Experimental validation of propulsion fault detection system using moving horizon estimation in quad-maran unmanned vessel

Kota Mukai, Naoyuki Hara and Keiji Konishi

Department of Electrical and Information Systems, Osaka Prefecture University, Sakai, Osaka, Japan

ABSTRACT
A quad-marun unmanned vessel is a new type of unmanned surface vessel for the automatic collection of various environmental data in aquaculture fields. We propose a fault detection and control scheme for a possible thrust decrease failure of quad-maran vessels. The fault detection and control are based on moving horizon estimation and model predictive control. The proposed estimation and control scheme is experimentally validated.

1. Introduction
Fault detection and fault-tolerant control are important aspects of practical control systems. Faults in systems cause a loss of control performance and/or lead to catastrophic damage to the components of control systems. For example, a sudden loss or damage of propellers in mult rotor copers may result in crashing into the ground. Faults can occur during operation, and fault detection needs to be performed in real time based on the measurements of a plant. Various fault detection and fault-tolerant control methods have been proposed and applied to many different systems in different industry sectors [1]. One approach to fault detection is to use moving horizon estimation (MHE) [2–4], and several MHE-based fault detection/control methods have been reported [5–9]. The MHE approach uses an estimation horizon that moves along with the time evolution, and the estimation is performed by solving an optimization problem.

The authors have been engaged in the control of a new type of unmanned surface vessel, a quad-marun unmanned vessel [10–13]. This vessel is mainly developed for environmental monitoring [14] in aquaculture fields, and an observation project has been conducted at oyster farms in Nanao Bay, Ishikawa Pref., Japan (Figure 1); initial observation results are analysed and reported in [15]. There have been problems such as shellfish dying in aquaculture farms recently because of changes in the water environment [16]. The vessel is expected to make it possible to obtain frequent and high-resolution observation data, such as temperature, oxygen level, and salinity, in aquaculture fields. Such data can play important roles in providing high-level services for aquaculture; for example, predicting the occurrence of hypoxia water mass. Quad-marun unmanned vessels have a feature that is suitable for operation in aquaculture fields; namely, four independently rotating hulls with propellers. This capability allows for good sailing and dynamic positioning ability in aquaculture fields where many obstacles such as buoys and rafts for farming are present. While fault detection and fault-tolerant control are essential features required for the control system of such vessels, they have yet to be implemented. During a sea trial, the propeller of the hull (Figure 2(a)) became entangled in a floating object (Figure 2(b)). The failed propeller could no longer generate thrust, and this failure could occur without significantly affecting rotation speed and motor current. We were thus unable to detect this type of fault simply by observing propeller system data; the motion of the vessel also needs to be taken into account for fault detection. This fault results in control performance deterioration or complete stalling depending on its severity.

In this paper, we propose a fault detection system for quad-marun vessels to identify faults in the propellers. We use a simple model of the vessel during dynamic positioning and formulate the fault detection problem based on MHE. We then develop a controller using model predictive control (MPC) where the vessel model and tuning parameters are adjusted according to the current estimated fault condition. The fault detection and control are established by solving optimization problems, a mixed-integer quadratic program (MIQP), and a quadratic program (QP), in real time. The MHE-based fault detection with the MPC-based controller is experimentally tested and validated using quad-marun vessels.
The contributions of the paper are twofold:

(1) Most existing studies on real-time fault detection and fault-tolerant control [1, 5–9] use simulations for evaluation, but there is always a gap between a plant model and an actual plant (such as model mismatch and disturbances). Experimental validation is necessary to assess whether or not the methods actually work in a real environment. We propose a fault detection system and carry out an experiment, serving as an experimental case study in this field, yielding information on implementation and insight from the experimental results.

(2) The proposed fault detection scheme adds an essential feature to quad-maran vessels. The feature serves to increase the reliability and safety of the vessel operation in aquaculture fields. This ultimately contributes to the IoT/ICT-assisted advanced aquaculture and fishery industry [17].

The preliminary version of this paper was presented at a conference [18]. Details of the model identification, MPC-based controller, and additional experimental results have been added to this paper.

The paper is structured as follows. In Section 2, we briefly explain the quad-maran vessel and introduce a model of the vessel. In Section 3, we represent possible fault conditions based on the vessel model and propose an MHE-based fault detection system with an MPC-based controller. In Section 4, we show the experimental results of the proposed method.

