Article

Chemical Spill Encircling using a Quadrotor and Autonomous Surface Vehicles: a Distributed Cooperative Approach

Marcelo Jacinto [1,*], António Pascoal [1] and Rita Cunha [1]

[1] Laboratory of Robotics and Systems in Engineering and Science (LARSyS), Instituto Superior Técnico, University of Lisbon, Portugal

[*] Correspondence: marcelo.jacinto@tecnico.ulisboa.pt

Abstract: This article addresses the problem of formation control of a quadrotor and one (or more) marine vehicles operating at the surface of the water with the end goal of encircling the boundary of a chemical spill, enabling such vehicles to carry and release chemical dispersants used during ocean cleanup missions to break-up oil molecules. Firstly, the mathematical models of the Medusa class of marine robots and quadrotor aircrafts are introduced, followed by the design of single vehicle motion controllers that allow these vehicles to follow a parameterized path individually using Lyapunov based techniques. At a second stage, a distributed controller using event triggered communications is introduced, enabling the vehicles to perform cooperative path following missions according to a pre-defined geometric formation. In the next step, a real time path planning algorithm is developed that makes use of a camera sensor, installed on-board the quadrotor. This sensor enables the detection in the image of which pixels encode parts of a chemical spill boundary and use them to generate and update in real time a set of smooth B-spline based paths for all the vehicles to follow cooperatively. The performance of the complete system is evaluated by resorting to 3-D simulation software, making it possible to simulate visually a chemical spill. Results from real water trials are also provided for parts of the system, where two Medusa vehicles are required to perform a static lawn-mowing path following mission cooperatively at the surface of the water.

Keywords: Quadrotor control; Autonomous Surface Vehicle control; Cooperative Path Following; Online Path Planning; Chemical Spill Boundary Encircling

1. Introduction

The problem of perimeter detection, boundary search and encircling has been a widely researched topic with a variety of practical applications, ranging from monitoring of wildfire spread in forests[1], to the control and encircling of oil spills [2] and harmful invasive algae blooms [3] at the surface of the ocean. In this paper we will focus on the problem of chemical spill encircling.

The two main phenomena that contribute to the transportation and spread of hazardous chemicals over water, such as oil, are advection and diffusion. In the first, the chemical is transported due to the flow of water while the second refers to the motion of the fluid caused by the existence of concentration gradients. One way of modelling the flow field of the incompressible fluid is by solving iteratively the convection-diffusion equations [4]. In the literature, many works address the problem of dynamic boundary tracking at the surface of the ocean by proposing control schemes which require for (at least one) surface vessel to measure the concentration gradient of a hazardous contaminant. These measurements of the chemical plume are used by potential field controllers with the end goal of steering the robots to the boundary of the plume [5,6]. A completely different approach adopted by Saldaña et al. [7] is to consider that a general environmental boundary can be approximated by a closed curve that is slowly-varying over time and can be described by a general parametric equation. In his research, the author proposes a model for the curve described spatially by a truncated Fourier series that changes its shape smoothly over time. To achieve this, it is assumed that a team of Autonomous Surface Vehicles (ASV)s are distributed equally around the chemical spill and every vehicle is capable of taking local measurements of the boundary as it moves around it. These local measurements are then used to update the shape of the closed curve using recursive least squares. Although this is a very general solution to the problem, it can be argued that the use of a truncated Fourier series to represent a path for underactuated vehicles to follow is a rather poor choice of function as the resulting curve can self-intersect and exhibit substantial oscillations. Moreover, it does not take into consideration the physical constraints imposed by the vehicles. In order to lift the limitations imposed by this method, more stable parametric curves could be considered, such as Bernstein polynomials or B-Splines [8].

In recent years, there has been a massive development and demand for Autonomous Underwater Vehicles (AUV)s, due not only to their agility when it comes to the execution of scientific and comercial missions, but also due to their low cost when compared to traditional ships which require a crew on board to be operated. Additionally, there has also been an exponential growth in demand for Unmanned Aerial Vehicles (UAV)s with special emphasis on multirotor systems, which usually offer high quality camera sensors at low market prices. Aerial vehicles can have a top-down view of the environment, making them the tool par excellence for surveillance and maintenance missions. On the other hand, AUVs and ASVs can be used to carry and release chemical dispersants used in cleanup missions to break oil molecules [9]. Together, these unmanned vehicles have a huge potential to automate and reduce the cost of ocean cleanup operations.

In this paper we address the problem of chemical spill encircling and focus on the development of a set of control and path planning tools that allow a team of robots constituted of a quadrotor (equipped with an onboard camera) and ASVs to detect and encircle closely the dynamical boundary of a chemical spill as depicted in Figure 1. In our proposal, the quadrotor is responsible for detecting in real time the boundary of a chemical spill in the image stream produced by its onboard camera and producing a path that itself and one or more ASVs are required to follow cooperatively. To achieve this, we start by proposing a set of single vehicle motion control laws based on non-linear Lyapunov techniques that allow individual ASVs to follow a pre-defined parametric curve, based on previous works by Aguiar et al. [10–12]. These control techniques are then extended to the case of quadrotor vehicles. Borrowing from the work of N. Hung and F. Rego [13], a distributed controller using event triggered communications is presented, allowing the vehicles to perform Cooperative Path Following (CPF) missions, according to a pre-defined geometric formation. Finally, a new real time path-planning framework using growing unclamped (and uniform) cubic B-splines is proposed that fits a 2-D point cloud generated from the drone's image stream.

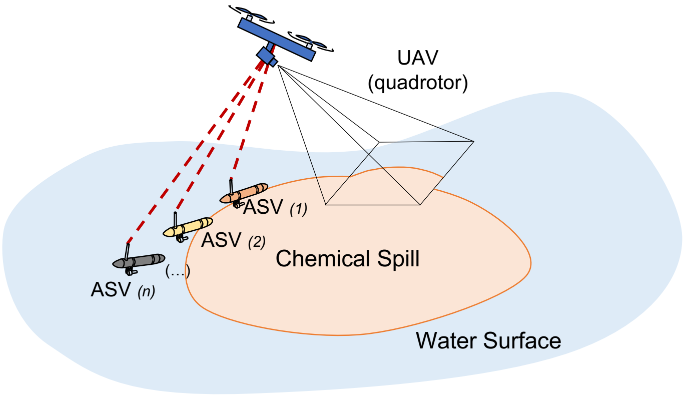


Figure 1. Cooperative path following along an environmental boundary

A set of real experiments are performed with the Medusa-class of marine vehicles [14] (property of ISR-DSOR) to access the real life performance of the proposed path following and CPF algorithms. Additionally, the complete path planning solution is evaluated by resorting to the Gazebo 3-D simulator, PX4-SITL [15] and UUVSimulator [16], using a dynamic model of a Medusa vehicle and an Iris quadrotor equipped with a virtual RGB camera.

2. Preliminaries

2.1. Notation

The unit vector $\mathbf{e}_3$ is defined as $\mathbf{e}_3 = [0,0,1]^T$. For a vector $\mathbf{x} \in \mathbb{R}^n$, the symbol x_i denotes the ith element of the vector. We shall use $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$ to denote the Euclidean norm of a vector. The notation $K \succeq 0$ is used to denote a matrix $K \in \mathbb{R}^{n\times n}$ that is positive semi-definite. The symbol I is used to denote the identity matrix and $\mathbf{1}$ a vector with all elements equal to one. The symbols $\lfloor x \rceil, x \in \mathbb{R}$ denotes x nearest integer, $\lfloor x \rfloor$ denotes the floor of x and $\lceil x \rceil$ denotes the ceiling of x. The symbol $R(.)$ is used to denote a rotation matrix with properties: $R^T = R^{-1}$ and $det(R) = 1$. The map $S(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n\times n}, n = 2,3$ yields a skew-symmetric matrix $S(\mathbf{x})\mathbf{y} = \mathbf{x} \times \mathbf{y}, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. When considering an estimator for an unknown variable x, we use the hat nomenclature $\hat{x}$ to denote its estimate and $\tilde{x}$ when referring to the estimation error.

2.2. Graph Theory

A weighted digraph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$ consists of a set of N vertices $\mathcal{V} = [V_1, ..., V_N]^T$, a set of directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and a weighted adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N\times N}$ such that $a_{ij} > 0$ if the edge that connects vertex i to j belong to the graph and 0 otherwise. The set of in-neighbours of a vertex i is given by $\mathcal{N}_i^{in} = \{j \in \mathcal{V} : (j,i) \in \mathcal{E}\}$ and the set of out-neighbours by $\mathcal{N}_i^{out} = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$. The in- and out-degree matrices D^{in} and D^{out} are a set of diagonal matrices defined by

$$D^{in/out} = diag(d_i^{in/out}), \text{ with } d_i^{in} = \sum_{j\in\mathcal{N}_i^{in}} a_{ij} \text{ and } d_i^{out} = \sum_{j\in\mathcal{N}_i^{out}} a_{ji}. \quad (1)$$

A graph $\mathcal{G}$ is undirected if communication links are unidirectional. If $\mathcal{G}$ is an undirected graph, then $\mathcal{G}$ is also balanced, i.e. $D^{in} = D^{out} := D$ and its Laplacian matrix L is symmetric, positive semi-definite and defined according to $L := (D - \mathcal{A})$. In these conditions it is well known that L has a simple eigenvalue at zero associated with eigen vector $\mathbf{1}$ with the remaining eigen values positive. Moreover $L\mathbf{1} = \mathbf{0}$.

Remark: With the graph definition given above, we adopt the convention that an agent i can receive information from its neighbors in $\mathcal{N}_i^{in}$ and send information to its neighbors in $\mathcal{N}_i^{out}$.

2.3. Uniform B-Spline curves

A 2-D B-Spline curve of degree k+1 in $\mathbb{R}^2$ is a piecewise polynomial function formed by several components of degree k, defined as

$$C(\gamma) = \sum_{i=0}^{n} B_{i,k}(\gamma)P_i, \quad (2)$$

where $\mathcal{P} = \{P_i \in \mathbb{R}^2, i = 0, ..., n\}$ are a set of control points and $B_{i,k}(\gamma)$ are the B-Spline basis functions. It follows from the Cox-De Boor's recursive algorithm [8], that:

$$B_{i,0}(\gamma) = \begin{cases} 1, \text{ if } \gamma_i \leq \gamma \leq \gamma_{i+1} \\ 0, \text{ otherwise} \end{cases}, \quad (3)$$

$$B_{i,j}(\gamma) = \frac{\gamma - \gamma_i}{\gamma_{i+j} - \gamma_i} B_{i,j-1}(\gamma) + \frac{\gamma_{i+j+1} - \gamma}{\gamma_{i+j+1} - \gamma_{i+1}} B_{i+1,j-1}(\gamma), \quad (4)$$

where the values γ_i belong to the m-dimensional knot vector $\mathbf{U} = \{\gamma_i\}_{i=0}^{m}$, with the number of knots related to the degree of the curve and the number of control points by $m = k + 1 + n$.

For the particular case of 2-D uniform, non-clamped cubic B-Splines with $n - k + 1$ segments, each segment's x and y-coordinates of the parametric curve can be described according to the vectorial notation [17], as follows:

$$C_i(\gamma) := \begin{bmatrix} C_i^x(\gamma) & C_i^y(\gamma) \end{bmatrix}^T, \tag{5}$$

with $C_i^x(\gamma)$ and $C_i^y(\gamma)$ computed according to

$$C_i^{x/y}(\gamma) := \underbrace{\frac{1}{6}\begin{bmatrix}(\gamma-i)^3 & (\gamma-i)^2 & (\gamma-i) & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}}_{\begin{bmatrix}B_{i,3}(\gamma) & B_{i+1,3}(\gamma) & B_{i+2,3}(\gamma) & B_{i+3,3}(\gamma)\end{bmatrix}}\begin{bmatrix}P_i^{x/y}\\ P_{i+1}^{x/y}\\ P_{i+2}^{x/y}\\ P_{i+3}^{x/y}\end{bmatrix}, \tag{6}$$

where $\gamma \in [0, n-k+1)$ and $i := \lfloor\gamma\rfloor$, such that $\gamma - i \in [0,1)$ and each curve segment is only defined by 4 distinct control points. Defining a unidimensional vector with all control points $\mathbf{P} = [P_0^x, ..., P_n^x, P_0^y, ..., P_n^y]^T \in \mathbb{R}^{2(n+1)}$, where both x and y-coordinates are concatenated, and a vector of distinct curve parameters $\boldsymbol{\gamma} = [\gamma_0, ..., \gamma_q] \in \mathbb{R}^{q+1}$ that we wish to evaluate our curve at, $\mathbf{C}(\boldsymbol{\gamma}) \in \mathbb{R}^{2(q+1)}$ is given by

$$\mathbf{C}(\boldsymbol{\gamma}) = B(\boldsymbol{\gamma}) \cdot \mathbf{P}, \tag{7}$$

where $B(\boldsymbol{\gamma}) \in \mathbb{R}^{2(q+1)\times 2(n+1)}$ is a diagonal by blocks matrix, and for each line of B only 4 basis functions are different then zero and computed according to (6).

3. Vehicle Modeling

Let $\{U\}$ denote an inertial reference frame and $\{B\}$ a body-fixed reference frame attached to the geometric center of mass of each vehicle, according to Figure 2.

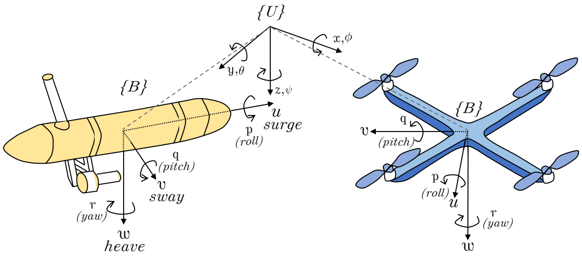


Figure 2. Adopted reference frames for a surface vehicle (left) and a quadrotor (right)

3.1. ASV Model

The ASV vehicle is modeled as a rigid body whose motion is restricted to a 2D plane at the surface of the water. Let the kinematic equations of the vehicle be given by

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}}_{\dot{\mathbf{p}}} = \underbrace{\begin{bmatrix}cos(\psi) & -sin(\psi)\\ sin(\psi) & cos(\psi)\end{bmatrix}}_{{}^U_B R(\psi)}\underbrace{\begin{bmatrix}u\\ v\end{bmatrix}}_{\mathbf{v}} + \underbrace{\begin{bmatrix}v_{cx}\\ v_{cy}\end{bmatrix}}_{\mathbf{v}_c}, \tag{8}$$

$$\dot{\psi} = r, \tag{9}$$

where $\mathbf{p} := [x, y]^T$ denotes the ASV position expressed in $\{U\}$, $\mathbf{v} := [u, v]^T$ the body-velocity vector, ${}^U_B R(\psi) \in \mathbb{R}^{2\times 2}$ rotation matrix and $\mathbf{v} := [v_{cx}, v_{cy}]^T$ the ocean current expressed in $\{U\}$ assumed to be constant, irrotational and bounded. The ASV model is considered to be underactuated with the input of the system being given by $\mathbf{u} = [u, r]^T \in \mathbb{R}^2$.

