Relative Vessel Motion Tracking using Sensor Fusion, Aruco Markers, and MRU Sensors

Article *in* Modeling, Identification and Control (MIC) · April 2017

DOI: 10.4173/mic.2017.2.3

2 authors:

Sondre Sanden Tørdal
University of Agder
18 PUBLICATIONS **122** CITATIONS

SEE PROFILE

Geir Hovland
University of Agder
97 PUBLICATIONS **1,344** CITATIONS

SEE PROFILE

Modeling, Identification and Control, Vol. 38, No. 2, 2017, pp. 79–93, ISSN 1890–1328

Relative Vessel Motion Tracking using Sensor Fusion, Aruco Markers, and MRU Sensors

Sondre Sanden Tørdal [1] and Geir Hovland [1]

[1] *Faculty of Engineering and Science, University of Agder, 4879 Grimstad, Norway. E-mail:* sondre.tordal@uia.no

Abstract

This paper presents a novel approach for estimating the relative motion between two moving offshore vessels. The method is based on a sensor fusion algorithm including a vision system and two motion reference units (MRUs). The vision system makes use of the open-source computer vision library OpenCV and a cube with Aruco markers placed onto each of the cube sides. The Extended Quaternion Kalman Filter (EQKF) is used for bad pose rejection for the vision system. The presented sensor fusion algorithm is based on the Indirect Feedforward Kalman Filter for error estimation. The system is self-calibrating in the sense that the Aruco cube can be placed in an arbitrary location on the secondary vessel. Experimental 6-DOF results demonstrate the accuracy and efficiency of the proposed sensor fusion method compared with the internal joint sensors of two Stewart platforms and the industrial robot. The standard deviation error was found to be 31mm or better when the Arcuo cube was placed at three different locations.

Keywords: Sensor fusion, vision, offshore motion compensation, Kalman filter, Aruco.

1 Introduction

Complex offshore load operations are usually carried out by the current industry state-of-the-art Active Heave Compensated (AHC) cranes, which are capable of decoupling the floating vessel's motion and the hanging load's motion by using a Motion Reference Unit (MRU) to measure the vessel's motion in real-time. As a result, the AHC crane is capable of controlling the load's height above the seabed for instance, which again will reduce the risk of destroying the hanging load when lowering it onto the seabed. These AHC cranes and MRU sensors have been tested in the offshore industry for years, and have been an important enabler for more advanced and efficient offshore operations in general. In future offshore load handling operations, an increased level of complexity is expected due to an increased level of offshore activities such as: floating wind turbines, remote fish farms and autonomous shipping, to mention some. A common challenge for all these operations is that cargo, equipment and personnel have to be transferred between two floating installations. This is the main motivation for investigating the Vessel-to-Vessel Motion Compensation (VVMC) problem (see Figure 1 for illustration) which is seen as an extension of the AHC techniques used today. In VVMC, it is necessary to establish a method for measuring the relative motion between two floating vessels in real-time. As a result of measuring these motions precisely and efficiently, the load handling equipment such as a crane could be used to keep a hanging load in a fixed position relative to the secondary vessel. This is an important enabler for load transfer between two vessels during harsher weather conditions than allowed for today. Today such operations are limited by a so-called weather window, which only allows for any load or personnel to be transferred from one vessel to another if the significant wave height is below typically 2.5 meters (Kjelland (2016)). By introducing VVMC, load transfer between two vessels during harsher weather conditions may be allowed in the future. As a result, such operations can be carried out in a safer and more

doi:10.4173/mic.2017.2.3

efficient way. Also, the operation cost may be reduced, given the fact that the time spent in waiting for better weather conditions to actually carry out the load transfer may be drastically reduced.

As an initial step towards solving the VVMC problem, a suitable method for measuring the relative motion between two floating vessels is needed. A method for measuring these relative motions using MRU sensors is motivated by the fact that MRUs have been successfully used for measuring vessel motions for many years in the industry. In addition, extensive research towards robust filtering algorithms using Inertial Measurement Units (IMU) and gyroscopes have been investigated by Küchler et al. (2011b); Richter et al. (2014); Küchler et al. (2011a) to mention some. It is therefore assumed that MRUs or IMUs with accompanying filtering algorithms are capable of measuring the motions of the two independent vessels with sufficient accuracy. However, in VVMC it is not enough to only measure the motion of each vessel independently, but rather to measure the relative motion between them. It is therefore believed from our previous experience and research presented by Tørdal et al. (2016), that a third sensor like a camera vision system capable of measuring the absolute motions between the two floating vessels can be used together with two wirelessly connected MRUs placed onto the two vessels in order to track the relative motions in real-time. Extensive research within the field of motion tracking using camera vision has already been carried out, and efficient programming libraries such as OpenCV (Itseez (2015)) provide a lot of functions which can be used for tracking rigid objects seen by a camera. One of the most recent methods which is capable of measuring all 6 Degrees of Freedom (DOF) of a fiducial marker is provided by the Aruco add-on library for OpenCV presented by Garrido-Jurado et al. (2014). By using this programming library together with a suitable camera and two MRUs, the authors believe that a proof-of-concept for tracking the relative vessel motions is realizable.

In this paper, a sensor fusion algorithm combining the measurements acquired from two independent MRUs placed onto two moving vessels, and a camera vision tracking system capable of measuring all 6 DOFs are presented. An extended Kalman filter has been used to improve the camera vision tracking performance, and a linear Kalman filter has been applied to estimate the error between the vision and the MRU measurements. In addition, the presented method is self-calibrating, meaning that the secondary MRU can be placed in any arbitrary location at the secondary vessel. As a final experiment, the equipment in the Norwegian Motion Laboratory is used to verify the effectiveness and accuracy of the proposed solution to the VVMC motion tracking problem.

2 Problem Formulation and Experimental Lab Setup

In VVMC, the main goal is to safely transport either personnel or cargo between two floating vessels which can be two ships, offshore platforms, floating wind turbines etc. The detection of the relative motions between the two floating vessels in real-time is considered a crucial task which has to be solved in order to carry out the motion compensation task using offshore cranes or other robot-like equipment. The relative motion between the two floating ships consists of a positional offset, and an orientation offset. In robotics, these motions are often described using homogeneous transformation matrices, which describe the geometric transformation between the two body coordinates of body-A $\{b_A\}$ relative to body-B $\{b_B\}$. An illustration of two floating vessels laying alongside each other at sea is given by Figure 1.

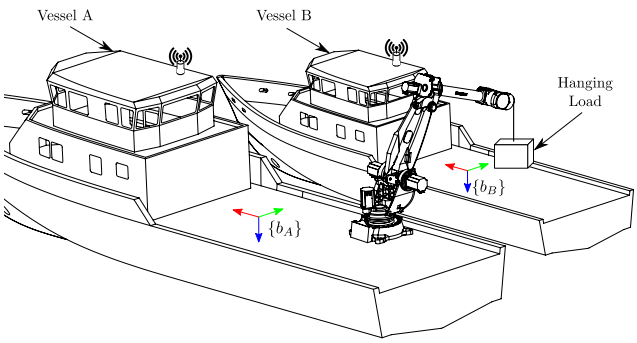

Figure 1: The VVMC problem where a load is supposed to be transported from vessel/ship A onto vessel/ship B.

The figure illustrates both the body-fixed coordinate systems of vessels A and B, a load handling equipment (industrial robot), and a suspended load hanging in the end of the industrial robot (end-effector). The industry standard equipment used to detect the motion of floating vessels relies heavily on the use of MRU sensors to measure the heave, roll and pitch motions of the vessel. These sensors are proven to give precise and reliable measurements and have been used for many years in the application of offshore AHC cranes. However, in VVMC, it is desired to keep the hanging load in some desired height above the secondary vessel's ship deck, or even in a given orientation relative to some coordinate system defined on the secondary floating vessel. To experimentally test and investigate

the effectiveness of the proposed method, the lab setup shown in Figure 4 is used.

