Centralised versus Decentralised Control Reconfiguration for Collaborating Underwater Robots

Lidia Furno [*] **Mikkel Cornelius Nielsen** [**] **Mogens Blanke** [*,**]

[*] *Automation and Control Group, Department of Electrical Engineering, Technical University of Denmark, Kgs.Lyngby, Denmark (e-mail: {furno,mb}@elektro.dtu.dk)*
[**] *AMOS CoE, Institute of Technical Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway (e-mail:mikkel.cornelius.nielsen@itk.ntnu.no)*

Abstract: The present paper introduces an approach to fault-tolerant reconfiguration for collaborating underwater robots. Fault-tolerant reconfiguration is obtained using the virtual actuator approach (Steffen, 2005). The paper investigates properties of a centralised versus a decentralised implementation and assesses the capabilities under communication constraints between the individual robots. In the centralised case, each robot sends information related to its own status to a unique virtual actuator that computes the necessary reconfiguration. In the decentralised case, each robot is equipped with its own virtual actuator that is able to accommodate both local faults and faults within a collaborating unit. The paper discusses how this is done through exploiting structural information (e.g. thruster configuration) for each participant in the cooperation. A test scenario is presented as a case in which an underwater drill need be transported and positioned by three collaborating robots as part of an underwater autonomous operation.

Keywords: Collaborating robots, underwater robotics, fault tolerance control, actuator fault, reconfiguration, distributed system.

1. INTRODUCTION

In recent years, the interest in collective robotics, which provides a new perspective based on cooperation between multiple robots, has grown significantly. The gain is that tasks can be performed in a more efficient and faster way compared to a single robot. In order to accomplish a specific goal, the interaction of robots working together has to be controlled through a centralised unit that takes over the individual local controllers to ensure optimal coordination and synchronization or through decentralised units to have low impact on communication.

In underwater environments, robots are subject to failures during their mission, so it is fundamental to examine the fault-tolerant aspects. Fault-tolerance is the property that enables the system to continue operating properly in case of non functional elements. A common source of failures are thrusters, due to their duct that can be easily blocked by seaweed or underwater life that can be sucked into the water flow.

The fault reconfiguration problem is a very active research topic, hence many different solutions have been proposed.

Yang et al. (1998, 1999) proposed a method for fault-tolerant reconfiguration where a failure of a thruster meant eliminating a thruster entirely from the equation to make the thruster configuration matrix invertible if sufficient redundancy remained in the system. The method considered complete thruster failures only. This problem was further addressed in Podder et al. (2000), where a weighted pseudo-inverse was introduced to optimize the thruster force distribution. The principle of reconfiguration through optimization of the control allocation gained interest over the last decade as the computational power increased (Indiveri and Parlangeli, 2006). Sarkar et al. (2002) used the pseudo-inverse method to obtain fault accommodation, and it is noted that in case the faulty thruster configuration matrix does not contain full rank, the pseudo-inverse method cannot fulfil the reconfiguration goal. Omerdic and Roberts (2004) introduced a fault-tolerant control system using the pseudo-inverse method and to avoid infeasible solutions performed approximations through truncation and scaling to achieve an optimal feasible allocation. The method considered three types of faults: jammed propeller, heavily-jammed propeller and broken propeller. Zhu et al. (2008) employed a paradigm that considered faults to develop gradually and used fault magnitude estimation to compensate for thruster loss of efficiency. Sliding mode control was employed in (Corradini et al., 2011), where the inherent robustness of the sliding mode controller was shown to be able to account for a range of thruster faults.

The present paper adopts a reconfiguration block (virtual actuator) to accommodate faults in a cluster of multiple underwater robots who collaborate to perform a task. A centralised and a decentralised implementation of a virtual actuator are presented. The former reconfigures the par-

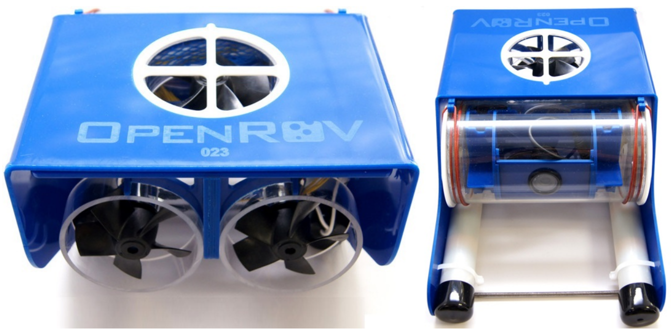
Fig. 1. OpenROV, an open source underwater robot for educational purposes.

ticipating robots which inform the unique virtual actuator about the status of their thrusters. The latter exploits structural information (e.g., where each robot is located in the cluster) to accommodate faults through a local virtual actuator. Choosing one implementation over the other is shown to depend on requirements and capabilities of the system (e.g., available communication channels). The faulty system together with the virtual actuator is shown to provide approximately the same output as the nominal system. It is demonstrated that almost the same path following capability is achieved in case of actuator faults, for both centralised and decentralised implementations, even if the control performance may be degraded (less agile as less thrust being available).

The paper is organised as follows. An example scenario is first introduced to motivate the concept of collaborating UAVs. UAV dynamics and kinematics is then revisited and a conventional path control is selected as a standard controller for both centralised and individual UAV robot control. Handling of thruster faults is then discussed selecting a model matching technique and the virtual actuator approach for reconfiguration. Centralised and decentralised control and reconfiguration architectures are then analysed in view of the limited communication channel capabilities under water. The main result is to show that an architecture for distributed reconfiguration with limited inter-robot communication is feasible.