2. Quad-maran unmanned vessel

2.1. Overview

A quad-maran unmanned vessel is composed of four hulls and a central structure that joins them (Figure 3). Each hull can change its orientation individually and has a propeller at its rear. The rotation direction and speed of the propellers can be changed individually. These features contribute to the dynamic positioning capability and good maneuverability during sailing. For high-speed sailing, all the hulls are parallel, as shown in Figure 3(a). When performing dynamic positioning, the hulls take an X form, as shown in Figure 3(b). In the configuration of Figure 3(b), the vessel can move in all directions, making it suitable for positioning at an observation point.

The position of the vessel is obtained from a GNSS (Global Navigation Satellite System) device, and the azimuth can be obtained from a magnetometer. Anemometers can also be installed if necessary. Details of the vessel equipment are found in [19].

2.2. Model in dynamic positioning

2.2.1. Linear state-space model

A model is required for MHE-based fault detection. As a simple model, we use the following linear state-space model for the X configuration in Figure 3(b).

\[ \dot{x} = A_c x + B_c u, \]
where $x \in \mathbb{R}^4$ is a state vector, $u \in \mathbb{R}^2$ is a control input vector (propeller speed reference), and $y \in \mathbb{R}^2$ is an observation vector (vessel’s position). The vessel’s coordinate system is shown in Figure 4. $p_x$ indicates the vessel position parallel to the longitudinal axis of hulls No. 1 and No. 3. The $p_y$ axis is orthogonal to the $p_x$ axis. The two propellers of the two aligned hulls are paired, and the same control input is given to them. $u^1$ is the voltage reference for the propeller rotation speed of hulls No. 1 and No. 3. $u^2$ is the same, but for hulls No. 2 and No. 4. When $u^1 > 0$, the propellers of hulls No. 1 and No. 3 are commanded to rotate in the direction of increasing $p_x$. The parameters $a$ and $d$ in $A_c$ are proportional coefficients of hydrodynamic drag that increases in proportion to the speeds of respective axes. The parameters $e$ and $h$ in $B_c$ are proportional coefficients of thrust that increases in proportion to the speeds of respective propellers.

2.2.2. System identification experiment

The parameter values of $a$, $d$, $e$, and $h$ in $A_c$ and $B_c$ are required. We perform a system identification experiment and identify the parameters $a, d, e, h, w_1$, and $w_2$ in the model,

\[
\begin{bmatrix}
\dot{p}_x \\
\dot{p}_y \\
\dot{p}_x \\
\dot{p}_y
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & a & 0 \\
0 & 0 & 0 & d
\end{bmatrix}
\begin{bmatrix}
p_x \\
p_y \\
p_x \\
p_y
\end{bmatrix}
\]
This model comes from (1) with the wind-effect terms. $w_x$ [m/s] is the wind speed component aligned with the $p_x$ axis. $w_y$ is the same but aligned with the $p_y$ axis. $w_x$ and $w_y$ are obtained by decomposing the wind speed vector to each axis, $p_x$ and $p_y$. In the model (3), we consider the motion of each axis to be independent, and thus we identified each axis $p_x$ and $p_y$ independently. In the identification, we use a discrete-valued input signal, taking $-5.0$ V and $5.0$ V alternately every $8.0$ s.

The identification data on the $p_x$ axis are shown in Figure 5. Figure 5(b) shows that the wind speed $w_x$, and it is mostly positive, giving a positive wind thrust. With these data, we estimated the parameters $a$, $e$, $w_1$ by using the MATLAB `ssest` function (see Appendix A for an excerpt of MATLAB script used in the identification). The parameters were identified by using the data from 10.0 to 30.0 s. The estimated parameters are as follows:

$$a = -0.1289, \quad e = 0.01509, \quad w_1 = 0.002755.$$  

For validation, other data not used in the identification were compared with the behaviour of the model with the estimated parameters $a$, $e$, $w_1$. The comparison results are shown in Figure 6, and the model output resembles the measured data for the first 20.0 s.

Even after 20.0 s, the model can reproduce similar behaviour. The same experiment was performed on the $p_y$ axis, and the following parameters were obtained:

$$d = -0.1388, \quad h = 0.01719, \quad w_2 = 0.0005284.$$  

We performed the similar validation test for $p_y$ axis and obtained the result shown in Fig. 7, which indicates that the model output resembles the measured data.

Although the same identification method is used for both $p_x$ and $p_y$ axis models, only the validation result of $p_x$ axis shows the off-set deviation (Fig. 6), occurring at around 25 s. We suspect the gust happened somewhere between 20–25 s as the cause of the error. The anemometer used in the system identification experiment has some smoothing property, and it is not suited to measure the sudden change in the wind like gust. When the gust is not captured well in the anemometer, an off-set like error occurs between the measured data and model output; in Fig. 6, an off-set occurs at around $25$ s and the amount of the off-set is kept at almost the same level after that.