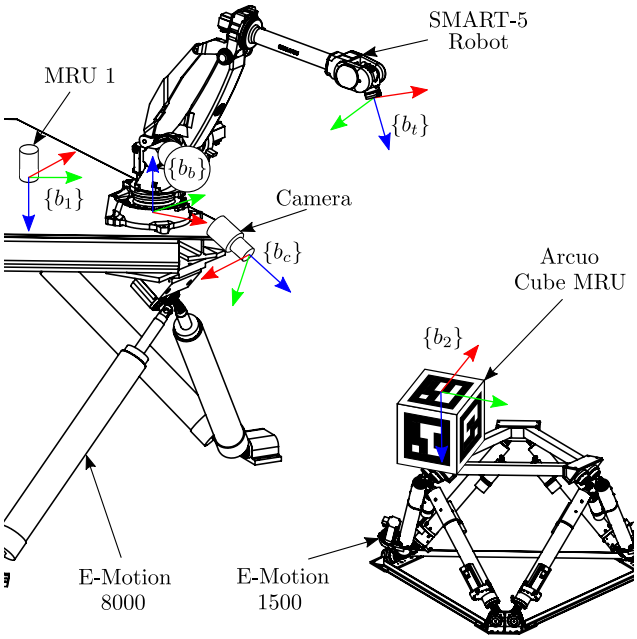

Figure 2: The Norwegian Motion Laboratory located at the University of Agder.

Figure 2 illustrates the experimental lab setup used to carry out the VVMC experiments presented in this paper. As seen from the figure, the camera (Logitech C930e) is mounted onto the biggest Stewart platform (E-Motion 8000) in front of the industrial robot (Comau SMART-5 NJ 110-3.0) used to simulate the load handling equipment. One of the MRUs, namely MRU1, is placed on top of the biggest Stewart platform, and a secondary MRU (MRU2) is placed inside the Aruco cube. By placing the second MRU inside the Aruco cube it is possible to use the camera to measure the absolute orientation and position offset between the two body-fixed coordinates of MRU1 and MRU2. This absolute measurement is an important enabler for the proposed sensor fusion system which is combining the MRUs and the vision system. The proposed sensor fusion algorithm is illustrated in Figure 3.

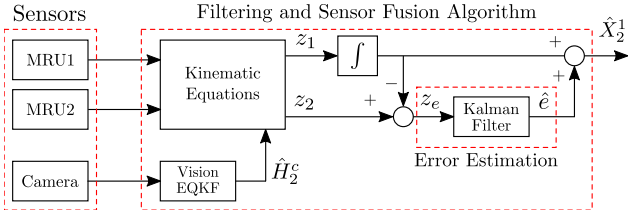

Figure 3: Illustration of the proposed sensor fusion algorithm combining the camera and the MRU measurements.

The algorithm relies on the measurements acquired from two MRUs and a vision tracking system used to measure the absolute position and orientation offset between the two body-fixed coordinates of the MRUs. In addition, an important feature of the proposed solution is the fact that the Aruco cube can be placed at any location onto the secondary vessel's ship deck. This enables for easier installation and more flexible operation, since the proposed solution has the ability for automatic self-calibration.

To carry out the VVMC lab-experiment, a common control and logging interface is needed. Figure 4 illustrates how the laboratory equipment is connected to the main control unit delivered by Beckhoff Automation (CX2040).

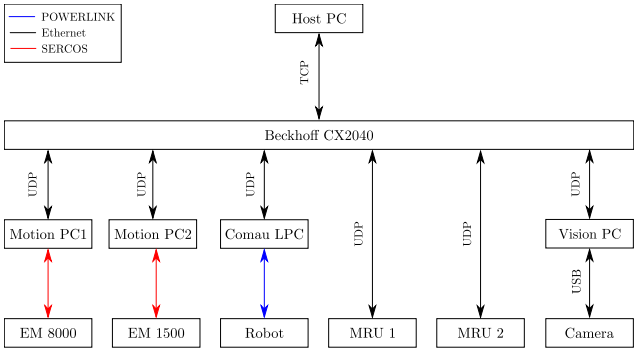

Figure 4: Communication network used to operate all the lab equipment from a common control unit (Beckhoff CX2040).

The communication network consists of three different types of communication: SERCOS, POWERLINK, and Ethernet. While SERCOS and POWERLINK are both hard real-time and deterministic communication protocols, the UDP communication protocol using a conventional Ethernet connection is not proven to be real-time, nor deterministic. However, the author's experience shows that the UDP interface has demonstrated to give satisfactory performance since the wave motions to be simulated in the laboratory are found in the range of 8-20s in typical wave periods.

3 Vision Tracking of Aruco Cube

In order to use the acquired measurements from both the MRUs placed onto the two vessels, a third sensor is needed (Tørdal et al. (2016)) to measure the absolute position and orientation H_2^1 between the two body-fixed coordinate frames $\{b_1\}$ and $\{b_2\}$ of MRU1 and MRU2 (see Figure 2). A camera vision tracking method capable of measuring all 6 DOFs between the camera body fixed coordinate system $\{b_c\}$ and the sec-

ondary MRU's body-fixed coordinate system $\{b_2\}$ is proposed as a possible solution. A picture showing the lab set-up used to simulate the suggested VVMC sensor fusion algorithm is given in Figure 5.

Figure 5: Experimental lab setup for VVMC in the Norwegian Motion Laboratory.

3.1 Aruco Marker Detection using OpenCV

To measure all 6 DOFs using the camera placed in front of the robot, it was chosen to use a square cube and place an Aruco marker on each of the cube sides except the side which is facing down. The reason for having a cube is motivated by the fact that it is possible to detect one of the markers given any orientation of the cube, as long as the bottom of the cube is facing downwards. The considered Aruco cube which is a square cube with an edge length of 39 cm is illustrated in Figure 6.

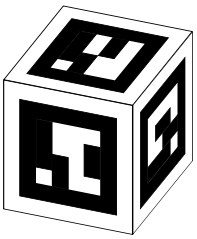

Figure 6: Aruco cube with an MRU placed inside the cube.

By using the Aruco add-on library supported by OpenCV (Garrido-Jurado et al. (2014)), it is fairly straightforward to make a C++ program capable of tracking each of the Aruco markers: identification, position and pose relative to the camera at a typical update cycle ranging in between 40-100ms. The update speed of the algorithm is like for most vision applications, heavily influenced by the selected image resolution. The position and orientation of the i'th Aruco marker can be defined as:

$$H_i^c = \begin{bmatrix} R_q(q_i) & p_i \\ 0 & 1 \end{bmatrix} \qquad i \in [0, N_m] \tag{1}$$

where H_i^c is the homogeneous transformation matrix between the i'th marker and the camera (also known as the extrinsic parameters), $R_q(q_i)$ and q_i is the i'th rotation matrix and the quaternion respectively, describing the i'th marker's pose given in the camera coordinate system, p_i is the positional vector describing the position of the i'th marker, and N_m is the number of Aruco markers detected in the processed picture. $R(q_i)$ is the function converting a quaternion into a rotation matrix as:

$$\underbrace{2\begin{bmatrix} \frac{1}{2} - q_y^2 + q_z^2 & q_x q_y - q_0 q_z & q_0 q_y + q_x q_z \\ q_0 q_z + q_x q_y & -q_x^2 - q_z^2 + \frac{1}{2} & q_y q_z - q_0 q_x \\ q_x q_z - q_0 q_y & q_0 q_x + q_y q_z & -q_x^2 - q_y^2 + \frac{1}{2} \end{bmatrix}}_{R_q(q)} \tag{2}$$

where q_0, q_x, q_y, q_z are the quaternion coefficients describing the quaternion. The quaternion is defined by:

$$q := q_0 + q_x i + q_y j + q_z k \tag{3}$$

$$i^2 + j^2 + k^2 := -1 \;\text{ when }\; i \neq j \neq k \tag{4}$$

where i, j, k are the fundamental quaternion units which all square to -1. To further understand how the quaternion can be used to describe the orientation of a rigid body relative to a given coordinate system, the following definition is used:

$$q = \underbrace{\cos(\theta/2)}_{q_0} + \underbrace{\hat{n}\sin(\theta/2)}_{q_x i + q_y j + q_z k}, \qquad ||\hat{n}|| = 1 \tag{5}$$

where $\hat{n}$ is a unit axis in SO(3) and θ is the rotation of the rigid body around axis $\hat{n}$ in radians. The quaternion has many useful and superior properties compared to Euler-angles especially. However, the reason for using quaternions is motivated by the fact that several (N_m) Aruco markers are measured and an average of these measurements are desired. The averaging method is presented in the next subsection.