2. EXAMPLE SCENARIO

A test scenario has been created to expose the properties of a centralised versus decentralised control reconfiguration for collaborating underwater robots. The scenario has a background in a real life case where cracks occur in the structure of oil platforms due to severe weather, constant presence of extreme stresses and design flaws (e.g.,the North Sea Oil platform, Siri). We focus on the transportation phase in which a drill is carried through a cut-out in the wall of the chamber in which repair service is needed. The drill is supported by three identical AUVs that are envisaged to have an attachment system that allows autonomous pick up and connection. Each robot has similar chassis and thruster configuration as an OpenROV (see Fig. 1). Each robot is equipped with three fixed direction (pitch) thrusters: two are mounted in parallel in a rear-facing configuration and one is transversal. This thruster configuration allows the robots to move in three dimensions but the single robot is under-actuated by having only

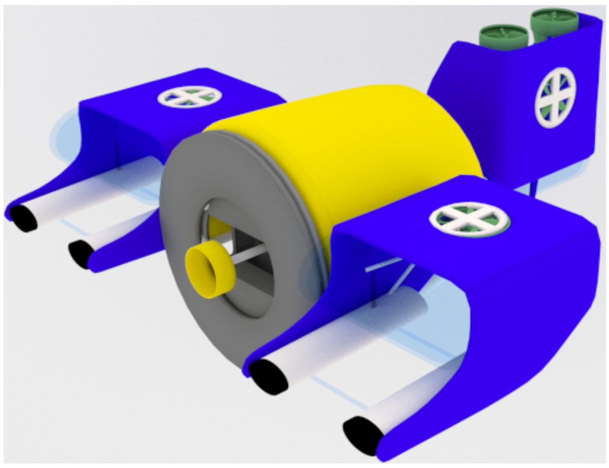
Fig. 2. The cluster comprising three AUVs (*S1*, *S2* and *S3* are the robot on the left, right and back, respectively) attached to the drill. The earth reference frame is NED.

3 Degrees of Freedom (3DOF): surge, yaw and heave. It follows that the single robot lacks manoeuvrability in sway, roll and pitch.

If a tool needs to be carried to a desired position, the robots attach to the tool in defined positions. A configuration is shown in Fig. 2: the robots on the left, right and in back are referred to as *S1*, *S2* and *S3*. When firmly engaged, the cluster has nine thrusters: five are "horizontal"; four are "vertical", seen in body coordinates. The robot that comprises the cluster of the tool and three robots is over-actuated and is able to move in 6DOF, (see Table 1).

Table 1. SNAME notation for marine crafts

DOF	τ	ν	η
Translation along x-axis (surge)	X	u	x
Translation along y-axis (sway)	Y	v	y
Translation along z-axis (heave)	Z	w	z
Rotation about x-axis (roll)	K	p	ϕ
Rotation about y-axis (pitch)	M	q	θ
Rotation about z-axis (yaw)	N	r	ψ

3. DYNAMICS

The dynamics of an individual robot follows its' own equations of motion while free, but follows the dynamics of the cluster after attachment. Following Fossen (1994), the non-linear equations of motion are, in body-fixed coordinates,

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \tag{2}$$

$\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^T$ refers to the position and orientation vector with coordinates in the North-East-Down(NED) earth-fixed frame. Orientation is represented by Euler angles: roll(ϕ), pitch(θ) and yaw(ψ).

$\boldsymbol{\nu} = [u, v, w, p, q, r]^T$ refers to the linear and angular velocity vector with coordinates in the body-fixed frame, which centre overlaps with the centre of gravity of the robot.

M is the inertia matrix which consists of the rigid-body mass matrix $\boldsymbol{M_{RB}}$ and the added mass matrix $\boldsymbol{M_A}$ due to the inertia of the surrounding fluid and can be written as:

$$\boldsymbol{M} = \boldsymbol{M_{RB}} + \boldsymbol{M_A} \tag{3}$$

$\boldsymbol{C(\nu)}$ is the matrix of Coriolis and centripetal terms (comprising added mass). Due to the low velocity at which the robots move, its effect on the dynamics is not worthy of being considered.

$\mathbf{D}(\boldsymbol{\nu})$ is the damping matrix that has the general form:

$$\boldsymbol{D(\nu)} = \boldsymbol{D_P(\nu)} + \boldsymbol{D_V(\nu)} + \boldsymbol{D_N(\nu)}, \tag{4}$$

where $\boldsymbol{D_P(\nu)}$ is damping due to the lost energy generated by surface waves (neglected for an AUV), $\boldsymbol{D_V(\nu)}$ and $\boldsymbol{D_N(\nu)}$ denote, respectively, the linear and nonlinear viscous damping caused by skin friction, vortex shedding and lift/drag effects, Fossen (2011).

$\mathbf{g}(\boldsymbol{\eta})$ is the vector of buoyant and gravitational forces (restoring forces) that, respectively, act through the centre of gravity and centre of buoyancy of the vehicle.

$\boldsymbol{J(\eta)}$ is a transformation matrix which is related through the functions of Euler angles. It allows to change reference frames easily.

$\boldsymbol{\tau} = [X, Y, Z, K, M, N]^T$ refers to forces and moments influencing the vehicle in the body-fixed frame used to achieve the desired motion.

$\boldsymbol{T} \in \Re^{d \times h}$ is the thrust configuration matrix that represents the position and orientation of each thruster in the system. $\boldsymbol{T}$ is defined as (Omerdic and Roberts, 2004):

$$\boldsymbol{T} = \begin{bmatrix} {}^1\boldsymbol{e_x} & & {}^j\boldsymbol{e_x} & & {}^h\boldsymbol{e_x} \\ {}^1\boldsymbol{e_y} & \dots & {}^j\boldsymbol{e_y} & \dots & {}^h\boldsymbol{e_y} \\ {}^1\boldsymbol{e_z} & & {}^j\boldsymbol{e_z} & & {}^h\boldsymbol{e_z} \\ ({}^1\boldsymbol{l} \times {}^1\boldsymbol{e})_x & & ({}^j\boldsymbol{l} \times {}^j\boldsymbol{e})_x & & ({}^h\boldsymbol{l} \times {}^h\boldsymbol{e})_x \\ ({}^1\boldsymbol{l} \times {}^1\boldsymbol{e})_y & \dots & ({}^j\boldsymbol{l} \times {}^j\boldsymbol{e})_y & \dots & ({}^h\boldsymbol{l} \times {}^h\boldsymbol{e})_y \\ ({}^1\boldsymbol{l} \times {}^1\boldsymbol{e})_z & & ({}^j\boldsymbol{l} \times {}^j\boldsymbol{e})_z & & ({}^h\boldsymbol{l} \times {}^h\boldsymbol{e})_z \end{bmatrix} \tag{5}$$

where d is the number of DOF, h is the number of thrusters, $\boldsymbol{e} = [e_x \; e_y \; e_z]^T$ is a unit vector that defines the orientation of a thruster and $\boldsymbol{l} = [l_x \; l_y \; l_z]^T$ is the position vector of a thruster with respect to the centre of gravity. $\boldsymbol{\tau}$ is obtained by exploiting $\boldsymbol{T}$ as shown in sub-section 4.3.

