sensors

MDPI

Article

Energy-Efficient Configuration and Control Allocation for a Dynamically Reconfigurable Underwater Robot

Tho Dang [1,*], Lionel Lapierre [1,*], Rene Zapata [1] and Benoit Ropars [2]

[1] Laboratory of Informatics, Robotics and MicroElectronics (LIRMM) (UMR 5506 CNRS—UM), Université Montpellier, 161 rue Ada, CEDEX 5, 34392 Montpellier, France; zapata@lirmm.fr
[2] Reeds Company, 199 rue Hélène Boucher, 34170 Castelnau-Le-Lez, France; ropars@lirmm.fr
* Correspondence: danghuu@lirmm.fr (T.D.); lapierre@lirmm.fr (L.L.)

Abstract: A dynamically reconfigurable underwater robot, which can vary its configuration during a mission, would be useful for confined environment exploration and docking because of its versatility. A mission can be performed by choosing among different configurations, and the energy cost may increase, owing to the reconfigurability of the robot. Energy saving is the critical issue in long-range missions with underwater robots. Moreover, control allocation must be considered for a redundant system and input constraints. We propose an approach for an energy-efficient configuration and control allocation for a dynamically reconfigurable underwater robot that is built for karst exploration. The proposed method is based on sequential quadratic programming, which minimizes an energy-like criterion with respect to robotic constraints, i.e., mechanical limitations, actuator saturations, and a dead zone. The optimization problem is solved in each sampling instant. Two popular tasks for underwater robots, i.e., path-following and station-keeping (observation) problems, are simulated, and the simulation results show the efficiency of the method. Moreover, an experiment is carried out to highlight the results.

Keywords: autonomous underwater robot; dynamically reconfigurable underwater robot; control allocation; optimization

Citation: Dang, T.; Lapierre, L.; Zapata, R.; Ropars, B. Energy-Efficient Configuration and Control Allocation for a Dynamically Reconfigurable Underwater Robot. *Sensors* **2023**, *23*, 5439. https://doi.org/10.3390/s23125439

Academic Editors: Thor I. Fossen and Enrico Meli

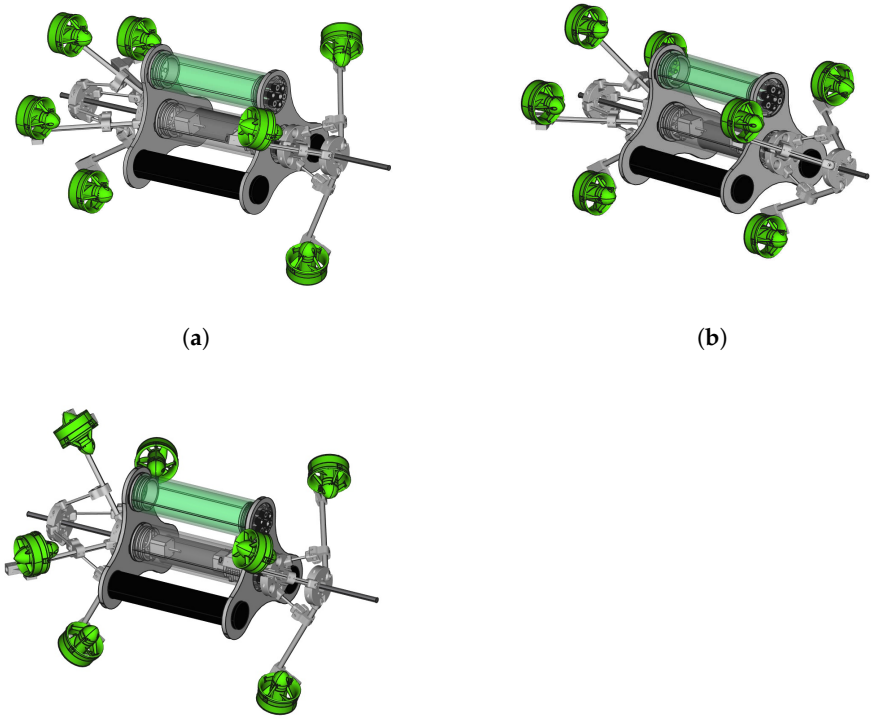
Received: 28 March 2023
Revised: 21 May 2023
Accepted: 6 June 2023
Published: 8 June 2023

1. Introduction

1.1. A Dynamically Reconfigurable Underwater Robot and Perspectives

In robotic fields, reconfigurable robots are an attractive area because of their versatility. They can change their shape or configuration corresponding to specific mission requirements. Therefore, the building cost can be reduced by one robot performing several tasks. Moreover, reconfigurable robots can be applied for complex tasks requiring adaptive configurations, such as karst exploration or space applications. Robustness is also an advantage of reconfigurable robots in terms of the flexibility. Overviews of these aspects and other issues of modular self-reconfigurable robot systems are available in the literature [1,2].

A dynamically reconfigurable underwater robot was built in our laboratory at Montpellier University. Readers can refer to [3] for more details. The robot consists of seven thrusters (three forward and four backward thrusters), and its configuration can be dynamically varied. Figure 1 shows some configurations of the robot, i.e., the robot has an open state for the forward branch and a close state for the backward branch, called an open–close state. The similarities are shown in Figure 1b,c, corresponding to close–close and open–open states, respectively. The change in the robot configuration can be viewed at https://youtu.be/yBBCu1z3q-0 (accessed on 7 February 2023). In the close–close state, it operates as a torpedo-shaped robot; in the open–open state, it operates as an isotropic system. Our robot can operate as an over-actuated system.



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

(**c**)

Figure 1. A 3D model of dynamically reconfigurable robot with three different configurations [3]. (**a**) Robot in open–close state. (**b**) Robot in close–close state. (**c**) The robot in open–open state.

One of the challenges of an over-actuated system is control allocation, in which the distribution of the applied forces on actuators, the so-called applied force vector, is found when a desired control vector (output from a controller) is given. In the following, we describe control allocation methods for an over-actuated system.

1.2. Control Allocation

The basic property of an over-actuated system is that the number of actuators is larger than the controllable degrees of freedom (DOFs). The problem is how to map the desired actuation on the DOFs to forces on the actuators through a *configuration matrix*. In the literature, two approaches are developed to solve this problem. The first method is to divide the control design into two levels. In the first level, the control laws for each DOF are designed. The outputs of this level, called the desired control vector, are the inputs of the second level. In the second level, a control allocation algorithm is designed to assign the control inputs for actuators to optimize one or some cost functions with respect to redundancy and actuator limitations. The problem at the second level is called the control allocation (CA) problem. With the second method, the control inputs (normally with constraints) are directly considered in the control design process. This issue arises in the model predictive control (MPC) method because control allocation is considered a constraint in the MPC formulation. However, this increases the computational cost, so it is the most challenging issue in the MPC problem.

The control allocation problem is one of the main tasks in the control design of over-actuated or redundant systems. Normally, the actuators of a system are constrained with mechanical and electrical limitations, such as saturation or a dead zone. The role of the control allocation block in a control loop is displayed in Figure 2, in which the input is the desired control vector ($\mathbf{F}_B^d$), and the output is the applied force vector ($\mathbf{F}_m$).

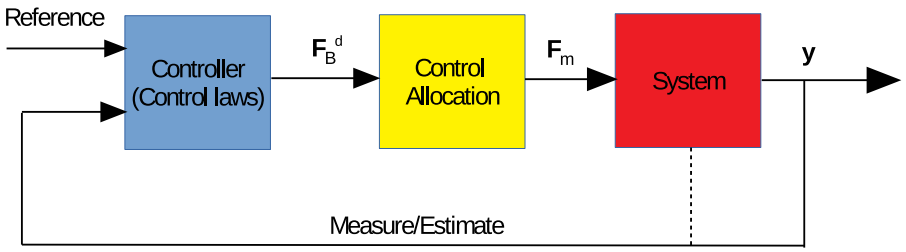


Figure 2. Control allocation block in general control loop.

The many available control allocation methods are divided into two groups: pseudo-inverse- and optimization-based methods, and with or without constraints. Without constraints, the problem is easier. However, the unconstrained control allocation problem provides the basic ideas for many constrained control allocation problems. Most of the control allocation methods are based on optimization techniques, either explicitly or implicitly. Depending on the application, the appropriate control allocation method is chosen.

Some surveys have been conducted on the control allocation problem in recent years. In [4], the authors compared many control allocation algorithms with closed- and open-loop measures. In [5], the authors evaluated the performance and computational cost of the optimization methods of the control allocation problem. In [6,7], control allocation methods for ships and underwater vehicles were investigated. A survey was published in 2013 [8], in which many control allocation methods and applications were presented and discussed. With the advances in neural networks (NNs), NN-based control allocation approaches have been developed [9,10].