3.2. Quadrotor Model

The kinematic equations that describe the motion of a rigid body in 3-D space can be described by a double integrator model according to:

$$\ddot{\mathbf{p}} := \underbrace{g\mathbf{e}_3 - \frac{1}{m}{}^U_B R(\boldsymbol{\theta}) T \mathbf{e}_3}_{\mathbf{u}} + \mathbf{d}. \tag{10}$$

where $\mathbf{p} := [x, y, z]^T$ denotes the quadrotor's position expressed in $\{U\}$, $\boldsymbol{\theta} := [\phi, \theta, \psi]^T$ denotes the orientation vector of $\{B\}$ expressed in $\{U\}$ and $\mathbf{u} \in \mathbb{R}^3$ can be regarded as the input of the system, comprising both the attitude of the vehicle and the total thrust T. The vector $\mathbf{d} \in \mathbb{R}^3$ represents unmeasured external disturbances, such as wind, acting on the vehicle, assumed to be constant and bounded such that $\|\mathbf{d}\| \leq d_{max}$.

4. Path Following

The Path Following (PF) problem concerns the problem of making a vehicle move along a desired path $\mathbf{p}_d(\gamma)$ parameterized by a variable γ (for example, the arc-length of the curve). The key ideia is that each vehicle must approach a virtual target that moves along the path with a desired speed profile $v_d(\gamma)$, according to Figure 3. Since the end goal is to have more than one vehicle performing path following with a pre-defined inter-vehicle formation, this speed profile can be given by:

$$v_d(\gamma) := v_L(\gamma) + v^{coord}, \text{ with } |v_L(\gamma)| \leq v_L^{max}, \tag{11}$$

where $v_L(\gamma)$ is a desired speed profile defined as a function of the path, v_L^{max} a pre-defined speed upper-bound and v^{coord} a speed coordination term that will be used in section 5 to enable the CPF behaviour. It is important to notice that the desired speed profile $v_L(\gamma)$ should be the same for all the vehicles, enabling them to follow a given path at the same rate. On the other hand, the speed coordination term v^{coord} will not be the same for all vehicles and will be used to adjust the progression speed of each individual robot based on how aligned they are with each other.

Remark: The speed profile $v_d(\gamma)$ might not correspond directly to an inertial speed, especially if the curve is not parameterized in terms of the arc-length. Nonetheless, a relation between the inertial speed and the desired speed profile is addressed in detail in section 6.4.

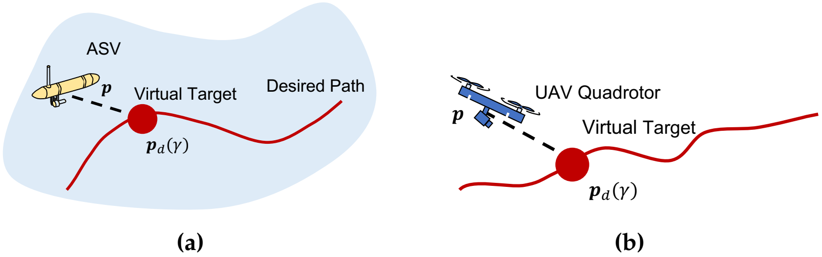


Figure 3. Path following schematic: (**a**) ASV path following. (**b**) Quadrotor path following.

Problem 1. *Given a generic vehicle (ASV or quadrotor), consider the geometric path $\mathbf{p}_d(\gamma) : [0, \infty) \to \mathbb{R}^2/\mathbb{R}^3$ for the ASV/quadrotor respectively, parameterized by a continuous variable $\gamma \in \mathbb{R}$ and $v_d(\gamma, t) \in \mathbb{R}$ a desired speed profile for a virtual target moving along the desired path. Furthermore, consider $\mathbf{p}_d(\gamma)$ to be C^2 and have its first and second derivatives with respect to γ bounded. Assume the vehicle is equipped with inner-loop controllers allowing it to track a desired control reference $\mathbf{u}_d \in \mathbb{R}^2/\mathbb{R}^3$, assumed to be bounded, by recruiting the appropriate forces and* *torques to apply to the vehicle. Design a feedback control law for the system input* $\mathbf{u}_d$ *and virtual target* $\ddot{\gamma}$ *such that:*

- *the vehicle's position converges to a tube around the desired position that can be made arbitrarily small, i.e.* $\|\mathbf{p}(t) - \mathbf{p}_d(\gamma)\|$ *converges to a neighbourhood of the origin;*
- *the speed of the virtual target moving along the path converges to the desired speed profile, i.e.* $|\dot{\gamma} - v_d(\gamma, t)| \to 0$ *as* $t \to \infty$.

4.1. ASV Path Following

Following the approach proposed by Aguiar et al. [10–12], consider the global diffeomorphic coordinate transformation which expresses the position error defined in the body-frame of the vehicle $\{B\}$ as

$$\mathbf{e}_p(t) := {}^{B}_{U}R(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma)) \tag{12}$$

and let the speed-tracking error be defined as

$$e_\gamma := \dot{\gamma} - v_d(\gamma, t). \tag{13}$$

With these definitions, the body-fixed position error dynamics are given by

$$\dot{\mathbf{e}}_p(t) = {}^{B}_{U}\dot{R}(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma)) + {}^{B}_{U}R(\psi)(\dot{\mathbf{p}}(t) - \dot{\mathbf{p}}_d(\gamma)). \tag{14}$$

We recall that the derivative of a rotation matrix can be expressed as the product of a skew-symmetric matrix with the transposed rotation matrix, that is,

$${}^{B}_{U}\dot{R}(\psi) = -S(r){}^{B}_{U}R(\psi). \tag{15}$$

Replacing (15) in (14) yields the position error dynamics expressed in the body-fixed frame as

$$\dot{\mathbf{e}}_p(t) = -S(r)\underbrace{{}^{B}_{U}R(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma))}_{\mathbf{e}_p(t)} + \mathbf{v} + \underbrace{{}^{B}_{U}R(\psi)\mathbf{v}_c}_{\boldsymbol{v}_c} - {}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma}. \tag{16}$$

Since there is no direct control in the sway motion, the goal is to generate surge speed and heading rate control references. Therefore, we must make these references appear explicitly in the error expression. By introducing an offset $\boldsymbol{\delta} = [0, \delta]^T \in \mathbb{R}^2$ (with $\delta < 0$) in the standard position error, it is possible to re-write (16) as:

$$\dot{\mathbf{e}}_p(t) = -S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -\delta \end{bmatrix}}_{\Delta} \underbrace{\begin{bmatrix} u \\ r \end{bmatrix}}_{\mathbf{u}} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v_c} - {}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma}. \tag{17}$$

Consider that each ASV is equipped with a Doppler Velocity Logger (DVL) capable of providing the vehicle's relative velocity with respect to the water $\mathbf{v}_m$, expressed in $\{B\}$ and a Global Positioning System (GPS) unit which provides measurements of the position of the vehicle $\mathbf{p}_m$, expressed in $\{U\}$. To estimate the ocean current, Pascoal et al. [18] and Sanches et al. [19] propose the use of a complementary filter. Consider the process model given by (8) such that $\mathbf{p}_m = \mathbf{p}$ and $\mathbf{v}_m = \mathbf{v}$. The candidate complementary filter model is described by

$$\mathcal{F} := \begin{cases} \dot{\hat{\mathbf{p}}} = k_1(\mathbf{p}_m - \hat{\mathbf{p}}) + {}^{U}_{B}R(\psi)\mathbf{v}_m + \hat{\mathbf{v}}_c \\ \dot{\hat{\mathbf{v}}}_c = k_2(\mathbf{p}_m - \hat{\mathbf{p}}) \end{cases} \tag{18}$$

with k_1 and k_2 positive constants. The proposed complementary filter is asymptotically stable. For a formal stability analysis of this complementary filter, refer to Pascoal et al.[18].

At this point it is important to notice that the current velocity v_c and the requested input $\mathbf{u}_d$ to be applied to vehicle's kinematic model cannot be estimated and tracked, respectively with infinite precision. For this reason, we define the current estimation error and the inner-loop tracking error given by

$$\begin{aligned}\tilde{v}_c &:= v_c - \hat{v}_{\mathbf{c}},\\ \tilde{\mathbf{u}} &:= \mathbf{u} - \mathbf{u}_d.\end{aligned} \tag{19}$$

Consider the proposition 1 introduced bellow, in which a solution to problem 1 applied to an ASV is provided, along with convergence guarantees in the presence of bounded estimation and tracking errors.

Proposition 1. *Consider the system described by the kinematics in (8) with the outer-loop control laws given by*

$$\mathbf{u}_d := \Delta^{-1}\left(- K_p \sigma(\mathbf{e}_p - \delta) - \begin{bmatrix} 0 \\ v \end{bmatrix} - \hat{v}_c + {}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} v_d(\gamma, t) \right), \tag{20}$$

$$\ddot{\gamma} := -k_\gamma e_\gamma + \dot{v}_d(\gamma, t) + (\mathbf{e}_p - \delta)^{T}{}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}, \tag{21}$$

where $K_p \succeq 0$, $k_\gamma > 0$ and $\sigma(\mathbf{e}_p) = tanh(\|\mathbf{e}_p\|)\frac{\mathbf{e}_p}{\|\mathbf{e}_p\|}$ is a saturation function. The closed-loop system is input-to-state stable (ISS) with respect to $\Delta\tilde{\mathbf{u}} + \tilde{v}_{\mathbf{c}}$, and the proposed control law solves problem 1 for the ASV vehicle.

Proof: See appendix A.

4.2. Quadrotor Path Following

Given that the quadrotor system is modelled by a double integrator in the inertial frame $\{U\}$ as stated in (10), consider the position and velocity errors defined in $\{U\}$ as

$$\mathbf{e}_p := \mathbf{p}(t) - \mathbf{p}_d(\gamma), \tag{22}$$

$$\mathbf{e}_v := \dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t), \tag{23}$$

and a virtual target speed tracking error defined by (13). Consider also a new auxiliary error $\mathbf{z}$ defined as

$$\mathbf{z} := \mathbf{e}_v + K_1 \mathbf{e}_p, \tag{24}$$

where $K_1 \succeq 0$ is a gain matrix. The position and velocity error dynamics can be written as

$$\dot{\mathbf{e}}_p = \dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{\gamma}, \tag{25}$$

$$\dot{\mathbf{e}}_v = \ddot{\mathbf{p}} - \frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t)\right). \tag{26}$$

Furthermore, consider the time derivative introduced in (26), the desired virtual target speed function (11), and the virtual target speed tracking error function (13). Then, the time derivative term introduced in (26) can be expanded as

$$\frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t)\right) = \underbrace{\left[\frac{\partial^2 \mathbf{p}_d}{\partial \gamma^2} v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\frac{\partial v_L(\gamma)}{\partial \gamma}\right]}_{\mathbf{h}(\gamma)}(e_\gamma + v_d(\gamma, t)) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t). \tag{27}$$

Replacing (10) and (27) in (26) yields

$$\dot{\mathbf{e}}_v = \mathbf{u} + \mathbf{d} - \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) - \frac{\partial \mathbf{p}_d}{\partial \gamma} v^{coord}(t). \tag{28}$$

Unlike the case of the ASVs where current estimates are given by a complementary filter, in the case of a quadrotor a different direction is taken towards estimating disturbances such as wind. According to Xie and Cabecinhas et al [20,21], straightforward implementations of estimators can lead to windup and result in unbounded growth of an external disturbance estimate. To avoid such problems, Xie and Cabecinhas propose the use of a sufficiently smooth projection operator in the estimator design. Consider the disturbance observer given by

$$\dot{\hat{\mathbf{d}}} := K_d \text{Proj}(\mathbf{z}, \hat{\mathbf{d}}) = \mathbf{z} - \frac{\eta_1 \eta_2}{2(\beta^2 + 2\beta d_{max})^{n+1} d_{max}^2} \hat{\mathbf{d}}, \tag{29}$$

where K_d denotes a diagonal gain matrix and

$$\eta_1 = \begin{cases} (\hat{\mathbf{d}}^T \hat{\mathbf{d}} - d_{max}^2)^{n+1}, \text{ if } (\hat{\mathbf{d}}^T \hat{\mathbf{d}} - d_{max}^2) > 0 \\ 0, \text{ otherwise} \end{cases} \tag{30}$$

and

$$\eta_2 = \hat{\mathbf{d}}^T \mathbf{z} + \sqrt{(\hat{\mathbf{d}}^T \mathbf{z})^2 + \varsigma^2}, \tag{31}$$

where $\varsigma, \beta > 0$ are arbitrary constants. This projection operator, first proposed in Cai et al [22] enjoys the usefull properties

$$\tilde{\mathbf{d}}^T Proj(\mathbf{z}, \hat{\mathbf{d}}) \geq \tilde{\mathbf{d}}^T \mathbf{z}. \tag{32}$$

and

$$\left\| \hat{\mathbf{d}} \right\| \leq d_{max} + \beta, \forall t \geq 0. \tag{33}$$

Once again, consider the inner-loop tracking error and disturbance estimation error given by

$$\tilde{\mathbf{u}} := \mathbf{u} - \mathbf{u}_d, \tag{34}$$

$$\tilde{\mathbf{d}} := \mathbf{d} - \hat{\mathbf{d}}. \tag{35}$$

Proposition 2. *Consider the system described by* (10)*, the disturbance estimator dynamics given by* (29) *and the inner-loop tracking error given by* (34)*. Furthermore, consider the control law given by*

$$\mathbf{u}_d := -\hat{\mathbf{d}} + \mathbf{h}(\gamma) v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma} v^{coord}(t) - \mathbf{e}_v K_v - \mathbf{e}_p K_p. \tag{36}$$

$$\ddot{\gamma} := -k_\gamma e_\gamma + \dot{v}_d(\gamma, t) + \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} + \mathbf{z}^T \left(\mathbf{h}(\gamma) + K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma} \right), \tag{37}$$

where $K_p, K_v \succeq 0$ and k_γ is a positive gain. For sufficiently small initial position and velocity errors ($\mathbf{e}_p$, $\mathbf{e}_v$), and a sufficiently large separation between the time-scales of the inner and outer loop systems, it can be guaranteed that the system error converges to a neighbourhood of zero. The proposed control law solves problem 1 for the quadrotor vehicle.

