sensors

MDPI

Article

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

Marcelo Jacinto *, Rita Cunha and António Pascoal

Laboratory of Robotics and Systems in Engineering and Science (LARSyS), Instituto Superior Técnico, University of Lisbon, 1049-001 Lisboa, Portugal; rita@isr.tecnico.ulisboa.pt (R.C.); antonio@isr.tecnico.ulisboa.pt (A.P.)

* 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 parameterised 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 visually simulate 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

Citation: Jacinto, M.; Cunha, R.; Pascoal, A. Chemical Spill Encircling Using a Quadrotor and Autonomous Surface Vehicles: A Distributed Cooperative Approach. *Sensors* **2022**, *22*, 2178. https://doi.org/10.3390/s22062178

Academic Editor: Maria Gabriella Xibilia

Received: 4 February 2022
Accepted: 7 March 2022
Published: 10 March 2022

1. Introduction

The problems of perimeter detection, boundary searching, and encircling have been widely researched topics with a variety of practical applications, ranging from the 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 (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 that

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 (ASVs) 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 of and demand for Autonomous Underwater Vehicles (AUVs), due not only to their agility when it comes to the execution of scientific and comercial missions, but also to their low cost when compared to traditional ships, which require an on-board crew to be operated. Additionally, there has also been an exponential growth in demand for Unmanned Aerial Vehicles (UAVs), with a 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 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 the dynamical boundary of a chemical spill closely, 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 that uses growing unclamped (and uniform) cubic B-splines is proposed, which fits a 2-D point cloud generated from the drone's image stream.

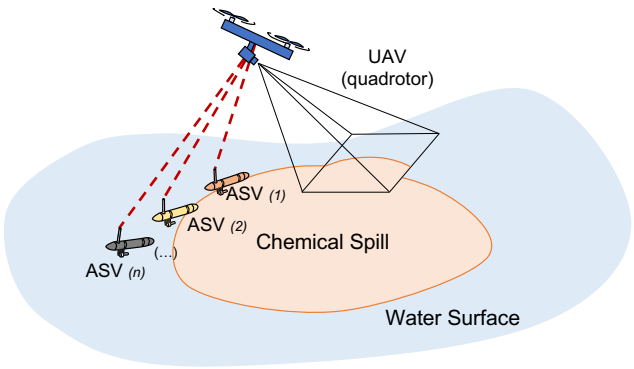


Figure 1. Cooperative path following along an environmental boundary.

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 $\mathbf{1}$ is a vector with all elements equal to one. The symbols $\lfloor x \rceil, x \in \mathbb{R}$ denote the 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 belongs 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:

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

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 being 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:

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

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, according to L. Piegl and W. Tiller ([8], Chapter 2.2), that:

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

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

where the index $j = 0, ..., k$ and 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 four 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 $\boldsymbol{\gamma} = [\gamma_0, ..., \gamma_q] \in \mathbb{R}^{q+1}$ at which we wish to evaluate our curve, $\mathbf{C}(\boldsymbol{\gamma}) \in \mathbb{R}^{2(q+1)}$ 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 four basis functions are different then zero and computed according to (6).

3. Vehicle Modelling

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.

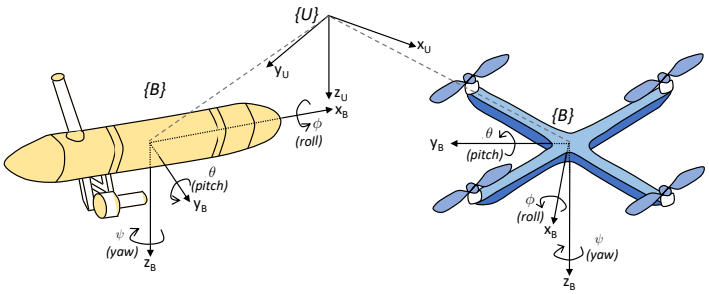


Figure 2. Adopted reference frames for a surface vehicle (**left**) and a quadrotor (**right**).

3.1. ASV Model

The ASV vehicle is modeled as a rigid body whose motion is restricted to a 2-D plane at the surface of the water, such that the roll and pitch angles are zero, i.e., $\phi = \theta = 0$. 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 $\{U\}$, $\mathbf{v} := [u, v]^T$ denotes the body-velocity vector, ${}^U_B R(\psi) \in \mathbb{R}^{2\times 2}$ denotes the rotation matrix, and $\mathbf{v} := [v_{cx}, v_{cy}]^T$ denotes the ocean current, expressed in $\{U\}$ and 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 $\|\mathbf{d}\| \leq d_{max}$. The 3-D rotation matrix adopted for the quadrotor model follows the Z-Y-X convention, and is given by:

$${}^U_B R(\boldsymbol{\theta}) = R_z(\psi) R_y(\theta) R_x(\phi). \tag{11}$$

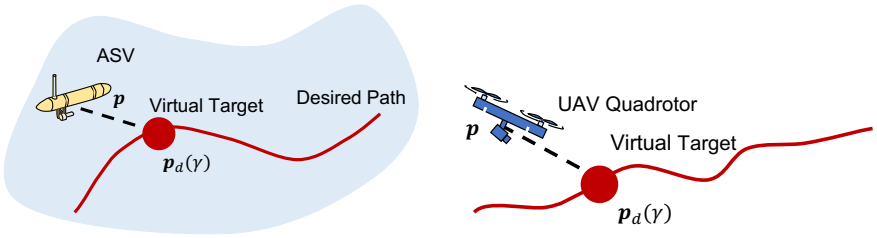
4. Path Following

The path following (PF) problem concerns the problem of making a vehicle move along a desired path $\mathbf{p}_d(\gamma)$ parameterised by a variable γ (for example, the arc-length of the curve). The key idea 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 is given as the sum of another single-vehicle speed profile and an inter-vehicle coordination term, according to:

$$v_d(\gamma) := v_L(\gamma) + v^{coord}, \text{ with } |v_L(\gamma)| \leq v_L^{max}, \tag{12}$$

where $v_L(\gamma)$ is a desired speed profile defined only as a function of the path, v_L^{max} is a pre-defined speed upper-bound, and v^{coord} is the 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 1. *Speed profile $v_d(\gamma)$ might not correspond directly to an inertial speed, especially if the curve is not parameterised 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.*



(**a**) (**b**)

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, parameterised 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* $\dot{\gamma}$ *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:

$$\mathbf{e}_p(t) := {}^B_U R(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma)), \tag{13}$$

and let the speed-tracking error be defined as:

$$e_\gamma := \dot{\gamma} - v_d(\gamma, t). \tag{14}$$

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

$$\dot{\mathbf{e}}_p(t) = {}^B_U \dot{R}(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma)) + {}^B_U R(\psi)(\dot{\mathbf{p}}(t) - \dot{\mathbf{p}}_d(\gamma)). \tag{15}$$