1.3. The Singularity of Control Allocation

As mentioned above, many approaches can be used to solve the CA problem. However, in most cases, a configuration matrix is constant and remains unchanged during the robot operation. When the configuration matrix is varied, it may yield a singular or a near-singular configuration; therefore, some DOFs are not controllable. This was previously discussed [11] by researchers who proposed an approach to penalize the singularity of the configuration. However, some advantages of the singular configuration have been addressed, such as when facing disturbances and to achieve optimal energy. Owing to [12], we can investigate different control allocation methods for near-singular configurations. In this situation, the minimum singular value of the configuration matrix is too small. This yields a pseudo-inverse that is too large (which is easily observed with the singular value decomposition of the configuration matrix) and causes a large error if pseudo-inverse-based CA methods are used. Hence, optimization-based CA methods are suitable for dynamically reconfigurable robots.

1.4. Control Allocation with Varying Configuration Matrix

Control allocation methods with a varying configuration matrix have been introduced [13–15] for fault tolerance, in which the control performance is guaranteed when the efficiency of the actuators is lost. In another direction, the configuration matrix contains variables that must be found to minimize an objective function, which is normally power consumption. In the literature, this concept was only implicitly introduced in one study [11], in which azimuths were found to optimize energy consumption. In this study, the problem was formulated and approximated as locally convex quadratic programming. In particular, a sack variable was added to guarantee that the optimization problem always had a feasible solution and, in each sample, the nontrivial part (updating part) of the optimization problem was found by linearizing the objective function and constraints on the optimal solution of the previous sample.

For dynamically reconfigurable robots, the configuration matrix can be varied during robot operations. One question is how to achieve an optimal energy criterion with respect to the parameters of the configuration matrix and the control allocation problem. Motivated by such robots, a real-time technique in nonlinear MPC, so-called one-iteration optimization [16], and optimization-based control allocation methods, we developed an approach to achieve energy-efficient configuration and control allocation, which is different from the aforementioned method, for a dynamically reconfigurable robot that can vary its configuration during missions. The main contributions of the paper are summarized as follows:

1. Propose an energy-efficient configuration problem for a dynamically reconfigurable robot with respect to its constraints.
2. Propose an integration of a one-iteration optimization technique and a control allocation method to solve the energy-efficient configuration problem.

The remainder of this paper is organized as follows: basic notations are summarized in Appendix B. The energy-efficient configuration and control allocation problem are presented in Section 2. A proposed solution is introduced in Section 3. The simulation results are shown and discussed in Section 4. The experimental result is discussed in Section 5; finally, concluding remarks are provided in Section 6.

2. Energy-Efficient Configuration and Control Allocation Problem

Without loss of generality and to ensure ease of understanding, our robot is used to formulate the problem. Some additional notations are illustrated in Figure 3, i.e., body frame, X_B, Y_B, Z_B; linear velocities and angular rates expressed in body frames u, v, w and p, q, r, respectively; and two angles α_F, α_B, for changing the robot configuration, which can be changed during robot operations.

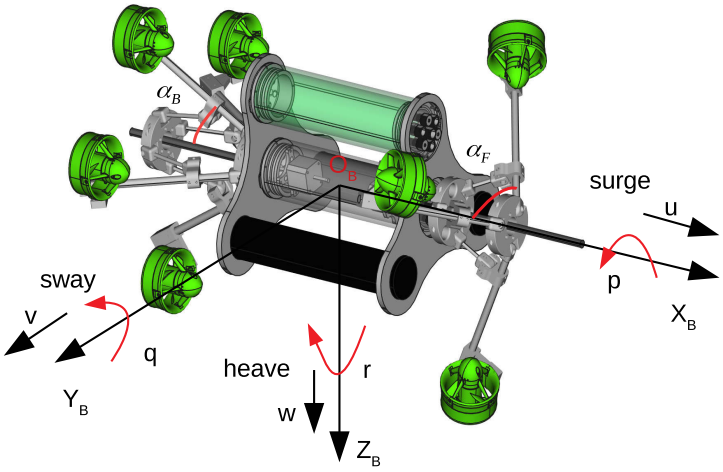


Figure 3. Notations and definitions of two angles α_F and α_B.

The relationship between the resulting control vector, including force and torque elements, in the body frame, denoted as $\mathbf{F}_B$, and the force vector applied on the thrusters, denoted as $\mathbf{F}_m$, is described as a mapping through the *configuration matrix*, denoted as $\mathbf{A}$, which describes the geometric organization of thrusters in the body frame:

$$\mathbf{F}_B = \mathbf{A}(\alpha_F, \alpha_B)\mathbf{F}_m = \begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix} \tag{1}$$

where $\mathbf{F}_B \in \mathbb{R}^6$, $\mathbf{A} \in \mathbb{R}^{6\times m}$, $\mathbf{F}_m = [F_{m,1} \quad F_{m,2} \quad \dots \quad F_{m,m}]^T \in \mathbb{R}^m$, and m is the number of thrusters, $m = 7 > 6$. Because the system has 6 DOFs with 7 actuators, the actuation system is said to be redundant or over-actuated.

From the scheme of the robot, the configuration matrix is as follows:

$$\begin{aligned}\mathbf{A} &= \begin{pmatrix} \mathbf{u}_1^B & \mathbf{u}_2^B & \cdots & \mathbf{u}_m^B \\ \mathbf{r}_1^B \otimes \mathbf{u}_1^B & \mathbf{r}_2^B \otimes \mathbf{u}_2^B & \cdots & \mathbf{r}_m^B \otimes \mathbf{u}_m^B \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{u}_1^B & \mathbf{u}_2^B & \cdots & \mathbf{u}_m^B \\ \boldsymbol{\tau}_1^B & \boldsymbol{\tau}_2^B & \cdots & \boldsymbol{\tau}_m^B \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}\end{aligned} \tag{2}$$

where $m = 7$ and $\mathbf{u}_1^B, \ldots, \mathbf{u}_7^B$ and $\mathbf{r}_1^B, \ldots, \mathbf{r}_7^B$ are shown in Appendix A. The basic idea for the computing matrix $\mathbf{A}$ is to use transformation matrices between the coordinate systems. Because of space limitations, this computation is not shown in this paper. When two angles, α_F and α_B (Figure 3), are varied, the robot's configuration ($\mathbf{A}$ matrix) changes.

In this section, we consider an energy-efficient configuration and the control allocation problem, in which the objective function is defined using the Euclidean norm of the applied force vector, $\mathbf{F}_m$ with respect to mechanical constraints: the thruster limitations. In particular, the problem is formulated as

$$\min_{\alpha_F, \alpha_B, \mathbf{F}_m} J = \|\mathbf{F}_m\|^2 \tag{3a}$$

$$s.t \quad 45^\circ \leq \alpha_F, \alpha_B \leq 90^\circ \tag{3b}$$

$$\mathbf{F}_m \in \mathbb{F} \tag{3c}$$

$$\mathbf{F}_B^d - \mathbf{A}(\alpha_F, \alpha_B)\mathbf{F}_m = 0 \tag{3d}$$

where $\mathbf{F}_B^d$ is the desired control vector (output from the controller), and $\mathbb{F}$ is a feasible set of thruster forces. The constraint (3b) is the mechanical limitation on the robot, in which two angles can vary from 45° to 90°.

The objective function is chosen as the Euclidean norm of the applied force vector. This is reasonable because of the nearly linear characteristics of the thrusters used in our robot (see more details in the simulation section). The problem objective is to find two angles, α_F, α_B and applied force vector $\mathbf{F}_m$, to minimize function J and to satisfy the constraints. This is a nonlinear optimization problem that is solved at each sampling time (called online optimization) because the desired control vector $\mathbf{F}_B^d$ is changed in each time step in the general case. Note that in our problem, the configuration matrix $\mathbf{A}$ is dynamic and has two angles, α_F and α_B (see Figure 3).

Other perspectives we needed to consider are the reactivity of the robot (the time for state propagation or system response) and the time delay of the changing configurations. If the system response is too fast, we cannot apply online optimization. Our objective was to solve the online optimization problem; therefore, we assumed that the time required for solving at least one iteration of the optimization problem is less than the time that the system needs from the current to the next state. In our case of an underwater robot, this assumption is reasonable.