Remark: A more in-depth and quantitative overall stability analysis can be conducted for the inner-outer loop control system, but this will be dependent directly on the type of inner-loop adopted. This results from the fact that the desired accelerations $\mathbf{u_d}$ must be decoupled in a set of desired thrust and attitude for the quadrotor to track. Given that this analysis is out of the scope of this work, we assume that quadrotor is equipped with a generic inner-loop that is capable of keeping the tracking error $\tilde{\mathbf{u}}$ small and bounded.

Proof: See appendix B.

5. Cooperative Path Following

In this section, the problem of CPF is addressed. The end goal is to have an algorithm that allows one quadrotor and multiple ASVs to perform a path following mission cooperatively, using a distributed architecture. The vehicles are required to execute their mission according to a fixed geometric configuration. To cope with limitations imposed by real environments where inter-vehicle communications are discrete, an Event-Triggered Communications (ETC) mechanism is adopted, based on previous work developed by A. Aguiar and A. Pascoal [23] and N. Hung and F. Rego [13].

5.1. Synchronization Problem with Event-Triggered Communications

Consider a group of $N \in \mathbb{R}^+ \setminus \{1\}$ autonomous vehicles/agents in a network that can be described mathematically by a digraph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$, consisting on N vertices, a set of directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where the edge ε_{ij} represents the flow of information from agent i to agent j, and a weighted adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$. Furthermore, each vehicle i is able to receive information from its neighbours in $\mathcal{N}_i^{in}$ and send information to its neighbours in $\mathcal{N}_i^{out}$, i.e. $\mathcal{G}$ is undirected. Moreover, consider that the communication topology of the vehicles is fixed, hence the Laplacian L associated to $\mathcal{G}$ is constant. Let the state vector of the system be composed by the path parameter of each individual vehicle $\gamma = [\gamma_1, ..., \gamma_N]^T$. In addition, each vehicle is equipped with PF controllers proposed in section 4 and has an assigned path to follow, appropriately parameterized in order to ensure that a given desired formation between the vehicles is met. The CPF problem consists in designing a distributed control scheme that adjusts the speed of the vehicles such that all path parameters γ reach a consensus. Consider the problem formulation below.

Problem 2. *For each agent i, with $i = 1, ..., N$ derive a consensus protocol for the speed correction term $\mathbf{v}^{\mathbf{coord}} = [v_1^{coord}, ..., v_N^{coord}]^T$ such that $\lim_{t \to \infty} |\gamma_i - \gamma_j| = 0, \forall j \in N_i^{in}$, and the formation of vehicles achieves the desired speed assignment $\mathbf{v_L}(\gamma) = [v_{L1}, ..., v_{LN}]^T$ as $t \to \infty$.*

Note that, according to the previously developed (PF) controllers, for each vehicle i, $|\dot{\gamma}_i - v_d(\gamma, t)| = 0$ is only guaranteed as $t \to \infty$, as the controlled variable is $\ddot{\gamma}_i$ and not $\dot{\gamma}_i$. Having this fact in mind, and assuming that the vehicles have already converged to their desired paths, i.e. $e_p \approx 0$ (and $e_v \approx 0$ in the case of the quadrotor), then the following simplifying assumption can be made:

Assumption 1. *The speed progression of all the virtual targets along the desired path is always assumed to be modelled by a single integrator system, which can be expressed in vectorial form as*

$$\dot{\gamma} = \mathbf{v_d}(\gamma, t) = \mathbf{v_L}(\gamma) + \mathbf{v^{coord}}. \tag{38}$$

Let the synchronization error vector be defined as $\boldsymbol{\varepsilon} = [\varepsilon_1, ..., \varepsilon_N]^T$ where, for each i,

$$\varepsilon_i := \sum_{j \in \mathcal{N}_i^{in}} a_{ij}(\gamma_i - \gamma_j), \tag{39}$$

with a_{ij} elements of the weighted adjacency matrix that describes the vehicle network. This error can also be expressed in vectorial form as

$$\boldsymbol{\varepsilon} := L\gamma, \tag{40}$$

where ε_i denotes the coordination error between vehicle i and its neighbours. With the above notation, the coordination error dynamics of the multi-vehicle system are given by

$$\dot{\boldsymbol{\varepsilon}} := L\dot{\boldsymbol{\gamma}}. \tag{41}$$

In the work of N. Hung and F. Rego [13], the authors propose a scheme where each agent i has a set of estimators $\hat{\gamma}_j, j \in \mathcal{N}_i^{in}$ for the true state of each in-neighbour virtual target γ_j. In addition, each agent i has an estimator for its own state $\hat{\gamma}_i$ which is reset whenever vehicle i broadcasts its true state γ_i. The other estimators are reset whenever agent i receives the true state from its in-neighbours $j \in \mathcal{N}_i^{in}$. In this work, a time-dependent broadcast condition is adopted.

Proposition 3. *Consider the distributed control law given by*

$$v_i^{coord} := -k_\varepsilon \sum_{j\in\mathcal{N}_i^{in}} a_{ij}(\gamma_i - \hat{\gamma}_j), \tag{42}$$

where $k_\varepsilon > 0$ and $\hat{\gamma}_j$ is vehicle's i estimate of vehicle's j real virtual target value. Consider also that the bank of estimators that each vehicle i is running is described by the dynamics equation

$$\dot{\hat{\gamma}}_i := v_L(\hat{\gamma}_i). \tag{43}$$

At any time instant t, under negligible transmission delays, the vehicle's j self-state estimate $\hat{\gamma}_j$ is equal to vehicle's i estimate of $\hat{\gamma}_j$, which allows us to express the estimator dynamics using vectorial notation as

$$\dot{\hat{\boldsymbol{\gamma}}} := \mathbf{v_L}(\hat{\boldsymbol{\gamma}}), \tag{44}$$

where, $\hat{\boldsymbol{\gamma}} = [\hat{\gamma}_1, ..., \hat{\gamma}_N]^T$ is the self-estimate of the virtual target of each vehicle. Let $\tilde{\boldsymbol{\gamma}} = [\tilde{\gamma}_1, ..., \tilde{\gamma}_N]^T$ denote the local estimation errors of each vehicle, such that $\tilde{\boldsymbol{\gamma}} = \boldsymbol{\gamma} - \hat{\boldsymbol{\gamma}}$. Then, $\mathbf{v^{coord}}$ can also be given in vectorial notation, according to

$$\mathbf{v^{coord}} := -k_\varepsilon[D\boldsymbol{\gamma} - \mathcal{A}\hat{\boldsymbol{\gamma}}] = -k_\varepsilon(\boldsymbol{\varepsilon} + \mathcal{A}\tilde{\boldsymbol{\gamma}}). \tag{45}$$

where D is a diagonal matrix and A the graph adjacency matrix. Consider also a triggering function used to define when to broadcast the along-path position of the virtual target of each vehicle, defined as

$$\begin{cases} \delta_i(t) := |\tilde{\gamma}_i(t)| - g_i(t) \\ \tilde{\gamma}_i(t) = \hat{\gamma}_i(t) - \gamma_i(t) \end{cases}, \tag{46}$$

where $\tilde{\gamma}_i(t)$ is the local estimation error of agent i and $g_i(t)$ is a time-dependent threshold function, such that if the estimation error exceeds this threshold, i.e. $\delta_i(t) \geq 0$, vehicle i broadcasts its state to the out-neighbours $\mathcal{N}_i^{out}$ and resets its local estimator. Furthermore, consider $g_i(t)$ to belong to a class of non-negative functions, given by

$$g_i(t) = c_i + b_i e^{-\alpha_i t}, \tag{47}$$

with c_i, b_i and α_i positive constants and $\mathbf{g}(t) = [g_1, ..., g_N]^T$ the collection of functions g_i for each individual vehicle i. Consider also that $\mathbf{v_L}(\gamma) = v_L\mathbf{1} + \tilde{\mathbf{v}}_\mathbf{L}$, where $\tilde{\mathbf{v}}_\mathbf{L}$ is a bounded and arbitrarily small term that accounts for a transient period in which the vehicles are on different sections of the path, with slightly different desired speed profiles. Then, under Assumption 1, the system is ISS with respect to the error vector $\boldsymbol{\varepsilon}$ and the inputs $\tilde{\boldsymbol{\gamma}}$ and $\tilde{\mathbf{v}}_\mathbf{L}$.

Proof: See appendix C.

The proposed control scheme used for achieving CPF using ETC is summarized in algorithm 1.

Algorithm 1 Event Triggered Communication for vehicle i (adapted from [24])

```
At every time instant $t$, each vehicle $i$ follows the procedure:
procedure COORDINATION AND COMMUNICATION
    if $\delta_i(t) \geq 0$ where $\delta_i$ is computed using (46) and (47) then
        Broadcast $\gamma_i(t)$;
        Reset the estimator $\hat{\gamma}_i$;
    if Receive a new message from agent $j \in \mathcal{N}_i^{in}$ then
        Reset $\hat{\gamma}_j(t)$;
    Run the estimators according to (43);
    Update the first order control protocol $\boldsymbol{u}_i$ using (42);
```

Given the general distributed control scheme, we now elaborate and address a specific formation in the context of this work in the sections that follow.

6. Path Planning

This section addresses the problem of generating a set of smooth and planar reference paths for each individual vehicle to follow with the end goal of encircling the boundary of a chemical spill. In order to make the vehicles follow the dynamic boundary according to a pre-defined formation (such as a triangle) multiple paths should be generated from one reference path that encodes the boundary. Borrowing from the work of Saldaña et al. [7], we start by presenting a rigorous mathematical definition of a dynamic boundary below.

Definition 1. *A dynamic boundary is a set of planar points Ω_t such that $\forall z \in \Omega_t$, and for any $\xi > 0$, the open disk centered at point z with radius ξ contains points of Ω_t and its complement set Ω_t^C. Moreover, the dynamic boundary can be approximated by a parametric closed curve (Jordan Curve) $\mathbf{C}(\gamma, t) : [0, \infty) \times [0, \infty) \to \mathbb{R}^2$, mapped by a parameter $\gamma \in \mathbb{R}_0^+$ and time $t \in \mathbb{R}_0^+$. The curve is continuous with no self-intersecting points, and changes smoothly with respect to both time t and parameter γ, as depicted in Figure 4 a).*

Since the chemical spill boundary is assumed to be dynamic, a path planning problem can be formulated in which a quadrotor is actively re-planning the path that the ASVs should follow at the water surface, as the group vehicles moves along it and more up-to-date data is acquired by the quadrotor's vision system. Consider therefore problem 3.

Problem 3. *Consider a quadrotor flying over a body of water at a pre-defined fixed altitude, equipped with a camera sensor pointing downwards with a fixed pitch angle relative to the vehicle's body reference frame $\{B\}$. Consider also that the vehicle is capable of detecting the boundary of a chemical spill in the 2-D image provided by the camera sensor. Furthermore, one or more ASVs at the surface of the water are required to follow a path dictated by the quadrotor, according to a pre-defined vehicle formation. As the quadrotor detects the dynamic boundary in the image:*

1. *use the data provided by its navigation system to convert the pixels to a 2-D point cloud expressed in the inertial frame $\{U\}$;*
2. *remove outliers and perform pre-processing on the 2-D point cloud;*
3. *generate a smooth and planar reference path, by formulating an online optimization problem that fits the data with open uniform B-splines;*
4. *send the updated path to the vehicle network;*
5. *make each vehicle generate an unique path for itself, capturing the pre-defined vehicle formation;*
6. *repeat the process.*

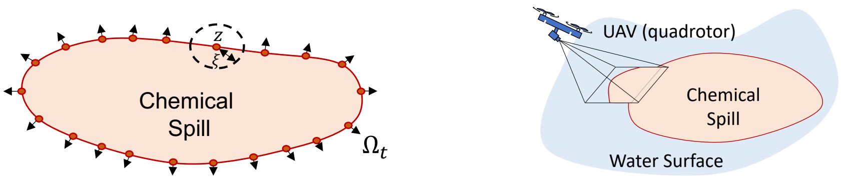
In order to solve problem 3 a few simplifying assumptions are made:

Assumption 2. *The dynamic boundary is located at the ocean's surface, assumed to be a 2-D plane at $Z_U = 0$ in the inertial frame of reference $\{U\}$.*

Assumption 3. *The quadrotor has a navigation system that can track the vehicle's pose with good accuracy.*

Assumption 4. *The quadrotor has a limited field of view of the environment, i.e, the camera sensor might not be able to capture the entire chemical spill boundary, but rather sections of it, according to Figure 4 b).*

Assumption 5. *The detection of the pixels that encode the boundary in the image frame is a sub-system that is assumed to be already available, such as the one proposed in [25].*



(a) **(b)**

Figure 4. Dynamic Boundary schematic: (**a**) Boundary formal definition (**b**) Drone's field of view.

6.1. Planar Point Cloud Generation

The camera model adopted is characterized by: i) a set of extrinsic parameters, which model the conversion between coordinates expressed in the world/inertial reference frame $\{U\}$ and the camera reference frame $\{C\}$; ii) intrinsic parameters which describe how a set of points in $\{C\}$ are represented in the image frame, according to Figure 5.