Table 1 shows the SNAME notation used in this paper. Appendix A provides matrices and parameters used to simulate the example scenario in section 2.

4. NOMINAL CONTROLLER ARCHITECTURE

The standard control structure is shown in Fig. 3. The output of the controller is the desired vector of forces and moments $\boldsymbol{\tau_d}$. Then, the control allocation block maps $\boldsymbol{\tau_d}$ into a vector control inputs $\boldsymbol{u_c}$ with as many elements as the actual number of thrusters. Control inputs are exploited to generates a vector of propulsion forces and moments $\boldsymbol{\tau}$ that is the input to the dynamics of the robot. The main objective of the control allocation is to ensure that the condition $\boldsymbol{\tau_d} = \boldsymbol{\tau}$ is satisfied for all attainable $\boldsymbol{\tau_d}$, Omerdic and Roberts (2004).

The system comprises an heading controller, a forward speed controller and a vertical controller. In the present paper, a 2D path control is used for simplification.

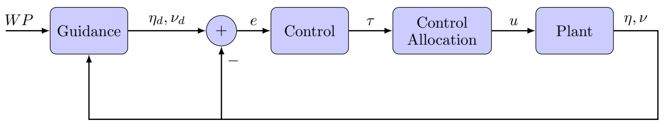

Fig. 3. The structure comprises a guidance block generating the path following, a control block that regulates the outputs to the desired values and a control allocator block for obtaining the vector of control input $\boldsymbol{u}$ to feed the plant.

4.1 Heading controller

The controller acts on the vehicle in order to perform path tracking. In our scenario, the robot has to reach the given waypoints (provided by a human or another robot) in order to get as close as possible to the crack that must be prevented from extending further. It is supposed that an inspection robot has been sent before the transportation mission to identify the exact position of the crack.

The LOS guidance look ahead approach in Breivik and Fossen (2011) exploits the specified waypoints and the current positions in the earth fixed-frame in order to generate a reference ψ_r. Afterwards, a low pass filter (LP) is used as a linear reference model for trajectory generation, defined as follows:

$$\psi_d(s) = h_{LP}(s)\psi_r(s) \tag{6}$$

A PID is used to compare the desired heading with the measured heading and to provide the yaw moment τ_N.

According to Fossen (2011), a feed-forward term τ_{FF} should be determined such that perfect tracking during course-changing manoeuvres is obtained and it is implemented according to:

$$\tau_{FF} = m_{PID}(\dot{r}_d + \frac{1}{T_{PID}} r_d), \tag{7}$$

where $m_{PID} = I_z - N_{\dot{r}}$ and $T_{PID} = -\frac{m_{PID}}{N_r}$.

The controller for the yaw moment τ_N is defined as:

$$\tau_N(s) = \tau_{FF}(s) - K_P(1 + T_D s + \frac{1}{T_I s})e_\psi(s), \tag{8}$$

where $e_\psi = \psi - \psi_d$ is the heading error, $K_P > 0$ is the PID proportional gain constant, $T_D > 0$ is the PID derivative time constant and $T_I > 0$ is the PID integral time constant.

When a decentralised controller is used, just one robot uses the PID control law previously explained. The other robots use a PD type control law with a droop switch around the integrator term in the PID that was active before physical connection of the robots was activated.

4.2 Control allocation

The input to the control allocation block is the vector of desired forces and moments $\boldsymbol{\tau_d}$, in which τ_N is the controlled moment. The control allocator maps $\boldsymbol{\tau_d}$ into control inputs $\boldsymbol{u_c}$ that indicates the number of revolutions (rpm) that each thruster have to perform to achieve the desired forces and moments. It is defined as:

$$\boldsymbol{u_c} = \boldsymbol{K^{-1}T^{\dagger}\tau_d} = \boldsymbol{H\tau_d} \tag{9}$$

$\boldsymbol{T}$ is the thrust configuration matrix defined in (5). $\boldsymbol{K} \in \Re^{h \times h}$ is the force coefficient matrix:

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_{hh} \end{bmatrix} \tag{10}$$

and:

$$k_{jj} = \frac{T_{j_{max}}}{\rho D^4 \left|n_{j_{max}}\right| n_{j_{max}}} \qquad j = 1, \ldots, h \tag{11}$$

where $T_{j_{max}}$ is the maximum thrust provided, ρ is the water density, D is the propeller diameter, $u_{j_{max}} = |n_{j_{max}}| n_{j_{max}}$ is the maximum thruster revolution (rpm) and h is the number of thrusters. The force coefficients are shown in Appendix A.3. A general overview of approaches used in thruster allocation is available in Johansen and Fossen (2013).

4.3 Computation of $\boldsymbol{\tau}$

The vector of propulsion forces and moments $\boldsymbol{\tau}$ is defined as:

$$\boldsymbol{\tau} = \boldsymbol{TKu_c} = \boldsymbol{Bu_c}, \tag{12}$$

which is the input to the dynamics of the system.

Both centralised and decentralised controllers utilize the methods previously explained. Regarding the example scenario in section 2, the former considers all the thrusters of the system (i.e., nine) and the latter comprises the local thrusters of the individual robots (i.e., three).

5. RECONFIGURATION BY MEANS OF A VIRTUAL ACTUATOR

A fault causes a variation in the characteristics of a component such that the mode of operation or performance is changed in an undesired way. It has to be well distinguished from a failure that describes the inability of a system or a component to accomplish its function. A fault-tolerant control has to prevent a fault from causing a failure at the system level, Blanke et al. (2006).