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:

$${}^B_U \dot{R}(\psi) = -S(r) {}^B_U R(\psi). \tag{16}$$

Replacing (16) in (15) yields the position error dynamics expressed in the body-fixed frame as:

$$\dot{\mathbf{e}}_p(t) = -S(r) \underbrace{{}^B_U R(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma))}_{\mathbf{e}_p(t)} + \mathbf{v} + \underbrace{{}^B_U R(\psi)\mathbf{v}_c}_{v_c} - {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} \dot{\gamma}. \tag{17}$$

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 (17) as:

$$\dot{\mathbf{e}}_p(t) = -S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -\delta \end{bmatrix}}_{\Delta} \underbrace{\begin{bmatrix} u \\ r \end{bmatrix}}_{\mathbf{u}} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v_c} - {}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma}. \tag{18}$$

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 $\mathbf{v}$, expressed in $\{B\}$, and a Global Positioning System (GPS) unit which provides measurements of the position of the vehicle $\mathbf{p}$, 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) and the candidate complementary filter model described by:

$$\mathcal{F} := \begin{cases} \dot{\hat{\mathbf{p}}} = k_1(\mathbf{p} - \hat{\mathbf{p}}) + {}^{U}_{B}R(\psi)\mathbf{v} + \hat{\mathbf{v}}_c \\ \dot{\hat{\mathbf{v}}}_c = k_2(\mathbf{p} - \hat{\mathbf{p}}), \end{cases} \tag{19}$$

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 $\boldsymbol{v_c}$ and the requested input $\mathbf{u}_d$ that are to be applied to a 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:

$$\begin{aligned} \tilde{\boldsymbol{v}}_c &:= \boldsymbol{v}_c - \hat{\boldsymbol{v}}_c, \\ \tilde{\mathbf{u}} &:= \mathbf{u} - \mathbf{u}_d. \end{aligned} \tag{20}$$

Consider the Proposition 1 introduced below, 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:*

$$\mathbf{u}_d := \Delta^{-1}\left(-K_p\boldsymbol{\sigma}(\mathbf{e}_p - \boldsymbol{\delta}) - \begin{bmatrix} 0 \\ v \end{bmatrix} - \hat{\boldsymbol{v}}_c + {}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}v_d(\gamma, t)\right), \tag{21}$$

$$\ddot{\gamma} := -k_\gamma e_\gamma + \dot{v}_d(\gamma, t) + (\mathbf{e}_p - \boldsymbol{\delta})^T {}^{B}_{U}R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}, \tag{22}$$

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

Proof. 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:

$$\mathbf{e}_p := \mathbf{p}(t) - \mathbf{p}_d(\gamma), \tag{23}$$

$$\mathbf{e}_v := \dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma}v_d(\gamma, t), \tag{24}$$

and a virtual target speed tracking error defined by (14). Consider also a new auxiliary error $\mathbf{z}$, defined as:

$$\mathbf{z} := \mathbf{e}_v + K_1 \mathbf{e}_p, \tag{25}$$

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

$$\dot{\mathbf{e}}_p = \dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{\gamma}, \tag{26}$$

$$\dot{\mathbf{e}}_v = \ddot{\mathbf{p}} - \frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t)\right). \tag{27}$$

Furthermore, consider the time derivative introduced in (27), the desired virtual target speed function (12), and the virtual target speed tracking error function (14). Then, the time derivative term introduced in (27) can be expanded as:

$$\frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t)\right) = \underbrace{\left[\frac{\partial^2 \mathbf{p}_d}{\partial \gamma^2} v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\frac{\partial v_L(\gamma)}{\partial \gamma}\right]}_{\mathbf{h}(\gamma)} (e_\gamma + v_d(\gamma, t)) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t). \tag{28}$$

Replacing (10) and (28) in (27) yields:

$$\dot{\mathbf{e}}_v = \mathbf{u} + \mathbf{d} - \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) - \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t). \tag{29}$$

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:

$$\dot{\hat{\mathbf{d}}} := K_d \mathrm{Proj}(\mathbf{z}, \hat{\mathbf{d}}) = \mathbf{z} - \frac{\eta_1 \eta_2}{2(\beta^2 + 2\beta d_{max})^{n+1} d_{max}^2}\hat{\mathbf{d}}, \tag{30}$$

where K_d denotes a diagonal gain matrix and:

$$\eta_1 = \begin{cases} (\hat{\mathbf{d}}^T\hat{\mathbf{d}} - d_{max}^2)^{n+1}, \text{ if } (\hat{\mathbf{d}}^T\hat{\mathbf{d}} - d_{max}^2) > 0 \\ 0, \text{ otherwise,} \end{cases} \tag{31}$$

and:

$$\eta_2 = \hat{\mathbf{d}}^T\mathbf{z} + \sqrt{(\hat{\mathbf{d}}^T\mathbf{z})^2 + \varsigma^2}, \tag{32}$$

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

$$\tilde{\mathbf{d}}^T Proj(\mathbf{z}, \hat{\mathbf{d}}) \geq \tilde{\mathbf{d}}^T\mathbf{z}, \tag{33}$$

and:

$$\left\|\hat{\mathbf{d}}\right\| \leq d_{max} + \beta, \forall t \geq 0. \tag{34}$$

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

$$\tilde{\mathbf{u}} := \mathbf{u} - \mathbf{u}_d, \tag{35}$$

$$\tilde{\mathbf{d}} := \mathbf{d} - \hat{\mathbf{d}}. \tag{36}$$

Proposition 2. *Consider the system described by (10), the disturbance estimator dynamics given by (30), and the inner-loop tracking error given by (35). Furthermore, consider the control law given by:*

$$\mathbf{u}_d := -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - \mathbf{e}_v K_v - \mathbf{e}_p K_p, \tag{37}$$

$$\ddot{\gamma} := -k_\gamma e_\gamma + \dot{v}_d(\gamma,t) + \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} + \mathbf{z}^T\left(\mathbf{h}(\gamma) + K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma}\right), \tag{38}$$

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 2. *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 the quadrotor is equipped with a generic inner loop that is capable of keeping the tracking error $\tilde{\mathbf{u}}$ small and bounded.*

Proof. 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].