3.2 Position and Quaternion Averaging

Given that we have N_m measurements of the Aruco marker position and orientation relative to the camera, it is straightforward to calculate the average position $\bar{p}$ as:

$$\bar{p} = \frac{1}{A}\sum_{i=1}^{N_m} a_i p_i, \qquad A = \sum_{i=1}^{N_m} a_i \tag{6}$$

where a_i is the area of the i'th marker in the picture frame, p_i is the measured position vector to marker number i, and A is the total area of all the markers found in the current picture. The area of the i'th marker is found by using the shoelace formula (Braden (1986)) which calculates the area formed by four image points as:

$$\begin{aligned} a = \frac{1}{2}|&x_1y_2 + x_2y_3 + x_3y_4 + x_4y_1 \cdots \\ &-x_1y_4 - x_2y_1 - x_3y_2 - x_4y_3| \end{aligned} \tag{7}$$

where $\{x_1 \cdots x_4, y_1 \cdots y_4\}$ are the x- and y-pixel coordinates of the Aruco corner points returned by the Aruco detection function in OpenCV. The location of the Aruco corner points is further illustrated in Figure 7.

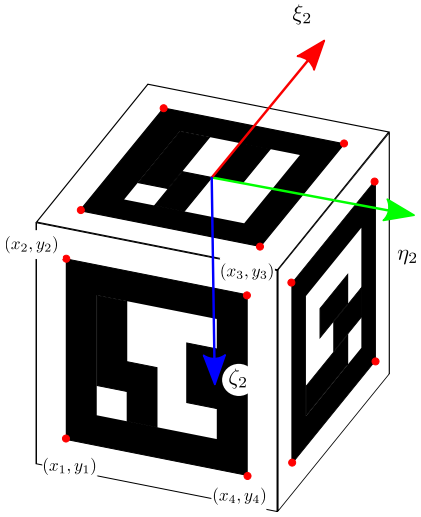

Figure 7: Corner points of each Aruco marker as they are represented in the OpenCV Aruco library.

As for the positions p_i to each of the Aruco markers, also the orientation of each marker q_i is processed to obtain the average quaternion $\bar{q}$ of all N_m detected markers. However, averaging a quaternion is not as straightforward as for the positions since quaternions are not linearly independent. The quaternion averaging method of Markley et al. (2007) has therefore been applied to solve this problem. The principle used is that all the measured quaternions are stacked into a measurement matrix Q using the following expression:

$$Q = \frac{1}{A}\sum_{n=1}^{N_m} a_i q_i q_i^T, \qquad A = \sum_{i=1}^{N_m} a_i \tag{8}$$

where $q_i q_i^T$ is defined by Equation (9).

$$qq^T = \begin{bmatrix} q_0^2 & q_0q_x & q_0q_y & q_0q_z \\ q_0q_x & q_x^2 & q_xq_y & q_xq_z \\ q_0q_y & q_xq_y & q_y^2 & q_yq_z \\ q_0q_z & q_xq_z & q_yq_z & q_z^2 \end{bmatrix} \tag{9}$$

To find the averaged quaternion $\bar{q}$, the eigenvalue problem of measurements matrix Q has to be solved using the eigenvalue decomposition. The eigenvalue problem is defined as:

$$Qv = \lambda v \tag{10}$$

where the average quaternion $\bar{q}$ is given by the eigenvector v corresponding to the smallest eigenvalue λ. As an enabler to carry out the quaternion averaging in real-time, the Eigen library (Guennebaud and Jacob (2010)) supported by C++ is used to implement the quaternion averaging algorithm.

3.3 Image Model and Camera Calibration

To obtain correct physical measurements of the Aruco cube position and orientation (extrinsic parameters), the camera's intrinsic parameters have to be calibrated properly. These parameters describe the relation between the 2D image pixels (x_i, y_i) and the real world coordinates (x, y, z). The relationship between the image pixels and the world coordinates is modeled using the pinhole camera model given by:

$$\begin{bmatrix} x_i \\ y_i \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{11}$$

where f_x and f_y are the camera focal lengths in the x- and y-direction, (c_x, c_y) is the optical center expressed in pixels, and K is known as the camera calibration matrix which can be found from taking several pictures of a checkerboard and process them using the Matlab camera calibration toolbox. The OpenCV library also

features a camera calibration functionality, but it is not as user friendly as the Matlab toolbox. One of the pictures used to calibrate the camera is given in Figure 8.

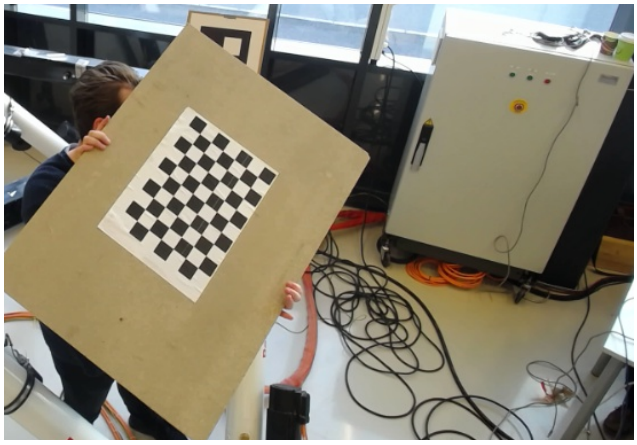

Figure 8: One of the pictures used to calibrate the camera using the Matlab camera calibration toolbox.

In addition to calculate the camera calibration matrix K, it is also necessary to remove the radial image distortion. The mathematical relationship between the corrected image pixels (x_i, y_i) and the radial distorted pixels (x_d, y_d) is given by:

$$x_i = x_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{12}$$

$$y_i = y_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{13}$$

$$r = \sqrt{x_d^2 + y_d^2} \tag{14}$$

where $\{k_1, k_2, k_3\}$ are the radial distortion coefficients which are also found using the Matlab calibration toolbox. In addition to the radial distortion, tangential distortion may occur if the camera lens is not perfectly parallel to the image sensor plane. However, it is in general common to ignore this since it can be assumed that the lens is more or less parallel to the image sensor.

3.4 Extended Quaternion Kalman Filter (EQKF) for Bad Pose Rejection

Subsection 3.2 presented an averaging method of all the detected markers in the camera picture. The observed noise in the averaged measurements $\bar{p}$ and $\bar{q}$ was not considered small enough to be used directly in the sensor fusion algorithm which will be discussed later. Since the orientation problem is described using a quaternion, an Extended Quaternion Based Kalman Filter (EQKF) is used to estimate the Aruco cube position and orientation. The approach presented here is inspired by the approach used by Kraft (2003), Pawlus et al. (2016) and Marins et al. (2001). The state-vector of the combined position and orientation estimation problem is:

$$x = \begin{bmatrix} p \\ \dot{p} \\ \ddot{p} \\ q \\ \omega \\ \dot{\omega} \end{bmatrix} \tag{15}$$

where $p, \dot{p}$ and $\ddot{p}$ are the position, velocity and acceleration of the Aruco cube, q is the orientation quaternion, and $\omega, \dot{\omega}$ are the rotational velocities and accelerations in the body-fixed coordinate system of the Aruco cube. The vector components of p, q and ω are given by Equation (16).

$$p = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \qquad q = \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix} \qquad \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{16}$$

It is also required that the orientation quaternion is kept unitary at all times, meaning that both the estimated quaternion $\hat{q}$ and the measured quaternion $\bar{q}$ are constrained as:

$$\|q\| = \sqrt{q_0^2 + q_x^2 + q_y^2 + q_z^2} = 1. \tag{17}$$

3.5 Process and Measurement Model

The state transition and observation model is given as:

$$\begin{aligned} x_k &= f(x_{k-1}) + w_{k-1} \\ z_k &= h(x_k) + v_k \end{aligned} \tag{18}$$

where x_k is the current state, w_{k-1} is the previous process noise, z_k is the current measurement vector, $h(x_k)$ is the current measurement model, and v_k is the current measurement noise. Both the process and the measurement noise are assumed to have covariances Q_k and R_k with zero mean value. The non-linear state transition function is defined by the following equation:

$$f(x) = \begin{bmatrix} p + \dot{p}\Delta t + \frac{1}{2}\ddot{p}\Delta t^2 \\ \dot{p} + \ddot{p}\Delta t \\ \ddot{p} \\ q + \dot{q}\Delta t \\ \omega + \dot{\omega}\Delta t \\ \dot{\omega} \end{bmatrix} + w \tag{19}$$

where Δt is the time step of the proposed vision algorithm (65 ms), and $\dot{q}$ is the time-differentiated quaternion which is defined by:

$$\dot{q} = \frac{1}{2} \underbrace{\begin{bmatrix} 0 \\ \omega + \dot{\omega}\Delta t \end{bmatrix}}_{q_\omega} \otimes q \tag{20}$$

where q_w is the rotational velocity vector ω treated as a quaternion with zero scalar component, and $\otimes$ represents the quaternion product. The measurement function $h(x)$ is fairly simple since both the position p and the orientation q are measured directly as:

$$h(x) = \begin{bmatrix} \bar{p} \\ \bar{q} \end{bmatrix} + v \tag{21}$$

where both the measured position and the orientation are obtained from the averaged measurements as they are defined in Subsection 3.2.