The identification input in Fig. 5(a) is a single-frequency pulse, but not that it contains only a single frequency component. The MALAB function `pexcit` with default tolerance parameters is used to compute the degrees of persistence of excitation of the identification inputs, and they are $16$ ($u^1$ and $u^2$) and $20$ ($w_x$ and $w_y$). Thus it is not fully inappropriate for identifying the second order system with these identification inputs.

3. Fault detection system

3.1. Model discretization

We discretize the model (1), (2), giving

$$\begin{align*}
x_{t+1} &= Ax_t + Bu_t + w_t, \\
y_t &= Cx_t + v_t,
\end{align*}$$  

where $w_t \in \mathbb{R}^4$ and $v_t \in \mathbb{R}^2$ are newly introduced, and they represent the process and observation noises, respectively. Note that the wind disturbance terms in (3) are not explicitly included in (4), but instead, the model (4) has the general process noise term. Our fault detection method, described in the following sections, uses the model (4).

3.2. Representation of fault situations

When a thrust decrease occurs during the dynamic positioning configuration shown in Figure 3(b), this fault can be regarded as a change in the input matrix
partially fault: One propeller pair has a significant thrust decrease.

(2) Near-total fault: Both propellers in a pair have a significant thrust decrease.

A model incorporating these fault conditions into (4) is given by

\[
\begin{align*}
x_{t+1} &= Ax_t + B_0 x_0^0 + \sum_{j=1}^{2} B_j x_j^i + w_t, \\
y_t &= C x_t + v_t, \\
x_j^i &= \begin{bmatrix} \delta_j^{i1} u_t^1 \\ \delta_j^{i2} u_t^2 \end{bmatrix}, \\
\sum_{j=0}^{2} \delta_j^i &= 1,
\end{align*}
\]

where \(B_j \in \mathbb{R}^{4 \times 2}\) (\(B_0 = B\)) is the input matrix corresponding to the fault condition \(j \in \{0, 1, 2\}\), and \(\delta_j^i\) is the 0–1 integer variable that takes 1 when the assumed fault \(j\) at time \(t\) occurs on the control input \(i\) and 0 when it does not. The equality constraint (8) means that only one fault condition can arise for each control input. For example, when \(\delta_i^{10} = \delta_i^{20} = 1\) and \(\delta_i^j = 0\) \((i = 1, 2, j = 1, 2)\), the model (5) is equivalent to (4), which indicates no faults. When \(\delta_i^{12} = \delta_i^{21} = 1\) and the other 0–1 variables are 0, the fault conditions \(j = 2\) and \(j = 1\) occur on the control inputs \(u^1_t\) and \(u^2_t\), respectively.

### 3.3. Fault detection with MHE

We use MHE (Moving Horizon Estimation) to identify the current fault condition in real time based on the data set of the past outputs and inputs over a certain time horizon [5] (see Figure 8). At time step \(k\), the state variables \(x_{k-N+1}, \ldots, x_k\) and \(\delta_{k-N}^j, \ldots, \delta_{k-1}^j\) are estimated from \(u\) and \(y\) between time steps \(k-1\) and \(k-N\). We estimate the states and 0–1 fault variables by solving the following optimization problem.

\[
\begin{align*}
\min_{\hat{x}, \hat{w}, \hat{v}, \hat{\delta}} & \quad J(\hat{x}, \hat{w}, \hat{v}, \hat{\delta}) \\
\text{s.t.} & \quad \hat{x}_{t+1} = A \hat{x}_t + B_0 \hat{x}_0^0 + \sum_{j=1}^{2} B_j \hat{x}_j^i + \hat{w}_t,
\end{align*}
\]

where \(\hat{x}_{k-N} = \hat{x}_{k-N+1} + \cdots + \hat{x}_k\) for \(k-N\) time steps. The cost function \(J(\hat{x}, \hat{w}, \hat{v}, \hat{\delta})\) is defined as follows:

\[
J = \sum_{t=k-N}^{k-1} \|\hat{w}_t\|^2_{Q^{-1}} + \sum_{t=k-N}^{k-1} \|\hat{v}_t\|^2_{R^{-1}} + \|\hat{x}_{k-N} - \hat{x}_{k-N}\|^2_{\pi^{-1}} + \sum_{t=k-N}^{k-1} \left( \sum_{j=1}^{2} \|\hat{\delta}_j^i\|^2_{\delta w_j} \right),
\]

where \(\hat{\delta}_j^i\) is defined as

\[
\hat{\delta}_j^i = \begin{bmatrix} \hat{\delta}_j^{i1} \\ \hat{\delta}_j^{i2} \end{bmatrix}.
\]