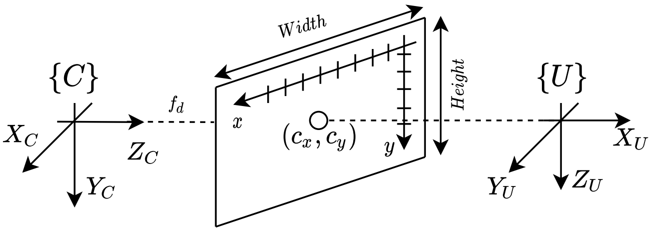


Figure 5. Camera model and reference frames

The intrinsic parameters consist of the focal distance f_d, the scale factors (s_x, s_y) in the X and Y-axis respectively, and (c_x, c_y) which correspond to the offset of the focal point in the image plane. These parameters can be obtained a priori by resorting to a camera calibration process, described in detail in [26]. Combining together the matrices of intrinsic parameters K, also known as the full-rank calibration matrix, and the matrix of external parameters ${}^{C}_{U}[R|T]$ and expressing the inertial frame coordinates as homogeneous coordinates, the transformation between inertial frame and camera plane is described by

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_d s_x & 0 & c_x \\ 0 & f_d s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} {}^{C}_{U}R & {}^{C}_{U}\boldsymbol{T} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}}_{{}^{C}_{U}[R|T]} \begin{bmatrix} X_U \\ Y_U \\ Z_U \\ 1 \end{bmatrix}, \tag{48}$$

where x and y denote the coordinates in the image frame and λ is a scale factor. It is important to mention that ${}^{C}_{U}[R|T]$ results from a series of successive rigid-body transformations (rotations and translations) given by

$$ {}^{C}_{U}[R|T] = {}^{C}_{B}[R|T]\,{}^{B}_{U}[R|T], \tag{49} $$

where ${}^{B}_{U}[R|T]$ denotes the conversion of coordinates expressed in the inertial frame $\{U\}$ to the quadrotor's body frame $\{B\}$, provided by its navigation system and ${}^{C}_{B}[R|T]$ is a matrix known a priori, as the camera attached to the vehicle is assumed to be fixed. The intrinsic and extrinsic parameters can be aggregated in a matrix Ω according to

$$ \Omega = K \cdot {}^{C}_{U}[R|T]. \tag{50} $$

In order to convert a given set of pixels (x, y) that encode the chemical spill boundary in the image frame to a point cloud expressed in the inertial frame, depth information about the scene is required. Taking into consideration assumption 2, all the points in the inertial frame will lie on the plane described by $Z_U = 0$, which solves the depth requirement. Moreover, from assumption 3 it can be concluded that the linear system of equations (48) is well defined and can be inverted such that for each pixel representing the boundary of the chemical spill, X_U and Y_U are extracted from

$$ \frac{1}{\lambda}\begin{bmatrix} X_U \\ Y_U \\ 1 \end{bmatrix} = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_4 \\ \Omega_5 & \Omega_6 & \Omega_8 \\ \Omega_9 & \Omega_{10} & \Omega_{12} \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{51} $$

Remark: This methodology relies heavily on the assumption that the quadrotor has a good navigation system, since small estimation errors in the altitude of the vehicle can lead to errors of several meters in the generated point cloud.

6.2. Pre-Processing the Planar Point Cloud

Before using the 2-D point cloud to generate a path, it is important to pre-process the information provided in it. Consider for instance, the example in Figure 6 where the quadrotor produces a 2-D point cloud, representing the boundary to be followed, at an arbitrary time-step t_k. In the point cloud, some points represent the chemical spill boundary in a region that is close to the vehicle - the region of interest, i.e. where the main cluster of points is expected to be located (in region B). The separation between regions A and B is defined by drawing a normal to path at the point where the re-planning starts. Some points are outliers and other points represent regions of the boundary that were partially occluded. The latter are seen as disconnected from the main cluster and should also be interpreted as outliers, disregarding them in the path planning process.

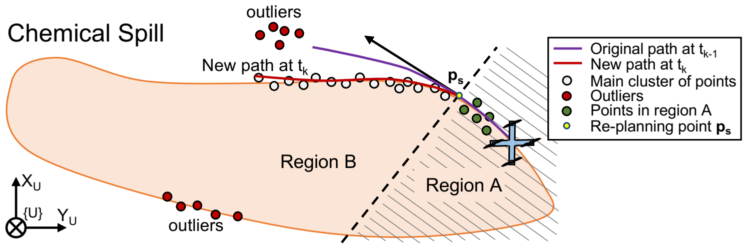


Figure 6. Pre-processing the point cloud and re-planning schematic

Unlike conventional motion planning problems, the main cluster of points does not have an explicit ordering yielding a sequence of waypoints that the vehicle should visit sequentially in time - this information must be inferred. On the other hand, it is possible to define explicitly where the path re-planning process starts - at a point $\mathbf{p}_s := C(\gamma_s)$ arbitrarily further ahead of the drone's position on the current curve $C(\gamma)$, such that

$\gamma_{drone} \leq \gamma_s$. Motivated by this example, and inspired by the work of Liu Y. et al. [27], the following pre-processing steps are introduced:

- Remove unused points that are behind the point $\mathbf{p}_s$, i.e. points in region A;
- Order the remaining set of points and remove outliers in region B;

6.2.1. Removing Unused Points

Consider $\mathbf{p}_s \in \mathbb{R}^2$ to be the point at which the path re-planning starts. In order to remove the points that are in region A, consider that ψ_s is the tangent angle to the current path at $\mathbf{p}_s$. A coordinate transformation can be applied to the 2-D point cloud $\boldsymbol{X} := \{\boldsymbol{X}_m\}_{m=1}^{M} \in \mathbb{R}^2$, such that in a new reference frame, points that are behind $\mathbf{p}_s$ (in region A) have a negative X-coordinate. This coordinate transformation is given by

$$\boldsymbol{X}_m^{\circ} = R(\psi_s) \cdot (\boldsymbol{X}_m - \mathbf{p}_s), \forall m = 1, ..., M \tag{52}$$

where $\boldsymbol{X}_m^{\circ} = [\boldsymbol{X}_m^{\circ x}, \boldsymbol{X}_m^{\circ y}]^T$. Each point $\boldsymbol{X}_m$ is discarded if $\boldsymbol{X}_m^{\circ x} < 0$. The points that belong to set $\boldsymbol{X}$ and are not discarded, should be saved in a new set $\boldsymbol{X}^{\star} := \{\boldsymbol{X}_j\}_{j=1}^{J} \in \mathbb{R}^2$ with $J \leq M$. The pseudocode is shown in algorithm 2.

Algorithm 2 Remove points "behind" the re-planning point

```
Obtain a new 2-D point cloud $\boldsymbol{X} := \{\boldsymbol{X}_m\}_{m=1}^{M} \in \mathbb{R}^2$;
Define $\mathbf{p}_s$ as the desired initial point for the re-planning to start;
Define $\psi_s$ as the tangent angle to the current path at $t_k$ at $\mathbf{p}_s$;
Follow the procedure:
procedure REMOVE UNUSED POINTS($\boldsymbol{X}$, $\mathbf{p}_s$, $\psi_s$)
    for $m = 1, ..., M$ do
        Compute $\boldsymbol{X}_m^{\circ}$ according to (52);
        if $\boldsymbol{X}_m^{\circ x} < 0$ then
            Discard $\boldsymbol{X}_m$;
    return the new set $\boldsymbol{X}^{\star} := \{\boldsymbol{X}_j\}_{j=1}^{J} \in \mathbb{R}^2$ with $J \leq M$.
```

6.2.2. Ordering a Set of Points and Removing Outliers

In order to avoid clustering outliers, reduce the point cloud to a curve-like shape and extract some implicit ordering from the data, Lee I. [28] proposes an algorithm that seeks to extract a structure "as simple as possible" from the data, by resorting to an Euclidean Minimum Spanning Tree (EMST). Consider the unordered set of points $\boldsymbol{X}^{\star}$ obtained previously and a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $\mathcal{V} = \{\boldsymbol{X}_j = (x_j, y_j) | j = 1, ..., J\}$ and $\mathcal{E} = \{(\boldsymbol{X}_i, \boldsymbol{X}_j) | i, j = 1, ..., J, i \neq j\}$. The EMST is a tree that connects all points in $\mathcal{G}$ with the weight of its edges corresponding to the Euclidean distance between each pair of points, that can be computed according to the very popular Kruskal's algorithm. In order to reduce the time complexity of this process, a threshold distance N_J can be used to define whether each pair of points is connected and use a KDTree [29] to compute a sparse graph $\mathcal{G}$ where each point has a limited set of neighbours, as shown in Figure 7.

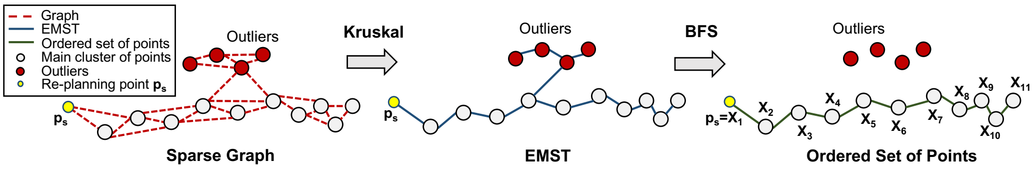


Figure 7. From sparse graph to an ordered list of points (example)

To get rid of outliers and define a coarse path to follow, Breadth First Search (BFS) can be applied to the EMST, starting from $\mathbf{p}_s$. The resulting ordered list of points that forms the path with the highest number of points should be saved in a new ordered set $\boldsymbol{X}^{\dagger} := \{X_k\}_{k=1}^{K} \in \mathbb{R}^2$. The proposed steps are summarized in algorithm 3.

Algorithm 3 Order a set of 2-D points

```
Add the desired initial point for the path $\mathbf{p}_s$ to $\boldsymbol{X}^{\star}$;
Define a threshold distance for the neighbours $N_J$;
Follow the procedure:
procedure ORDER POINTS($\boldsymbol{X}^{\star}$, $N_J$)
    Construct a KDTree from $\boldsymbol{X}^{\star}$ and use $N_J$ as a distance threshold;
    Create a graph $\mathcal{G}$ with $J$ vertices and no edges;
    for $\boldsymbol{X}_j$, $j = 1, ..., J$ do
        Query the KDTree for the neighbours of $\boldsymbol{X}_j$ and their euclidean distances;
        Add the corresponding edges to the graph $\mathcal{G}$;
    Compute the MST of the graph $\mathcal{G}$ starting from vertex corresponding to $\mathbf{p}_s$;
    Compute the path with the highest number of points, starting at $\mathbf{p}_s$ using BFS;
    return the new ordered set of points $\mathbf{X}^{\dagger} := \{\mathbf{X}_k\}_{k=1}^{K} \in \mathbb{R}^2$.
```

6.3. Path Generation - Approximating the point cloud with a parametric curve

In order to have a suitable representation of a path that the proposed controllers can follow, it is a requirement to generate a curve that is smooth and at least $\mathcal{C}^2$. In order to fulfil this requirement, the ordered set of points produced previously can be approximated by non-clamped uniform cubic B-Splines, composed of multiple spline segments where each segment is paramaterized by $\gamma \in [0, 1)$.

6.3.1. Define the number of segments to use

Consider now the ordered sequence of K points obtained via the application of algorithms 2 and 3 to the original 2-D point cloud. In order to fit the points with a parametric curve we are required to attribute to each point $\boldsymbol{X}_k \in \mathbb{R}^2$ a corresponding γ_k in the target parametric curve. This problem could be formulated as a nonlinear optimization problem - computationally demanding to solve for real-time applications. A non-optimal but more efficient solution proposed by Liu M. et al. [30] for Simultaneous Localization and Mapping (SLAM) applications is to consider D_X to be the total distance between the points, given by

$$D_X := \sum_{k=2}^{K} \|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|, \tag{53}$$

and the corresponding vector of parametric values $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_k]^T$ to be given by

$$\begin{cases} \gamma_1 = 0, \\ \gamma_k = \gamma_{k-1} + \frac{\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|}{D_X}\gamma_{max}, k = 2, ..., K, \end{cases} \tag{54}$$

where γ_{max} is the maximum parameter value of the parametric curve. For cubic B-splines, this number depends directly on the number of control points N_C that the target curve will have, such that $\gamma_{max} = N_C - 3$. The number of control points also dictates how many spline segments are used for the fitting problem. The optimal number of control points can be obtained by solving yet another nonlinear optimization problem, but due to the real time nature of the problem this option is disregarded. Given that a uniform cubic B-spline must have at least 4 control points to define one segment and that a low number of sections can under-fit a long set of points whilst a high number lead to over-fitting issues, this number should not be a static constant either. A non-optimal, yet dynamic way of defining the number of control points N_C is by taking:

$$N_C := max\left\{ \left\lfloor \frac{D_X}{\rho} \right\rceil, 4 \right\}, \tag{55}$$

with $(1/\rho) > 0$ a control points density (tunning parameter defined a priori). A smaller ρ leads to a higher N_C. Applying this method to the previous example, and considering $N_C = 7$, then $\gamma_{max} = \gamma_{11} = 4$.

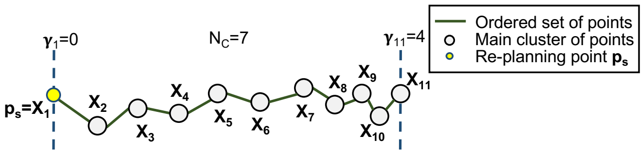


Figure 8. Ordered set of points with parametric values associated to them (example)

6.3.2. Fitting the points with a uniform cubic B-spline

For fitting the ordered set of points $\boldsymbol{X}^{\dagger}$ with a non-clamped uniform cubic B-Spline $C(\gamma, \mathbf{P})$, an optimization problem is formulated. Consider the objective function given by

$$f(\mathbf{P}) := \underbrace{\sum_{k=1}^{K} \|\mathbf{C}(\gamma_k, \mathbf{P}) - \mathbf{X}_k\|^2}_{goal} + F_r, \tag{56}$$

with

$$F_r = \underbrace{\lambda \int_0^{\gamma_{max}} \left\| \frac{\partial \mathbf{C}(\gamma, \mathbf{P})}{\partial \gamma} \right\|^2 d\gamma + \beta \int_0^{\gamma_{max}} \left\| \frac{\partial^2 C(\gamma, \mathbf{P})}{\partial \gamma^2} \right\|^2 d\gamma}_{\text{regularization term}}, \tag{57}$$

where $\mathbf{P} = [P_0^x, ..., P_{N_c-1}^x, P_0^y, ..., P_{N_c-1}^y]^T \in \mathbb{R}^{2N_c}$ is the vector of control points that defines the target curve. The first term minimizes the distance between the target B-Spline curve and the set of points whilst F_r is a regularization term and $\alpha, \gamma \geq 0$ are the regularization variables. The integral of the L_2^2 norm of the first derivative penalizes the total length of the curve and while the integral of the L_2^2 norm of the second derivative penalizes bends in the path. This objective function can also be expressed using vector notation, according to

$$f(\mathbf{P}) = \underbrace{\|B(\gamma)\mathbf{P} - \mathbf{X}\|^2}_{goal} + \underbrace{\lambda \mathbf{P}^T R_1 \mathbf{P} + \beta \mathbf{P}^T R_2 \mathbf{P}}_{\text{regularization term}}, \tag{58}$$

where $\mathbf{X} = [X_1^x, ..., X_K^x, X_1^y, ..., X_K^y]$ denotes points to fit, and R_1, R_2 are constant matrices that can be computed numerically (see appendix D).