3.6 State Estimation and Update Equations

The discrete-time model of the process and measurement model is defined as as:

$$\begin{aligned} \hat{x}_k &= F_{k-1}\hat{x}_{k-1} + w_{k-1} \\ z_k &= H_k\hat{x}_k^- + v_k \end{aligned} \tag{22}$$

where F_{k-1} is the linearized state transition matrix evaluated at the previous state estimate $\hat{x}_{k-1}$, and H_k is the linearized measurement matrix evaluated at the current predicted state $\hat{x}_k^-$. The state transition matrix and measurement matrix are given by the following two equations:

$$F_{k-1} = \frac{\partial f}{\partial x}(\hat{x}_{k-1}, 0) \tag{23}$$

$$H_k = \frac{\partial h}{\partial x}(\hat{x}_k^-, 0). \tag{24}$$

Given these equations it is possible to predict and correct the state estimation. The prediction equations are defined as:

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1} \tag{25}$$

$$P_k^- = F_{k-1}P_{k-1}F_{k-1}^T + Q_{k-1} \tag{26}$$

where $\hat{x}_k^-$ is the current state prediction, and P_k^- is the predicted covariance. Based on the prediction, the model may be corrected given that the measurement z_k is occurring. The measurement correction equations are defined as:

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k + R_k\right)^{-1} \tag{27}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H_k\hat{x}_k^-\right) \tag{28}$$

$$P_k = \left(I - K_k H_k\right) P_k^- \tag{29}$$

where K_k is the current Kalman gain, $\hat{x}_k$ is the current state estimate, and P_k is the current covariance matrix. The process and measurement covariance matrices are constant for all k and are defined as identity matrices with a constant gain on each of the diagonal elements. The covariance matrices Q_k and R_k are defined as:

$$Q_k = \sigma_q^2 I, \qquad R_k = \sigma_r^2 I \tag{30}$$

where the process variance σ_q^2 and the measurement variance σ_r^2 are manually tuned to give satisfactory performance and noise rejection. The actual parameter values for σ_q^2 and σ_r^2 are given in the Appendix. The C++ source files for the Aruco box tracking algorithm are available at: https://github.com/sondre1988/vision-tracker/tree/master.

3.7 Hand-Eye Camera Calibration

In addition to the Aruco marker detection and the proposed EQKF algorithm, it is also necessary to know where the camera is relative to the robot base coordinate system $\{b_b\}$. To find these calibration parameters, a two-step procedure has to be carried out; first is to find the location of a marker relative to the end-effector coordinate system $\{b_t\}$ of the robot, second is to find the camera coordinate system $\{b_c\}$ relative to the robot base $\{b_b\}$. The first problem is known as the hand-eye calibration problem, and has been investigated using both closed form solutions and iterative approaches to successfully solve the matrix equation:

$$A_n X = X B_n \tag{31}$$

where A_n is the n'th movements of the robot end effector, B_n is the n'th movements of the marker attached to the robot end-effector, and X is the unknown location and orientation of the marker relative to the robot end-effector. By measuring k robot poses $H_{t,k}^b$ and k camera observations $H_{m,k}^c$ of the marker attached to the robot end-effector. The matrix A_n and B_n are found by using the following two equations:

$$A_n = \left(H_{t,k}^b\right)^{-1} H_{t,k-1}^b \tag{32}$$

$$B_n = \left(H_{m,k}^c\right)^{-1} H_{m,k-1}^c. \tag{33}$$

To actually solve the hand-eye calibration problem given that k measurements are carried out, the closed-form solution presented by Park and Martin (1994) is used since it serves as an efficient and elegant solution when several measurements are present. Later on, also Baillot et al. (2003) have demonstrated the efficiency and practical use of the hand-eye calibration algorithm presented by Park and Martin. The unknown homogeneous transformation matrix describing the marker location relative to the robot end-effector is given by:

$$X = \begin{bmatrix} R_X & t_X \\ 0 & 1 \end{bmatrix}, \qquad X \in SE(3) \tag{34}$$

where R_X is the unknown rotation matrix, and t_X is the unknown translation vector. The proposed least-squares solution presented by Park and Martin utilizes the following equations to solve the hand-eye calibration problem. The logarithm of a matrix is defined as:

$$\log(R) = \frac{\theta}{2\sin(\theta)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \qquad R \in SO(3) \tag{35}$$

where $\log(R)$ is defined as the logarithm of an orthogonal rotation matrix R. The angle θ is defined as:

$$\cos(\theta) = \frac{\mathrm{Tr}(R) - 1}{2} \tag{36}$$

where $\mathrm{Tr}(R)$ is the trace of matrix R. In order to solve for the rotation matrix R_X, it is necessary to take the logarithm of the rotation matrices R_{A_n} and R_{B_n} and stack them into a measurement matrix, according to:

$$M = \sum_{i=1}^{n} \log(R_{B_i}) \log(R_{A_i})^T \tag{37}$$

where M is the measurement matrix used to solve for the unknown rotation matrix R_X using:

$$R_X = (M^T M)^{-\frac{1}{2}} M^T \tag{38}$$

where $(M^T M)^{-\frac{1}{2}}$ can be found using the eigenvalue decomposition or the singular value decomposition (SVD). The eigenvalue decomposition is used as described by $(M^T M)^{-\frac{1}{2}} = V D^{-\frac{1}{2}} V^{-1}$. When the rotation matrix R_X has been determined, the unknown position t_X can be found from applying the least-squares solution described as:

$$t_X = (C^T C)^{-1} C^T d \tag{39}$$

where matrix C and vector d are given by the following equations:

$$C = \begin{bmatrix} I - R_{A_1} \\ I - R_{A_2} \\ \vdots \\ I - R_{A_n} \end{bmatrix}, \qquad d = \begin{bmatrix} t_{A_1} - R_X t_{B_1} \\ t_{A_2} - R_X t_{B_2} \\ \vdots \\ t_{A_n} - R_X t_{B_n} \end{bmatrix} \tag{40}$$

A Matlab function of the hand-eye calibration algorithm has been developed and is available at https://github.com/sondre1988/matlab-functions/blob/master/src/HandEyeParkMartin.m

4 MRU and Vision Sensor Fusion

As motivated in both the introduction and the problem formulation, the desire of tracking the relative motions between two floating vessels in real-time is of high importance when it comes to solving the VVMC problem. In this section, a sensor fusion system utilizing the visual tracking system described in Section 3 and the MRU sensors will be presented.

4.1 Kinematic Model

A kinematic model of the camera relative to the MRUs and the robot base is needed in order to utilize the acquired measurements from the camera and the MRUs in a suitable way. Figure 9 is used to illustrate the kinematic model and the accompanying coordinate systems which are involved.

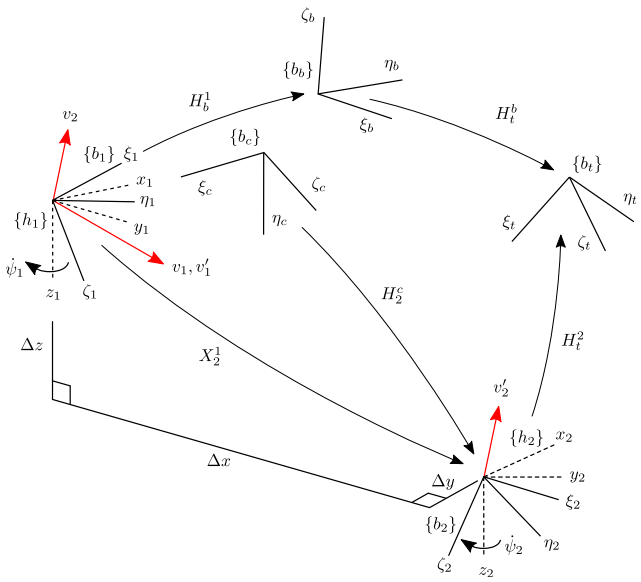

Figure 9: Coordinate systems as they are used in the sensor fusion algorithm to describe the relative position and orientation between the MRUs.