In our scenario, a robot is most likely to have faults in actuators because of seaweed that can be stuck into the propeller's duct or sea life that can be sucked into the water flow. An actuator fault changes or interrupt the effect of the controller on the plant which laws remain intact.

This paper explores a reconfiguration method, the virtual actuator approach, illustrated in Fig. 4. The figure shows a nominal controller with input $\boldsymbol{\eta_c}$ and output $\boldsymbol{u_c}$, a plant with input $\boldsymbol{u_f}$ and measured output $\boldsymbol{\eta_f}$. A reconfiguration block constitutes the virtual actuator. Finally, $\boldsymbol{\eta_{ref}}$ is the reference signal. In the nominal case, the I/O relations are:

$$\begin{aligned} \boldsymbol{u_c} &= \boldsymbol{u_f} \\ \boldsymbol{\eta_c} &= \boldsymbol{\eta_f} \end{aligned} \tag{13}$$

5.1 Centralised reconfiguration

A virtual actuator replaces the effect of the faulty actuator by exploiting the other control inputs and it implicates the use of diverse I/O relations between the nominal controller and the plant.

The idea of reconfiguration is to build a unit for handling faults in the already existing control structure. The nominal controller is part of the reconfigured control. Thus, the controller that has been specifically designed to ensure the desired goals can still be used when needed. Finally, the purpose of the virtual actuator approach is to satisfy the fault hiding goal, which hides faults from the controller. Other actuator reconfiguration approaches are available that aim at model matching, including Yang and Blanke (2000) and Yang et al. (2007), but this paper will focus on the virtual actuator approach Steffen (2005).

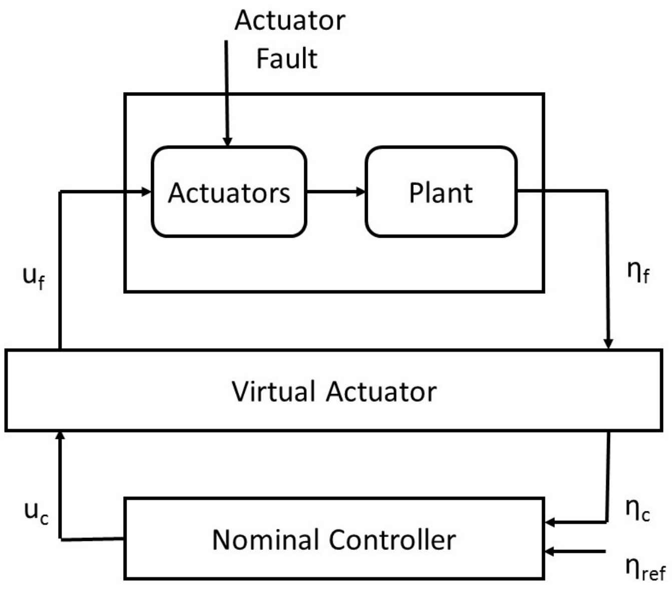

Fig. 4. Virtual Actuator for actuator faults.

The virtual actuator is described as the following state space system, Steffen (2005):

$$\dot{\boldsymbol{\nu}}_{\boldsymbol{\Delta}} = \boldsymbol{A_\Delta \nu_\Delta} + \boldsymbol{B_\Delta u_c}, \qquad \boldsymbol{\nu_\Delta(0)} = \boldsymbol{\nu_{\Delta 0}} \tag{14}$$
$$\boldsymbol{u_f} = \boldsymbol{C_\Delta \nu_\Delta} + \boldsymbol{D_\Delta u_c} \tag{15}$$
$$\boldsymbol{\eta_c} = \boldsymbol{C\nu_\Delta} + \boldsymbol{\eta_f} \tag{16}$$

with matrices:

$$\boldsymbol{A_\Delta} = \boldsymbol{A} - \boldsymbol{B_f M_s} \tag{17}$$
$$\boldsymbol{B_\Delta} = \boldsymbol{B} - \boldsymbol{B_f N} \tag{18}$$
$$\boldsymbol{C_\Delta} = \boldsymbol{M_s} \tag{19}$$
$$\boldsymbol{D_\Delta} = \boldsymbol{N} \tag{20}$$
$$\boldsymbol{A} = \boldsymbol{D} \tag{21}$$

$\boldsymbol{D}$ is the damping matrix (see appendix A.2).

$\boldsymbol{A}$ is the state matrix in which the restoring terms are not considered because of the stability of roll and pitch.

The matrix $\boldsymbol{B}$ maps the actuators' inputs to the forces and moments to be used in the plant (see section 4.3).

The matrix $\boldsymbol{B_f}$ is constructed in case of a total fault (failure) in the j_{th} actuator. $\boldsymbol{B_f}$ is the same as $\boldsymbol{B}$ besides the j_{th} column that is deleted.

The matrix $\boldsymbol{M_s}$ has to be chosen so that the matrix $\boldsymbol{A} - \boldsymbol{B_f M_s}$ has eigenvalues with negative real parts in order to ensure the stability of the reconfigured closed loop system, Blanke et al. (2006). Thus, the pair ($\boldsymbol{A}$,$\boldsymbol{B_f}$) is stabilizable.

The matrix $\boldsymbol{N}$ can be chosen such as:

$$\boldsymbol{N} = (\boldsymbol{B_f^T B_f})^{-1} \boldsymbol{B_f^T B} \tag{22}$$

In order to ensure that the I/O relations are the same as (13), it is preferred to choose $\boldsymbol{N}$ such as:

$$\boldsymbol{B_\Delta} = \boldsymbol{B} - \boldsymbol{B_f N} = \boldsymbol{O} \tag{23}$$

The condition (23) is satisfied when:

$$rank\ \boldsymbol{B_f} = rank\ ([\boldsymbol{B}, \boldsymbol{B_f}]) \tag{24}$$

In this case, the virtual actuator can be reduced to a static reconfiguration block:

$$\boldsymbol{u_f} = \boldsymbol{N u_c} \tag{25}$$

$$\boldsymbol{\eta_f} = \boldsymbol{\eta_c} \tag{26}$$

and the matrix $\mathbf{N}$ satisfies the property:

$$\sum_{j=1}^{h} n_{jw} = 1 \qquad w = 1, \ldots, h \tag{27}$$

where h is the number of thrusters. The property (27) produces $\boldsymbol{u_f} = \boldsymbol{u_c}$.

