{R}^{3\times n_\mathcal{R}}$.

The blending filter is redefined in $\mathbb{R}^{3n}$ as $\boldsymbol{N} = \boldsymbol{I}_{3n} - \boldsymbol{J}^\dagger\boldsymbol{J}$, where $\boldsymbol{J}^\dagger$ indicates the Moore-Penrose pseudoinverse of $\boldsymbol{J} \in \mathbb{R}^{3k\times 3n}$, with $k < n$, and $\boldsymbol{J}$ is defined the same as $\mathbb{R}^2$ case, such that $\boldsymbol{J}\boldsymbol{N} = \mathbb{0}_{3k\times 3n}$.

Autonomous corrector and Human guidance signals are defined in $\mathbb{R}^{3n}$ as $\boldsymbol{u}_a \in \mathbb{R}^{3n}$ and $\boldsymbol{u}_h \in \mathbb{R}^{3n}$, respectively. Regarding human guidance in $\mathbb{R}^3$, one needs three components of $\boldsymbol{q}$ to act on translation, three components to act on rotation, and another component of $\boldsymbol{q}$ for scaling. Thus, a haptic device with at least 7 DoFs is needed in order to achieve full canonical transformation path manipulation; and the canonical transformations translation, rotation, and scaling w.r.t. a point $\bar{\boldsymbol{p}} \in \mathbb{R}^3$ might be represented as follows

$$\begin{aligned}\boldsymbol{Q}_1(\boldsymbol{x}_h) &= \boldsymbol{I}_3\\ \boldsymbol{Q}_2(\boldsymbol{x}_h) &= \boldsymbol{x}_h - \mathbb{1}_n \otimes \bar{\boldsymbol{p}}\\ \boldsymbol{Q}_3(\boldsymbol{x}_h) &= \operatorname{diag}(\underbrace{\bar{\boldsymbol{I}}_3, \cdots, \bar{\boldsymbol{I}}_3}_{n \text{ times}})\,(\boldsymbol{x}_h - \mathbb{1}_n \otimes \bar{\boldsymbol{p}})\,,\end{aligned}$$

in which $\bar{\boldsymbol{I}}_3 = \begin{pmatrix}0 & -1 & 1\\ 1 & 0 & -1\\ -1 & 1 & 0\end{pmatrix}$ comes from the infinitesimal rotation definition.

Finally, the required function for reinitialization is:

$$\begin{aligned}\boldsymbol{d} : \mathbb{R}^3 \times \mathbb{R}^{3n} \times S &\to \mathbb{R}^3\\ \boldsymbol{o}, \boldsymbol{x}, s &\mapsto \boldsymbol{d}(\boldsymbol{o}, \boldsymbol{x}, s) = \boldsymbol{o} - \boldsymbol{\gamma}_S(\boldsymbol{x}, s)\end{aligned}$$

The rest of the framework along with its properties and proofs are the same as 2D case.

Appendix C: Proofs

Proof of Property 1. To prove Property 1 we refer to the B-spline structure described in Appendix 12. From Remark 4 it follows that only the basis function derivatives $\frac{d^j B_{i-\lambda}^\lambda}{d s^j}, \ldots, \frac{d^j B_i^\lambda}{d s^j}$ for $j = 0, \ldots, k$ can be not null. Therefore, expression (17) of matrix $\boldsymbol{J}$ becomes

$$\boldsymbol{J} = \left[\begin{array}{c|c|c} \mathbb{0}_{2k\times 2(n-i-\lambda)} & \boldsymbol{M} & \mathbb{0}_{2k\times 2(n-i)}\end{array}\right] \tag{40}$$

where $\boldsymbol{M} \in \mathbb{R}^{2k\times 2(\lambda+1)}$ is the following matrix

$$\boldsymbol{M} = \begin{bmatrix} B_{i-\lambda}^\lambda & 0 & \cdots & B_i^\lambda & 0\\ 0 & B_{i-\lambda}^\lambda & & 0 & B_i^\lambda\\ & & \vdots & & \\ \frac{d^k B_{i-\lambda}^\lambda}{d s^k} & 0 & \cdots & \frac{d^k B_i^\lambda}{d s^k} & 0\\ 0 & \frac{d^k B_{i-\lambda}^\lambda}{d s^k} & & 0 & \frac{d^k B_i^\lambda}{d s^k}\end{bmatrix}. \tag{41}$$

where we did not write the dependency from $(\boldsymbol{s}, s)$ to have a compact notation. Expression (41) proves that $\dim(\Im(\boldsymbol{J})) \leq 2(\lambda+1)$.