The matrix \(Q^{-1} \in \mathbb{R}^{4 \times 4}\) is a weight for process noise, \(R^{-1} \in \mathbb{R}^{2 \times 2}\) is a weight for observation noise, \(\pi^{-1} \in \mathbb{R}^{4 \times 4}\) is a weight for the estimation error of the state at time step \(k-N\), and \(\delta w_j \in \mathbb{R}^{2 \times 2}\) is a weight that adjusts the detection sensitivity of the fault condition \(j\). The
form of cost function $J$ is typical in MHE problems. The state estimates do not appear explicitly in (10), but they are evaluated because the state-space equation is included in the constraints of the optimization problem.

At each time step, the optimization problem (9) is solved, and the fault condition is identified. The optimization problem requires the state variable $\tilde{x}_{k-N}$ at the starting point of the estimation horizon. For the first estimation, $\tilde{x}_{k-N}$ is initialized with an appropriate value, e.g. the zero vector. After the second estimation, $\tilde{x}_{k-N}$ is initialized with a state estimate obtained at the previous estimation, i.e., being $\delta_t$.

In practical environment. There could be a system-over the entire estimation horizon, which is unrealistic fault conditions are not detected at the same time, and $T_h$ should be less than 1; here 1 is not allowable because 1 indicates that there are no estimation errors in $\delta_t$ over the entire estimation horizon, which is unrealistic in practical environment. There could be a systematic way for choosing appropriate $T_h$ in advance if a certain property on the estimation performance under disturbances and model mismatch is available. However, currently no such method is established, and $T_h$ needs to be chosen by trial and error.

The discrete-time model (5)–(8) incorporating the fault condition, $F_i$, is set to 1 if a fault condition $j$ occurs at the control input $u_t$ at time $t$, and 0 if it does not. A fault indicator variable, $F_i$, is set to $j$ if a fault condition $j$ occurs at the control input $u_t$ at time $t$ as follows:

$$F_i^j = \frac{\sum_{t=k-N}^{k-1} \delta_t^j}{N} \geq T_h. \quad (12)$$

For example, if $F_i^j$ is zero, it indicates that the control input $u_t$ is normal at time $t$. If $F_i^j$ is 1, it indicates that the control input $u_t$ is in a fault condition 1 at time $t$.

In practice, the threshold $T_h$ is considered as a tuning parameter for adjusting the sensitivity of fault detection, and should be chosen from an interval (0.5, 1). $T_h$ should be larger than 0.5 in order to ensure that two fault conditions are not detected at the same time, and $T_h$ should be less than 1; here 1 is not allowable because 1 indicates that there are no estimation errors in $\delta_t$ over the entire estimation horizon, which is unrealistic in practical environment. There could be a systematic way for choosing appropriate $T_h$ in advance if a certain property on the estimation performance under disturbances and model mismatch is available. However, currently no such method is established, and $T_h$ needs to be chosen by trial and error.

The discrete-time model (5)–(8) incorporating the fault conditions is in fact described by the form of switched linear systems

$$x_{t+1} = Ax_t + B(\sigma_t)u_t + w_t$$

$$y_t = Cx_t + v_t,$$

where $\sigma_t \in \{1, 2, \ldots, 9\}$ denotes the discrete-valued variable representing the fault condition. This is because we have three conditions (normal, partial fault, and near-total fault) for each of two input channels, which totals nine fault patterns. Since we assume that a fault is characterized by a change in the $B$ matrix, nine fault patterns can be represented by $B(1), B(2), \ldots, B(9)$. Thus the MHE problem (9) can be alternatively posed as an estimation problem for the switched linear system where we seek to find the continuous-valued state $x_t$ and discrete-valued switching variable $\sigma_t$ over the estimation horizon based on the measurements (past outputs and inputs). Theoretical analysis on the moving horizon estimation of switched linear systems has been conducted in [20]; the main result (Theorem 2 in [20]) is that the estimation error of the continuous-valued state at $t = N + \alpha (\alpha > 0)$ is bounded from above under some assumptions. Note that the formula was presented for the switched linear system without input term, which is not directly applicable to our case. Also the estimation performance on the current state $x_t$ and discrete-valued variable have not been made clear in [20]. In our paper, the estimation performance of the proposed scheme is not fully supported by the theoretical results; however, we have verified the estimation performance of the current state and 0-1 variable $\delta_t^j$ by numerical simulations (Appendix B).