Synchronisation 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 of 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 the PF controllers proposed in Section 4, and has an assigned path to follow, appropriately parameterised 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 $\mathbf{v^{coord}} = [v_1^{coord}, ..., v_N^{coord}]^T$, 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:*

$$\dot{\boldsymbol{\gamma}} = \mathbf{v_d}(\boldsymbol{\gamma}, t) = \mathbf{v_L}(\boldsymbol{\gamma}) + \mathbf{v^{coord}}. \tag{39}$$

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

$$\varepsilon_i := \sum_{j \in \mathcal{N}_i^{in}} a_{ij}(\gamma_i - \gamma_j), \tag{40}$$

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

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

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:

$$\dot{\boldsymbol{\varepsilon}} := L\dot{\boldsymbol{\gamma}}. \tag{42}$$

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:*

$$v_i^{coord} := -k_\varepsilon \sum_{j \in \mathcal{N}_i^{in}} a_{ij}(\gamma_i - \hat{\gamma}_j), \tag{43}$$

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

$$\dot{\hat{\gamma}}_i := v_L(\hat{\gamma}_i). \tag{44}$$

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

$$\dot{\hat{\boldsymbol{\gamma}}} := \mathbf{v_L}(\hat{\boldsymbol{\gamma}}), \tag{45}$$

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:

$$\mathbf{v^{coord}} := -k_\varepsilon[D\boldsymbol{\gamma} - \mathcal{A}\hat{\boldsymbol{\gamma}}] = -k_\varepsilon(\boldsymbol{\varepsilon} + \mathcal{A}\tilde{\boldsymbol{\gamma}}), \tag{46}$$

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:

$$\begin{cases} \delta_i(t) := |\tilde{\gamma}_i(t)| - g_i(t) \\ \tilde{\gamma}_i(t) = \hat{\gamma}_i(t) - \gamma_i(t), \end{cases} \tag{47}$$

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:

$$g_i(t) = c_i + b_i e^{-\alpha_i t}, \tag{48}$$

with c_i, b_i and α_i being positive constants and $\mathbf{g}(t) = [g_1, ..., g_N]^T$ being 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. Appendix C. □

The proposed control scheme used for achieving CPF using ETC is summarised in Algorithm 1.

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

1: At every time instant t, each vehicle i follows the procedure:
2: **procedure** COORDINATION AND COMMUNICATION
3: **if** $\delta_i(t) \geq 0$ where δ_i is computed using (47) and (48) **then**
4: Broadcast $\gamma_i(t)$;
5: Reset the estimator $\hat{\gamma}_i$;
6: **if** Receive a new message from agent $j \in \mathcal{N}_i^{in}$ **then**
7: Reset $\hat{\gamma}_j(t)$;
8: Run the estimators according to (44);
9: Update the first order control protocol $\boldsymbol{u}_i$ using (43).

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 4a.*

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 of 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 optimisation 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 4b.*

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].*

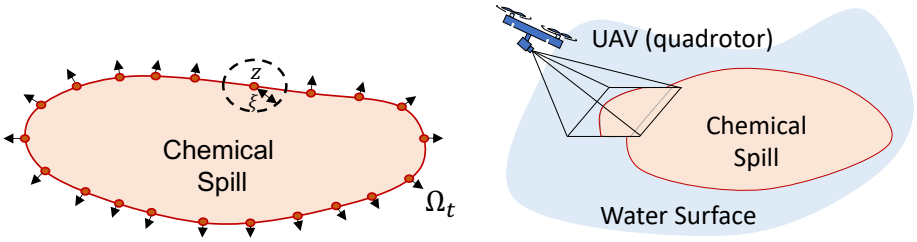


Figure 4. Dynamic Boundary schematic: (**a**) Boundary formal definition. (**b**) Drone's field of view.

6.1. Planar Point Cloud Generation

The camera model adopted is characterised 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.

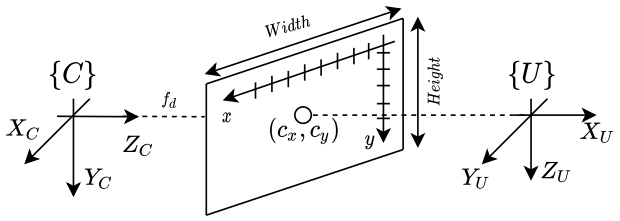


Figure 5. Camera model and reference frames.

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 corresponds 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 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:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_d s_x & 0 & c_x \\ 0 & f_d s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} {}^{C}_{U}R & {}^{C}_{U}\boldsymbol{T} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}}_{{}^{C}_{U}[R|T]} \begin{bmatrix} X_U \\ Y_U \\ Z_U \\ 1 \end{bmatrix}, \tag{49}$$

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:

$${}^{C}_{U}[R|T] = {}^{C}_{B}[R|T]{}^{B}_{U}[R|T], \tag{50}$$

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:

$$\Omega = K \cdot {}^{C}_{U}[R|T]. \tag{51}$$

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 Equation (49) 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:

$$\frac{1}{\lambda} \begin{bmatrix} X_U \\ Y_U \\ 1 \end{bmatrix} = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_4 \\ \Omega_5 & \Omega_6 & \Omega_8 \\ \Omega_9 & \Omega_{10} & \Omega_{12} \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{52}$$

Remark 3. *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, 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 the path at the point where the re-planning starts (defined formally in Section 6.3.1). Some points are outliers as a result of either noisy measurements or regions of the boundary that are not entirely captured by the field of view of the camera. The latter can be seen as disconnected from the main cluster and should be disregarded in the path planning process. According to Figure 6, the original path (in purple) obtained at time t_{k-1} should be re-planned in order to obtain a new one (in red) that better fits the main cluster of points.

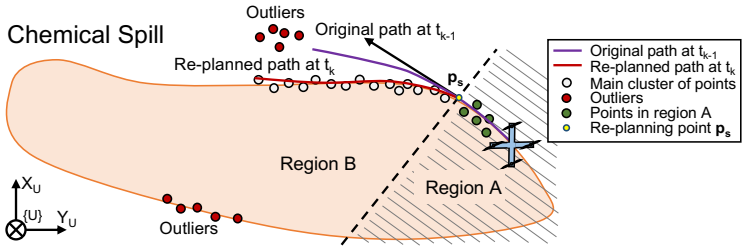


Figure 6. Pre-processing the point cloud and re-planning schematic.

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 $\boldsymbol{X} := \{\boldsymbol{X}_l\}_{l=1}^{L} \in \mathbb{R}^2$, 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:

$$\boldsymbol{X}_l^{\circ} = R(\psi_s) \cdot (\boldsymbol{X}_l - \mathbf{p}_s), \forall l = 1, ..., L, \tag{53}$$

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

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

1: Obtain a new 2-D point cloud $\boldsymbol{X} := \{\boldsymbol{X}_l\}_{l=1}^{L} \in \mathbb{R}^2$;
2: Define $\mathbf{p}_s$ as the desired initial point for the re-planning to start;
3: Define ψ_s as the tangent angle to the current path at t_k at $\mathbf{p}_s$;
4: Follow the procedure:
5: **procedure** REMOVE UNUSED POINTS($\boldsymbol{X}$, $\mathbf{p}_s$, ψ_s)
6: **for** $l = 1, ..., L$ **do**
7: Compute $\boldsymbol{X}_l^{\circ}$ according to (53);
8: **if** $\boldsymbol{X}_l^{\circ x} < 0$ **then**
9: Discard $\boldsymbol{X}_l$;
10: **return** the new set $\boldsymbol{X}^{\star} := \{\boldsymbol{X}_j\}_{j=1}^{J} \in \mathbb{R}^2$ with $J \leq L$.

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 a KDTree [29] can be used to compute a sparse graph $\mathcal{G}$ where each point has a limited set of neighbours, as shown in Figure 7.