In order to define the new path, it would not suffice to discard the previously planned curve defined after γ_s and minimize the objective function with respect to the control points. To guarantee $\mathcal{C}^2$ continuity between the previous path and the newly planned one, linear equality constraints should be imposed on the values of $\mathbf{C}^{new}(0)$, $\mathbf{C}^{new'}(0)$ and $\mathbf{C}^{new''}(0)$ of the new curve. Moreover, it is a requirement to save the old curve up to γ_s, as it may still be in use by other vehicles in the network.

Consider the re-planning point $\mathbf{p}_s$ introduced previously, chosen such that it corresponds to the transition between the spline segment the virtual target of the drone is "sitting on" and the next segment, according to

$$\mathbf{p}_s = \mathbf{C}^{old}(\gamma_s) \text{ with } \gamma_s = \lceil \gamma_{drone} \rceil, \tag{59}$$

where γ_{drone} corresponds to the quadrotor's virtual target at time instant t_k. With this choice of γ_s it is possible to take advantage of the local support property of B-splines and simplify the equality constraints of the problem while at the same time simplifying the storage of the curves in memory.

Considering $\mathbf{p}_s$ dictated by (59), the old curve segments that are described by parametric values such as $\gamma \geq \gamma_s$ should be discarded and replaced by a newer curve. Since each curve segment is defined by only 4 control points, then discarding those segments is equivalent to removing control points with indexes $i \geq \gamma_s + 3$ from the old control points vector. This operation results in a vector given by

$$\mathbf{P}^{old} = [P_0^x, P_1^x, ..., P_{\gamma_s}^x, P_{\gamma_s+1}^x, P_{\gamma_s+2}^x, P_0^y, P_1^y, ..., P_{\gamma_s}^y, P_{\gamma_s+1}^y, P_{\gamma_s+2}^y]^T. \tag{60}$$

For the particular example in Figure 9, spline 1 (in green) should be discarded given that $\gamma_{drone} \in [0, 1)$, hence $\gamma_s = 1$ and spline 0 kept. To achieve this, all the control points with indexes $i \geq 1 + 3$ should be removed from the control points vector $\mathbf{P}^{old}$, i.e. $P_4 = (P_4^x, P_4^y)$.

Making use of the local support property once more, it is known that $\mathcal{C}^2$ continuity between two consecutive cubic spline segments is guaranteed, as long as the last 3 control points of the first segment coincide with the first 3 control points of the second segment. A trivial way of generating a new B-Spline with guarantees of $\mathcal{C}^2$ continuity in the transition with the old curve, without explicitly defining equality constrains on the derivatives of the function, is to solve the following optimization problem:

$$\mathbf{P}^{new} = \underset{\mathbf{P}^{new}}{\operatorname{argmin}} \, f(\mathbf{P}^{new})$$

subject to

$$\begin{bmatrix} P_0^{x\ new} \\ P_1^{x\ new} \\ P_2^{x\ new} \\ P_0^{y\ new} \\ P_1^{y\ new} \\ P_2^{y\ new} \end{bmatrix} = \begin{bmatrix} P_{\gamma_s}^x \\ P_{\gamma_s+1}^x \\ P_{\gamma_s+2}^x \\ P_{\gamma_s}^y \\ P_{\gamma_s+1}^y \\ P_{\gamma_s+2}^y \end{bmatrix}, \tag{61}$$

where $\mathbf{P}^{new} = [P_0^{x\ new}, ..., P_{N_C-1}^{x\ new}, P_0^{y\ new}, ..., P_{N_C-1}^{y\ new}]^T$ is a new control points vector.

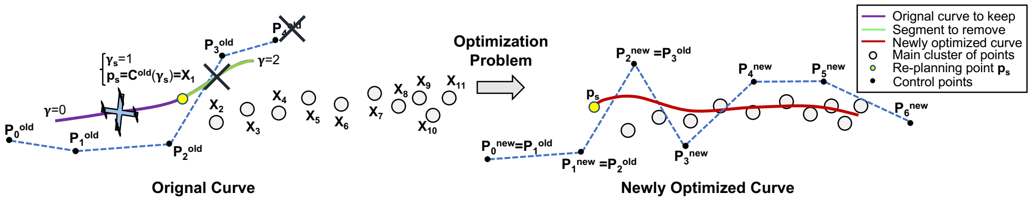


Figure 9. Solving the optimization problem (example)

To keep track of old and new curves, it is possible to just concatenate the new control points vector $\mathbf{P}^{new}$ with the old control points vector $\mathbf{P}^{old}$, ignoring the first three control points, i.e. P_0^{new}, P_1^{new} and P_2^{new}, which are repeated as a result of the equality constraints imposed by (61). Applying this methodology to the previous example, the final control points vector is given according to Figure 10.

doi:10.20944/preprints202202.0119.v1

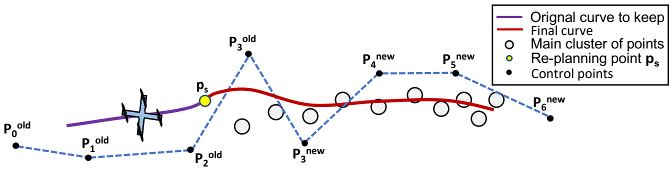


Figure 10. Final curve with control points concatenated (example)

These series of procedures are summarized in algorithm 4. For the sake of simplicity, the separation between X and Y-coordinates of the control points was omitted.

Algorithm 4 Fitting the points - growing uniform cubic B-spline

```
Compute $D_X$, $\gamma$ and $N_C$ according to equations (53), (54) and (55) respectively
Consider $\gamma_s = \lceil \gamma_{t_k} \rceil$ and the original control points vector:
    $$\mathbf{P} = \left[P_0, P_1, ..., P_{\gamma_s}, P_{\gamma_s+1}, P_{\gamma_s+2}, P_{\gamma_s+3}, P_{\gamma_s+4}..., P_n\right]^T; \quad (62)$$
Remove control points (corresponding to splines to be re-planned) from the original control points vector, such that:
    $$\mathbf{P}^{old} = \left[P_0, P_1, ..., P_{\gamma_s}, P_{\gamma_s+1}, P_{\gamma_s+2}\right]^T; \quad (63)$$
Solve the optimization problem in (61) and obtain a new vector with $N_C$ control points:
    $$\begin{gathered} \mathbf{P}^{new} = \left[P_0^{new}, P_1^{new}, P_2^{new}, ..., P_{N_C-1}^{new}\right]^T, \\ \text{with } P_0^{new} = P_{\gamma_s}, P_1^{new} = P_{\gamma_s+1}, P_2^{new} = P_{\gamma_s+2}; \end{gathered} \quad (64)$$
Concatenate the new vector with the old vector (ignoring the first 3 control points which are repeated):
    $$\mathbf{P}^{final} = \left[P_0, P_1, ..., P_{\gamma_s}, P_{\gamma_s+1}, P_{\gamma_s+2}, P_3^{new}, ..., P_{N_C-1}^{new}\right]^T. \quad (65)$$
```

6.4. From 2-D Path to Vehicle Formation

To generate individual paths for each vehicle to follow, we can consider a reference path (obtained via the application of the previous algorithms) and offsetting each point according to an expression that captures a desired vehicle formation. Start by considering a formation vector denominated $\boldsymbol{\mu}_i \in \mathbb{R}^3$ for each vehicle i, with each distance defined in the tangential reference frame $\{T\}$ to the virtual target's position in the original curve, according to Figure 11. According to Xie et al. [31], it is possible to define a desired path for each vehicle given by

$$\mathbf{p}_{di}(\gamma_i) = \mathbf{C}(\gamma_i) + {}^U_T R(\gamma_i)\boldsymbol{\mu}_i, \quad (66)$$

where $\mathbf{p}_{di}$ is the desired path for the vehicle i, $\mathbf{C}(\gamma_i)$ the planned curve and ${}^U_T R(\gamma_i)$ a rotation matrix computed according to

$${}^U_T R(\gamma_i) = [\mathbf{r}_1(\gamma_i), \mathbf{r}_3(\gamma_i) \times \mathbf{r}_1(\gamma_i), \mathbf{r}_3(\gamma_i)], \quad (67)$$

with

$$\mathbf{r}_1(\gamma_i) = \frac{\partial \mathbf{p}_d / \partial \gamma}{\|\partial \mathbf{p}_d / \partial \gamma\|}, \text{with } \|\partial \mathbf{p}_d / \partial \gamma\| \neq 0 \qquad \mathbf{r}_3(\gamma_i) = \frac{\mathbf{r}_d - (\mathbf{r}_d \cdot \mathbf{r}_1(\gamma_i))\mathbf{r}_1(\gamma_i)}{\|\mathbf{r}_d - (\mathbf{r}_d \cdot \mathbf{r}_1(\gamma_i))\mathbf{r}_1(\gamma_i)\|} \quad (68)$$

such that $\mathbf{r}_1$ is the tangent to the curve. Moreover, since all vehicles will only be required to operate in a 2-D plane, a trivial definition for one of the axis of the tangential frame $\{T\}$ is $\mathbf{r}_d = [0, 0, 1]^T$.

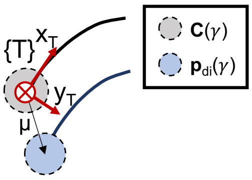


Figure 11. Formation vector

A path might not be not parameterized according to the arc length and, for B-Splines in particular, each spline segment is such that $\gamma \in [0, 1)$. Therefore, it is commonplace to define the a constant required speed $V \leq V_{max}$ for the vehicle and let the desired speed profile for the virtual targets be given by:

$$v_L(\gamma) = \frac{V}{\|\mathbf{p}_d'(\gamma)\|}. \tag{69}$$

7. Implementation Details

To evaluate the performance of the proposed PF and CPF algorithms applied to marine ASVs, real water trials were conducted at Doca dos Olivais (Lisbon, Portugal) using the Medusa-class of underactuated marine vehicles [14], shown in Figure 12. The vehicles used in the real trials were equipped with a GPS Astech MB100, a NavQuest600 Micro DVL and a Vectornav VN-100T Attitude and Heading Reference System (AHRS). The operating system used during development was Ubuntu 18.04LTS along with ROS Melodic.

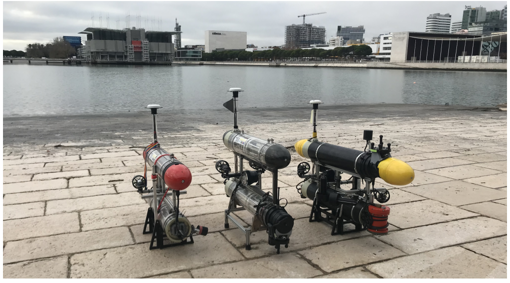


Figure 12. Real Medusa vehicles at Doca dos Olivais, Lisbon (Portugal)

To analyse the performance of the proposed online path planning algorithm, a realistic simulation environment that closely resembles the Doca dos Olivais site was developed and incorporated into Gazebo simulator. Given the main goal of having a fleet of vehicles encircling a chemical spill, is was necessary to overlay a red stain mesh on top of the ocean's surface.

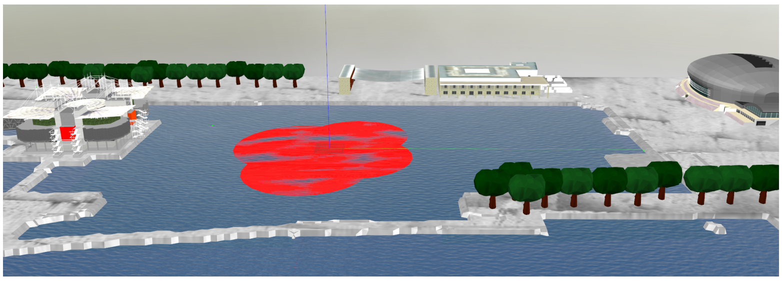
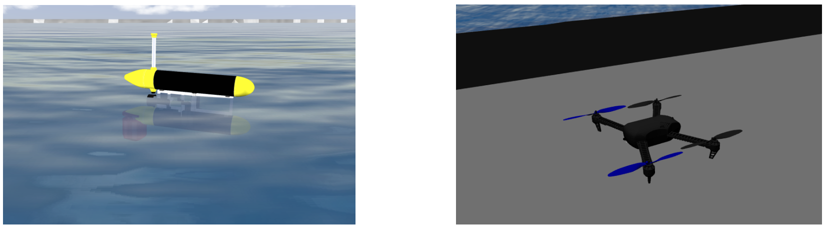


Figure 13. Simulated world of Doca dos Olivais with red chemical spill in gazebo

For simulating the Medusa ASVs, a CAD model of the vehicles was incorporated into the simulator, Figure 14 a). The virtual vehicle was also equipped with DVL, AHRS and GPS sensors provided by UUVSimulator plugin [16]. To simulate the quadrotor, the Iris vehicle provided by the PX4 SITL Gazebo Plugin [15] was used, see Figure 14 b).