Let $\boldsymbol{N}_\lambda \in \mathbb{R}^{2(\lambda+1)\times 2(\lambda+1)}$ indicate the matrix $\boldsymbol{N}_\lambda = \boldsymbol{I}_{2(\lambda+1)} - \boldsymbol{M}^\dagger\boldsymbol{M}$. By substituting (40) and (41) into (16), $\boldsymbol{N}$ has the following structure

$$\boldsymbol{N} = \begin{bmatrix}\boldsymbol{I}_{2(n-i-\lambda)} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{N}_\lambda & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{2(n-i)}\end{bmatrix}, \tag{42}$$

where the terms $\boldsymbol{0}$ indicate matrices of zeros of appropriate size that complete the non-diagonal blocks of $\boldsymbol{N}$, which concludes the proof.

Appendix D: B-Splines

B-splines are linear combinations of independent polynomial basis functions that are completely described by:

1. a sequence of scalars $s_1, \dots, s_l$ (*knots* sequence), with $s_j \leq s_{j+1}$ for $j = 1, \dots, l-1$. The knots sequence determines the pool of basis functions that define the spline.
2. a parameter $\lambda \in \mathbb{N}_{>0}$ (*spline degree*). It is related to the differentiability at the knots because the B-spline is $\lambda - k$ times continuously differentiable at a knot with multiplicity k.
3. a vector of planar[3] points $\boldsymbol{x} = [\boldsymbol{x}_1 \ \dots \ \boldsymbol{x}_n]^T$ (*control points*), with $\boldsymbol{x}_j \in \mathbb{R}^2$ and $j = 1, \dots, n$, which determines how the basis functions are combined.

The relation between the number l of knots, the number n of control points and the degree λ of the B-spline is

$$\begin{aligned} \textit{open B-spline} \quad & l = n - \lambda + 1 \\ \textit{cyclic B-spline} \quad & l = n \geq \lambda. \end{aligned}$$

For the computation of the basis functions it is useful to introduce the knots vector $\boldsymbol{s}$:

$$\begin{aligned} \textit{open B-spline} \quad & \boldsymbol{s} = [\underbrace{s_1, \dots, s_1}_{\lambda+1}, s_2, \dots, s_{l-1}, \underbrace{s_l, \dots, s_l}_{\lambda+1}] \\ \textit{cyclic B-spline} \quad & \boldsymbol{s} = [s_1, s_2, \dots, s_l]. \end{aligned}$$

The basis functions of degree λ are denoted as $B_j^\lambda(\boldsymbol{s}, s)$ with $j = 1, \dots, n$ and $s \in [s_1, \ s_l]$, and they are defined recursively

$$\begin{aligned} B_j^0(\boldsymbol{s}, s) &= \begin{cases} 1, & \text{if } s_j \leq s < s_{j+1} \\ 0, & \text{otherwise} \end{cases} \\ B_j^\lambda(\boldsymbol{s}, s) &= \frac{s - s_j}{s_{j+\lambda} - s_j} B_j^{\lambda-1}(\boldsymbol{s}, s) + \frac{s_{j+\lambda+1} - s}{s_{j+\lambda+1} - s_{j+1}} B_{j+1}^{\lambda-1}(\boldsymbol{s}, s). \end{aligned} \tag{43}$$

If the B-spline is cyclic all knot subtractions and knot index additions in (43) are replaced by modulus operations on $[s_1, \ s_l]$ and $[1, \ l]$.