In the present paper, the measurement compensation in (16) is not considered since the controller should refer to the actual position and orientation.

A virtual actuator ensures the same (or approximately the same) performance to the system. By having a centralised controller and a virtual actuator, sharing information related to the status of each thruster is essential. The virtual actuator assumes that each thruster is completely functional unless it receives a message stating the contrary. The message consists in the contribution of the faulty robot to the matrix $\boldsymbol{N}$ in (22). Afterwards, the virtual actuator asks for the contribution of the other robots participating in the cooperation. Once it collects their status, it constructs the $\boldsymbol{N}$ matrix by joining the information, computes the new inputs $\boldsymbol{u_f}$ and sends them to the robots.

Algorithm *Centralised reconfiguration*

Given:

o robots named $Si,\ i = 1, \ldots, o$,

h_{Si} thrusters corresponding to robot Si and $h = \sum_{i=1}^{o} h_{Si}$,

f_{Si} actuator faults in robot Si, $f_{Si} \leq h_{Si}$, $f = \sum_{i=1}^{o} f_{Si}$,

matrices $\boldsymbol{B_{Si}}$ and $\boldsymbol{B_{f_{Si}}}$ defined in (12) and subsection 5.1,

a centralised controller (see section 4) and a virtual actuator located in one of the robots.

Problem:

Find $\boldsymbol{u_f}$ defined in (25) such that the fault hiding goal (see subsection 5.1) and the performance indicator ($\|\boldsymbol{\eta} - \boldsymbol{\eta_f}\| \simeq 0$) is satisfied.

Solution:

- The faulty robot Si sends a triangular matrix ${}^{\boldsymbol{t}}\boldsymbol{S_{f_{Si}}}$ to the virtual actuator, obtained by the symmetric property of $\boldsymbol{S_{f_{Si}}}$,which is a part of $\mathbf{N}$:

  $$\boldsymbol{S_{f_{Si}}} = \boldsymbol{B^T_{f_{Si}} B_{Si}} \tag{28}$$

  ${}^{\boldsymbol{t}}\boldsymbol{S_{f_{Si}}}$ is sent in order to reduce the use of the communication channel.
- The virtual actuator asks the other robots for their local contribution as in (28).
- The robots receiving the request construct their local contribution and send it to the virtual actuator. In case of a fault-free robot, the $\boldsymbol{B_{Si}}$ has to be considered instead of $\boldsymbol{B_{f_{Si}}}$.
- Once the virtual actuator obtains the information needed, it constructs the matrix $\boldsymbol{S}$ as:

  $$\boldsymbol{S} = \boldsymbol{S_{S1}} \circ \cdots \circ \boldsymbol{S_{So}} \tag{29}$$
- The virtual actuator computes the global matrix $\boldsymbol{B_f^T B_f}$ by deleting the f columns of $\boldsymbol{S}$ corresponding to actuator faults.
- The virtual actuator computes $\boldsymbol{N}$ as in (22) and finds $\boldsymbol{u_f}$.

Example scenario: Fault on $S1$'s vertical thruster

Contribution from $S1$:

$$\boldsymbol{B^T_{f_{S1}} B_{S1}} = {}^{\boldsymbol{t}}\boldsymbol{S_{f_{S1}}} = \begin{bmatrix} s_{S1_1} & 1 & 0 & 0 \\ s_{S1_1} & 1 & 1 & 0 \\ s_{S1_1} & 1 & 1 & 1 \end{bmatrix}$$

Contribution from S2:

$$\boldsymbol{B^T_{S2} B_{S2}} = {}^{\boldsymbol{t}}\boldsymbol{S_{S2}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ s_{S2_1} & s^2_{S2_1} & 0 & 0 \\ 1 & s_{S2_1} & 1 & 0 \\ 1 & s_{S2_1} & 1 & 1 \end{bmatrix}$$

Contribution from S3:

$$\boldsymbol{B^T_{S3} B_{S3}} = {}^{\boldsymbol{t}}\boldsymbol{S_{S3}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ s_{S3_1} & s_{S3_1} & s^2_{S3_1} & 0 \\ s_{S3_2} & s_{S3_2} & s_{S3_1}s_{S3_2} & s^2_{S3_2} \end{bmatrix}$$

Note that subscripts identify thrusters in the same robot. Finally, horizontal thrusters are omitted for clarity.

5.2 Decentralised reconfiguration

Communication is one of the most challenging aspect in underwater environments and, when it is possible, sharing modest information is a key rule. It might be possible to use a dedicated optical underwater communication system could reach a bandwidth of 10 Mbit/s, Hanson and Radic (2008), over short ranges, but the bandwidth available with off the shelf acoustic modems is in the 1-5 kBaud range. With readily available equipment, bandwidth is hence a serious concern.

To reduce even more the information to be shared to have a satisfactory fault-tolerant system, a decentralised controller and a decentralised virtual actuator are considered. It is assumed that robots store information related to force coefficients in (11), thruster configuration, position and orientation of each team member during the cooperation.

The inputs to the thrusters $\boldsymbol{u^t_{f_{Si}}}$ are defined as:

$$\boldsymbol{u^t_{f_{Si}}} = \boldsymbol{u_{f_{Si}}} + \boldsymbol{N_{f_{Si}} u_c} \tag{30}$$

$\boldsymbol{u_{f_{Si}}}$ represents the inputs of robot Si as defined in (15). Note that just local thrusters are considered for its computation in order to satisfy the fault hiding goal. If there are no local faults, $\boldsymbol{u_{f_{Si}}} = \boldsymbol{u_{c_{Si}}}$.