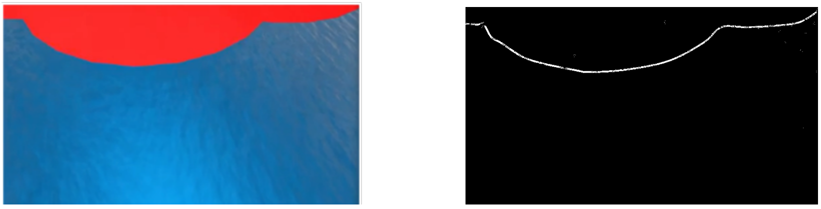
(a) (b)

Figure 14. Simulated vehicles in gazebo: (**a**) Medusa ASV (**b**) Iris quadrotor with a fixed camera.

The simulated quadrotor was equipped with a virtual camera mounted 21mm below the vehicle's center of mass with a pitch angle of -45° pointing downwards and produced an image with a resolution of 640 × 480 px, according to Figure 15 a). Its intrinsic parameters are given by

$$\begin{cases} (c_x, c_y) = (320.5, 240.5) \\ (f_d s_x, f_d s_y) = (381.4, 381.4) \end{cases} . \tag{70}$$

Given assumption 5, the detection of the boundary region between the spill and the ocean surface was out of the scope of this work. Therefore, we resorted to OpenCV library[32] to mask and threshold the red colours in the image feed. After this step, the Canny edge detection algorithm was applied to the binary image to retrieve the pixels corresponding to the boundary, according to Figure 15 b). To solve the optimization problem proposed in section 6.3.2, we resorted to Scipy's SQP solver [33].



(a) (b)

Figure 15. Simulated camera feed: (**a**) Quadrotor's camera output (**b**) Binary image.

The entire system architecture is shown in Figure 16. The inner-loop controls adopted for the quadrotor were the ones already provided by PX4 while for the ASV we resorted to PID inner-loop controllers to steer the vehicles.

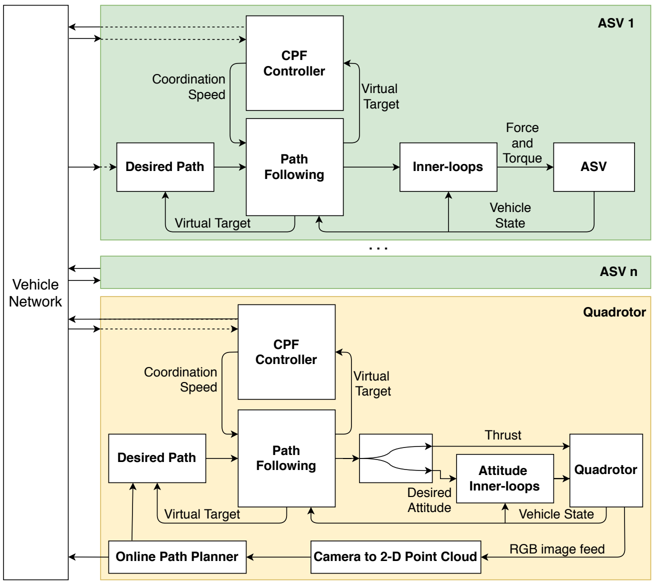


Figure 16. Planning and control architecture

doi:10.20944/preprints202202.0119.v1

8. Experimental and Simulation Results

In this section, we present some real experimental results regarding the PF and CPF controllers applied to two Medusa ASV vehicles. In addition, realistic 3-D simulation results are also presented for the case study where two Medusa vehicles were required to perform a CPF mission on a pre-defined path with a quadrotor, in a leader-follower formation. Finally, a third case study is presented, where a simulated quadrotor had to detect the boundary of a chemical spill, and plan in real-time a path for both itself and a Medusa ASV to follow cooperatively. The control gains adopted are available in appendix E.

8.1. CPF with ETC between 2 Medusa Vehicles (real)

For the real trial, performed at Doca dos Olivais (Lisbon, Portugal), two Medusa vehicles were required to perform a lawn-mowing mission cooperatively at the surface of the water, according to Figure 17. The black vehicle (Medusa 1) was required to follow the leader (Medusa 2) according to the formation vector $\boldsymbol{\mu} = [-5, -5, 0]^T$. Both vehicles were required to follow the path at $V = 0.5\text{m/s}$ and communications were bi-directional.

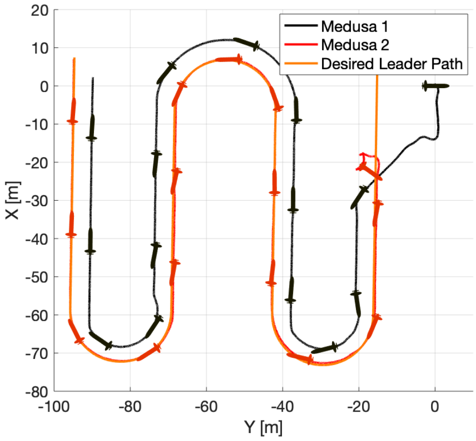
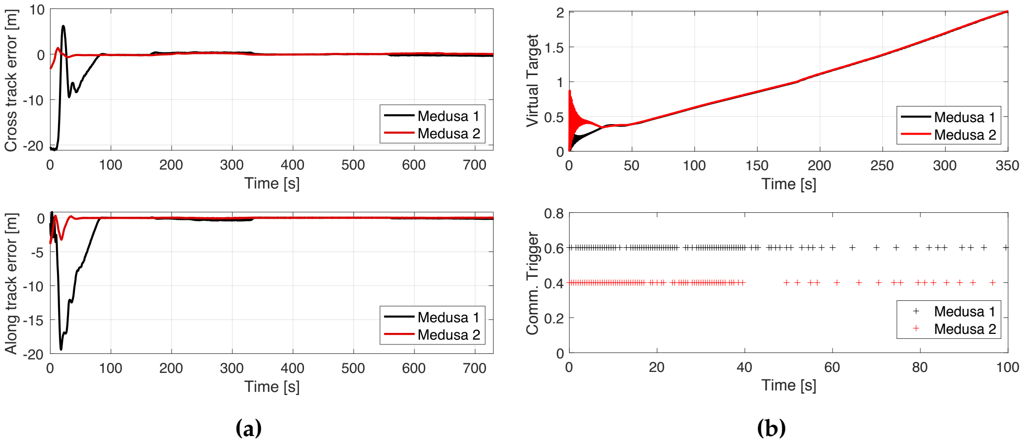


Figure 17. Real CPF mission with 2 Medusa vehiclse

According to the results in Figure 18 a), the along-track error of Medusa 1 increases quickly as the virtual target tries not only to minimize the distance to the vehicle but also its distance to its neighbour's (Medusa 2) virtual target. As the vehicles start to move, this error starts to decrease, and according to Figure 18 b), after approximately 50s, the vehicles align themselves according to the desired formation, approach the desired speed profile and, as a consequence, the rate of information exchange decreases.

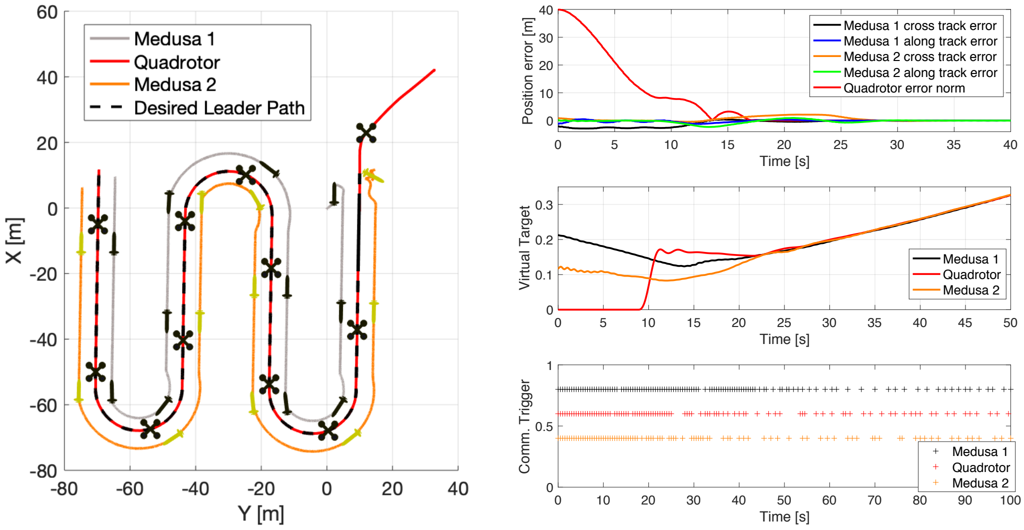


(a) **(b)**

Figure 18. CPF with 2 real Medusa vehicles: (**a**) X-Y view (**b**) Communication metrics.

8.2. CPF with ETC between a Quadrotor and Medusa Vehicles (simulation)

For this case study, a CPF mission was performed such that a simulated quadrotor and two Medusa vehicles were required to perform a lawn-mowing mission, according to Figure 19 a). In this experiment, the aircraft was required to fly at a fixed-altitude of 30m and the formation vector for Medusa 1 was given by $\boldsymbol{\mu}_1 = [-5, 5, 0]^T$m and for Medusa 2 $\boldsymbol{\mu}_2 = [-5, -5, 0]^T$m, leading to a triangular formation with 2 ASVs side by side, behind the quadrotor. In this experiment there was bi-directional communication between the pairs of vehicles: (quadrotor, Medusa 1) and (quadrotor, Medusa 2). From the results in Figure 19 b), it is observable that the vehicles converge to their desired formation at around 25s. After this period of time, the position error converges to a neighbourhood of zero and the virtual target speeds converge to their desired value. As a consequence, the number of communication events between the vehicles drops as the bank of observers in each vehicle can more accurately track the state of the virtual target of their peers.



(a) **(b)**

Figure 19. CPF with simulated Iris and Medusa vehicles: (**a**) X-Y view (**b**) Performance metrics.

8.3. Boundary Encircling with a Quadrotor and a Medusa Vehicle (simulation)

For the last simulated experiment, the quadrotor was required to start the same lawn-mowing that was adopted for the mission with one Medusa ASV. As soon as a chemical spill boundary was detected in the drone's image stream, the quadrotor was required to start the path planning algorithm at a rate of 1Hz and send the most up-to-date path to the ASV, according to Figure 20. The drone was required to follow the path at 30m of altitude with a desired constant speed of 0.5m/s. Since the quadrotor was equipped with a fixed-mounted camera, it was also required to align its yaw angle with the tangent to the path in order not to lose sight of the boundary being followed.

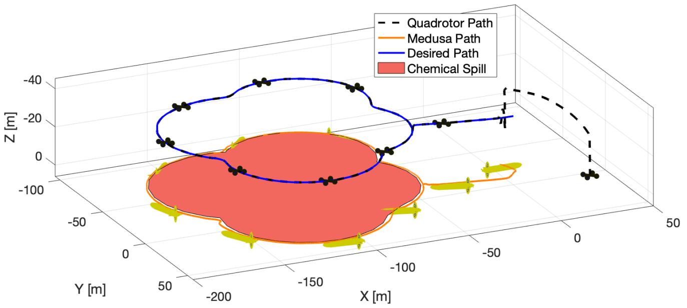
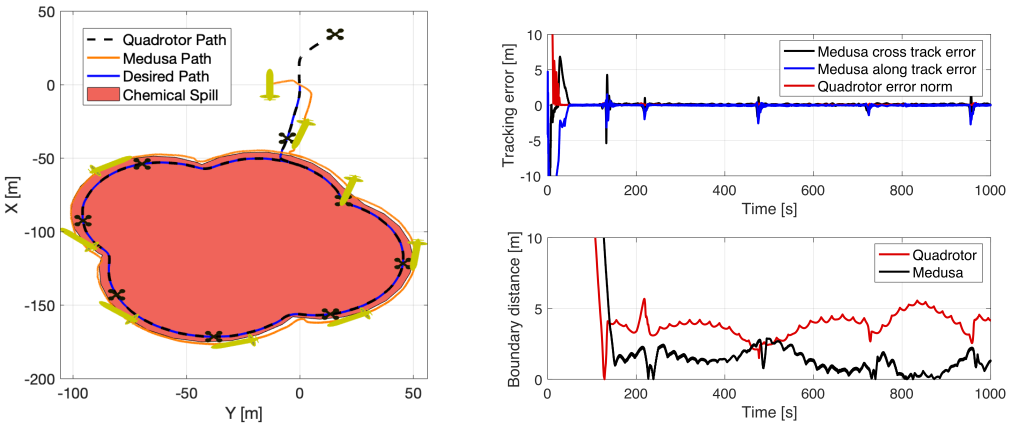


Figure 20. 3-D view of simulated boundary encircling mission with Iris and Medusa vehicles

In order to guarantee that the path further ahead could be generated for the ASV to follow, it was desirable for the marine vehicle to follow the the quadrotor from behind, i.e. with a formation vector $\mu = [-5, 5, 0]^T$m. In Figure 21 a) a top-down view of the executed mission is shown. In Figure 21 b), plots of the PF errors are provided along with the norm of the horizontal distance of each vehicle to the real boundary being followed. It is observable that the tracking error only increased in zones where the chemical spill had a crease. This is justified by the fact that the Medusa vehicle, when performing tight turns, was not able to cope with its virtual target speed and slowed down, leading to sudden spikes in along track error. These tracking errors were instantly compensated by the adaptive virtual target dynamics which attempted to minimize the distance between itself and the vehicle. It is also observable that the norm of the distance between the marine vehicle and the chemical spill is much lower, when compared to its aerial counterpart, with the Medusa always following the boundary from its outskirts, due to the formation vector adopted.



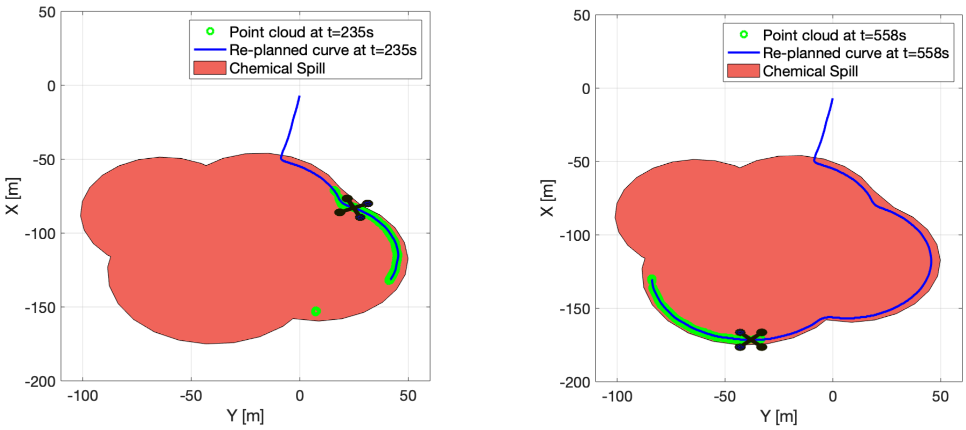
(a)

(b)

Figure 21. Boundary encircling with simulated Iris and Medusa vehicles: (**a**) X-Y view (**b**) Performance metrics.