The following notation is assumed in Figure 9: $\{b_1\}$ and $\{b_2\}$ are the body-fixed coordinates of MRU1 and MRU2, $\{h_1\}$ and $\{h_2\}$ are the heading coordinates of MRU1 and MRU2 which are equal to the body-fixed coordinates when both the roll and pitch angles are equal to zero. The z-axis of both heading coordinates is always pointing downwards. The camera body coordinate is denoted as $\{b_c\}$, the robot base $\{b_b\}$ and the robot end-effector $\{b_t\}$ are only denoted using their body-fixed coordinates. The offset position and orientation between the two heading frames $\{h_1\}$ and $\{h_2\}$ is given by:

$$X_2^1 = T_x(\Delta x) T_y(\Delta y) T_z(\Delta z) R_z(\Delta \psi) \tag{41}$$

where Δx, Δy and Δz are the offset positions, and $\Delta \psi$ is the offset heading angle. To fully determine the

body-fixed location and orientation between the two MRUs, the roll and pitch measurements from both the MRUs are simply added as:

$$H_2^1 = (R_x(\phi_1)R_y(\theta_1))^{-1} X_2^1 R_x(\phi_2)R_y(\theta_2) \tag{42}$$

where H_2^1 is the offset between the two body-fixed MRU coordinates.

4.2 Sensor Fusion Algorithm

The objective of the sensor fusion algorithm is to estimate the homogeneous transformation matrix X_2^1 between the two heading frames as precisely as possible combining the measurements of the two MRUs and the camera vision system presented in Section 3. The sensor fusion algorithm which is shown in Figure 3 estimates X_2^1 using the measurements z_1 and z_2. This filter structure is named Indirect Feedforward Kalman Filter for error estimation (Sasiadek and Hartana (2000)). The filter algorithm utilizes two measurements: z_1 which represents the changes in X_2^1 and z_2 which serves as an absolute measurement of X_2^1. By simply integrating z_1 it is possible to define an error measure z_e as:

$$z_e = z_2 - \int z_1 dt \tag{43}$$

and use a linear Kalman filter to find an estimated error $\hat{e}$ and add this to measurement z_1 such as:

$$\hat{X}_2^1 = f(z_1, \hat{e}) \tag{44}$$

where $\hat{X}_2^1$ represents the estimated position and orientation offset between the two MRU heading frames. The Kalman filter state-vector is then defined as:

$$x = \begin{bmatrix} e \\ \dot{e} \\ \ddot{e} \end{bmatrix}, \qquad e = \begin{bmatrix} \Delta x_e \\ \Delta y_e \\ \Delta z_e \\ \Delta \psi_e \end{bmatrix} \tag{45}$$

where Δx_e, Δy_e and Δz_e are the position errors and $\Delta\psi_e$ is the heading angle error. It is assumed that a linear process model describing the error measurements is satisfactory since the change in $\Delta\psi_e$ is most likely to be smaller than $\pm 5°$. The error process model is given by:

$$f(x) = \begin{bmatrix} e + \dot{e}\Delta t + \frac{1}{2}\ddot{e}\Delta t^2 \\ \dot{e} + \ddot{e}\Delta t \\ \ddot{e} \end{bmatrix} \tag{46}$$

where Δt is the time step used in the sensor fusion algorithm ($\Delta t = 4ms$). The measurement model is described as:

$$h(x) = e \tag{47}$$

indicating that the error can be measured directly using the camera and the MRUs. However, this is not completely true if the kinematic equations describing X_2^1 and H_2^1 are considered in detail. Anyway, as an engineering approach it is assumed that the measured roll and pitch angles from both the MRU sensors are very accurate and reliable, and hence the following equations can be used to calculate the two measurements z_1 and z_2 which are defined as:

$$z_1 = \begin{bmatrix} \Delta\dot{x}_1 \\ \Delta\dot{y}_1 \\ \Delta\dot{z}_1 \\ \Delta\dot{\psi}_1 \end{bmatrix}, \qquad z_2 = \begin{bmatrix} \Delta x_2 \\ \Delta y_2 \\ \Delta z_2 \\ \Delta\psi_2 \end{bmatrix} \tag{48}$$

where $\Delta\dot{x}_1$, $\Delta\dot{y}_1$ and $\Delta\dot{z}_1$ can be calculated as:

$$\begin{bmatrix} \Delta\dot{x}_1 \\ \Delta\dot{y}_1 \\ \Delta\dot{z}_1 \end{bmatrix} = v_2 - v_1 \tag{49}$$

where v_1 and v_2 are the measured velocity vectors of MRU1 and MRU2 represented in the first heading coordinate system $\{h_1\}$ of MRU1. These two velocities are found by the following two equations:

$$v_1 = R_x(\phi_1)R_y(\theta_1)v_1' \tag{50}$$

$$v_2 = R_z(\Delta\psi_2)R_x(\phi_2)R_y(\theta_2)v_2' \tag{51}$$

where v_1' and v_2' are the linear velocities measured along the body-fixed coordinate axes of MRU1 and MRU2, respectively. The turn rate difference $\Delta\dot{\psi}_1$ between the two heading coordinate systems is given by:

$$\Delta\dot{\psi}_1 = \dot{\psi}_2 - \dot{\psi}_1 \tag{52}$$

where $\dot{\psi}_1$ and $\dot{\psi}_2$ are the two MRU measured turn rates as illustrated in Figure 9.

The second measurement used in the sensor fusion algorithm is found by measuring X_2^1 directly using the following equation:

$$X_2^1 = R_x(\phi_1)R_y(\theta_1)H_c^1\hat{H}_2^c\left(R_x(\phi_2)R_y(\theta_2)\right)^{-1} \tag{53}$$

where H_c^1 is the known camera location and orientation relative to MRU1, and $\hat{H}_2^c$ is the EQKF estimated orientation and position of MRU2 relative to the camera body-fixed coordinate system. Using Eq. (41) it is possible to write X_2^1 as:

$$X_2^1 = \begin{bmatrix} \cos(\Delta\psi_2) & -\sin(\Delta\psi_2) & 0 & \Delta x_2 \\ \sin(\Delta\psi_2) & \cos(\Delta\psi_2) & 0 & \Delta y_2 \\ 0 & 0 & 1 & \Delta z_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{54}$$

By comparing the coefficients of Eq. (53) and Eq. (54) it is fairly straightforward to calculate the second measurement z_2 as:

$$z_2 = f(\phi_1, \theta_1, \phi_2, \theta_2, H_c^1, \hat{H}_2^c). \tag{55}$$

The resulting equations is not given in detail since they are found by utilizing the symbolic toolbox of Matlab and are fairly long.

By using the proposed linear process model, measurement model, and the error measurements z_e it is possible to find the estimated version of $\hat{X}_2^1$ using the following prediction and correction equations:

$$\hat{x}_k^- = A\hat{x}_{k-1} + w_{k-1} \tag{56}$$
$$z_k = H\hat{x}_k^- + v_k \tag{57}$$

where the state transition matrix A and the measurement matrix H are found from taking the partial derivative of Eq. (46) and Eq. (47) respectively:

$$A = \frac{\partial f}{\partial x}, \qquad H = \frac{\partial h}{\partial x}. \tag{58}$$

The prediction equations are given as:

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{59}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{60}$$

and the corresponding correction equations as:

$$K_k = P_k^- H^T \left(HP_k^- H + R\right)^{-1} \tag{61}$$
$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \tag{62}$$
$$P_k = \left(I - K_k H\right)P_k^- \tag{63}$$

where Q and R are the diagonal process and measurement covariance matrices given by:

$$Q = \sigma_q^2 I, \qquad R = \sigma_r^2 I. \tag{64}$$

The actual values for σ_q^2 and σ_r^2 are manually tuned to give satisfactory performance and the resulting values are given in the Appendix.