For the time delay when changing configurations, assume that at time step k, we have two angles: α_{Fk} and α_{Bk}. At the next time step, $k+1$, assume that we obtain a solution from the optimization problem with two angles $\alpha_{F(k+1)}$ and $\alpha_{B(k+1)}$. Physically, the robot requires time, $\triangle t_\alpha$, to change from α_{Fk} to $\alpha_{F(k+1)}$ and from α_{Bk} to $\alpha_{B(k+1)}$, which is the new optimal configuration. This changing time cannot be too fast, given the limitations of the DC motors used for changing the robot configuration. However, this change must be completed before the next time step, $k+1$. If not, the configuration matrix will not be associated with the correct corresponding time step $k+1$. In other words, the time needed for changing, $\triangle t_\alpha$, must be less than the sampling time. Therefore, the consecutive values of these two angles must be small enough.

To solve our problem, two assumptions were applied, as follows:

Assumption 1. *The reactivity of the system is long enough, i.e., to solve the online optimization problem and basic mathematical operations. In particular, the sampling time of our underwater*

robot is 0.1 (s). The computational time of one-iteration optimization and other basic mathematical operations is in centiseconds, e.g., 0.01 (s) (this depends on the computation system) and the system reactivity of an underwater robot, which normally depends on its shape and drag coefficients, is also in centiseconds.

Assumption 2. *The time for changing the mechanical system between two consecutive angles is fast enough in one sampling time.*

In practice, Assumption 1 is reasonable for underwater robots. Assumption 2 can be satisfied if the derivation between two consecutive angles is small enough.

Remark 1. *The problem* (3) *is a nonconvex parametric optimization problem, which has to be solved at each sampling time. Thus, as obtaining exact optimal solution is too challenging or probably impossible, the idea is to find an approximate solution.*

Remark 2. *The deviation of two angles α_F, α_B in two consecutive time steps is small enough, owing to mechanical and electrical limitations and Assumption 2.*

3. Solution

The procedure of finding the approximate solution of the problem is divided into two steps: *Predictor* and *corrector*. In the *predictor* step, the problem is quickly solved (in one iteration) to obtain the approximately optimal two angles and applied force vector. However, the hard constraint (3d) cannot be easily satisfied with this applied force vector. With these two angles, we can instantly compute the configuration matrix and move to the next step, *corrector*. In this step, an algorithm is used to find a better applied force vector with respect to this hard constraint. The two following subsections describe some basic results from real-time model predictive control, which we used in the *predictor* step.

3.1. Sequential Quadratic Programming (SQP)

Consider a nonlinear optimization problem (NLP):

$$\min_{\mathbf{x}} \quad J(\mathbf{x}) \tag{4a}$$

$$s.t \quad G(\mathbf{x}) = 0 \tag{4b}$$

$$H(\mathbf{x}) \geq 0 \tag{4c}$$

Sequential quadratic programming (SQP) is an iterative method to find a Karush–Kuhn–Tucker (KKT) point of an NLP. In particular, starting with an initial guess $\mathbf{y}_0 = (\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ (where $\mathbf{x}_0$ is a primal variable, and $\boldsymbol{\lambda}_0, \boldsymbol{\mu}_0$ are Lagrangian multipliers), an SQP method iterates:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \delta_k \Delta \mathbf{y}_k \tag{5}$$

where $\delta_k \in (0, 1]$, and

$$\Delta \mathbf{y}_{k+1} = \begin{pmatrix} \Delta \mathbf{x}_k \\ \Delta \boldsymbol{\lambda}_k \\ \Delta \boldsymbol{\mu}_k \end{pmatrix} := \begin{pmatrix} \Delta \mathbf{x}_k \\ \tilde{\boldsymbol{\lambda}}_k - \boldsymbol{\lambda}_k \\ \tilde{\boldsymbol{\mu}}_k - \boldsymbol{\mu}_k \end{pmatrix} \tag{6}$$

Equation (6) is obtained from the solution point $(\Delta \mathbf{x}_k, \tilde{\boldsymbol{\lambda}}_k, \tilde{\boldsymbol{\mu}}_k)$ of the following quadratic programming:

$$\min_{\Delta \mathbf{x} \in \Omega_k} \quad \frac{1}{2} \Delta \mathbf{x}^T \mathbf{A}_k \Delta \mathbf{x} + \nabla_x J(\mathbf{x}_k)^T \Delta \mathbf{x} \tag{7a}$$

$$s.t \quad G(\mathbf{x}_k) + \nabla_x G(\mathbf{x}_k)^T \Delta \mathbf{x} = 0 \tag{7b}$$

$$H(\mathbf{x}_k) + \nabla_x H(\mathbf{x}_k)^T \Delta \mathbf{x} \geq 0 \tag{7c}$$

Readers can refer to [17] for more details. The choice of the step length δ_k, Hessian matrix $\mathbf{A}_k$, and set $\Omega_k \subset \mathbb{R}^{n_x}$ derives variants of existing SQP methods. If we choose $\delta_k := 1, \Omega_k := \mathbb{R}^{n_x}$ and the Hessian matrix as in Equation (8), this is a full-step exact Hessian SQP method, which is appealing and important [18]:

$$\mathbf{A}_k := \nabla_x^2 L(\mathbf{x}_k, \lambda_k, \mu_k) \tag{8}$$

This choice has an advantage, in that the full-step exact Hessian SQP method shows the same high-quality local convergence behavior as the Newton–Raphson method in the vicinity of a solution of the KKT system (for an illustration for cases with equality constraints, see [17]). Note that a good initial guess is required not only for the primal variable $\mathbf{x}$ but also for the multipliers $\boldsymbol{\lambda}, \boldsymbol{\mu}$ (for a proof, refer to [19]).

3.2. Parametric Nonlinear Optimization

In this section, we describe a parametric optimization problem, $P(\mathbf{t})$, in which a parameter is changed as follows:

$$\min_{\mathbf{x}} \quad J(\mathbf{x}, \mathbf{t}) \tag{9a}$$
$$s.t \quad G(\mathbf{x}, \mathbf{t}) = 0 \tag{9b}$$
$$H(\mathbf{x}, \mathbf{t}) \geq 0 \tag{9c}$$

where $\mathbf{t}$ is a parametric variable. Note that this is not a time variable.

The problem (9) is equivalent to the following problem $P(\bar{\mathbf{t}})$:

$$\min_{\mathbf{x},\mathbf{t}} \quad J(\mathbf{x}, \mathbf{t}) \tag{10a}$$
$$s.t \quad \mathbf{t} - \bar{\mathbf{t}} = 0 \tag{10b}$$
$$G(\mathbf{x}, \mathbf{t}) = 0 \tag{10c}$$
$$H(\mathbf{x}, \mathbf{t}) \geq 0 \tag{10d}$$

The problem (10) is only different from (9) by reforming variable $\mathbf{t}$, which is fixed as an additional constraint $\mathbf{t} - \bar{\mathbf{t}} = 0$. This is introduced to show that *the first iteration of the SQP approach of the problem is the first-order approximation of the solution manifold of this problem*. The following theorem is extracted to highlight this point. Readers can refer to [18] for the proof and more details.

Theorem 1 (First-order prediction by exact Hessian SQP [18])**.** *Let us assume that we found a KKT point* $(\mathbf{x}^*(0), \boldsymbol{\lambda}^* 0, \boldsymbol{\mu}^*(0))$ *of problem* $P(0)$ *that satisfies the sufficient optimality conditions. If a full-step SQP algorithm with an exact Hessian for the solution of the problem* $P(\epsilon)$*, with* $\epsilon > 0$ *being sufficiently small, is started with this solution as an initial guess, then the nontrivial part of the first SQP step* $(\Delta\mathbf{x}, \Delta\boldsymbol{\lambda}, \boldsymbol{\Delta}\mu)$ *is identical to* ϵ *times the one-sided derivative of the solution manifold* $(\mathbf{x}^*(\mathbf{t}), \boldsymbol{\lambda}^*(\mathbf{t}), \boldsymbol{\mu}^*(\mathbf{t}))$ *of the problem* $P(\mathbf{t})$*, i.e.,*

$$\lim_{\mathbf{t}\to,\mathbf{t}>0} \frac{1}{\mathbf{t}} \begin{pmatrix} \mathbf{x}^*(\mathbf{t}) - \mathbf{x}^*(0) \\ \boldsymbol{\lambda}^*(\mathbf{t}) - \boldsymbol{\lambda}^*(0) \\ \boldsymbol{\mu}^*(\mathbf{t}) - \boldsymbol{\mu}^*(0) \end{pmatrix} = \begin{pmatrix} \delta\mathbf{x} \\ \delta\boldsymbol{\lambda} \\ \delta\boldsymbol{\mu} \end{pmatrix} = \frac{1}{\epsilon} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\mu} \end{pmatrix} \tag{11}$$

3.3. Online Optimization Observation in SQP

Following Theorem 1, an observation is that the first quadratic programming solution of a full-step exact Hessian SQP algorithm provides a good approximation of the exact solution if the algorithm is initialized in a neighborhood of this solution. In an online optimization scenario, it would probably be better to use an approximation of the first correction instead of waiting until the SQP algorithm converges. After the first SQP

iteration, we correct the solution if disturbances exist that affect the initial guess or violate the constraints; if no further disturbance occurs, the algorithm can continue to improve the outcome of the previous iterations. In our case of an energy-efficient configuration of the robot, the change in the two angles α_F, α_B in each time step is small. Hence, from an initial guess, two angles can be chosen after one iteration of SQP. After that, we can correct the applied force vector to obtain a better solution.