From Figure 21 b), it is evident that the horizontal distance between the real drone's position and the boundary is bounded by 6m. This result is to be expected as the altitude estimates are mainly provided by the simulated GPS system and small errors in the estimated attitude, especially yaw angle, will lead to errors of several meters in the generated 2-D point cloud. Due to the type of application at hand, and given that it is typical to have errors of several meters in underwater scenarios, these errors are considered within an acceptable range. In addition, the small oscillations in the boundary distance plot result from the simulated chemical spill boundary mesh being a composition of discrete lines which are picked up by the drone's camera.

In Figure 22, a plot of the point cloud generated by the algorithm is shown at two different time instants (in green), as well as the corresponding planned B-spline paths (in blue). Note that in Figure 22 a) some of the green dots further away from the vehicle were discarded by the planning algorithm as they were too far away from the main cluster of points and therefore discarded.



(a) (b)

Figure 22. Path generation (**a**) Time=235s (**b**) Time=558s.

9. Conclusion and Future Work

This paper addressed the problem of encircling an environmental boundary caused by a chemical spill using a team of robots composed of an aerial quadrotor and Medusa marine vehicles. The path following problem was introduced, and a non-linear control law derived for the ASV, exploiting the technique described in P. Aguiar and F. Vanni [10–12]. Inspired by this control law, a new one was derived for a quadrotor following the same methodology, with some key differences due to the nature of the aircraft. For the section that followed, the CPF problem was formulated and a proposal to solve the problem was presented, such that the synchronization controller was distributed and the same for all vehicles (aerial and marine) using event-triggered communications based on previous work by N. Hung and F. Rego [13]. In addition, a new real-time path planning algorithm was developed that made use of the camera sensor onboard of the quadrotor to have a local view of the boundary and generate a point cloud expressed in the inertial frame. This data was then used to solve an optimization problem which generates a B-spline based path that grows dynamically as the drone moves along the boundary and acquires more data. The path is then shared with all ASV vehicles in the network in real time. The proposed algorithms were implemented in ROS and a 3-D virtual scenario was generated, allowing for a mixture of real and simulated results. Future work includes making the height at which the quadrotor operates dynamic and introducing curvature limits as inequality constraints to the path planning problem as well as obstacle avoidance before carrying out integrated experiments with real vehicles.

Author Contributions: The individual contributions of the authors are as follows: Conceptualization, software, formal analysis, investigation, writing, review and editing by M.J.; Conceptualization, validation, review, editing, project administration and funding acquisition by A.P and R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the H2020 EU Marine Robotics Research Infrastructure Network (Project ID 731103), the H2020-EU.1.2.2 FET Proactive RAMONES project (Grant agreement ID: 101017808). Funded was also provided by FCT, FEDER, and ERDF through the LARSyS UIDB/50009/2020, LISBOA-01- 0145-FEDER-031411, and EMSO-PT projects.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The path planning library that implements the algorithms proposed in section 6 is available at https://github.com/MarceloJacinto/BSplineFit.

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Preposition 1

Proof. Consider the candidate Lyapunov Function given by

$$V_1(\mathbf{e}_p) = \frac{1}{2}(\mathbf{e}_p - \delta)^T(\mathbf{e}_p - \delta) \tag{A1}$$

Taking the first time derivative of (A1) and replacing in (16) and (13) yields leads to

$$\dot{V}_1(\mathbf{e}_p) = (\mathbf{e}_p - \delta)^T\left(- S(r)(\mathbf{e}_p - \delta) + \Delta\mathbf{u} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}(e_\gamma + v_d(\gamma, t))\right). \tag{A2}$$

Taking into account the properties of the skew-symmetric matrix S, then

$$(\mathbf{e}_p - \delta)^T S(r)(\mathbf{e}_p - \delta) = 0. \tag{A3}$$

Replacing (19) and (20) in $\dot{V}_1$ yields

$$\begin{aligned} \dot{V}_1(\mathbf{e}_p) &= (\mathbf{e}_p - \delta)^T\left(\Delta(\tilde{\mathbf{u}} + \mathbf{u}_d) + \begin{bmatrix} 0 \\ v \end{bmatrix} + \tilde{\boldsymbol{v}}_c + \hat{\boldsymbol{v}}_c - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}(e_\gamma + v_d(\gamma, t))\right) \\ &= -(\mathbf{e}_p - \delta)^T K_p \sigma(\mathbf{e}_p - \delta) - (\mathbf{e}_p - \delta)^T {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} e_\gamma + \Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c. \end{aligned} \tag{A4}$$

By taking a backstepping approach, consider a second candidate Lyapunov function

$$V_2(\mathbf{e}_p, e_\gamma) = V_1(\mathbf{e}_p) + \frac{1}{2}e_\gamma^2. \tag{A5}$$

Taking the first derivative and replacing in the control law (21) for the virtual target we obtain

$$\begin{aligned} \dot{V}_2(\mathbf{e}_p, e_\gamma) =& \dot{V}_1(\mathbf{e}_p) + e_\gamma(\ddot{\gamma} - \dot{v}_d(\gamma, t)) \\ =& - (\mathbf{e}_p - \delta)^T K_p \sigma(\mathbf{e}_p - \delta) - k_\gamma e_\gamma^2 + \Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c \\ \leq& - (1 - \theta)(\mathbf{e}_p - \delta)^T K_p \sigma(\mathbf{e}_p - \delta) - \theta(\mathbf{e}_p - \delta)^T K_p \sigma(\mathbf{e}_p - \delta) \\ & - k_\gamma |e_\gamma|^2 + \|\mathbf{e}_p - \delta\| \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \end{aligned} \tag{A6}$$

where $0 < \theta < 1$. The term

$$\begin{aligned} & - \theta(\mathbf{e}_p - \delta)^T K_p \boldsymbol{\sigma}(\mathbf{e}_p - \delta) + \|\mathbf{e}_p - \delta\| \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \\ & = -\theta(\mathbf{e}_p - \delta)^T K_p \frac{\mathbf{e}_p - \delta}{\|\mathbf{e}_p - \delta\|}\sigma(\|\mathbf{e}_p - \delta\|) + \|\mathbf{e}_p - \delta\| \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\|, \end{aligned} \tag{A7}$$

will be ≤ 0 if

$$\theta\lambda_{min}(K_p)\sigma(\|\mathbf{e}_p - \delta\|) \geq \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\|, \tag{A8}$$

which in turn implies that

$$\|\mathbf{e}_p - \delta\| \geq \sigma^{-1}\left(\frac{1}{\theta\lambda_{min}(K_p)}\|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\|\right), \tag{A9}$$

and

$$\dot{V}_2 \leq -(1 - \theta)(\mathbf{e}_p - \delta)^T K_p \sigma(\mathbf{e}_p - \delta) - k_\gamma |e_\gamma|^2, \tag{A10}$$

as the right side of inequality (A9) can be made arbitrarily small through the choice of the gain matrix K_p. It follows directly from H. Khalil [34, Theorem 4.19] that the controlled system is ISS. □

Appendix B. Proof of Preposition 2

Proof. Consider the candidate Lyapunov function given by

$$V_1(\mathbf{e}_p) := \frac{1}{2}\mathbf{e}_p^T\mathbf{e}_p. \tag{A11}$$

By taking the first derivative of (A11) and replacing in (23), (24) and (25), yields

$$\dot{V}_1(\mathbf{e}_p) = -\mathbf{e}_p^T K_1 \mathbf{e}_p + \mathbf{e}_p^T\left(\mathbf{z} - \frac{\partial \mathbf{p}_d}{\partial \gamma}e_\gamma\right). \tag{A12}$$

With a view to applying backstepping techniques, consider:

$$V_2(\mathbf{e}_p, \mathbf{e}_v) := V_1(\mathbf{e}_p) + \frac{1}{2}\mathbf{z}^T\mathbf{z}. \tag{A13}$$

Replacing (25), (26) and (28) in $\dot{V}_2$ yields

$$\begin{aligned}\dot{V}_2 = &- \mathbf{e}_p^T K_1 \mathbf{e}_p + \mathbf{e}_p^T\mathbf{z} - \mathbf{e}_p^T\frac{\partial \mathbf{p}_d}{\partial \gamma}e_\gamma \\ &+ \mathbf{z}^T\left(\mathbf{u} + \mathbf{d} - \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) - \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) + K_1\mathbf{e}_v - K_1\frac{\partial \mathbf{p}_d}{\partial \gamma}e_\gamma\right).\end{aligned} \tag{A14}$$

By replacing (34), (35) and (36) in the Lyapunov time derivative it follows that

$$\dot{V}_2 = -\mathbf{e}_p^T K_1 \mathbf{e}_p - \mathbf{z}^T K_2 \mathbf{z} - \mathbf{e}_p^T\frac{\partial \mathbf{p}_d}{\partial \gamma}e_\gamma - \mathbf{z}^T\left(\mathbf{h}(\gamma) + K_1\frac{\partial \mathbf{p}_d}{\partial \gamma}\right)e_\gamma + \mathbf{z}^T(\tilde{\mathbf{u}} + \tilde{\mathbf{d}}). \tag{A15}$$

Consider a third candidate Lyapunov obtained by backstepping, defined as

$$V_3 := V_2 + \frac{1}{2}e_\gamma^2. \tag{A16}$$

Taking its derivative with respect to time and replacing in (37) we obtain

$$\dot{V}_3 = -\mathbf{e}_p^T K_1 \mathbf{e}_p - \mathbf{z}^T K_2 \mathbf{z} - k_\gamma e_\gamma^2 + \mathbf{z}^T(\tilde{\mathbf{u}} + \tilde{\mathbf{d}}). \tag{A17}$$

Considering one last backstepping that involves the construction of

$$V_4 = V_3 + \frac{1}{2}\tilde{\mathbf{d}}^T K_d^{-1}\tilde{\mathbf{d}}. \tag{A18}$$

Taking its time derivative and taking into consideration (32),

$$\begin{aligned}\dot{V}_4 &= -\mathbf{e}_p^T K_1 \mathbf{e}_p - \mathbf{z}^T K_2 \mathbf{z} - k_\gamma e_\gamma^2 + \underbrace{\tilde{\mathbf{d}}^T(\mathbf{z} - \mathrm{Proj}(\mathbf{z}, \hat{\mathbf{d}}))}_{\leq 0} + \mathbf{z}^T\tilde{\mathbf{u}} \\ &\leq -W(\mathbf{e}_p, \mathbf{e}_v, e_\gamma) + \mathbf{z}^T\tilde{\mathbf{u}}.\end{aligned} \tag{A19}$$

Assuming that the quadrotor is equipped with a generic inner-loop that is capable of keeping the tracking error $\tilde{\mathbf{u}}$ small and bounded, then the right side of inequality (A19) can be made small enough such that the controlled system is stable. A more in-depth stability analysis can be conducted for the inner-outer loop control system, but this will be dependent directly on the type of inner-loop adopted. This results from the fact that the desired accelerations $\mathbf{u_d}$ must be decoupled in a set of desired thrust and attitude for the quadrotor to track.

In order to simplify the designed control law $\mathbf{u}_d$, consider the final algebraic manipulation

$$\begin{aligned}\mathbf{u}_d^\diamond &= -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - K_1\mathbf{e}_v - \mathbf{e}_p - K_2\mathbf{z} \\ &= -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - \mathbf{e}_v\underbrace{(K_1+K_2)}_{K_v} - \mathbf{e}_p\underbrace{(I+K_1K_2)}_{K_p}.\end{aligned} \tag{A20}$$

□

Appendix C. Proof of Preposition 3

Proof. Considering that

$$\mathbf{v_L}(\gamma) = v_L\mathbf{1} + \tilde{\mathbf{v}}_\mathbf{L}, \tag{A21}$$

where $\tilde{\mathbf{v}}_\mathbf{L}$ is a bounded and arbitrarily small term that accounts for a transient period in which the vehicles are on different sections of the path, with slightly different desired speed profiles. Replacing (38), the speed correction term proposed in (45) and (A21) in (41) yields

$$\begin{aligned}\dot{\varepsilon} &= L(\mathbf{v_L}(\gamma) - k_\varepsilon(\varepsilon + \mathcal{A}\tilde{\gamma})) \\ &= v_L\cancelto{0}{L\mathbf{1}} + L\tilde{\mathbf{v}}_\mathbf{L} - k_\varepsilon L(\varepsilon + \mathcal{A}\tilde{\gamma}) \\ &= -k_\varepsilon L(\varepsilon + \mathbf{d}) \text{ with } \mathbf{d} = \frac{\tilde{\mathbf{v}}_\mathbf{L}}{k_\varepsilon} + \mathcal{A}\tilde{\gamma},\end{aligned} \tag{A22}$$

where $\mathbf{d}$ is a disturbance that results from combining the terms dependent on $\tilde{\mathbf{v}}_\mathbf{L}$ and $\tilde{\gamma}$. Consider the Laplacian matrix L expressed in canonical Jordan form as

$$L = V\Lambda V^{-1}, \tag{A23}$$

and the change of variables

$$\bar{\varepsilon} = V^{-1}\varepsilon. \tag{A24}$$

Applying (A24) to (A22) yields

$$\dot{\bar{\varepsilon}} = -k_\varepsilon\Lambda(\bar{\varepsilon} + \bar{\mathbf{d}}), \text{ with } \bar{\mathbf{d}} = V^{-1}\mathbf{d}. \tag{A25}$$