4.3 End-effector Position Relative to the Secondary Vessel

Recalling Figure 9, it is seen from the coordinate systems that the robot end-effector is supposed to be kept in a constant position relative to the secondary MRU placed onto the secondary vessel's ship deck. The kinematic control of the robot is not considered the main contribution in this work, a simplified user input H_t^2 is therefore suggested and used for the final experimental verification. The user input is given as:

$$H_t^2 = T_x(x_u)T_y(y_u)\underbrace{R_y(-\theta_2)R_x(-\phi_2)}_{\text{MRU Measurements}}T_z(-z_u) \tag{65}$$

where x_u, y_u and z_u are the user defined positions of the robot end-effector relative to the body-fixed coordinate of MRU2 . The roll and pitch movements of MRU2 are used to always ensure that the last defined user input z_u is placing the end-effector in a desired height above the secondary ship deck in the direction of the world fixed z-axis. To finally carry out the VVMC task, the robot pose H_t^b has to be calculated using:

$$H_t^b = \left(H_b^1\right)^{-1}\hat{H}_2^1 H_t^2 \tag{66}$$

where $\hat{H}_2^1$ is found from Eq. (42) where X_2^1 is substituted with $\hat{X}_2^1$ which is a result of the proposed sensor fusion algorithm. Finally, the joint space angles q of the robot are found through applying the Inverse Kinematics (IK) algorithm given by:

$$q = f_{IK}(H_t^b). \tag{67}$$

The IK algorithm is not described in detail since it depends on the specific load handling equipment used to carry out the VVMC task.

5 Experimental Results

In this section, the result of applying the proposed sensor fusion algorithm is to be investigated. The motions of both the Stewart platforms are prescribed by stochastic wave motions defined by the Pierson-Moskowitz wave spectrum as presented by Pierson and Moskowitz (1964). The two Stewart platforms move asynchronously including all 6 DOF according to the wave spectrum prescribed by a typical wave period ranging in between 8 to 14 s. A more complex model to describe the motions of both the Stewart platforms could be investigated. However, it is assumed that the asynchronous stochastic wave motions is sufficient to describe the vessel motions.

5.1 Camera Filter Performance

The proposed EQKF algorithm presented in Section 3 is supposed to reduce the noise in the averaged position $\bar{p}$ and orientation $\bar{q}$ measurement of the Aruco cube. As a verification of the proposed algorithm, several plots comparing the averaged and the estimated curves are acquired from the physical experiments. The resulting x-, y- and z-direction EQKF performance curves are given in Figures 10, 11 and 12.

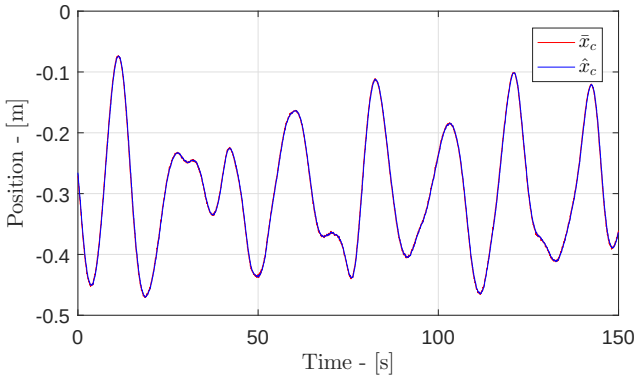

Figure 10: Positional comparison in the x-direction.

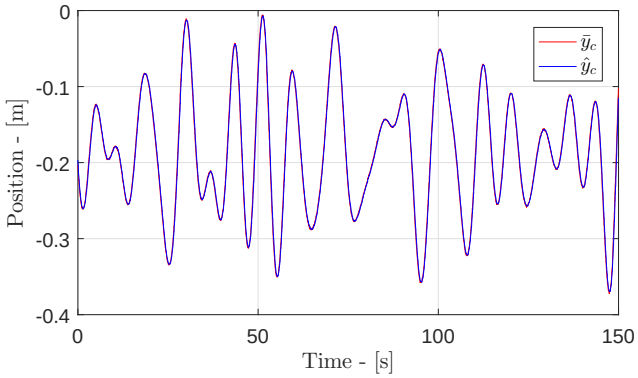

Figure 11: Positional comparison in the y-direction.

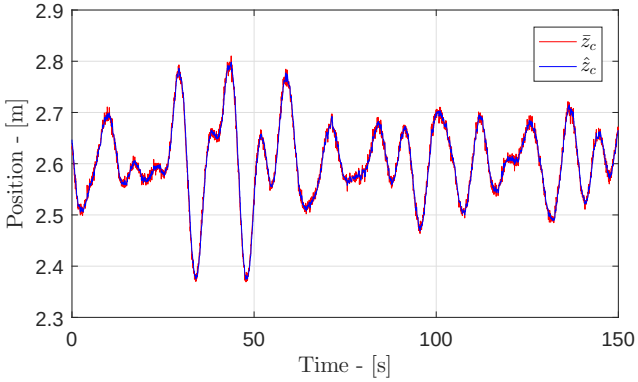

Figure 12: Positional comparison in the z-direction.

As seen from the figures, it is clear that both the x- and y-positional measurements are significantly more precise than the z-position measurements. This result is expected since the measurement accuracy in computer vision systems depends on the image pixel density. Therefore, the z-position contains more noise since small changes in the z-direction will cause the marker edges to move across more pixels in the image if compared to x- and y-movements. The image resolution used in our experiments was set to 960×540 pixels since this resolution gave an acceptable update rate at 65ms and sufficient measurement accuracy. The filter performance for the orientation quaternion is also analyzed in the same way as for the positions, and is shown in Figures 13, 14, 15 and 16

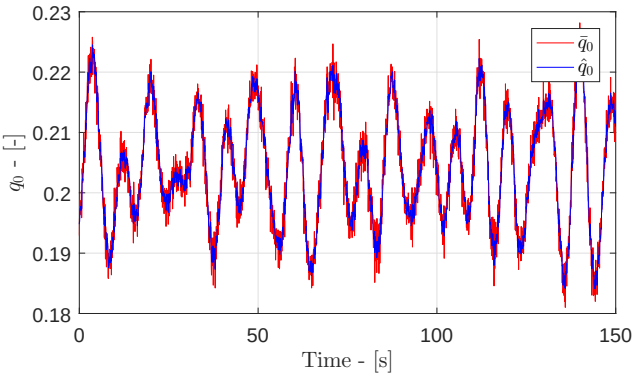

Figure 13: EQKF performance for quaternion coefficient q_0.

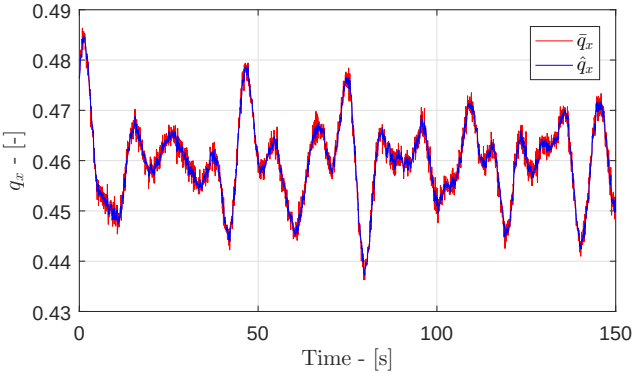

Figure 14: EQKF performance for quaternion coefficient q_x.

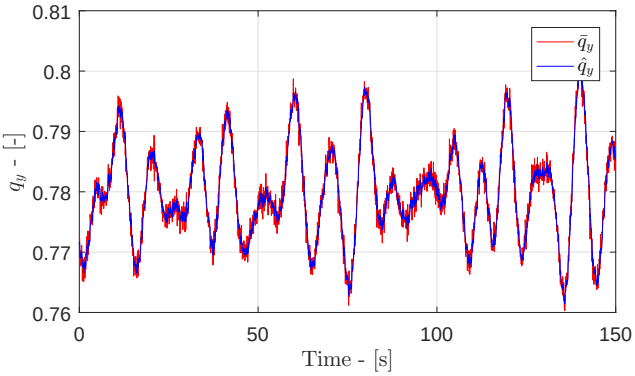

Figure 15: EQKF performance for quaternion coefficient q_y.

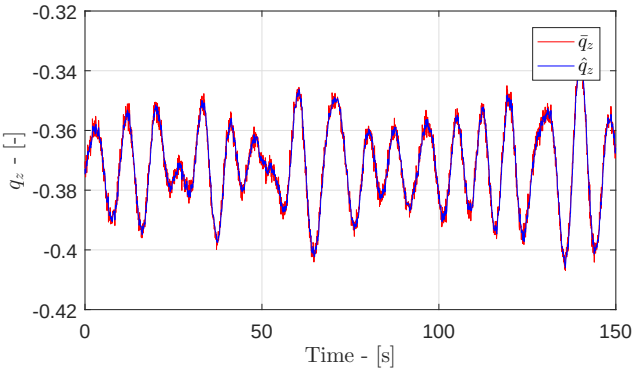

Figure 16: EQKF performance for quaternion coefficient q_z.