### 3.4. Dynamic positioning system with fault detection

We use model predictive control (MPC) for a controller that generates the control input $u_t$ such that the vessel is regulated to a reference state $x^*$. In MPC, control inputs over a future time horizon are computed by solving an optimization problem. We use the first component of the computed inputs over the horizon as an actual input.

The model used in MPC is adjusted to the current identified fault condition $F_i^j$. We split the matrices $B_0(= B), B_1, B_2$ in (5) as follows:

$$B_0 = [b_0^1, b_0^2], \quad B_1 = [b_1^1, b_1^2], \quad B_2 = [b_2^1, b_2^2]. \quad (14)$$

We define

$$\bar{B} = [b_0^{F_1}, b_1^{F_1}, b_2^{F_1}], \quad (15)$$

as the input coefficient matrix in the dynamics equation in MPC. This is updated at each sampling time with the newly estimated fault condition $F_i^j$. We solve the following optimization problem at each sampling time:

$$\min_{x, u} J(x, u) \quad (16)$$
\[ x_{t+1} = Ax_t + Bu_t \]
\[ -5 \leq u_i^t \leq 5 \quad (i = 1, 2) \]  
\[ x_0 = \bar{x} \]
\[ t = 0, 1, \ldots, N_c - 1 \]

where \( N_c - 1 \) is the length of the prediction horizon and the cost function \( J(x, u) \) is defined as follows:

\[ J = \sum_{t=0}^{N_c-1} \| x_t - x_f^t \|^2_{Q_{\text{MPC}}} + \sum_{t=0}^{N_c-1} \| u_t \|^2_{R_{\text{MPC}}}. \]

The estimated current state by MHE (denoted by \( \bar{x} \) in the above optimization problem) is used as an initial condition. This optimization problem is a quadratic programming (QP) problem. The inequality constraints in (17) are the upper and lower limits of the voltage reference for the propeller rotation speed. The matrix \( Q_{\text{MPC}} \in \mathbb{R}^{4 \times 4} \) in (18) is a weight for the error from the reference state \( x_f^t \), and \( R_{\text{MPC}} \in \mathbb{R}^{2 \times 2} \) is a weight for the control inputs. The weight matrices are adjusted to the identified fault condition.

Figure 9 shows the overall structure of the MHE-based fault detection system incorporated into the MPC-based dynamic positioning system.

4. Fault detection experiments

4.1. Experimental system and configuration

The experiment was conducted in a pool at Osaka Prefecture University (Figure 10). The size of the pool is \( 50 \text{ m} \times 20 \text{ m} \). The average wind speed on the date of the experiment was 2.6 m/s. The dynamic positioning system with fault detection in Figure 9 was implemented on a laptop PC (CPU: Intel(R) Core(TM) i7-8550U 1.80GHz, RAM: 16.0GB) and Raspberry Pi 3B on the central structure of the vessel, as shown in Figure 11. The PC performs the numerical optimizations for MHE (MIQP) and MPC (QP). Raspberry Pi collects observation data and applies the control input to the propeller system. Raspberry Pi sends the current position \([p_x, p_y]\) to the PC. The PC receives this value and calculates the control inputs \([u_1, u_2]\) for the next time step. Raspberry Pi receives the control inputs and controls the propellers. The PC and Raspberry Pi are connected by Wi-Fi and communicate by TCP socket connection. The programs on the PC and Raspberry Pi are written in Python. The numerical optimizations for MHE and MPC are performed by calling IBM ILOG CPLEX Optimization Studio library functions from Python. The estimation horizon \( N \) was set to 14 steps. Because the sample time is 0.5 s, 14 steps are equivalent to 7.0 s. The weights \( Q^{-1}, R^{-1}, \pi^{-1}, \delta w_1, \) and \( \delta w_2 \) used in MHE were set as follows:

\[ Q^{-1} = I_4, \quad R^{-1} = I_2, \quad \pi^{-1} = I_4, \]
\[ \delta w_1 = O_2, \quad \delta w_2 = 10^{-3} \cdot I_2. \]

The threshold \( T_h \) in (12) was set to 0.75. This is chosen through the process of trial and error and observing the fact that 0.75 is a middle value over (0.5, 1). The length \( N_c \) of the prediction horizon for MPC was set to 20 steps, which is equivalent to 10.0 s. In the no-fault condition, the following weights were used:

\[ Q_{\text{MPC}} = 100 \cdot I_4, \quad R_{\text{MPC}} = I_2. \]
to 0.01 for partial fault and 0.001 for near-total fault. This choice is intended to make the propeller compensate for the decreased thrust. The reference state $x^f$ for MPC was set to be a zero vector.