3.4. Algorithm

3.4.1. *Predictor* Step

This problem (3) can be rewritten as

$$\min_{\alpha_F, \alpha_B, \mathbf{F}_m, \mathbf{F}_B} J = \|\mathbf{F}_m\|^2 \tag{12a}$$

$$s.t \quad 45^\circ \leq \alpha_F, \alpha_B \leq 90^\circ \tag{12b}$$

$$\mathbf{F}_m \in \mathbb{F} \tag{12c}$$

$$\mathbf{F}_B - \mathbf{A}(\alpha_F, \alpha_B)\mathbf{F}_m = 0 \tag{12d}$$

$$\mathbf{F}_B - \mathbf{F}_B^d = 0 \tag{12e}$$

where $\mathbf{F}_B$ is a parametric variable, and $\mathbf{F}_B^d$ is its constraint and is changed during a time instant (output from a controller). This problem has the form of problem (10) and is solved in one iteration using SQP.

By denoting $\mathbf{x} = [\alpha_F \quad \alpha_B \quad \mathbf{F}_m]^T$, the problem (12) can be formed as

$$\min_{\mathbf{x}, \mathbf{F}_B} \quad J(\mathbf{x}) \tag{13a}$$

$$s.t \quad G(\mathbf{x}, \mathbf{F}_B) = 0 \tag{13b}$$

$$H(\mathbf{x}) \leq 0 \tag{13c}$$

$$\mathbf{F}_B - \mathbf{F}_B^d = 0 \tag{13d}$$

where $\mathbf{F}_B$ replaces the role of $\mathbf{t}$ and $\mathbf{F}_B^d$ replaces the role of $\bar{\mathbf{t}}$ in problem $P(\bar{\mathbf{t}})$.

The quadratic programming used to find direction in SQP, $\Delta\mathbf{x}_k$, $\Delta\mathbf{F}_B$, is as follows:

$$\min_{\Delta\mathbf{x}, \Delta\mathbf{F}_B} \frac{1}{2}\Delta\mathbf{x}^T\nabla^2_{\mathbf{x}}\mathcal{L}\Delta\mathbf{x} + \nabla_{\mathbf{x}}J^T\Delta\mathbf{x} + \nabla_{\mathbf{F}_B}J^T\Delta\mathbf{F}_B + \frac{1}{2}\Delta\mathbf{F_B}^T\nabla^2_{\mathbf{F}_B}\mathcal{L}\Delta\mathbf{F}_B + \Delta\mathbf{F_B}^T\nabla^2_{\mathbf{x},\mathbf{F}_B}\mathcal{L}\Delta\mathbf{x} \tag{14a}$$

$$s.t \quad \mathbf{G} + \nabla_{\mathbf{F}_B}\mathbf{G}^T\Delta\mathbf{F}_B + \nabla_{\mathbf{x}}\mathbf{G}^T\Delta\mathbf{x} = 0 \tag{14b}$$

$$\mathbf{H} + \nabla_{\mathbf{F}_B}\mathbf{H}^T\Delta\mathbf{F}_B + \nabla_{\mathbf{x}}\mathbf{H}^T\Delta\mathbf{x} \leq 0 \tag{14c}$$

$$\Delta\mathbf{F}_B - \Delta\mathbf{F}_B^d = 0 \tag{14d}$$

where $\mathcal{L}$ is a Lagrangian function.

The problem (14) is quadratic: it can be efficiently solved by functions in MATLAB, CPLEX, or others. Note that in the *predictor* step, only one iteration of SQP is performed.

3.4.2. *Corrector* Step

By solving Problem (12) in the *predictor* step, we obtained a solution for two angles α_F, α_B and the applied force vector $\mathbf{F}_m$. The two angles can be applied to the robot; however, with one iteration, $\mathbf{F}_m$ does not easily satisfy the hard constraint (12c). With two angles, we have a configuration matrix, $\mathbf{A}$; the following optimization problem is solved to find a new applied force vector $\mathbf{F}_m$:

$$\min_{\mathbf{F}_m} \quad J = \|W_F\mathbf{F}_m\|^2 + \|W_B(\mathbf{A}\mathbf{F}_m - \mathbf{F}_B^d)\|^2 \tag{15a}$$

$$s.t \quad \mathbf{F}_m \in \mathbb{F} \tag{15b}$$

where W_F, W_B are weighting vectors.

The problem (15) can be considered a classical control allocation problem and be efficiently solved by optimization-based methods or pseudo-based methods [8]. The dead-zone compensation [20] is applied if necessary.

Algorithm 1 shows the procedure to solve the energy-efficient configuration and control allocation problem for our robot.

Algorithm 1 Energy-efficient and control allocation algorithm.

Input: Parametric variable $\mathbf{F}_B^d$ (output from the controller)
Output: Local optimal angles α_F, α_B and applied force vector $\mathbf{F}_m$

Predictor step:

1: Initial guess: primal–dual variable, $(\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$; Hessian matrix $\nabla^2_{\mathbf{x}}\mathcal{L}$, and $\nabla_x J$, G, $\nabla_x G$, $\nabla_x H$, H: all are evaluated at $(\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$.
2: Solve QP problem (14) and obtain corresponding solution $(\Delta\mathbf{x}_k, \tilde{\boldsymbol{\lambda}}_k, \tilde{\boldsymbol{\mu}}_k)$
3: Compute multipliers direction $\Delta\boldsymbol{\lambda}_k = \tilde{\boldsymbol{\lambda}}_k - \boldsymbol{\lambda}_k$, $\Delta\boldsymbol{\mu}_k = \tilde{\boldsymbol{\mu}}_k - \boldsymbol{\mu}_k$
4: Update $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$, $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \Delta\boldsymbol{\lambda}_k = \tilde{\boldsymbol{\lambda}}_k$, and $\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \Delta\boldsymbol{\mu}_k = \tilde{\boldsymbol{\mu}}_k$ ▷ *This is the initial guess for the next time step.*
5: Extract two angles α_F, α_B from vector $\mathbf{x}_{k+1}$
6: Compute configuration matrix, $\mathbf{A}$, corresponding to two angles α_F, α_B.

Corrector step:

7: Solve problem (15) to obtain applied force vector $\mathbf{F}_m$

Remark 3. *The difference between online optimization and real-time schemes is the initial strategy. This depends on each specific problem. However, in some systems, the initial values are fixed because of limitations or typical applications. For such systems, an offline optimization procedure may be conducted first to find a suitable initial guess. If not, the solution of the current step can be improved in comparison with the previous step. For our dynamically reconfigurable underwater robot, for instance, because of transportation and karst exploration, from the beginning of a mission, the initial configuration of the robot is chosen as* $\alpha_F = \alpha_B = 45^\circ$. *Then, the initial guess is* $\alpha_F = \alpha_B = 45^\circ$, $\mathbf{F}_m = \mathbf{0}$.

4. Simulation Results

This section presents the simulation results of the proposed approach for our simulated robot. The control diagram of the simulation is shown in Figure 4. A PID or quaternion-based controller was used to derive desired control vector $\mathbf{F}_B^d$. The proposed algorithm inside the energy-efficient control allocation block found the applied force vector, $\mathbf{F}_m$, corresponding to each desired control vector. The inputs of the thrusters (PWM) were interpolated from vector $\mathbf{F}_m$ by using inverse thruster characteristics. The next subsection presents more details about the simulated robot and thruster characteristics.

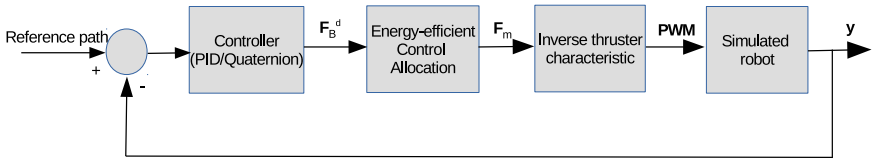


Figure 4. Control loop simulation.