$\boldsymbol{N_{f_{Si}}} \in \Re^{h_{Si} \times h}$ comprises the additional contribution of robot Si to the faulty robots. Each robot is able to construct its own copy of matrix $\boldsymbol{N}$ as in (22) by considering the thrusters of the whole system. Finally, $\boldsymbol{N_{f_{Si}}}$ is obtained by selecting in $\boldsymbol{N}$ the contribution to be provided by Si in order to accommodate the fault.

$\boldsymbol{u_c}$ refers to the nominal inputs of the whole system.

The decentralised virtual actuator decreases communication conspicuously, but the computational load on individual robots is increased.

Algorithm *Decentralised reconfiguration*

Given:

o robots named Si, $i = 1, \dots, o$,

h_{Si} thrusters corresponding to robot Si, $h = \sum_{i=1}^{o} h_{Si}$,

f_{Si} actuator faults in robot Si, $f_{Si} \leq h_{Si}$, $f = \sum_{i=1}^{o} f_{Si}$,

matrices $\boldsymbol{B_{Si}}$ and $\boldsymbol{B_{f_{Si}}}$ defined in (12) and subsection 5.1,

a local controller (see section 4) and a local virtual actuator.

Problem:

Find $\boldsymbol{u^t_{f_{Si}}}$ defined in (25) such that the fault hiding goal (see subsection 5.1) and the performance indicator ($\|\boldsymbol{\eta} - \boldsymbol{\eta_f}\| \simeq 0$) is satisfied.

Solution:

- The faulty robot Si activates its local virtual actuator.
- Si warns the other robots by communicating its actual status (i.e., fault identification).
- The robot Sj ($i \neq j$) receives the message.
- Sj computes $\boldsymbol{N_{f_{Sj}}}$ and finds $\boldsymbol{u^t_{f_{Sj}}}$ as in (30).

Example scenario: Fault on $S1$'s vertical thruster

Inputs to be sent to $S1$'s thrusters:

$$\boldsymbol{u^t_{f_{S1}}} = \boldsymbol{N_{S1} u_{c_{S1}}}$$

$$\boldsymbol{N_{S1}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Inputs to be sent to $S2$'s thrusters:

$$\boldsymbol{u^t_{f_{S2}}} = \boldsymbol{u_{c_{S2}}} + \boldsymbol{N_{f_{S2}} u_c}$$

$$\boldsymbol{N_{f_{S2}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{63} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Inputs to be sent to $S3$'s thrusters:

$$\boldsymbol{u^t_{f_{S3}}} = \boldsymbol{u_{c_{S3}}} + \boldsymbol{N_{f_{S3}} u_c}$$

$$\boldsymbol{N_{f_{S3}}} = \begin{bmatrix} 0 & 0 & n_{73} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{83} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note that n_{73} and n_{83} have opposite sign to avoid the whole system to rotate about the y-axis (pitch).

6. RESULTS

A static reconfiguration of the controller is possible for single faults (except for the horizontal thruster of *S3*), because the condition (24) is satisfied due to redundant thrusters with a similar effect as the faulty one. For the same reason, the static reconfiguration block defined in (25, 26) can be used in case of two faults (e.g., a fault in an horizontal thruster of *S1* and another one in a vertical thruster of *S3*) or even three faults (e.g., a whole robot breaks down).

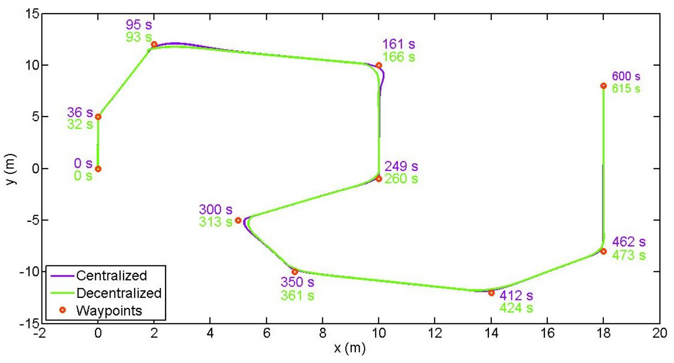

Fig. 5. Fault handling capability of the centralised (in violet) versus decentralised (in green) control reconfiguration in case of a failure of robot $S1$. The robot follows a trajectory passing through ten waypoints (red circles). The annotations indicate the time elapsed to reach a waypoint.

If a redundant thruster is not available for the reconfiguration, a dynamical block has to be used. This is the case for more than one fault in vertical thrusters. A fault in the horizontal thruster of robot *S3* causes loss of controllability since this is the only thruster in the system to provide heave (see Table 1).

In our example, the virtual actuator is able to ensure the same performance in case of single fault or almost the same performance up to two faults for vertical thrusters and two faults for horizontal thrusters, excluding the horizontal thruster of *S3*(e.g., two horizontal faults in *S1* and one horizontal fault in *S2* cannot be handled).

If simultaneous vertical faults occurs in *S1* and *S2*, the controllability requirement is not satisfied and the virtual actuator is not a solution to the problem.

Both the centralised and decentralised reconfigurations are able to accommodate the same combinations of faults, within the capabilities of thrust available in the two conditions. Since knowledge of the global system (i.e., thruster configuration and force coefficients) is exploited in both cases, performance requirements (i.e., $\|\boldsymbol{\eta} - \boldsymbol{\eta_f}\| \simeq 0$) can be made the same independent of the implementation.

Fig. 5 compares the performance of the centralised with the decentralised solutions when robot $S1$ fails. Differences in the path are due to the nominal controllers (see section 4).

Fig. 6 shows a comparison between the nominal force X_n (torque N_n) and the faulty force X_f (torque N_f) (see Table 1) in case of the total failure of robot $S1$ provided by the centralised system. Fig. 7 compares the nominal force X_n (torque N_n) and the the faulty force X_f (faulty torque N_f) (see Table 1) in case of the total failure of robot $S1$ provided by the decentralised system. Each change in force and torque corresponds to a new waypoint that has to be reached, so the robot adjusts according to it. Note that responses are time shifted due to the lower magnitude of total thrust available for propelling underwater robots.