### 4.2. Experimental results

In this experiment, two fault situations were assumed. One situation assumed in this experiment was that the

**Figure 12.** Results of fault detection experiment (fault on the propeller of the No. 1 hull). (a) Position. (b) Control inputs and (c) Estimated fault states.

**Figure 13.** Results of fault detection experiment (fault on both the propellers of the No. 2 and No. 4 hulls). (a) Position. (b) Control inputs and (c) Estimated fault states.
propeller of the No. 1 hull, corresponding to the control input $u^1_1$, has a fault (fault condition, $j = 1$). This situation was simulated by artificially stopping the propeller of the No. 1 hull regardless of the value of the control input $u^1_2$. The experimental results are shown in Figure 12. The fault starts at $t = 10.0 \text{s}$ and continues. Figure 12(a) shows that after the fault occurs, the vessel moves in the direction of the reference by the thrust loss on the control input 2, making the control input $p_x$ regulated to around 0 m. Figure 12(b) shows that the control input $u^2_1$ takes 5.0 V to get the vessel back to the origin from $t = 0.0 \text{s}$ to $t = 53.0 \text{s}$. Afterward, $u^2_1$ is controlled to keep $p_x$ at 0 m. Figure 12(c) shows that $F_1^2$ intermittently takes 1. This shows that the fault condition, which corresponds to the propeller fault on hull No. 1 or No. 3, was detected intermittently. Comparing $F_1^2$ (partial fault) to $F_2^2$ (no fault) in Figure 12(c), there is a different tendency between them. The result suggests that whether the fault is actually occurring can be identified based on the relatively long interval data rather than the short interval (or a single point in time) data.

Another situation assumed in this experiment was that the propellers of hulls No. 2 and No. 4, which correspond to the control input $u^2_2$, have a fault (fault condition, $j = 2$). This situation was simulated by artificially stopping the propellers regardless of the value of the control input $u^2_2$. The experimental results are shown in Figure 13. The fault occurs at $t = 10.0 \text{s}$ and continues. Figure 13(a) shows that for a few seconds after the fault occurs, the vessel moves in the positive $p_y$ direction because of inertia. Afterward, $p_y$ stops increasing at around 17.0 s. As time progressed further, $p_y$ began to decrease and went in the opposite direction of the reference. Figure 13(b) shows that the control input $u^2_2$ always takes 5.0 V trying to get the vessel back to the origin. Note that our fault scenario here is complete thrust loss on the control input 2, making the control effort useless after the fault occurs at 10 s. Figure 13(c) shows that $F_2^2$ intermittently takes 2. This shows that the fault condition, which corresponds to the propeller fault on hulls No. 2 and No. 4, was detected intermittently. From $t = 43.0 \text{s}$ to $t = 58.0 \text{s}$, $F_2^2$ continuously takes 0, and the propellers had been erroneously estimated to be normal. This estimation error was caused by the vessel moving toward the origin because of wind disturbance. Comparing the fault estimation result $F_1^1$ for the normal propellers (#1 and #3 hulls) to $F_2^1$, the tendency of each is clearly different.

5. Conclusion

In this study, a fault detection system using MHE in a quad-maran unmanned vessel was proposed. First, the model of the vessel in the X configuration for dynamic positioning was obtained by system identification experiments. Using the obtained model, the MHE-based fault detection system with an MPC-based controller was experimentally tested, and it was confirmed that the proposed scheme works for identifying the fault. The future challenge is to distinguish behaviours caused by actual faults and disturbances such as wind and waves to increase detection reliability. Also investigating the estimation performance of the MHE from the theoretical point of view is a challenge from the control theory side.

Acknowledgements

The authors thank Y. Nihei, S. Kitamura, N. Masuda, and H. Sakamoto for their support of the experiment.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was partially supported by the Ministry of Internal Affairs and Communications SCOPE [grant number 172107104].

Notes on contributors

Kota Mukai He graduated from the School of Electrical and Electric Engineering, College of Engineering, Osaka Prefecture University, in 2019.