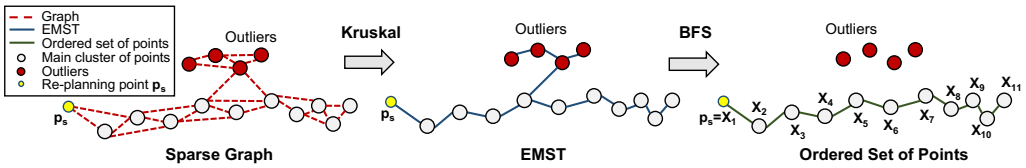


Figure 7. From sparse graph to an ordered list of points (example).

To remove outliers and define a coarse path to follow, Breadth First Search (BFS) can be applied to the EMST, starting from $\mathbf{p}_s$. This removal of outliers from the point cloud is crucial to avoid smaller clusters of points being considered later in the curve fitting problem. 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 summarised in Algorithm 3.

Algorithm 3 Order a set of 2-D points.

1: Add the desired initial point for the path $\mathbf{p}_s$ to $\boldsymbol{X}^{\star}$;
2: Define a threshold distance for the neighbours N_J;
3: Follow the procedure:
4: **procedure** ORDER POINTS($\boldsymbol{X}^{\star}$, N_J)
5: Construct a KDTree from $\boldsymbol{X}^{\star}$ and use N_J as a distance threshold;
6: Create a graph $\mathcal{G}$ with J vertices and no edges;
7: **for** $\boldsymbol{X}_j$, $j = 1, ..., J$ **do**
8: Query the KDTree for the neighbours of $\boldsymbol{X}_j$ and their euclidean distances;
9: Add the corresponding edges to the graph $\mathcal{G}$;
10: Compute the MST of the graph $\mathcal{G}$ starting from vertex corresponding to $\mathbf{p}_s$;
11: Compute the path with the highest number of points, starting at $\mathbf{p}_s$ using BFS;
12: **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 paramaterised 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 optimisation problem—which is computationally demanding to solve for real-time applications. A non-optimal, but more efficient solution, proposed by Liu M. et al. [30] for Simultaneous Localisation and Mapping (SLAM) applications, is to consider D_X to be the total distance between the points, given by:

$$D_X := \sum_{k=2}^{K} \|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|, \tag{54}$$

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

$$\begin{cases} \gamma_1 = 0, \\ \gamma_k = \gamma_{k-1} + \frac{\|X_k - X_{k-1}\|}{D_X} \gamma_{max}, k = 2, ..., K, \end{cases} \tag{55}$$

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 optimisation 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 four 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 leads 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:

$$N_C := max\left\{ \left\lfloor \frac{D_X}{\rho} \right\rceil, 4 \right\}, \tag{56}$$

with $(1/\rho) > 0$ being a control point's density (tunning parameter defined a priori). A smaller ρ leads to a higher N_C. Applying this method to the previous example, and considering $N_C = 7$, $\gamma_{max} = \gamma_{11} = 4$, the result in Figure 8 is obtained.

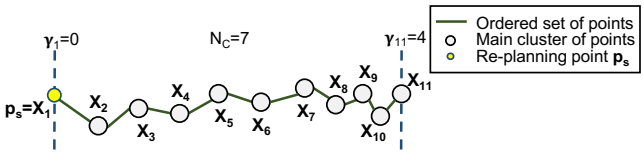


Figure 8. Ordered set of points with parametric values associated to them (example).

6.3.2. Fitting the Points with a Uniform Cubic B-Spline

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

$$f(\mathbf{P}) := \underbrace{\sum_{k=1}^{K} \|\mathbf{C}(\gamma_k, \mathbf{P}) - \mathbf{X}_k\|^2}_{goal} + F_r, \tag{57}$$

with:

$$F_r = \underbrace{\lambda \int_0^{\gamma_{max}} \left\| \frac{\partial \boldsymbol{C}(\gamma, \mathbf{P})}{\partial \gamma} \right\|^2 d\gamma + \beta \int_0^{\gamma_{max}} \left\| \frac{\partial^2 \boldsymbol{C}(\gamma, \mathbf{P})}{\partial \gamma^2} \right\|^2 d\gamma}_{\text{regularisation term}}, \tag{58}$$

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 minimises the distance between the target B-spline curve and the set of points, whilst F_r is a regularisation term and $\alpha, \gamma \geq 0$ are the regularisation variables. The integral of the L_2^2 norm of the first derivative penalises the total length of the curve, while the integral of the L_2^2 norm of the second derivative penalises bends in the path. This objective function can also be expressed using vector notation, according to:

$$f(\mathbf{P}) = \underbrace{\|B(\gamma)\mathbf{P} - \mathbf{X}\|^2}_{goal} + \underbrace{\lambda \mathbf{P}^T R_1 \mathbf{P} + \beta \mathbf{P}^T R_2 \mathbf{P}}_{\text{regularisation term}}, \tag{59}$$

where $\mathbf{X} = [X_1^x, ..., X_K^x, X_1^y, ..., X_K^y]$ denotes the 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 minimise 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 that the virtual target of the drone is "sitting on", and the next segment, according to:

$$\mathbf{p}_s = \mathbf{C}^{old}(\gamma_s) \text{ with } \gamma_s = \lceil \gamma_{drone} \rceil, \tag{60}$$

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 that $\mathbf{p}_s$ is dictated by (60), 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 four control points, 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:

$$\mathbf{P}^{old} = [P_0^x, P_1^x, ..., P_{\gamma_s}^x, P_{\gamma_s+1}^x, P_{\gamma_s+2}^x, P_0^y, P_1^y, ..., P_{\gamma_s}^y, P_{\gamma_s+1}^y, P_{\gamma_s+2}^y]^T. \tag{61}$$

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 are 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 three control points of the first segment coincide with the first three 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 constraints on the derivatives of the function, is to solve the following optimisation problem:

$$\begin{aligned} \mathbf{P}^{new} &= \underset{\mathbf{P}^{new}}{\operatorname{argmin}} \, f(\mathbf{P}^{new}) \\ &\text{subject to} \\ &\begin{bmatrix} P_0^{x\,new} \\ P_1^{x\,new} \\ P_2^{x\,new} \\ P_0^{y\,new} \\ P_1^{y\,new} \\ P_2^{y\,new} \end{bmatrix} = \begin{bmatrix} P_{\gamma_s}^x \\ P_{\gamma_s+1}^x \\ P_{\gamma_s+2}^x \\ P_{\gamma_s}^y \\ P_{\gamma_s+1}^y \\ P_{\gamma_s+2}^y \end{bmatrix}, \end{aligned} \tag{62}$$

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.

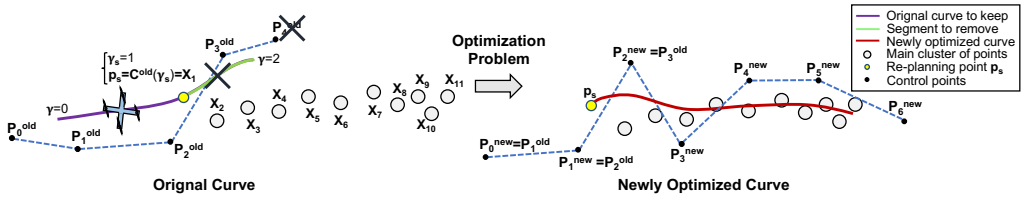


Figure 9. Solving the optimisation problem (example).