With this notation, the expression of the point $\boldsymbol{\gamma}(\boldsymbol{x}, s)$ of the B-spline curve $\boldsymbol{\gamma}(\boldsymbol{x}, \cdot)$ is

$$\boldsymbol{\gamma}(\boldsymbol{x}, s) = \sum_{j=1}^{n} \boldsymbol{x}_j B_j^\lambda(\boldsymbol{s}, s), \tag{44}$$

The k-th derivative of the B-spline curve (44) with respect to s is

$$\frac{d^k \boldsymbol{\gamma}(\boldsymbol{x}, s)}{d\, s^k} = \frac{d^k \boldsymbol{B}_{\boldsymbol{s}}(s)}{d\, s^k} \boldsymbol{x} = \sum_{j=1}^{n} \boldsymbol{x}_j \frac{d^k \, B_j^\lambda(\boldsymbol{s}, s)}{d\, s^k}, \tag{46}$$

where the basis function derivative $\frac{d^k \, B_j^\lambda(\boldsymbol{s}, s)}{d\, s^k}$ $B_j^\lambda(\boldsymbol{s}, s)$ is computed efficiently with the following recursive formula

$$\frac{d^k \, B_j^\lambda(\boldsymbol{s}, s)}{d\, s^k} = \frac{\lambda!}{(\lambda - k)!} \sum_{i=0}^{k} a_{k,i} B_{j+i}^{\lambda-k} \tag{47}$$

where $\boldsymbol{B}_{\boldsymbol{s}} \in \mathbb{R}^{2 \times n}$ is

$$\boldsymbol{B}_{\boldsymbol{s}}(s) = \begin{bmatrix} B_1^\lambda(\boldsymbol{s}, s) & 0 & \cdots & B_n^\lambda(\boldsymbol{s}, s) & 0 \\ 0 & B_1^\lambda(\boldsymbol{s}, s) & \cdots & 0 & B_n^\lambda(\boldsymbol{s}, s) \end{bmatrix}. \tag{45}$$

with $a_{0,0} = 1$, $a_{k,0} = \frac{a_{k-1,0}}{s_{j+\lambda-k+1} - s_j}$, $a_{k,i} = \frac{a_{k-1,i} - a_{k-1,i-1}}{s_{j+\lambda+i-k+1} - s_{j+i}}$, and $a_{k,k} = \frac{-a_{k-1,k-1}}{s_{j+\lambda+1} - s_{j+k}}$ for $i = 1, \dots, k-1$.

We also recall a significant property of B-spline functions (Biagiotti and Melchiorri 2008):

Remark 4. *On any knot span $[s_i, \ s_{i+1})$, with $i = 1, \dots, n-1$, at most $\lambda + 1$ basis functions are non zero, i.e., $B_{i-\lambda}^\lambda, \dots, B_i^\lambda$. As a consequence of (47) at any knot span $[s_i, \ s_{i+1})$ at most $\lambda + 1$ basis functions derivatives are non zero, i.e., $\frac{d^k \, B_{i-\lambda}^\lambda}{d\, s^k}, \dots, \frac{d^k \, B_i^\lambda}{d\, s^k}$, with $k = 1, \dots, \lambda$.*

Finally, in order to compute $\boldsymbol{x}_i^\star(\boldsymbol{x}, s)$, for $s \in S_i$ (cfr. Definition 2), we write (46) explicitly for $k = 1$.

$$\frac{d\, \boldsymbol{\gamma}(\boldsymbol{x}, s)}{d\, s} = \sum_{j=1, \, j \neq i}^{n} \boldsymbol{x}_j \frac{d\, B_j^\lambda(\boldsymbol{s}, s)}{d\, s} + \boldsymbol{x}_i \frac{d\, B_i^\lambda(\boldsymbol{s}, s)}{d\, s}.$$

Then, by imposing that the left side is $(0 \ 0)^T$ and rearranging, it follows

$$\boldsymbol{x}_i^\star(\boldsymbol{x}, s) = -\frac{\sum_{j=1, j \neq i}^{n} \boldsymbol{x}_j \frac{d\, B_j^\lambda(\boldsymbol{s}, s)}{d\, s}}{\frac{d\, B_i^\lambda(\boldsymbol{s}, s)}{d\, s}} \tag{48}$$

Observe that i) $\frac{d\, B_i^\lambda(\boldsymbol{s}, s)}{d\, s} \neq 0$ for $s \in S_i$, therefore on S_i $\boldsymbol{x}_i^\star(\boldsymbol{x}, s)$ is always defined and unique, and ii) $\boldsymbol{x}_i^\star(\boldsymbol{x}, s)$ does not depend on the value of $\boldsymbol{x}_i$.