By analyzing the curves representing the averaged quaternion $\bar{q}$ and the estimated quaternion $\hat{q}$ it is clear that also the orientation contains more noise due to the somewhat low resolution used in the proposed method. A higher camera resolution would give more accurate measurements for both the z-position and the orientation quaternion, but the resulting slower update rate would reduce the overall performance. However, an FPGA implementation of the vision tracking algorithm could result in both high measurement resolution and fast computation.

5.2 Sensor Fusion Performance

The sensor fusion algorithm presented in Section 4 heavily depends on the vision tracking of the Aruco cube. It is therefore interesting to investigate how much of this noise it is possible to remove using the proposed sensor fusion algorithm. To validate and investigate the sensor fusion performance, the estimated measurements and the unfiltered measurements are compared in Figures 17, 18, 19 and 20.

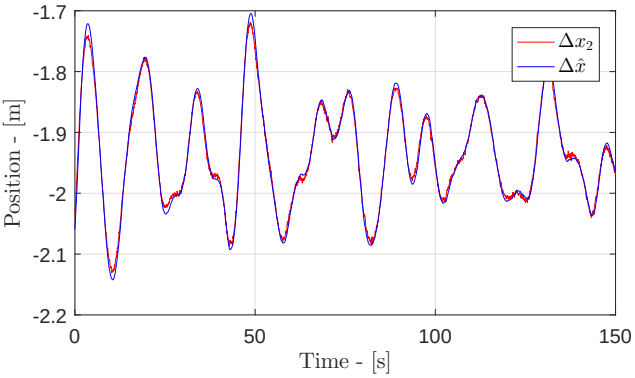

Figure 17: Sensor fusion performance in x-direction.

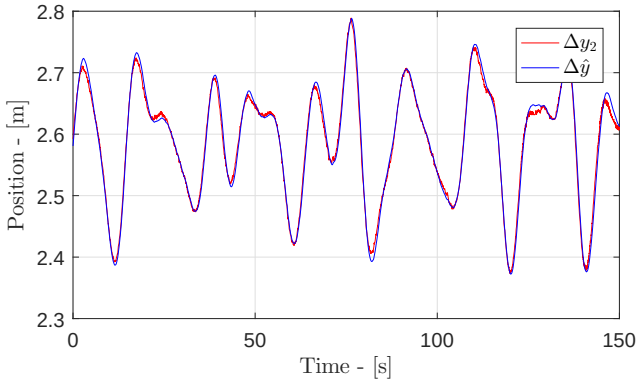

Figure 18: Sensor fusion performance in y-direction.

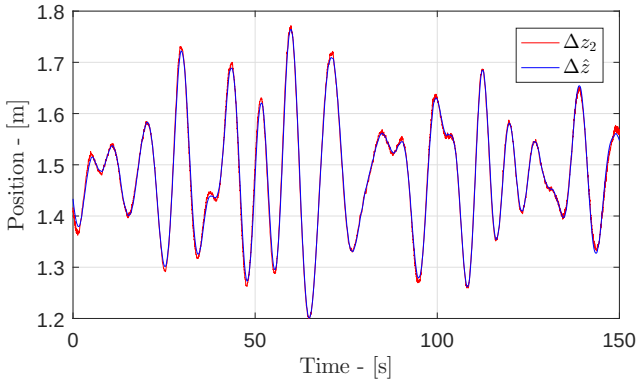

Figure 19: Sensor fusion performance in z-direction.

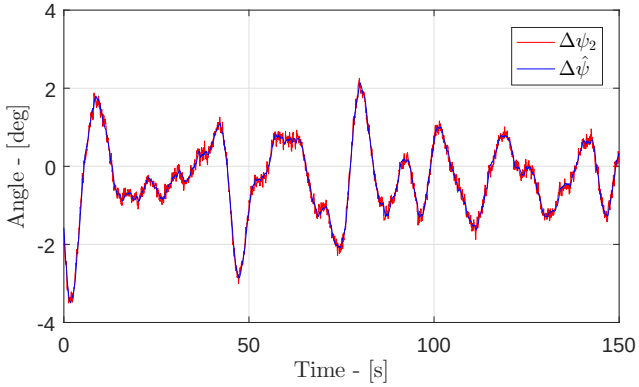

Figure 20: Sensor fusion performance in heading-direction.

5.3 Compensation Performance

The final goal of the proposed method is to see whether the Aruco box could be placed anywhere on the secondary vessel ship deck, where the only constraint is that the body-fixed z-axis of the secondary MRU sensor is pointing towards the vessel ship deck. Using the

resulting measurements, the robot end-effector should be kept in a given height above the plane spanned by the ship deck. The Aruco cube was therefore placed in three different locations in order to verify that the proposed solution was actually capable of self-calibrating. The robot end-effector was then controlled to carry out the VVMC task based on the resulting sensor fusion measurements. The three different test locations are presented in Figures 21, 22 and 23.

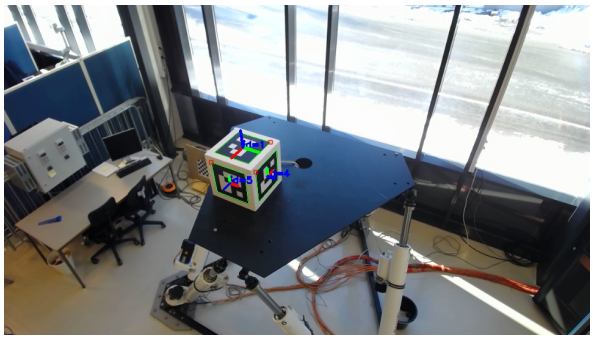

Figure 21: Experimental location number 1 seen from the camera.

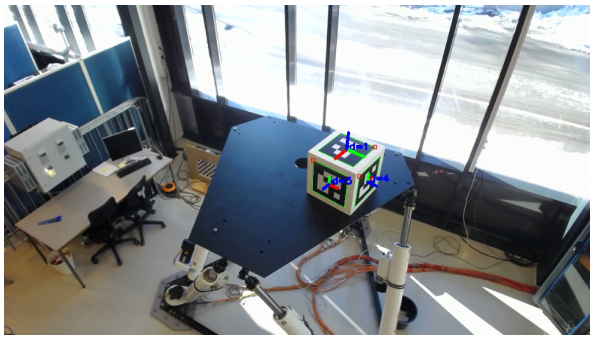

Figure 22: Experimental location number 2 seen from the camera.

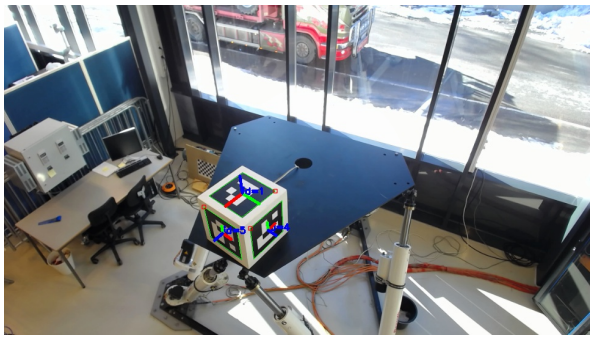

Figure 23: Experimental location number 3 seen from the camera.

A set of resulting curves is used to indicate how accurate the overall compensation task is, given that the Aruco box was placed in three random positions. The resulting curves are based on the feedback measurements taken from the robot and the two Stewart platforms, where the equipment is calibrated using a high precision FARO laser tracker featuring sub-millimeter precision. The resulting overall compensation performance is presented in Figures 24, 25 and 26.

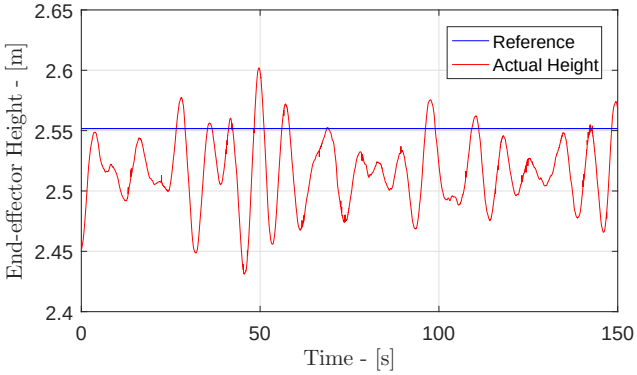

Figure 24: Experimental location 1.