To keep track of old and new curves, it is possible to concatenate only 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 (62). Applying this methodology to the previous example, the final control points vector is given according to Figure 10.

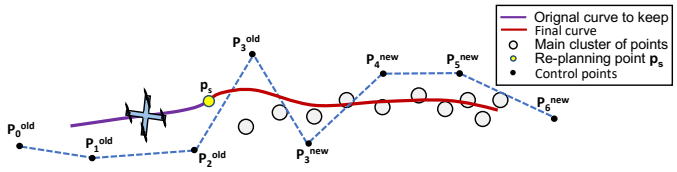


Figure 10. Final curve with control points concatenated (example).

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

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

1: Compute D_X, γ and N_C according to Equations (54), (55), and (56), respectively;
2: Consider $\gamma_s = \lceil \gamma_{t_k} \rceil$ and the original control points vector:

$$\mathbf{P} = \left[P_0, P_1, ..., P_{\gamma_s}, P_{\gamma_s+1}, P_{\gamma_s+2}, P_{\gamma_s+3}, P_{\gamma_s+4}..., P_n\right]^T; \tag{63}$$

3: Remove control points (corresponding to splines to be re-planned) from the original control points vector, such that:

$$\mathbf{P}^{old} = \left[P_0, P_1, ..., P_{\gamma_s}, P_{\gamma_s+1}, P_{\gamma_s+2}\right]^T; \tag{64}$$

4: Solve the optimisation problem in (62) and obtain a new vector with N_C control points:

$$\begin{aligned} \mathbf{P}^{new} &= \left[P_0^{new}, P_1^{new}, P_2^{new}, ..., P_{N_C-1}^{new}\right]^T, \\ \text{with } P_0^{new} &= P_{\gamma_s}, P_1^{new} = P_{\gamma_s+1}, P_2^{new} = P_{\gamma_s+2}; \end{aligned} \tag{65}$$

5: Concatenate the new vector with the old vector (ignoring the first three control points, which are repeated):

$$\mathbf{P}^{final} = \left[P_0, P_1, ..., P_{\gamma_s}, P_{\gamma_s+1}, P_{\gamma_s+2}, P_3^{new}, ..., P_{N_C-1}^{new}\right]^T. \tag{66}$$

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 offset 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:

$$\mathbf{p}_{di}(\gamma_i) = \mathbf{C}(\gamma_i) + {}^{U}_{T}R(\gamma_i)\boldsymbol{\mu}_i, \tag{67}$$

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

$${}^{U}_{T}R(\gamma_i) = [\mathbf{r}_1(\gamma_i), \mathbf{r}_3(\gamma_i) \times \mathbf{r}_1(\gamma_i), \mathbf{r}_3(\gamma_i)], \tag{68}$$

with:

$$\mathbf{r}_1(\gamma_i) = \frac{\partial \mathbf{p}_d/\partial\gamma}{\|\partial \mathbf{p}_d/\partial\gamma\|}, \text{ with } \|\partial \mathbf{p}_d/\partial\gamma\| \neq 0 \qquad \mathbf{r}_3(\gamma_i) = \frac{\mathbf{r}_d - (\mathbf{r}_d \cdot \mathbf{r}_1(\gamma_i))\mathbf{r}_1(\gamma_i)}{\|\mathbf{r}_d - (\mathbf{r}_d \cdot \mathbf{r}_1(\gamma_i))\mathbf{r}_1(\gamma_i)\|}, \tag{69}$$

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 $\mathbf{r}_d = [0, 0, 1]^T$.

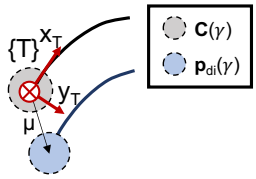


Figure 11. Formation vector.

A path might not be not parameterised 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 a constant required speed $V \leq V_{max}$ for the vehicle and let the desired speed profile for the virtual targets be given by:

$$v_L(\gamma) = \frac{V}{\|\mathbf{p}'_d(\gamma)\|}. \tag{70}$$

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 the 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).

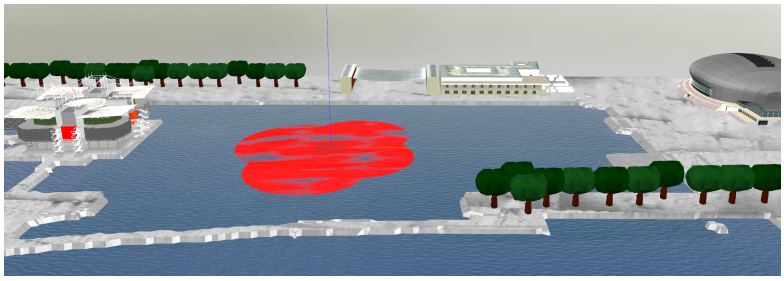
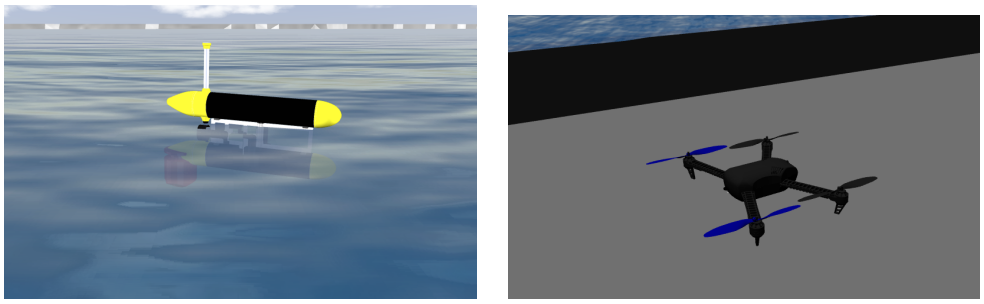


Figure 13. Simulated world of Doca dos Olivais with red chemical spill in Gazebo.

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



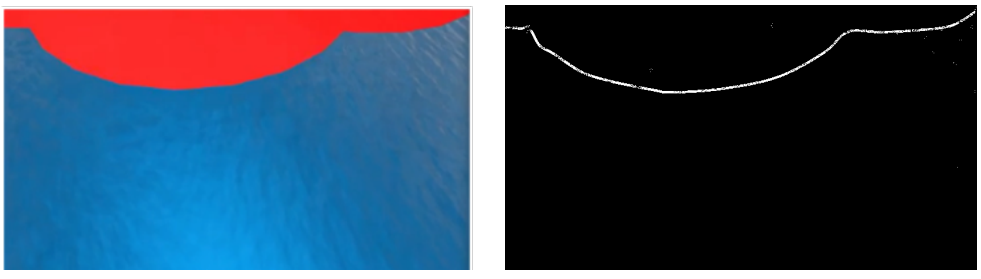
(**a**) (**b**)

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 21 mm below the vehicle's center of mass and with a pitch angle of $-45°$, pointing downwards, and produced an image with a resolution of 640×480 px, according to Figure 15a. Its intrinsic parameters are given by:

$$\begin{cases} (c_x, c_y) = (320.5, 240.5) \\ (f_d s_x, f_d s_y) = (381.4, 381.4). \end{cases} \tag{71}$$

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 15b. To solve the optimisation 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.