Naoyuki Hara He received B.Eng. and M.Eng. degrees from Tokyo Metropolitan Institute of Technology in 2003 and 2005, respectively, and the D.Eng. degree from Tokyo Metropolitan University in 2008. Since 2008, he has been with the Department of Electrical and Information Systems, Osaka Prefecture University, Osaka, Japan. His current research interests are in the field of control theory/applications and wind turbine control. He is a member of IEEJ, ISCE, IEICE, JWEA, RSJ, and IEEE.

Keiji Konishi He received the B.E., M.E. and Ph.D. degrees from Osaka Prefecture University, Osaka, Japan in 1991, 1993 and 1997, respectively. He is currently a professor with the Department of Electrical and Information Systems, Osaka Prefecture University, Osaka, Japan. His current research interests are in the areas of nonlinear science, network dynamics, chaotic systems, controlling complex systems, time-delay systems, and control systems. He is a member of APS, IEEE, IEICE, ISCE, IEEJ, and PSJ.

References

[1] Odgaard P, Aubrun C, Blanke M, editors. Special issue on industrial practice of fault diagnosis and fault tolerant control. Control engineering practice. Vols. 49–55. 2016.
[2] Rao CV, Rawlings JB, Lee JH. Constrained linear state estimation – a moving horizon approach. Automatica. 2001;37(10):1619–1628.

[3] Rawlings JB. Moving horizon estimation. Encyclopedia of systems and control. 2013. DOI:10.1007/978-1-4471-5102_9-4-1

[4] Zou L, Wang Z, Hu J, et al. Moving horizon estimation meets multi-sensor information fusion: development, opportunities and challenges. Inf Fusion. 2020;60:1–10.

[5] Bemporad A, Mignone D, Morari M. Moving horizon estimation for hybrid systems and fault detection. Proceedings of the American Control Conference; San Diego, California, USA: 1999. p. 2471–2475.

[6] Tyler ML, Asano K, Morari M. Application of moving horizon estimation based fault detection to cold tandem steel mill. Int J Control. 2000;73(5):427–438.

[7] Gatze EP, Doyle III FJ. Use of multiple models and qualitative knowledge for on-line moving horizon disturbance estimation and fault diagnosis. J Process Control. 2002;12(2):339–352.

[8] Izadi HA, Zhang Y, Gordon BW. Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation. IFAC Proc Vol. 2011;44(1):6343–6348.

[9] Wang Y, Keviczky T. Real-time fault-tolerant moving horizon air data estimation for the RECONFIG-URE benchmark. IEEE Trans Control Syst Technol. 2019;27(3):997–1011.

[10] Nihei Y, Nakada S, Hara N, et al. Quad-maran automated vessel and its application to oyster farms. Proceedings of the 61st Japan Joint Automatic Control Conference; Nagoya, Japan: 2018. p. 83–89.

[11] Zhang C, Srinivasamurthy S, Kitamura S, et al. Energy consumption of a sailing quad-maran automated vessel. Proceedings of the ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering; 2018. DOI:10.1115/OMAE2018-77946

[12] Nihei Y, Srinivasamurthy S, Sakamoto H, et al. Design methodology and development of an independently rotating multi-hull vessel. Proceedings of the ASME 2019 38th International Conference on Ocean, Offshore and Arctic Engineering; 2019. DOI:10.1115/OMAE2019-95401

[13] Srinivasamurthy S, Sakamoto H, Nishikawa T, et al. Numerical hull resistance and hydrodynamic characteristics of an independently rotating multi-hull vessel. Proceedings of the ASME 2019 38th International Conference on Ocean, Offshore and Arctic Engineering; 2019. DOI:10.1115/OMAE2019-95403

[14] Dunbabin M, Marques L. Robotics for environmental monitoring. IEEE Robot Autom Mag. 2012;19(1):24–39.

[15] Nihei Y, Tsurumi Y, Masuda N, et al. Automatic and frequent measurement of water quality at multiple points using quadmaran. J Japan Soc Civil Eng Ser B1. 2020;76(2):1-1039–1-104.

[16] Harada K, Ooei N. Present condition and issues of environment and fishery in the Nanaobay. Proceedings of the 61st Japan Joint Automatic Control Conference; Nagoya, Japan: 2018. p. 90–91.

[17] Special issue on IoT/ICT-based advanced technology for aquaculture and fishery industry. Systems, control, and information. 2021;65(1):2–32.