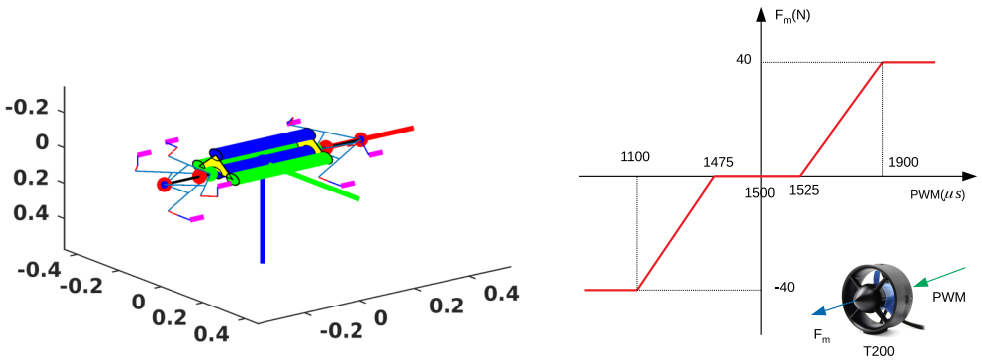
4.1. Simulated Robot

The simulated robot we built is shown in Figure 5a, which is the same as our prototype robot. Note that in the simulations, external disturbances and model uncertainties were not considered. The robot has three forward thrusters and four backward thrusters. The two blue cylinders are waterproof tubes containing electronic boards, and the two green cylinders are battery tubes. The thruster characteristics are shown in Figure 5b, which

approximates the T200 thruster of BlueRobotics [21]. The saturation values of the thrusters are 1100 µs and 1900 µs. The dead zone of the thrusters is [1475 µs − 1525 µs]. The robot can vary its configuration. The dynamic model of the robot is simplified as Equation (16). Assume that all feedback states are completely estimated:

$$F_u = m_u\dot{u} - d_u u \tag{16a}$$
$$F_v = m_v\dot{v} - d_v v \tag{16b}$$
$$F_w = m_w\dot{w} - d_w w \tag{16c}$$
$$\Gamma_p = m_p\dot{p} - d_p p \tag{16d}$$
$$\Gamma_q = m_q\dot{q} - d_q q \tag{16e}$$
$$\Gamma_r = m_r\dot{r} - d_r r \tag{16f}$$

where $m_u, m_v, m_w, m_p, m_q, m_r$ are the total masses (dry mass+added mass or inertia) along each motion axis; $d_u, d_v, d_w, d_p, d_q, d_r$ are the quadratic damping terms for each motion axis. Note that all coupling terms are neglected. Because the weight of our prototype robot in water is approximately 15 kg, the dynamic parameters of the robot are chosen as $m_u = m_v = m_w = 15$ kg, $m_p = m_q = m_r = 1$ kg, $d_u = d_v = d_w = d_p = d_q = d_r = 1$ kg.



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

Figure 5. Simulated robot and thruster characteristics. (**a**) Simulated robot, unit of three axis is meter. (**b**) Thruster characteristics.

In general, the robot operates with a control loop, in which a controller derives a desired control vector $\mathbf{F}_B^d$. In this section, we describe our simulations of the problem (3) when $\mathbf{F}_B^d$ is the dynamic parameter. We simulated two missions—path-following and station-keeping (observation) problems—which are important in underwater robotics.

4.2. Path-Following Problem

For the path-following problem, a line of sight (LoS)-based guidance method [22] was used in this simulation. We compared the energy-like criterion between the two static configurations and the dynamic one in this mission. A PID controller was used in this simulation. The chosen path is a spatial ellipse, which is parameterized as follows:

$$x = 60\cos(0.2618t) \tag{17}$$
$$y = 60\sin(0.2618t) \tag{18}$$
$$z = \sin(0.2618t) + 5 \tag{19}$$

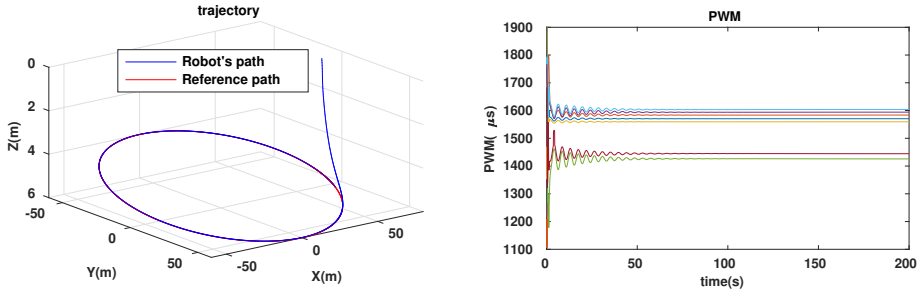
where t is a path parameter.

The desired composite speed is $\mathbf{U}_d = 2m/s$. The initial posture of the robot is $[x(0) \quad y(0) \quad z(0) \quad \phi(0) \quad \theta(0) \quad \psi(0)]^T = [64(m) \quad 3(m) \quad 0 \quad 0 \quad 0 \quad 3\pi/4]^T$. The initial speed of the AUV is $[u(0) \quad v(0) \quad w(0) \quad p(0) \quad q(0) \quad r(0)]^T = [1.5(m/s) \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$.

To evaluate the efficiency of our approach (dynamic configuration) in comparison with that of others (static configurations), we simulated the path-following problem for the three cases described in Table 1.

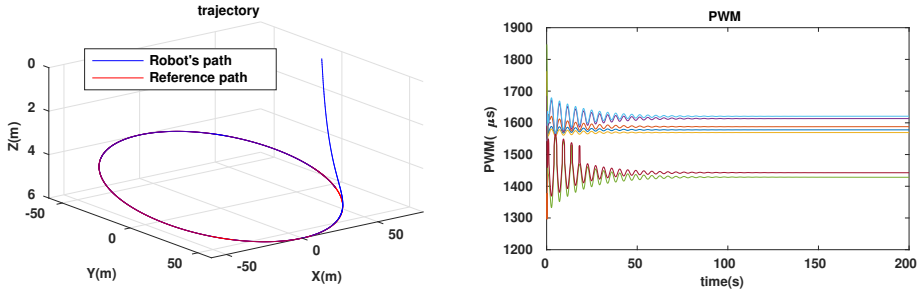
Table 1. Simulation cases for path-following problem.

No. Case	Two Angles α_F, α_B	Notes
1	$\alpha_F = \alpha_B = 70^0$	Simulation results in Figure 6
2	$\alpha_F = \alpha_B = 90^0$	Simulation results in Figure 7
3	dynamic	Simulation results in Figure 8



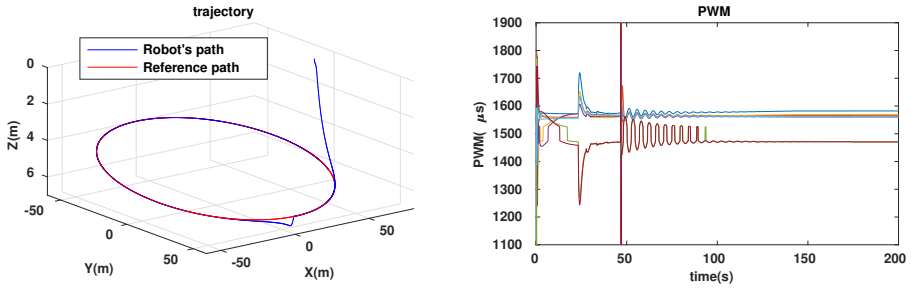
(**a**) (**b**)

Figure 6. Path following for ellipse with configuration ($\alpha_F = \alpha_B = 70^0$). (**a**) Trajectory of robot. (**b**) PWM of 7 thrusters.



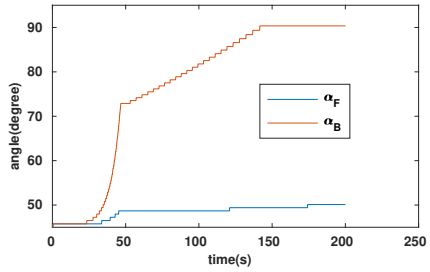
(**a**) (**b**)

Figure 7. Path following for ellipse with configuration ($\alpha_F = \alpha_B = 90^0$). (**a**) Trajectory of robot. (**b**) PWM of 7 thrusters.



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

Figure 8. *Cont.*



(**c**)

Figure 8. Path following for ellipse with dynamic configuration. (**a**) Trajectory of robot. (**b**) PWM of 7 thrusters. (**c**) Evolution of 2 angles.