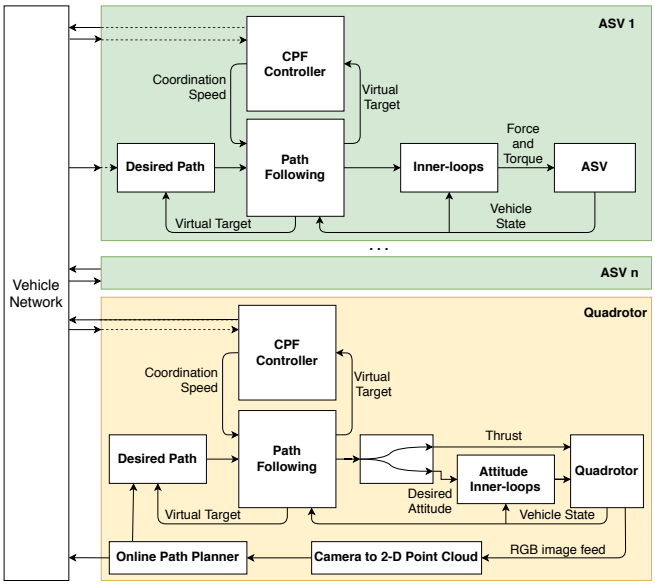


Figure 16. Planning and control architecture.

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.

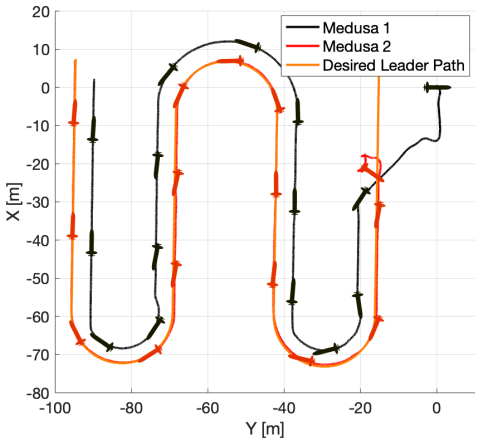


Figure 17. Real CPF mission with 2 Medusa vehicles.

According to the results in Figure 18a, the along-track error of Medusa 1 increases quickly as the virtual target tries not only to minimise 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 18b, after approximately 50 s, the vehicles align themselves according to the desired formation, approach the desired speed profile and, as a consequence, the rate of information exchange decreases. This decrease in the rate of communication is due to the bank of estimators for the virtual targets running in each vehicle being able to better predict the evolution of the virtual targets.

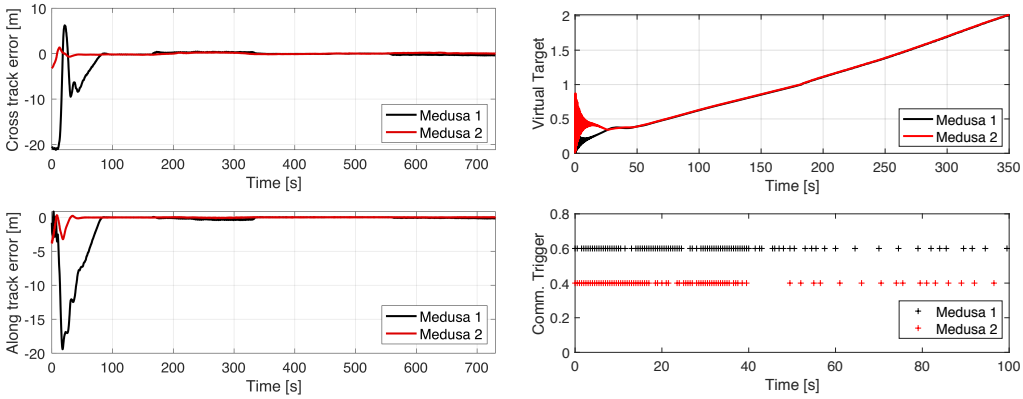


Figure 18. CPF with 2 real Medusa vehicles: (**a**) X–Y view. (**b**) Communication metrics.

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 19a. In this experiment, the aircraft was required to fly at a fixed altitude of 30 m; the formation vector for Medusa 1 was given by $\mu_1 = [-5, 5, 0]^T$m, and for Medusa 2, by $\mu_2 = [-5, -5, 0]^T$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 19b, it is observable that the vehicles converge to their desired formation at around 25 s. 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.

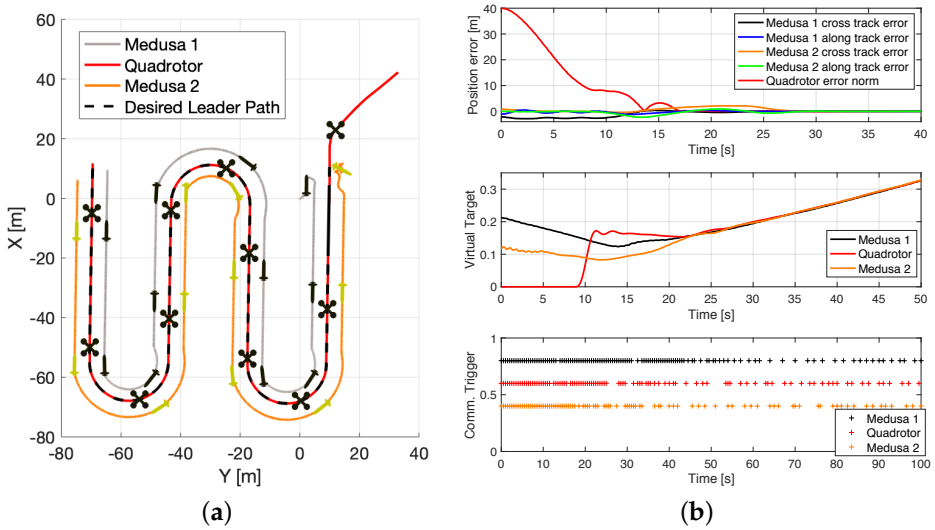


Figure 19. CPF with simulated Iris and Medusa vehicles: (**a**) X–Y view. (**b**). Performance metrics.

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 1 Hz and send the most up-to-date path to the ASV, according to Figure 20. The drone was required to follow the path at 30 m of altitude with a desired constant speed of 0.5 m/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.