It is possible to decompose the above equality according to the notation

$$\left[\begin{array}{c}\dot{\bar{\varepsilon}}_1 \\ \hline \dot{\bar{\varepsilon}}_2\end{array}\right] = \left[\begin{array}{c}0 \\ \hline -k_\varepsilon\Lambda_2(\bar{\varepsilon}_2 + \bar{\mathbf{d}}_2)\end{array}\right], \tag{A26}$$

where the first half of the vector denotes the term that depends on the null eigen value of the Laplacian while the second term is a vector that depends only on the positive eigen values of the Laplacian. Consider now the candidate Lyapunov function

$$V_{\bar{\varepsilon}_2} = \frac{1}{2}\bar{\varepsilon}_2^T\bar{\varepsilon}_2, \tag{A27}$$

and its time derivative given by

$$\begin{aligned}\dot{V}_{\bar{\varepsilon}_2} &= -k_\varepsilon\bar{\varepsilon}_2^T\Lambda_2(\bar{\varepsilon}_2 + \bar{\mathbf{d}}_2) \\ &= -(1-\theta)k_\varepsilon\bar{\varepsilon}_2^T\Lambda_2\bar{\varepsilon}_2 - \theta k_\varepsilon\bar{\varepsilon}_2^T\Lambda_2\bar{\varepsilon}_2 - k_\varepsilon\bar{\varepsilon}_2^T\Lambda_2\bar{\mathbf{d}}_2\end{aligned} \tag{A28}$$

where $0 < \theta < 1$. The term

$$-\theta k_\varepsilon\bar{\varepsilon}_2^T\Lambda_2\bar{\varepsilon}_2 - k_\varepsilon\bar{\varepsilon}_2^T\Lambda_2\bar{\mathbf{d}}_2 \tag{A29}$$

doi:10.20944/preprints202202.0119.v1

will be ≤ 0 if

$$\|\bar{\varepsilon}_2\| \geq \frac{1}{\theta}\|\bar{\mathbf{d}}_2\|, \tag{A30}$$

and therefore

$$\dot{V}_{\bar{\varepsilon}_2} \leq -(1-\theta)k_\varepsilon \bar{\varepsilon}_2^T \Lambda_2 \bar{\varepsilon}_2. \tag{A31}$$

The term $\|\tilde{\gamma}\|$ can be made arbitrarily small by controlling the gains that dictate the broadcasting scheme. Moreover, the term $\tilde{\mathbf{v}}_{\mathbf{L}}$ can be dominated by a proper choice k_ε. Hence, $\|\mathbf{d}\|$ can be made arbitrarily small and thus $\|\bar{\mathbf{d}}_2\|$ as well. It follows directly from H. Khalil [34, Theorem 4.19] that the controlled system is ISS with respect to the error vector ε and the inputs $\tilde{\gamma}$ and $\tilde{\mathbf{v}}_{\mathbf{L}}$. □

Appendix D. Computing the Regularization Term using Vectorial Notation

Consider the simplest unclamped uniform cubic B-spline with only 1 segment, such that $\gamma \in [0,1)$ and described by (6). Then, its first derivative $\mathbf{C}'(\gamma)$ is given by

$$\frac{\partial C^{x/y}}{\partial \gamma}(\gamma) = \underbrace{\underbrace{\begin{bmatrix}\gamma^2 & \gamma & 1 & 0\end{bmatrix}}_{\mathbf{T}(\gamma)} \underbrace{\frac{1}{6}\begin{bmatrix}3&0&0&0\\0&2&0&0\\0&0&1&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}-1&3&-3&1\\3&-6&3&0\\-3&0&3&0\\1&4&1&0\end{bmatrix}}_{M}}_{B'(\gamma)} \underbrace{\begin{bmatrix}P_0^{x/y}\\P_1^{x/y}\\P_2^{x/y}\\P_3^{x/y}\end{bmatrix}}_{\mathbf{P}}. \tag{A32}$$

Therefore, the term $\int_\gamma \|\mathbf{C}'(\gamma)\|^2 d\gamma$ is computed according to

$$\begin{aligned}\int_\gamma \|\mathbf{C}'(\gamma)\|^2 d\gamma &= \int_\gamma (B'(\gamma)\mathbf{P})^T (B'(\gamma)\mathbf{P}) d\gamma \\ &= \int_0^1 \mathbf{P}^T B'(\gamma)^T B'(\gamma)\mathbf{P} d\gamma \\ &= \mathbf{P}^T M^T \left[\int_0^1 \mathbf{T}(\gamma)^T \mathbf{T}(\gamma) d\gamma\right] M\mathbf{P}.\end{aligned} \tag{A33}$$

Further note that

$$\mathbf{T}(\gamma)^T\mathbf{T}(\gamma) = \begin{bmatrix}\gamma^4 & \gamma^3 & \gamma^2 & 0\\ \gamma^3 & \gamma^2 & \gamma & 0\\ \gamma^2 & \gamma & 1 & 0\\ 0&0&0&0\end{bmatrix} \tag{A34}$$

and, as a consequence,

$$\int_0^1 \mathbf{T}(\gamma)^T\mathbf{T}(\gamma)d\gamma = Q = \begin{bmatrix}1/5 & 1/4 & 1/3 & 0\\ 1/4 & 1/3 & 1/2 & 0\\ 1/3 & 1/2 & 1 & 0\\ 0&0&0&0\end{bmatrix}. \tag{A35}$$

Hence, for the simplest case of a single B-spline segment it is known that

$$\int_\gamma \|\mathbf{C}'(\gamma)\|^2 d\gamma = \mathbf{P}^T \underbrace{M^T Q M}_{R_1} \mathbf{P}. \tag{A36}$$

The easiest way to extend this technique to a B-spline with n segments, is to consider the modified vector $\mathbf{T}(\gamma) = [(\gamma-i)^2, (\gamma-i), 1, 0]^T$, where $i = \lfloor \gamma \rfloor$, according to the notation introduced in section 2.3. Then, since $(\gamma - i) \in [0,1)$, one can compute individually for each segment intermediate matrices R_1^i according to (A36). Due to the locality property of B-splines, it is possible to "stack" these intermediate matrices to form the final matrix R_1. An analogous rationale can be applied to compute R_2.

Appendix E. Controller Gains Adopted

The controller gains used to obtain the results in section 8 are presented in table A1.

Table A1. Controller and path planning gains

Currents Observer (ASV)		**Projection Operator (Quadrotor)**	
k_1	2.0	K_d	$diag(0.5, 0.5, 0.2)$
k_2	0.2	ς and β	10.0
Path Following (ASV)		**Path Following (Quadrotor)**	
K_p	$diag(0.5, 0.5)$	K_p	$diag(5.5, 5.5, 5.5)$
δ	-1.0	K_d	$diag(4.5, 4.5, 4.0)$
k_γ	0.5	k_γ	0.5
Cooperative Path Following		**Path Planning**	
k_ε	1.0	N_J	0.6m
c	0.001	$1/\rho$	4.0
b	5.0	λ	0.05
α	1.0	β	0.01

References

1. Casbeer, D.W.; Kingston, D.B.; Beard, R.W.; McLain, T.W. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science* **2006**, *37*, 351–360. doi:10.1080/00207720500438480.
2. Fahad, M.; Saul, N.; Guo, Y.; Bingham, B. Robotic simulation of dynamic plume tracking by Unmanned Surface Vessels. 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2654–2659. doi:10.1109/ICRA.2015.7139557.
3. Pettersson, L.; Pozdnyakov, D. *Monitoring of harmful algal blooms*, first ed.; Springer Science & Business Media, 2013; p. 309 pp. doi:10.1007/978-3-540-68209-7.
4. Sukhinov, A.; Chistyakov, A.; Nikitina, A.; Semenyakina, A.; Korovin, I.; Schaefer, G. Modelling of oil spill spread. 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV). IEEE, 2016, pp. 1134–1139. doi:10.1109/ICIEV.2016.7760176.
5. Li, S.; Guo, Y.; Bingham, B. Multi-robot cooperative control for monitoring and tracking dynamic plumes. 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 67–73. doi:10.1109/ICRA.2014.6906591.
6. Clark, J.; Fierro, R. Cooperative hybrid control of robotic sensors for perimeter detection and tracking. Proceedings of the 2005, American Control Conference, 2005., 2005, pp. 3500–3505 vol. 5. doi:10.1109/ACC.2005.1470515.
7. Saldaña, D.; Assunção, R.; Hsieh, M.A.; Campos, M.F.M.; Kumar, V. Cooperative prediction of time-varying boundaries with a team of robots. 2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS). IEEE, 2017, pp. 9–16. doi:10.1109/MRS.2017.8250925.
8. Piegl, L.; Tiller, W. *The Nurbs Book*, 1 ed.; Springer Science & Business Media, 1995; p. 646 pp. doi:10.1007/978-3-642-59223-2.
9. (IPIECA), I.P.I.E.C.A. Dispersants and their role in oil spill response. V. 5.
10. Aguiar, A.P.; Hespanha, J.P. Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty. *IEEE Transactions on Automatic Control* **2007**, *52*, 1362–1379. doi:10.1109/TAC.2007.902731.
11. Vanni, F.; Aguiar, A.P.; Pascoal, A.M. Cooperative path-following of underactuated autonomous marine vehicles with logic-based communication. *Proceedings Volumes* **2008**, *41*, 107–112. 2nd IFAC workshop on navigation, guidance and control of underwater vehicles, doi:https://doi.org/10.3182/20080408-3-IE-4914.00020.
12. Aguiar, A.P.; Ghabcheloo, R.; Pascoal, A.M.; Silvestre, C. Coordinated Path-Following Control of Multiple Autonomous Underwater Vehicles **2007**. *All Days*. ISOPE-I-07-006.
13. Hung, N.T.; Rego, F.C.; Pascoal, A.M. Event-Triggered Communications for the Synchronization of Nonlinear Multi Agent Systems on Weight-Balanced Digraphs. 2019 18th European Control Conference (ECC). IEEE, 2019, pp. 2713–2718. doi:10.23919/ECC.2019.8796277.
14. Abreu, P.; Botelho, J.; Góis, P.; Pascoal, A.; Ribeiro, J.; Ribeiro, M.; Rufino, M.; Sebastião, L.; Silva, H. The MEDUSA class of autonomous marine vehicles and their role in EU projects. OCEANS Shanghai, 2016, pp. 1–10. doi:10.1109/oceansap.2016.7485620.
15. Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R., Robot Operating System (ROS): The complete reference (Volume 1); Springer International Publishing: Cham, 2016; chapter RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. doi:10.1007/978-3-319-26054-9_23.
16. Manhães, M.; Scherer, S.A.; Voss, M.; Douat, L.R.; Rauschenbach, T. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. OCEANS 2016 MTS/IEEE Monterey, 2016, pp. 1–8. doi:10.1109/OCEANS.2016.7761080.

17. Dey, T.K. Course 784 notes - The Ohio State University - Lecture 7: Matrix Form for B-Spline Curves. https://web.cse.ohio-state.edu/~dey.8/course/784/note7.pdf (Accessed: 2022-01-16).
18. Pascoal, A.; Kaminer, I.; Oliveira, P. Navigation system design using time-varying complementary filters. *IEEE Transactions on Aerospace and Electronic Systems* **2000**, *36*, 1099–1114. doi:10.1109/7.892661.
19. Sanches, G. Sensor-Based Formation Control of Autonomous Marine Robots. M.sc. thesis, Instituto Superior Técnico, 2015.
20. Xie, W.; Cabecinhas, D.; Cunha, R.; Silvestre, C. Robust Motion Control of an Underactuated Hovercraft. *IEEE Transactions on Control Systems Technology* **2019**, *27*, 2195–2208. doi:10.1109/TCST.2018.2862861.
21. Cabecinhas, D.; Cunha, R.; Silvestre, C. A nonlinear quadrotor trajectory tracking controller with disturbance rejection. *Control Engineering Practice* **2014**, *26*, 1–10. doi:10.1016/j.conengprac.2013.12.017.
22. Cai, Z.; de Queiroz, M.; Dawson, D. A sufficiently smooth projection operator. *IEEE Transactions on Automatic Control* **2006**, *51*, 135–139. doi:10.1109/TAC.2005.861704.
23. Aguiar, A.P.; Pascoal, A.M. Coordinated path-following control for nonlinear systems with logic-based communication. 2007 46th IEEE Conference on Decision and Control. IEEE, 2007, pp. 1473–1479. doi:10.1109/CDC.2007.4434835.
24. Hung, N.T.; Pascoal, A.M. Consensus/synchronisation of networked nonlinear multiple agent systems with event-triggered communications. *International Journal of Control* **2020**, pp. 1–10. doi:10.1080/00207179.2020.1849806.
25. Odonkor, P.; Ball, Z.; Chowdhury, S. Distributed operation of collaborating unmanned aerial vehicles for time-sensitive oil spill mapping. *Swarm and Evolutionary Computation* **2019**, *46*. doi:10.1016/j.swevo.2019.01.005.
26. Szeliski, R. *Computer Vision: Algorithms and Applications*; Vol. 5, Springer Science & Business Media, 2011. doi:10.1007/978-1-84882-935-0.
27. Liu, Y.; Yang, H.; Wang, W. Reconstructing B-spline Curves from Point Clouds–A Tangential Flow Approach Using Least Squares Minimization. International Conference on Shape Modeling and Applications 2005 (SMI' 05). IEEE, 2005, pp. 4–12. doi:10.1109/SMI.2005.39.
28. Lee, I.K. Curve reconstruction from unorganized points. *Computer Aided Geometric Design* **2000**, *17*, 161–177. doi:10.1016/S0167-8396(99)00044-8.
29. Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509–517. doi:10.1145/361002.361007.
30. Liu, M.; Huang, S.; Dissanayake, G.; Kodagoda, S. Towards a consistent SLAM algorithm using B-Splines to represent environments. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 2065–2070. doi:10.1109/IROS.2010.5649703.
31. Xie, W.; Cabecinhas, D.; Cunha, R.; Silvestre, C. Cooperative Path Following Control of Multiple Quadcopters With Unknown External Disturbances. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2020**, *52*, 667–679. doi:10.1109/TSMC.2020.3032401.
32. Bradski, G. The OpenCV library. *Dr. Dobb's Journal of Software Tools* **2000**.
33. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods* **2020**, *17*, 261–272. doi:10.1038/s41592-019-0686-2.
34. Khalil, H. *Nonlinear systems*, 3 ed.; Always Learning, Pearson Education Limited, 2013.