[18] Mukai K, Hara N, Konishi K. Propulsion fault detection in quad-maran unmanned vessel using moving horizon estimation. Proceedings of the SICE Annual Conference; 2020. p. 1725–1728.

[19] Mukai K. Thruster fault detection in quad-maran vessel using MHE [undergraduate thesis]. Osaka Prefecture University; 2019.

[20] Alessandri A, Baglietto M, Battistelli G. Receding-horizon estimation for switching discrete-time linear systems, IEEE Trans Automat Control 2005;50(11): 1736–1748.

Appendix A Excerpt of MATLAB Script for Identification

Here is an excerpt of MATLAB script used in the identification.

```matlab
A1=[0 1; 0 -1];
B1=[0 0; 0.01 0.01];
C1=[1 0];
D1=0;

Indl=idatal; % input and output data set
ms1=idss(A1,B1,C1,D1);
ms1.Structure.a.Free=[0 0; 0 1];
ms1.Structure.b.Free=[0 0; 1 1];
ms1.Structure.c.Free=0;
ms1.Structure.d.Free=0;
ms1.Ts = 0;
ms1.x0=[0, 0];
LSM1=ssest(Indl,ms1,
ssestOptions('Display','on'));
```

Appendix B Estimation Performance Test

We performed numerical simulations and tested the estimation performance of the current state variables $x_k$ and 0-1 integer variable $\delta^o_{ij}$ for all fault patterns. The model and parameters ($Q^{-1}$, $R^{-1}$, $\pi^{-1}$, $\delta w_1$, $\delta w_2$, and the sample time) used in the simulations were the same as the ones used for estimation in Section 4. The estimation horizon $N$ was set to 20 steps. Taking into account the noises in real environment, uniformly distributed disturbances, process noise $w_t$ and observation noise $v_t$ were added to the model.

$$w_t \in [-0.001, 0.001], \ v_t \in [-0.01, 0.01].$$

An MPC controller was the same with the one used in Section 4 except that 0.1 in $R_{MPC}$ was used instead of 0.01 for partial fault compensation. The initial state $x_0$ was $[10, 10, 0, 0]$. The time length of the simulation was 120 steps for each simulation, and the estimation started from 20 steps. The fault occurred at 30 steps.

The results of the estimation performance are summarized in Table B1. For each element of the current state variables $x_k$, we calculated the mean square error. For $0-1$ integer variable $\delta^o_{ij}$, we computed the percentage of correct answers for all elements in the estimated horizon at each time.
Table B1. Estimation performance of $x_k$ and 0-1 integer variable $\delta_{ij}$.

| Fault cond. in $u^1$ | $0$ | $1$ | $2$ | $0$ | $0$ | $1$ | $2$ | $1$ | $2$ | $1$ | $2$ |
|----------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $p_x$ [m]            | 0.2233 | 0.1669 | 0.1385 | 0.2118 | 0.2191 | 0.1807 | 0.12 | 0.1826 | 0.1158 |
| $p_y$ [m]            | 0.205 | 0.2066 | 0.1995 | 0.1604 | 0.1243 | 0.1565 | 0.1153 | 0.1281 | 0.1602 |
| $p_x$ [m/s]          | 0.044 | 0.0429 | 0.0827 | 0.0463 | 0.0478 | 0.0401 | 0.0776 | 0.0423 | 0.0772 |
| $p_y$ [m/s]          | 0.0432 | 0.0391 | 0.0391 | 0.0465 | 0.0875 | 0.0467 | 0.0824 | 0.0867 | 0.0448 |
| $\delta_{10}$ [%]   | 82.3 | 70.15 | 92.25 | 76.35 | 72.4 | 72.8 | 90.95 | 80.7 | 90.35 |
| $\delta_{11}$ [%]   | 84.0 | 62.55 | 84.05 | 78.1 | 76.9 | 64.65 | 82.05 | 72.4 | 82.3 |
| $\delta_{12}$ [%]   | 98.3 | 91 | 79.2 | 98.25 | 95.5 | 89.95 | 77 | 91.1 | 77.45 |
| $\delta_{20}$ [%]   | 77.45 | 78 | 77.35 | 77.15 | 91.2 | 73.5 | 91.3 | 92.9 | 76.45 |
| $\delta_{21}$ [%]   | 77.5 | 78.05 | 78.4 | 67.45 | 77.7 | 61.85 | 80.95 | 81.2 | 66.9 |
| $\delta_{22}$ [%]   | 99.95 | 99.95 | 98.95 | 90.2 | 73.9 | 88.35 | 77.95 | 76.6 | 90.25 |