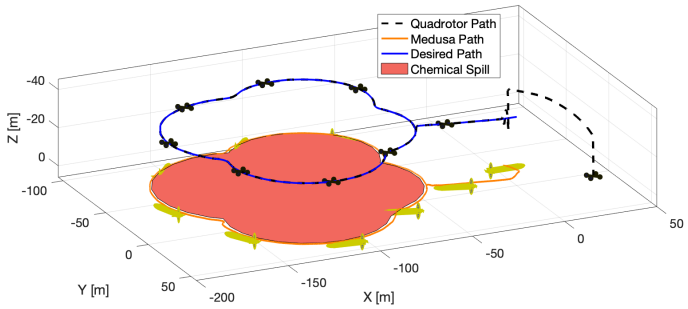


Figure 20. 3-D view of simulated boundary encircling mission with Iris and Medusa vehicles.

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 $\boldsymbol{\mu} = [-5, 5, 0]^T$m. In Figure 21a, a top-down view of the executed mission is shown. In Figure 21b, 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 the along-track error. These tracking errors were instantly compensated by the adaptive virtual target dynamics, which attempted to minimise 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 than its aerial counterpart, with the Medusa always following the boundary from its outskirts, due to the formation vector adopted.

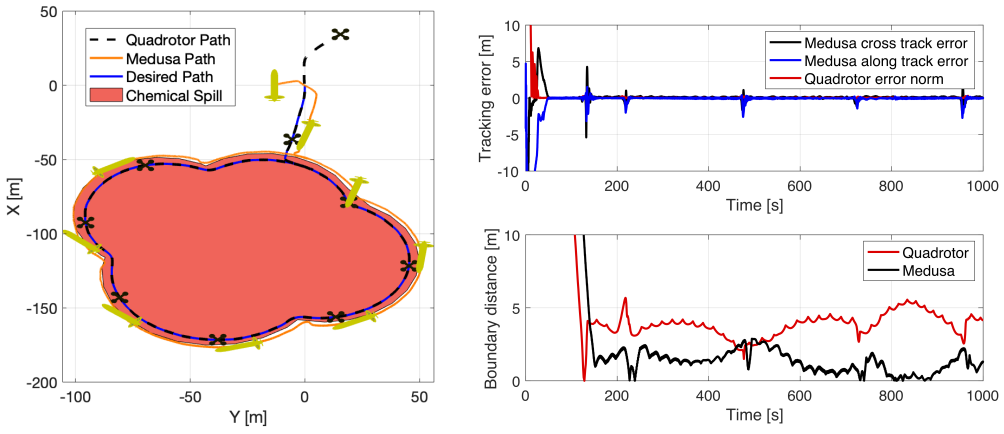


Figure 21. Boundary encircling with simulated Iris and Medusa vehicles: (**a**) X–Y view. (**b**) Performance metrics.

From Figure 21b, it is also 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 22a, 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.

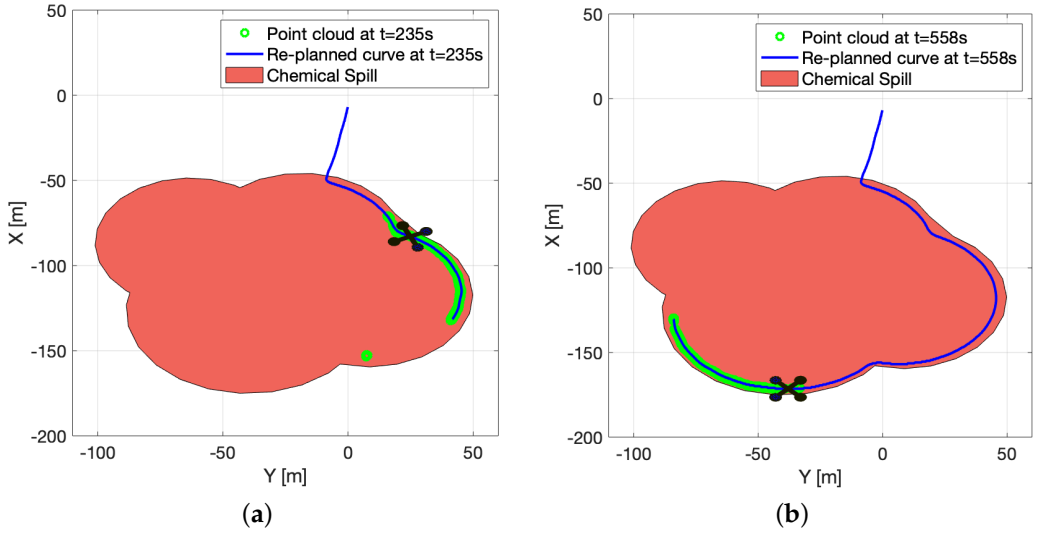


Figure 22. Path generation (**a**) Time = 235 s. (**b**) Time = 558 s.

9. Conclusions 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 synchronisation 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 optimisation 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: Conceptualisation, software, formal analysis, investigation, writing, review, and editing by M.J.; Conceptualisation, 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-INFRAIA-2017-1-two-stage EUMarineRobots-Marine Robotics Research Infrastructure Network (Grant agreement ID: 731103), the H2020-FETPROACT-2020-2 RAMONES-Radioactivity Monitoring in Ocean Ecosystems (Grant agreement ID: 101017808), H2020-MSCA-RISE-2018 ECOBOTICS.SEA-Bio-inspired Technologies for a Sustainable Marine Ecosystem (Grant agreement ID: 824043).

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 (accessed on 8 March 2022).

Acknowledgments: The authors of this work would like to express a deep gratitude to Nguyen Hung, Francisco Rego, João Quintas, and João Cruz for all the support provided during the preparation of the results presented in this work.

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

Appendix A. Proof of Preposition 1

Proof. Consider the candidate Lyapunov Function given by:

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

Taking the first time derivative of (A1) and replacing in (17) and (14) leads to:

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

Taking into account the properties of the skew-symmetric matrix S:

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

Replacing (20) and (21) in $\dot{V}_1$ yields:

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

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 (22) for the virtual target, we obtain:

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

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

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

will be ≤ 0 if:

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

which, in turn, implies that:

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

and:

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

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:

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

Taking the first derivative of (A11) and replacing in (24)–(26), 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 (26), (27), and (29) 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 (35)–(37) 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 (38), we obtain:

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

Consider 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 (33):

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

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, 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 thrusts and attitudes for the quadrotor to track.