The energy-like criterion evolutions of the path-following problem for the three simulation cases are shown in Figure 9. We found that the path-following performance was guaranteed for all three cases (see Figures 6a, 7a and 8a). However, with the dynamic configuration, from the energy perspective, the dynamic configuration showed better performance than the other two (see Figure 9). Note that this is only a local optimal solution, and in the configuration space, another optimal solution could exist. For each specific mission, from the initial values ($\alpha_F = \alpha_B = 45^\circ$), the robot configuration converges to a local optimal solution. If the mission is suddenly changed, this means that the desired control vector is largely disturbed. This will be carefully investigated and could be an interesting future work, so it is not mentioned in this paper. Following Figure 9, we have two instances at which the energy consumption of the dynamic configuration is larger than the fixed configuration. This happened because some disturbances were injected into the controller to investigate the response of the system with respect to the uncertainties, although this is out of the scope of this paper and will be in the future research. In particular, for the first instance, from $23.4(s)$ to $27.6(s)$, a small disturbance was injected into the controller, and for the second one, from $46.8(s)$ to $46.9(s)$, a very large disturbance was injected into the controller in a short time. With a small disturbance, the performance of the path-following problem was guaranteed. However, with a very large disturbance, following a path was not guaranteed (a peak in Figure 8a). Indeed, Theorem 1 was violated in this case.

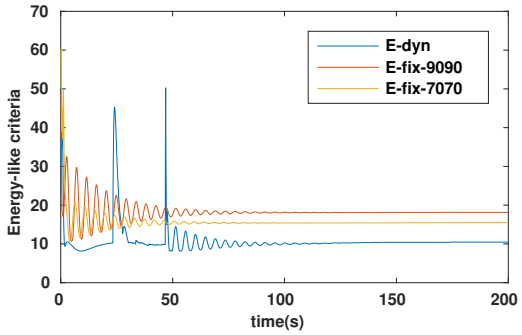


Figure 9. Energy-like criteria for path-following problem.

4.3. Station-Keeping (Observation) Problem

For the station-keeping (observation) problem, the robot normally has to rotate about some DOFs and maintain a constant position, e.g., constant depth. This probably could not be achieved by an under-actuated system, which has some uncontrollable DOFs. Owing to our robot's versatility, the dynamically reconfigurable robot can easily perform this mission. In this part, we present the simulation results of the observation problem with our robot, in which the robot dove to the desired depth and then rotated with the desired angular velocities, i.e., $[x^d \quad y^d \quad z^d](m)^T = [0 \quad 0 \quad 1]^T$ and $[p^d \quad q^d \quad r^d](rad/s)^T = [1 \quad 1 \quad 1]^T$.

The controller was designed with quaternion techniques to avoid singularities (gimbal lock) [23]. The simulations included fixed and dynamic configurations. The simulation results of the fixed configurations, in which $\alpha_F = \alpha_B = 90^0$, are shown in Figure 10. The simulation results of the dynamic configurations are depicted in Figure 11. Note that in the simulation, we assumed that all states of the robot could be completely measured or estimated.

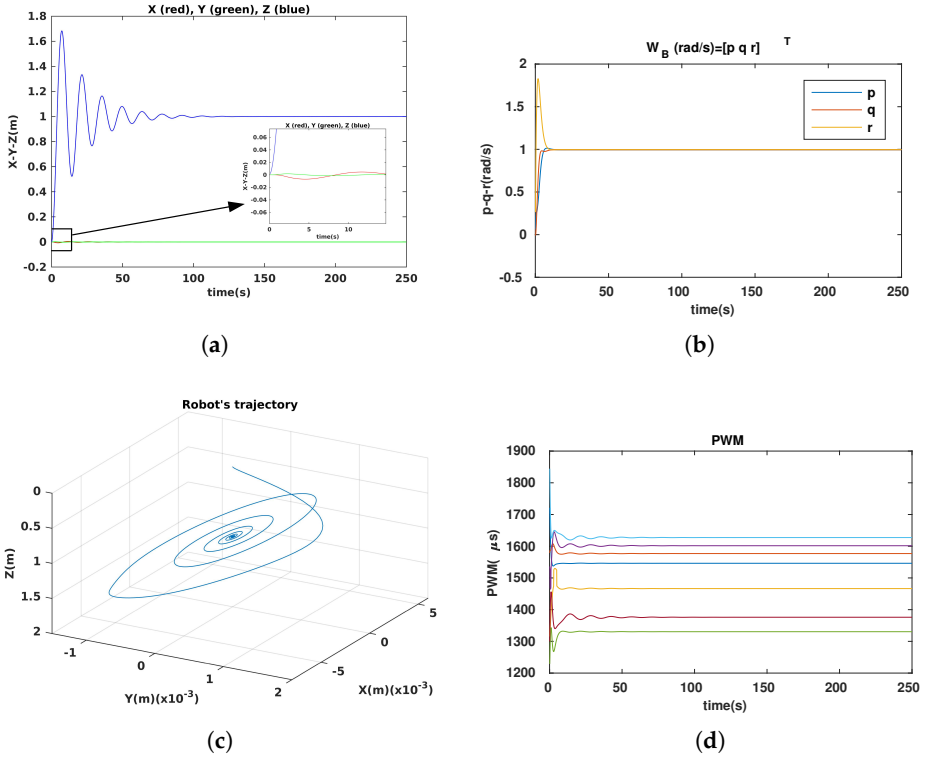


Figure 10. Simulation results with fixed configuration. (**a**) Positions of robot. (**b**) Angular velocities. (**c**) Robot trajectory. (**d**) PWM evolution of 7 thrusters.

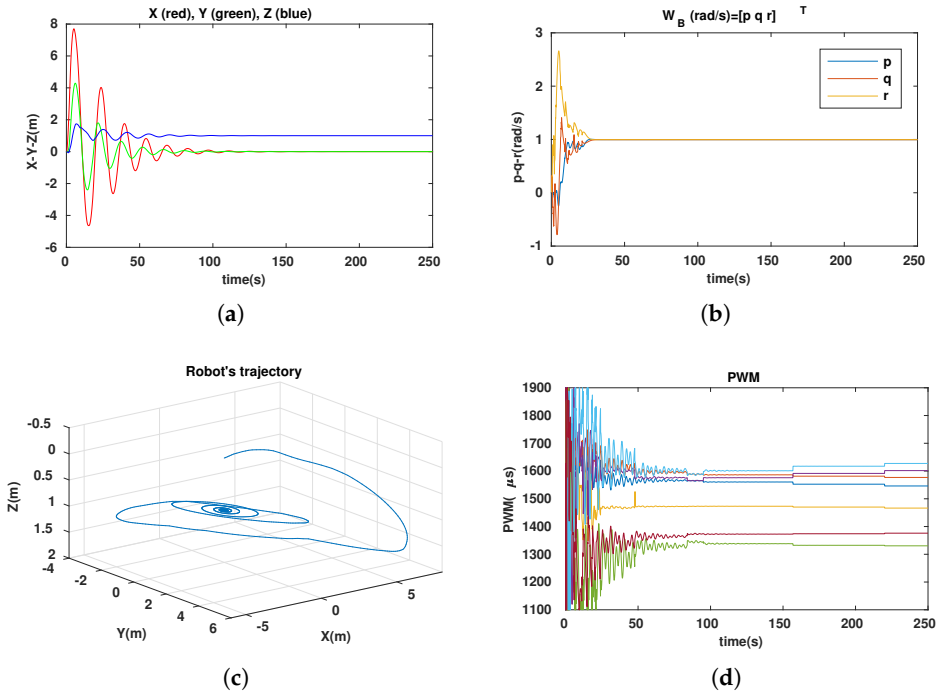
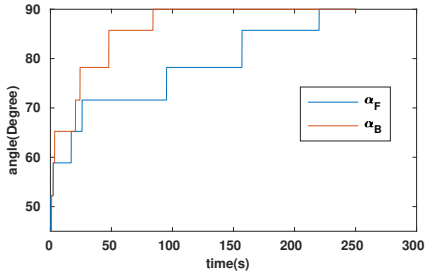


Figure 11. *Cont.*



(**e**)

Figure 11. Simulation results with dynamic configuration. (**a**) Positions of robot. (**b**) Angular velocities. (**c**) Robot trajectory. (**d**) PWM evolution of 7 thrusters. (**e**) Two angles' evolution: α_F-blue; α_B-red.