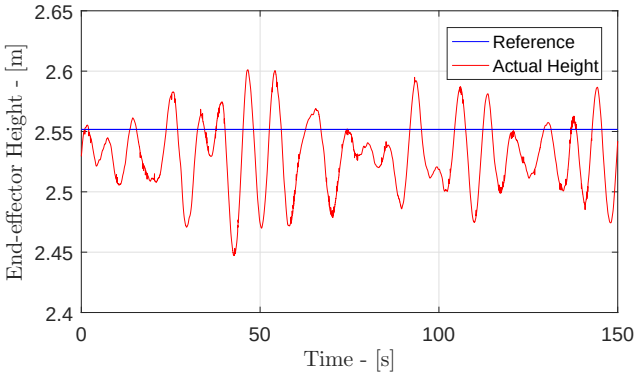

Figure 25: Experimental location 2.

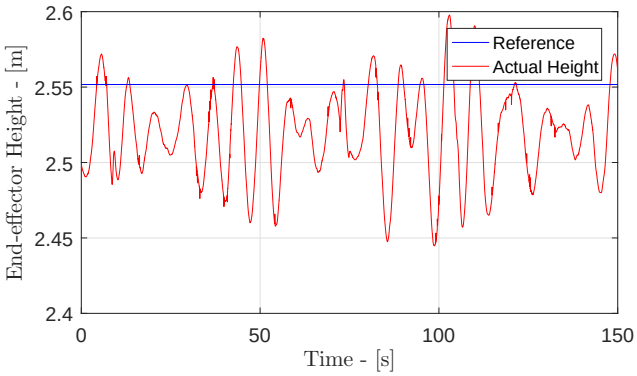

Figure 26: Experimental location 3.

The resulting curves indicate that the three different experimental locations of the Aruco cube give somewhat the same result in terms of overall compensation accuracy. Anyway, a more quantitative measure of the overall compensation accuracy is given in Table 1, where the accuracy is summarized in terms of RMS error, absolute maximum deviation, standard deviation and mean deviation.

Table 1: Compensation error.

Location	e_{RMS} [m]	$\lvert e\rvert_{\text{MAX}}$ [m]	σ_e [m]	μ_e [m]
1	0.046	0.121	0.030	0.036
2	0.037	0.105	0.030	0.021
3	0.044	0.107	0.031	0.032

From the table, it can be seen that the standard deviation is almost the same for all three test locations, while the mean deviation is indicating that there is a constant offset which is much likely to originate from some imperfections in the calibration matrices H_b^1 used to describe the offset between MRU1 and the robot base. This calibration matrix was measured by hand, and is therefore more likely to contain some errors. In future work, a more sophisticated method for determining these calibration parameters can be utilized, such as the one presented by Mirzaei and Roumeliotis (2008) for instance.

6 Discussion and Conclusions

In this paper, a sensor fusion algorithm capable of measuring all 6 DOF between two floating vessels is investigated. The proposed method utilized two MRU sensors and a camera vision system to track the relative motions in real-time. The camera vision system using an Aruco cube was able to provide an absolute measurement between the body-fixed coordinates of the two MRUs. This information was useful in terms of: ability to self-calibrate the motion tracking system, remove drift in the MRU measurements, and provide a visual feedback to the operator through the augmented reality indicating the detected coordinate system of the Aruco cube. The overall accuracy for all three test locations resulted in a maximum standard deviation of 31 mm. In addition, the sensor fusion algorithm demonstrated the ability to self-calibrate since the Aruco cube was placed in three different locations onto the secondary Stewart platform. However, it is observed that, the results suffer from a constant mean error of roughly 30 mm, which may indicate that some small errors might occur in the calibration parameters such as MRU1's location relative to camera, for instance. In addition to the calibration error, the overall time-delay of approximately 65 ms due to the camera processing is also contributing to the overall compensation error.

Future work in improving the relative motion tracking between two vessels should focus on reducing the overall time-delay in the camera processing algorithm and use a higher camera resolution for higher accuracy. Likewise, the proposed sensor fusion algorithm should be extended to also estimate the MRU drift in addition to only estimate the error between the MRUs and the camera vision system. This might enable for motion compensation even when the camera is not capable of detecting the Aruco cube for some small time periods.

Appendix

$$K = \begin{bmatrix} 582.64590 & 0 & 457.91989 \\ 0 & 581.19390 & 274.12724 \\ 0 & 0 & 1 \end{bmatrix}$$

Filter Description	σ_q	σ_r
Camera Vision EQKF	0.05	0.05
Sensor fusion error KF	0.0001	0.0001

Acknowledgments

The research presented in this paper has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

References

Baillot, Y., Julier, S. J., Brown, D., and Livingston, M. A. A tracker alignment framework for augmented reality. *Proceedings - 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2003*, 2003. pages 142–150. doi:10.1109/ISMAR.2003.1240697.

Braden, B. The Surveyor's Area Formula. *The College Mathematics Journal*, 1986. 17(4):326–337. doi:10.2307/2686282.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 2014. 47(6):2280–2292. doi:10.1016/j.patcog.2014.01.005.

Guennebaud, G. and Jacob, B. Eigen v3. 2010. URL http://eigen.tuxfamily.org.

Itseez. Open Source Computer Vision Library. 2015. URL https://github.com/itseez/opencv, doi:10.1016/j.cell.2011.11.001.

Kjelland, M. B. *Offshore Wind Turbine Access Using Knuckle Boom Cranes*. Ph.D. thesis, 2016.

Kraft, E. A Quaternion-base Unscented Kalman Filter for Orientation Tracking. *Proceedings of the Sixth International Conference of Information Fusion*, 2003. 1(1):47–54. doi:10.1109/ICIF.2003.177425.

Küchler, S., Eberharter, J. K., Langer, K., Schneider, K., and Sawodny, O. Heave motion estimation of a vessel using acceleration measurements. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2011a. 18(PART 1):14742–14747. doi:10.3182/20110828-6-IT-1002.01935.

Küchler, S., Pregizer, C., Eberharter, J. K., Schneider, K., and Sawodny, O. Real-Time Estimation of a Ship's Attitude. *IEEE American Control Conference*, 2011b. pages 2411–2416.

Marins, J., Yun, X. Y., Bachmann, E. R., McGhee, R. B., and Zyda, M. J. An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, 2001. 4:2003–2011. doi:10.1109/IROS.2001.976367.

Markley, F. L., Cheng, Y., Crassidis, J. L., and Oshman, Y. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 2007. 30(4):1193–1197. doi:10.2514/1.28949.

Mirzaei, F. M. and Roumeliotis, S. I. A Kalman Filter-based Algorithm for IMU-Camera Calibration. *IEEE Transactions on Robotics and Automation*, 2008. 25(4):1143–1156. doi:10.1109/IROS.2007.4399342.

Park, F. C. and Martin, B. J. Robot Sensor Calibration: Solving AX = XB on the Euclidean Group. *IEEE Transactions on Robotics and Automation*, 1994. 10(5):717–721. doi:10.1109/70.326576.

Pawlus, W., Kandukuri, S. T., Hovland, G., Choux, M., and Hansen, M. R. EKF-based estimation and control of electric drivetrain in offshore pipe racking machine. *Proceedings of the IEEE International Conference on Industrial Technology*, 2016. 2016-May:153–158. doi:10.1109/ICIT.2016.7474742.

Pierson, W. J. and Moskowitz, L. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. *Journal of Geophysical Research*, 1964. 69(24):5181–5190. doi:10.1029/JZ069i024p05181.

Richter, M., Schneider, K., Walser, D., and Sawodny, O. Real-Time Heave Motion Estimation Using Adaptive Filtering Techniques. *The International Federation of Automatic Control (IFAC) World Congress.*, 2014. 19(1):10119–10125. doi:10.3182/20140824-6-ZA-1003.00111.

Sasiadek, J. and Hartana, P. Sensor data fusion using Kalman filter. *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, 2000. 2:941–952. doi:10.1109/IFIC.2000.859866.

Tørdal, S. S., Løvsland, P.-O., and Hovland, G. Testing of Wireless Sensor Performance in Vessel-to-Vessel Motion Compensation. *42st Annual Conference of the IEEE Industrial Electronics Society*, 2016. doi:10.1109/IECON.2016.7793951.