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

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

□

Appendix C. Proof of Preposition 3

Proof. Consider 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 (39), the speed correction term proposed in (46), and (A21) in (42) yields:

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

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

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

and the change of variables:

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

Applying (A24) to (A22) yields:

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

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

$$\begin{bmatrix} \dot{\bar{\boldsymbol{\varepsilon}}}_1 \\ \hline \dot{\bar{\boldsymbol{\varepsilon}}}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \hline -k_\varepsilon \Lambda_2(\bar{\boldsymbol{\varepsilon}}_2 + \bar{\mathbf{d}}_2) \end{bmatrix}, \tag{A26}$$

where the first half of the vector denotes the term that depends on the null eigenvalue of the Laplacian, while the second term is a vector that depends only on the positive eigenvalues of the Laplacian. Consider now the candidate Lyapunov function:

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

and its time derivative, given by:

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

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

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

will be ≤ 0 if:

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

and, therefore:

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

The term $\|\tilde{\boldsymbol{\gamma}}\|$ 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, $\|\bar{\mathbf{d}}_2\|$ can be as well. It follows directly from H. Khalil ([34], Theorem 4.19) that the controlled system is ISS with respect to the error vector $\boldsymbol{\varepsilon}$ and the inputs $\tilde{\boldsymbol{\gamma}}$ and $\tilde{\mathbf{v}}_\mathbf{L}$. □

Appendix D. Computing the Regularisation Term Using Vectorial Notation

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

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

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. *Int. J. Syst. Sci.* **2006**, *37*, 351–360. [CrossRef]
2. Fahad, M.; Saul, N.; Guo, Y.; Bingham, B. Robotic simulation of dynamic plume tracking by Unmanned Surface Vessels. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2654–2659. [CrossRef]
3. Pettersson, L.; Pozdnyakov, D. *Monitoring of Harmful Algal Blooms*, 1st ed.; Springer Science & Business Media: Berlin, Germany, 2013; p. 309. [CrossRef]
4. Sukhinov, A.; Chistyakov, A.; Nikitina, A.; Semenyakina, A.; Korovin, I.; Schaefer, G. Modelling of oil spill spread. In Proceedings of the 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV), Dhaka, Bangladesh, 13–14 May 2016; pp. 1134–1139. [CrossRef]
5. Li, S.; Guo, Y.; Bingham, B. Multi-robot cooperative control for monitoring and tracking dynamic plumes. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 67–73. [CrossRef]
6. Clark, J.; Fierro, R. Cooperative hybrid control of robotic sensors for perimeter detection and tracking. In Proceedings of the 2005, American Control Conference, Portland, OR, USA, 8–10 June 2005; Volume 5, pp. 3500–3505. [CrossRef]
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. In Proceedings of the 2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Los Angeles, CA, USA, 4–5 December 2017; pp. 9–16. [CrossRef]
8. Piegl, L.; Tiller, W. *The Nurbs Book*, 1 ed.; Springer Science & Business Media: Berlin, Germany, 1995; p. 646. [CrossRef]
9. International Petroleum Industry Environmental Conservation Association (IPIECA). Dispersants and Their Role in Oil Spill Response. Volume 5. Available online: https://www.amn.pt/DCPM/Documents/DispersantsII.pdf (accessed on 8 March 2022).
10. Aguiar, A.P.; Hespanha, J.P. Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles with Parametric Modeling Uncertainty. *IEEE Trans. Autom. Control.* **2007**, *52*, 1362–1379. [CrossRef]
11. Vanni, F.; Aguiar, A.P.; Pascoal, A.M. Cooperative path-following of underactuated autonomous marine vehicles with logic-based communication. In Proceedings of the 2nd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, Killaloe, Ireland, 8–10 April 2008; Volume 41, pp. 107–112. [CrossRef]
12. Aguiar, A.P.; Ghabcheloo, R.; Pascoal, A.M.; Silvestre, C. Coordinated Path-Following Control of Multiple Autonomous Underwater Vehicles. In Proceedings of the Seventeenth International Offshore and Polar Engineering Conference, Lisbon, Portugal, 1–6 July 2007; 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. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 2713–2718. [CrossRef]
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. In Proceedings of the OCEANS Shanghai, Shanghai, China, 10–13 April 2016; pp. 1–10. [CrossRef]
15. Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R. *Robot Operating System (ROS): The Complete Reference (Volume 1)*; Springer International Publishing: Cham, Switzerland, 2016; pp. 595–625; chapter RotorS—A Modular Gazebo MAV Simulator Framework. [CrossRef]
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. In Proceedings of the OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, USA, 19–23 September 2016; pp. 1–8. [CrossRef]
17. Dey, T.K. Course 784 Notes—The Ohio State University—Lecture 7: Matrix Form for B-Spline Curves. Available online: https://web.cse.ohio-state.edu/~dey.8/course/784/note7.pdf (accessed on 16 January 2022).
18. Pascoal, A.; Kaminer, I.; Oliveira, P. Navigation system design using time-varying complementary filters. *IEEE Trans. Aerosp. Electron. Syst.* **2000**, *36*, 1099–1114. [CrossRef]
19. Sanches, G. Sensor-Based Formation Control of Autonomous Marine Robots. Master's Thesis, Instituto Superior Técnico, Lisbon, Portugal, 2015.
20. Xie, W.; Cabecinhas, D.; Cunha, R.; Silvestre, C. Robust Motion Control of an Underactuated Hovercraft. *IEEE Trans. Control. Syst. Technol.* **2019**, *27*, 2195–2208. [CrossRef]
21. Cabecinhas, D.; Cunha, R.; Silvestre, C. A nonlinear quadrotor trajectory tracking controller with disturbance rejection. *Control. Eng. Pract.* **2014**, *26*, 1–10. [CrossRef]
22. Cai, Z.; de Queiroz, M.; Dawson, D. A sufficiently smooth projection operator. *IEEE Trans. Autom. Control.* **2006**, *51*, 135–139. [CrossRef]
23. Aguiar, A.P.; Pascoal, A.M. Coordinated path-following control for nonlinear systems with logic-based communication. In Proceedings of the 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007; pp. 1473–1479. [CrossRef]
24. Hung, N.T.; Pascoal, A.M. Consensus/synchronisation of networked nonlinear multiple agent systems with event-triggered communications. *Int. J. Control.* **2020**, 1–10. [CrossRef]

25. Odonkor, P.; Ball, Z.; Chowdhury, S. Distributed operation of collaborating unmanned aerial vehicles for time-sensitive oil spill mapping. *Swarm Evol. Comput.* **2019**, *46*, 52–68. [CrossRef]
26. Szeliski, R. *Computer Vision: Algorithms and Applications*; Springer Science & Business Media: Berlin, Germany, 2011; Volume 5. [CrossRef]
27. Liu, Y.; Yang, H.; Wang, W. Reconstructing B-spline Curves from Point Clouds—A Tangential Flow Approach Using Least Squares Minimization. In Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05), Cambridge, MA, USA, 13–17 June 2005; pp. 4–12. [CrossRef]
28. Lee, I.K. Curve reconstruction from unorganized points. *Comput. Aided Geom. Des.* **2000**, *17*, 161–177. [CrossRef]
29. Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
30. Liu, M.; Huang, S.; Dissanayake, G.; Kodagoda, S. Towards a consistent SLAM algorithm using B-Splines to represent environments. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 2065–2070. [CrossRef]
31. Xie, W.; Cabecinhas, D.; Cunha, R.; Silvestre, C. Cooperative Path Following Control of Multiple Quadcopters with Unknown External Disturbances. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *52*, 667–679. [CrossRef]
32. Bradski, G. The OpenCV library. *Dr. Dobb's J. Softw. Tools* **2000**, *120*, 122–125.
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. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
34. Khalil, H. *Nonlinear Systems*, 3rd ed.; Always Learning; Pearson Education Limited: New York, NY, USA, 2013.