The control performance was guaranteed in both the fixed and dynamic configurations. The robot reached the desired depth (Figures 10a and 11a) and followed the desired angular velocities (Figures 10b and 11b). As shown in Figures 10d and 11d, the energy-like criterion of fixed configuration was better than that of dynamic configuration because the two angles $\alpha_F = \alpha_B = 90^\circ$ could be considered a local optimal solution in this mission (mainly the rotation task). The two angles' evolution is shown in Figure 11e, which converge to 90°, the same as in the fixed configuration. With initial value $\alpha_F = \alpha_B = 45^\circ$, the robot's configuration continues to improve through the samples. Following Figure 11 (observation problem), the normally rotating priority task is considered. A locally optimal fixed configuration (rotating priority) is chosen. The dynamic configuration converges to the chosen optimal fixed configuration through the time. This shows that our approach drove the system to the locally optimal configuration.

Remark 4. *The proposed method can be used to find the local energy-efficient configuration of a dynamically reconfigurable robot, as described in the previous sections, for problems in which the change in the parametric variable is small enough. However, missions requiring a large change in the parametric variable (the desired control vector) or one DOF to vary from uncontrollable to controllable or vice versa during configuration changes are outside the scope of the proposed method and will be a future research area in terms of system stability and controller design considering a switching mechanism between controllable and uncontrollable DOFs. The experiment we next describe highlights this remark.*

5. Experiment Results

The real robot was tested in a swimming pool with different configurations. In this test, the robot performed four tasks with two corresponding angles: travel straight (from point A to point B, $\alpha_F = \alpha_B = 45^\circ$), complete a turn 180° (around point B, $\alpha_F = 85^\circ, \alpha_B = 45^\circ$), dive to a predefined depth (from point B to point C, $\alpha_F = 85^\circ, \alpha_B = 85^\circ$), and perform a sway (from point C to point D, $\alpha_F = 85^\circ, \alpha_B = 85^\circ$). The robot trajectory is illustrated in Figure 12a. The values of the two angles (α_F, α_B) for each trajectory are depicted in Figure 12d. The desired control vector (output from the PID controller), including the desired force and torque elements, is shown in Figure 12b,c. Thanks to our robot, which can vary its configuration by changing two angles α_F, α_B, we can divide the feasible space of these two angles into four regions (Figure 13), in which the robot has priority in its DOFs: *surge priority*, *sway/heave priority*, and *rotating priority*. The term *priority* means that the robot prefers to use such a task in the priority region. For our experiment, the two angles converged to the corresponding priority region.

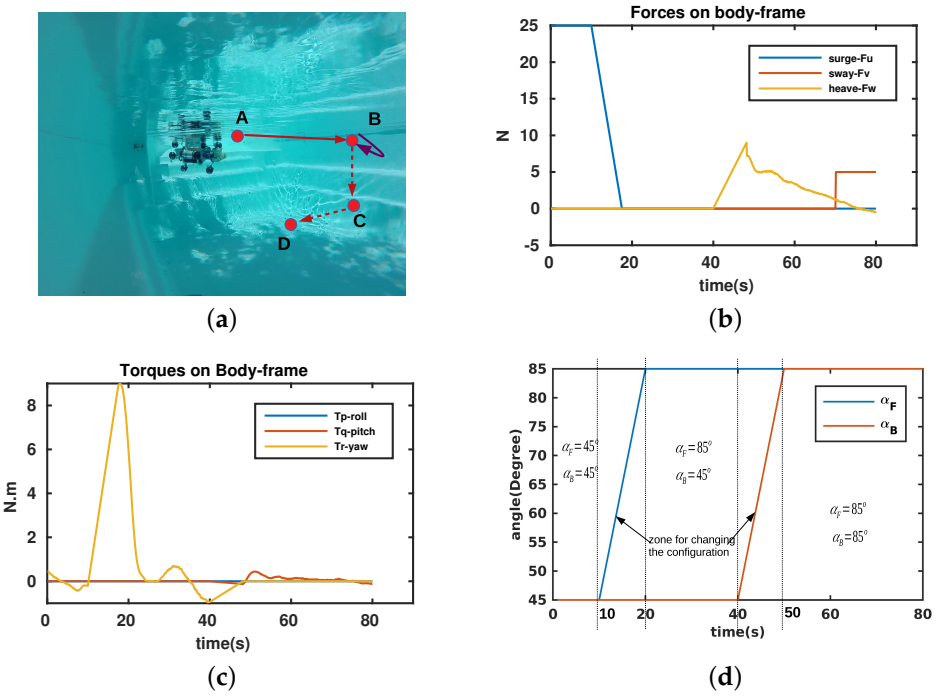


Figure 12. Experiment results. (**a**) Experiment descriptions. (**b**) Desired forces expressed in body frame. (**c**) Desired torques expressed in body frame. (**d**) Evolution of two angles.

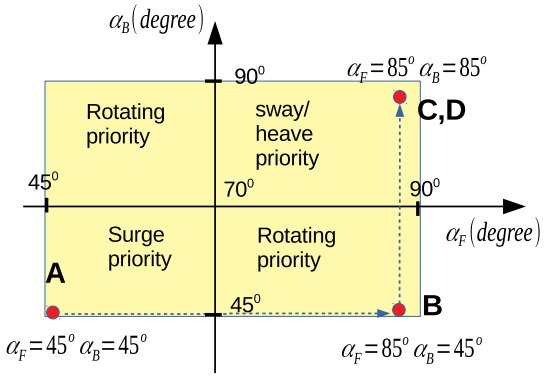


Figure 13. Two angles in feasible map and priority regions.

When we applied the proposed method to this trajectory, with initial values $\alpha_F = \alpha_B = 45°$, the trajectory was reasonable and optimal from point A to point B and when turning around point B. Nevertheless, to directly dive from point B to point C, the robot had to considerably change its configuration enough because the heave DOF is uncontrollable with $\alpha_F = \alpha_B = 45°$ and the proximity of these angle values. As such, the robot could not directly perform this dive. So, the robot can use a *hybrid mechanism* between dynamic and static states during its operations. For some parts of the trajectory, the robot can use the dynamic mechanism and, for others, the robot can use a static one with time spent to change the configuration.

6. Conclusions and Future Studies

In this paper, we proposed an approach for an energy-efficient configuration and the control allocation of a dynamically reconfigurable underwater robot that was built for karst or confined environment exploration. The energy-like criterion was minimized with respect to the robot constraints. The proposed method was solved online (for each sampling time), corresponding to the robot's dynamic configuration. The approach was

divided into two steps: a *predictor* step and a *corrector* step. In the *predictor* step, the solution of one iteration SQP was chosen for the robot configuration. In the *corrector* step, quadratic programming, as a classical CA method, was solved to adjust the applied force vector that could be assigned for the actuators. The simulation results showed the efficiency of the proposed method through the application of two problems: path following and station keeping (observation). In the future, we will perform real tests with this method. Moreover, external disturbances and model uncertainties will be considered in our subsequent studies.

Author Contributions: Conceptualization, T.D., L.L., R.Z.; methodology, T.D., L.L., R.Z. and B.R.; software, T.D., L.L. and R.Z.; validation, T.D., L.L. and R.Z.; writing—original draft preparation, T.D.; writing—review and editing, L.L.; supervision, L.L. and R.Z.; project administration, L.L.; funding acquisition, L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported by the LabEx NUMEV (ANR-10-LABX- 0020) within the I-SITE MUSE (ANR-16-IDEX-0006) and the Region Occitanie (french FEDER funds).

Acknowledgments: The authors would like to thank Numev Labex, MUSE, Montpellier University; Region Occitanie; and FEDER for supporting this study.

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

Abbreviations

The following abbreviations are used in this manuscript:

AUV	Autonomous Underwater Robot
SQP	Sequential Quadratic Programming
PWM	Pulse Width Modulation
DOF	Degree of Freedom
MPC	Model Predictive Control
CA	Control Allocation
PID	Proportional–Integral–Derivative controller

Appendix A. Configuration Matrix

The elements of the configuration matrix **A** are presented in Table A1. Note that d and $d_i, i \in \{F, e, t\}$ are geometrical distances corresponding to the shape of the robot. $\beta_i, i \in \{1, 2, ..., 7\}$ is an angle.

Table A1. Elements of **a** matrix.