Fig. 8 shows the sensitivity to output disturbances in surge speed u, depth z and heading angle ψ. The controllers are designed to suppress disturbances up to $0.05 rad/s$

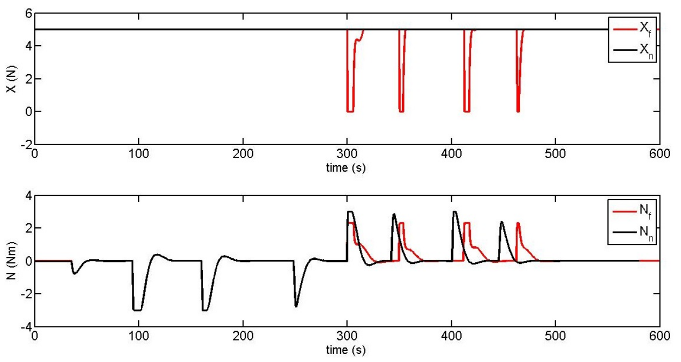

Fig. 6. The nominal force X_n and torque N_n are shown in black. The force X_f and torque N_f for a total failure of robot $S1$ are shown in red. Case: centralised control.

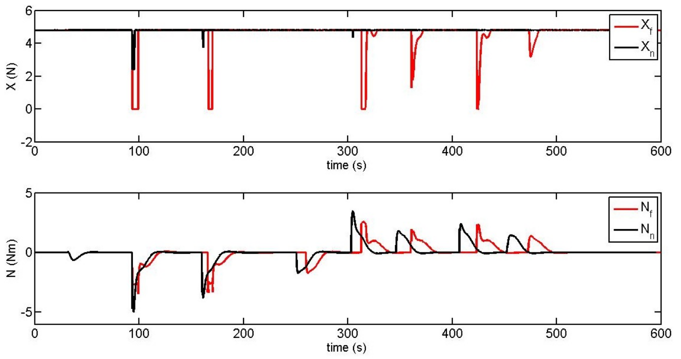

Fig. 7. The nominal force X_n and torque N_n are shown in black. The faulty force X_f and torque N_f for a total failure of robot $S1$ are shown in red. Case: decentralised control.

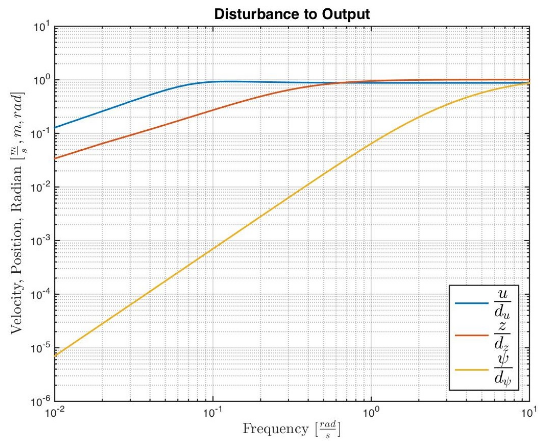

Fig. 8. The output disturbance sensitivity for surge speed, z-position and heading. Disturbance rejection bandwith is chosen to be $0.07 rad/s$ in u, 5 rad/s in yaw.

in speed, $0.3 rad/s$ in heave and $3 rad/s$ in yaw with no amplification of disturbances at any frequency.

7. CONCLUSIONS

This paper focused on actuator faults as these are quite frequent in underwater missions, often due to seaweed, marine life or wires that block a thruster. Centralised and decentralised reconfiguration methods were compared in a scenario where three robots perform the task of carrying a drill. Using a conventional line of sight control for path following, the solution comprised path control and control allocator. Faults were handled by means of a virtual actuator. The virtual actuator was shown to provide approximately unchanged performance as the nominal case for the majority of thruster fault cases.

In the centralised approach, one of the robots acted as a leader and coordinated the actions of the team. While allowing for global optimization and completeness, the central unit could fail or communication channel difficulties could prevent timely sharing of information at the sampling frequency.

The distributed version was based on every robot being autonomous and taking its own decisions. An algorithm was presented where only fault-status was communicated between robots.

As a future work, sensor faults and parameters uncertainty will be considered and a proof of equation (27) will be provided. Finally, a distributed algorithm which does not rely on global structural information will be proposed. It should guarantee fault-handling capability even in case of flexible configurations among robots.

ACKNOWLEDGEMENTS

We gratefully acknowledge support from DTU and from the NTNU Centre for Autonomous Marine Operations and Systems (AMOS), to which the Research Council of Norway is the main sponsor (grant 223254).

REFERENCES

Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M. (2006). *Diagnosis and Fault-Tolerant Control*. Springer, 2^{nd} edition.

Breivik, M. and Fossen, T.I. (2011). *Guidance Laws for Autonomous Underwater Vehicles*, chapter 4 in: Underwater Vehicles (ed: A. V. Inzartsev), pp 51–76. Intech.

Corradini, M.L., Monteri, A., and Orlando, G. (2011). An actuator failure tolerant control scheme for an underwater remotely operated vehicle. *IEEE Transactions on Control Systems Technology*, 19(5), 1036 – 1046.

Fossen, T.I. (1994). *Guidance and Control of Ocean Vehicles*. John Wiley & Sons.

Fossen, T.I. (2011). *Handbook of Marine Craft Hydrodynmics and Motion Control*. John Wiley & Sons.

Hanson, F. and Radic, S. (2008). High bandwidth underwater optical communication. *Applied Optics*, 47(2), 277–283.

Indiveri, G. and Parlangeli, G. (2006). On thruster allocation, fault detection and accommodation issues for underwater robotic vehicles. In *Proc. IEEE ISCCSP 2006 Second Int. Symp. on Communications, Control, and Signal Processing*.

Johansen, T.A. and Fossen, T.I. (2013). Control Allocation - A Survey. *Automatica*, 49, 1087–1103.

Omerdic, E. and Roberts, G. (2004). Thruster fault diagnosis and accomodation for open-frame underwater vehicles. *Control Engineering Practice*, 12, 1575 – 1598.

Podder, T., Antonelli, G., and Sarkar, N. (2000). Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: simulations and experiments.

In *Proc. ICRA'2000, IEEE Int. Conf. on Robotics and Automation*, volume 2.