$\mathbf{u}_1^B = \begin{pmatrix} \frac{1}{\sqrt{2}}(sin\alpha_F + cos\alpha_F) \\ \frac{\sqrt{3}}{2\sqrt{2}}(-cos\alpha_F + sin\alpha_F) \\ \frac{1}{2\sqrt{2}}(-cos\alpha_F + sin\alpha_F) \end{pmatrix}$	$\mathbf{r_1^B} = \begin{pmatrix} (d_F + L_F) - \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_1) \\ -\frac{d_t}{2} - \frac{\sqrt{3(d_e^2 + d^2)}sin(\alpha_F + \beta_1)}{2} \\ \frac{\sqrt{3}}{2}d_t - \frac{\sqrt{d_e^2 + d^2}sin(\alpha_F + \beta_1)}{2} \end{pmatrix}$
$\mathbf{u}_2^B = \begin{pmatrix} \frac{cos\alpha_F + sin\alpha_F}{\sqrt{2}} \\ 0 \\ \frac{cos\alpha_F - sin\alpha_F}{\sqrt{2}} \end{pmatrix}$	$\mathbf{r}_2^B = \begin{pmatrix} (d_F + L_F) - \sqrt{d^2 + d_e^2}cos(\alpha_F + \beta_2)) \\ d_t \\ \sqrt{d^2 + d_e^2}sin(\alpha_F + \beta_2) \end{pmatrix}$
$\mathbf{u}_3^B = \begin{pmatrix} \frac{1}{\sqrt{2}}(sin\alpha_F + cos\alpha_F) \\ \frac{\sqrt{3}}{2\sqrt{2}}(cos\alpha_F - sin\alpha_F) \\ \frac{1}{2\sqrt{2}}(sin\alpha_F - cos\alpha_F) \end{pmatrix}$	$\mathbf{r}_3^B = \begin{pmatrix} (d_F + L_F) - \sqrt{d_e^2 + d^2}.cos(\alpha_F + \beta_3) \\ \frac{d_t}{2} + \frac{\sqrt{3(d_e^2 + d^2)}}{2}.sin(\alpha_F + \beta_3) \\ \frac{\sqrt{3}}{2}d_t - \frac{\sqrt{d_e^2 + d^2}}{2}sin(\alpha_F + \beta_3) \end{pmatrix}$
$\mathbf{u}_4^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ \frac{-cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \end{pmatrix}$	$\mathbf{r}_4^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_4)) \\ -(\sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_4)) \\ -dt \end{pmatrix}$
$\mathbf{u}_5^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \\ \frac{cos\alpha_B - sin\alpha_B}{\sqrt{2}} \end{pmatrix}$	$\mathbf{r}_5^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2}cos(\alpha_B + \beta_5)) \\ d_t \\ \sqrt{d^2 + d_e^2}sin(\alpha_B + \beta_5) \end{pmatrix}$

Table A1. *Cont.*

$\mathbf{u}_6^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ \frac{cos\alpha_B - sin\alpha_B}{\sqrt{2}} \\ 0 \end{pmatrix}$	$\mathbf{r}_6^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2} cos(\alpha_B + \beta_6)) \\ \sqrt{d^2 + d_e^2} sin(\alpha_B + \beta_6) \\ -d_t \end{pmatrix}$
$\mathbf{u}_7^B = \begin{pmatrix} \frac{cos\alpha_B + sin\alpha_B}{\sqrt{2}} \\ 0 \\ \frac{-cos\alpha_B + sin\alpha_B}{\sqrt{2}} \end{pmatrix}$	$\mathbf{r}_7^B = \begin{pmatrix} -(d_F + \sqrt{d^2 + d_e^2} cos(\alpha_B + \beta_7)) \\ -d_t \\ -(\sqrt{d^2 + d_e^2} sin(\alpha_B + \beta_7)) \end{pmatrix}$

Appendix B. Notations

This section describes the notations used in the paper. However, further notations are introduced when needed.

$\mathbf{A}$	Configuration matrix
$\mathbf{u}_i^B$	(3×1) unit vector of direction of the *ith* thruster with respect to body frame
$\mathbf{r}_i^B$	(3×1) unit vector of position of the *ith* thruster with respect to body frame
$\mathbf{F}_m$	$(m \times 1)$ applied force vector of m thrusters
$F_{m,i}$	Applied force magnitude of the *ith* thruster
$\mathbf{F}_B^d$	(6×1) desired control vector (including force and torque) with respect to body frame
$\mathbf{F}_B = \begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix}$	(6×1) resulting control vector (including force and torque) with respect to body frame
$\otimes$	Cross product
$\|\cdot\|$	Euclidean norm
m	Number of thrusters
n	Number of degrees of freedom (DOFs)

References

1. Yim, M.; Shen, W.M.; Salemi, B.; Rus, D.; Moll, M.; Lipson, H.; Klavins, E.; Chirikjian, G.S. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robot. Autom. Mag.* **2007**, *14*, 43–52. [CrossRef]
2. Liu, J.; Zhang, X.; Hao, G. Survey on research and development of reconfigurable modular robots. *Adv. Mech. Eng.* **2016**, *8*, 1687814016659597. [CrossRef]
3. Dang, T.; Lapierre, L.; Zapata, R.; Ropars, B.; Gourmelen, G. A Dynamically Reconfigurable Autonomous Underwater Robot for Karst Exploration: Design and Experiment. *Sensors* **2022**, *22*, 3379. [CrossRef] [PubMed]
4. Page, A.B.; Steinberg, M.L. A closed-loop comparison of control allocation methods. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Dever, CO, USA, 14–17 August 2000; pp. 1760–1770.
5. Bodson, M. Evaluation of optimization methods for control allocation. *J. Guid. Control Dyn.* **2002**, *25*, 703–711 [CrossRef]
6. Fossen, T.I.; Johansen, T.A. A Survey of Control Allocation Methods for Ships and Underwater Vehicles. In Proceedings of the 2006 14th Mediterranean Conference on Control and Automation, Ancona, Italy, 28–30 June 2006; pp. 1–6.
7. Fossen, T.I.; Johansen, T.A.; Perez, T. *A Survey of Control Allocation Methods for Underwater Vehicles*; INTECH Open Access Publisher: London, UK, 2009.
8. Johansen, T.A.; Fossen, T.I. Control allocation—A survey. *Automatica* **2013**, *49*, 1087–1103. [CrossRef]
9. Skulstad, R.; Li, G.; Zhang, H.; Fossen, T.I. A Neural Network Approach to Control Allocation of Ships for Dynamic Positioning. *IFAC* **2018**, *51*, 128–133. [CrossRef]
10. Kang, J.; Choi, K. Development of an Artificial Neural Network Control Allocation Algorithm for Small Tailless Aircraft Based on Dynamic Allocation Method. *Int. J. Aeronaut. Space Sci.* **2022**, *23*, 363–378. [CrossRef]
11. Johansen, T.A.; Fossen, T.I.; Berge, S.P. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. *IEEE Trans. Control Syst. Technol.* **2004**, *12*, 211–216. [CrossRef]
12. Härkegård, O. Backstepping and Control Allocation with Applications to Flight Control. Ph.D. Thesis, Linköpings Universitet, Linköping, Sweden, 2003.
13. Casavola, A.; Garone, E. Fault-tolerant adaptive control allocation schemes for overactuated systems. *Int. J. Robust Nonlinear Control* **2010**, *20*, 1958–1980. [CrossRef]
14. Tohidi, S.S.; Yildiz, Y.; Kolmanovsky, I. Fault tolerant control for over-actuated systems: An adaptive correction approach. In Proceedings of the 2016 American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 2530–2535.
15. Cristofaro, A.; Polycarpou, M.M.; Johansen, T.A. Fault-Tolerant Control Allocation for Overactuated Nonlinear Systems. *Asian J. Control* **2018**, *20*, 621–634. [CrossRef]
16. Diehl, M.; Bock, H.G.; Schlöder, J.P. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control Optim.* **2005**, *43*, 1714–1736. [CrossRef]

17. Nocedal, J.; Wright, S. *Numerical Optimization*; Springer Science & Business Media: New York, NY, USA, 2006.
18. Diehl, M. Real-Time Optimization for Large Scale Nonlinear Processes. Ph.D. Thesis, Heidelberg University, Heidelberg, Germany , 2001.
19. Fletcher, R. *Practical Methods of Optimization*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
20. Ropars, B.; Lasbouygues, A.; Lapierre, L.; Andreu, D. Thruster's dead-zones compensation for the actuation system of an underwater vehicle. In Proceedings of the Control Conference (ECC), 2015 European, Linz, Austria, 15–17 July 2015; pp. 741–746.
21. BlueRobotics. Available online: https://bluerobotics.com/ (accessed on 4 March 2021).
22. Breivik, M.; Fossen, T.I. A unified control concept for autonomous underwater vehicles. In Proceedings of the 2006 American Control Conference, Minneapolis, MN, USA, 14–16 June 2006; pp. 7–pp.
23. Louis, S.; Lapierre, L.; Godary-Dejean, K.; Onmek, Y.; Claverie, T.; Villéger, S. *Quaternion Based Control for Robotic Observation of Marine Diversity*; OCEANS: Aberdeen, UK, 2017.