Sarkar, N., Podder, T., and Antonelli, G. (2002). Fault-accommodating thruster force allocation of an AUV considering thruster redundancy and saturation. *IEEE Transactions on Robotics and Automation*, 18, 223–233.

Steffen, T. (2005). *Control Reconfiguration of Dynamical Systems*. Springer.

Yang, K., Yuh, J., and Choi, S. (1998). Experimental study of fault-tolerant system design for underwater robots. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, 1015–1056.

Yang, K., Yuh, J., and Choi, S. (1999). Fault-tolerant system design of an autonomous underwater vehicle ODIN: An experimental study. *Int. Journal of Systems Science*, 30(9), 1011–1019.

Yang, Z., Blanke, M., and Verhagen, M. (2007). Robust control mixer method for reconfigurable control design using model matching. *IET Control Theory & Applications*, 1(1), 349–357.

Yang, Z. and Blanke, M. (2000). The robust control mixer module method for control reconfiguration. In *Proc. American Control Conference*, 3407–3411. IEEE Explore.

Zhu, D., Liu, Q., and Yang, Y. (2008). An Active Fault-Tolerant Control Method Of Unmanned Underwater Vehicles with Continuous and Uncertain Faults. *International Journal of Advanced Robotic Systems*, 5, 411–418.

Appendix A. MATRICES AND PARAMETERS USED TO SIMULATE THE EXAMPLE SCENARIO

The inertia matrix $\boldsymbol{M}$ and damping matrix $\boldsymbol{D}$ are not full because the number of parameters can be severely reduced by exploiting body symmetry conditions. In particular, the configuration in which the robots assemble is symmetrical on the xy plane(bottom/top symmetry) and on the xz plane (port/starboard symmetry).

A.1 Inertia Matrix:

$$\boldsymbol{M_{RB}} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & mx_g \\ 0 & 0 & m & 0 & -mx_g & 0 \\ 0 & 0 & 0 & I_x & 0 & 0 \\ 0 & 0 & -mx_g & 0 & I_y & 0 \\ 0 & mx_g & 0 & 0 & 0 & I_z \end{bmatrix} \tag{A.1}$$

$$\boldsymbol{M_A} = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & Y_{\dot{r}} \\ 0 & 0 & Z_{\dot{w}} & 0 & Z_{\dot{q}} & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & M_{\dot{w}} & 0 & M_{\dot{q}} & 0 \\ 0 & N_{\dot{v}} & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{A.2}$$

where m is the total mass of the robot, I_x, I_y and I_z are the moments of inertia around x_b, y_b and z_b (body fixed coordinate system), respectively, and x_g is the position of the centre of gravity around x_b.

Table A.1. $\boldsymbol{M_{RB}}$ parameters

m	I_x	I_y	I_z	x_g
47.5	1.7	3.5	2.3	0

Table A.2. $\boldsymbol{M_A}$ parameters

$X_{\dot{u}}$	$Y_{\dot{v}}$	$Y_{\dot{r}}$	$Z_{\dot{w}}$	$Z_{\dot{q}}$	$K_{\dot{p}}$	$M_{\dot{w}}$	$M_{\dot{q}}$	$N_{\dot{v}}$	$N_{\dot{r}}$
92.9	78.9	0.8	236.1	1.2	1.2	1.2	2.5	0.8	1.6

A.2 Damping Matrix:

$$\boldsymbol{D_V}(\boldsymbol{\nu}) = -\begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & 0 & Y_r \\ 0 & 0 & Z_w & 0 & Z_q & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & M_w & 0 & M_q & 0 \\ 0 & N_v & 0 & 0 & 0 & N_r \end{bmatrix} \tag{A.3}$$

$$\boldsymbol{D_N}(\boldsymbol{\nu}) = -\begin{bmatrix} X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v| & 0 & 0 & 0 & Y_{|r|r}r \\ 0 & 0 & Z_{|w|w}|w| & 0 & Z_{|q|q}|q| & 0 \\ 0 & 0 & 0 & K_{|p|p}|p| & 0 & 0 \\ 0 & 0 & M_{|w|w}|w| & 0 & M_{|q|q}|q| & 0 \\ 0 & N_{|v|v}|v| & 0 & 0 & 0 & N_{|r|r}|r| \end{bmatrix} \tag{A.4}$$

Table A.3. $\boldsymbol{D_V}(\boldsymbol{\nu})$ parameters

X_u	Y_v	Y_r	Z_w	Z_q	K_p	M_w	M_q	N_v	N_r
-11.5	-13.4	-1.7	-7.45	-0.7	-2	-0.3	-2	-0.3	-2

Table A.4. $\boldsymbol{D_{N(\nu)}}$ parameters

$X_{\|u\|u}\|u\|$	$Y_{\|v\|v}\|v\|$	$Y_{\|r\|r}r$	$Z_{\|w\|w}\|w\|$	$Z_{\|q\|q}\|q\|$
-29.04	-44.73	-6.6	-72.36	-7.1

$K_{\|p\|p}\|p\|$	$M_{\|w\|w}\|w\|$	$M_{\|q\|q}\|q\|$	$N_{\|v\|v}\|v\|$	$N_{\|r\|r}\|r\|$
-14.8	-0.3	-16.9	-1.1	-18.9

A.3 Force coefficients

Table A.5. Force coefficients for Graupner 2310.60 propellers

Horizontal k (*S1,S2*)	Horizontal k (*S3*)	Vertical k
7.7	26.1	27.4

A.4 Thrust configuration matrix

$$\boldsymbol{T} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & -0.25 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & -0.02 & 0 & 0 & -0.02 & 0.33 & 0.23 & 0 \\ 0.3 & 0.22 & 0 & -0.2 & -0.3 & 0 & 0 & 0 & 0.28 \end{bmatrix} \tag{A.5}$$

The first three columns refer to robot *S1*. The following three to *S2* and the final three to *S3*. If robots are in a non-cluster configuration, each one has its $\boldsymbol{T}$ reduced to a $3{\times}3$ matrix representing its own thruster configuration.