Journal of Marine Science and Technology

Volume 29 | Issue 4 Article 2

Integrated Simulation of Virtual Prototypes and Control Algorithms of Unmanned Surface Vehicles Based on a Robot Operating System

Hye-Won Lee
Research Institute of Marine Systems Engineering, Seoul National University, Seoul, Republic of Korea

Joo-Hyun Woo
Department of Naval Architecture and Ocean System Engineering, Korea Maritime and Ocean University, Republic of Korea

Myung-Il Roh
Seoul National University, miroh@snu.ac.kr

Seung-Ho Ham
School of Industrial and Naval Architecture, Changwon National University, Seoul, Republic of Korea

Luman Zhao
Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, Norway

See next page for additional authors

Integrated Simulation of Virtual Prototypes and Control Algorithms of Unmanned Surface Vehicles Based on a Robot Operating System

Authors

Hye-Won Lee, Joo-Hyun Woo, Myung-Il Roh, Seung-Ho Ham, Luman Zhao, Sol Ha, Nak-Wan Kim, and Chan-Woo Yu

RESEARCH ARTICLE

Integrated Simulation of Virtual Prototypes and Control Algorithms of Unmanned Surface Vehicles Based on a Robot Operating System

Hye-Won Lee [a], Joo-Hyun Woo [b], Myung-Il Roh [c,*], Seung-Ho Ham [d], Luman Zhao [e], Sol Ha [f], Nak-Wan Kim [g], Chan-Woo Yu [h]

[a] Research Institute of Marine Systems Engineering, Seoul National University, Seoul, Republic of Korea
[b] Department of Naval Architecture and Ocean System Engineering, Korea Maritime and Ocean University, Republic of Korea
[c] Department of the Naval Architecture and Ocean Engineering, and Research Institute of Marine Systems Engineering, Seoul National University, Seoul, Republic of Korea
[d] School of Industrial and Naval Architecture, Changwon National University, Seoul, Republic of Korea
[e] Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, Norway
[f] Department of Naval Architecture and Ocean Engineering, Mokpo National University, Republic of Korea
[g] Research Institute of Marine Systems Engineering, Seoul National University, Republic of Korea
[h] Agency for Defense Development, Republic of Korea

Abstract

Unmanned surface vehicles (USVs) require autonomous software with a high level of liberalization and automation to complete various missions. Therefore, experimental verification and validation of this software at the initial design stage are essential. However, such experiments are impossible before the actual hardware is developed, and the creation of an external environment for the mission requires substantial cost and time. In this study, virtual prototypes of the mission environment and hardware for the USV are developed. Then, a simulation environment for testing the control algorithms in the autonomous software is constructed. Data communication with the USV hardware is necessary for the autonomous software to acquire information from the sensors and to operate the actuators. Similarly, data communication between the programs of the virtual prototypes and the autonomous software is required for the integrated simulation environment. In this study, the robot operating system (ROS) software platform is adopted to construct the interface for this data communication. Finally, the integrated simulation environment of the control algorithms and the virtual prototypes are constructed based on ROS to verify the autonomous software of the USV. The applicability of the suggested simulation environment is evaluated by application to three scenarios: mine detection, path following, and port entry.

Keywords: Unmanned surface vehicle (USV), Autonomous software, Virtual prototype, Integrated simulation, Control algorithm

1. Introduction

As the patterns of warfare change, the applications of unmanned war systems (UWSs) are increasing considerably. The use of UWSs has many advantages, such as the reduction of human risk or loss and operational costs. Moreover, UWSs can be integrated with existing manned war systems to sustain monitoring mission performance. The classification of UWSs is shown in Fig. 1.

UWSs can be classified into three categories: unmanned ground systems (UGSs), unmanned aerial systems (UASs), and unmanned maritime systems (UMSs). UMSs, which perform maritime missions, are divided into unmanned underwater vehicles (UUVs) and unmanned surface vehicles (USVs)

Received 10 October 2019; revised 30 November 2020; accepted 9 December 2020.
Available online 3 September 2021.

* Corresponding author. Department of the Naval Architecture and Ocean Engineering, and Research Institute of Marine Systems Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 08826, Republic of Korea.
E-mail address: miroh@snu.ac.kr (M.-I. Roh).

https://doi.org/10.514/2709-6998.1583

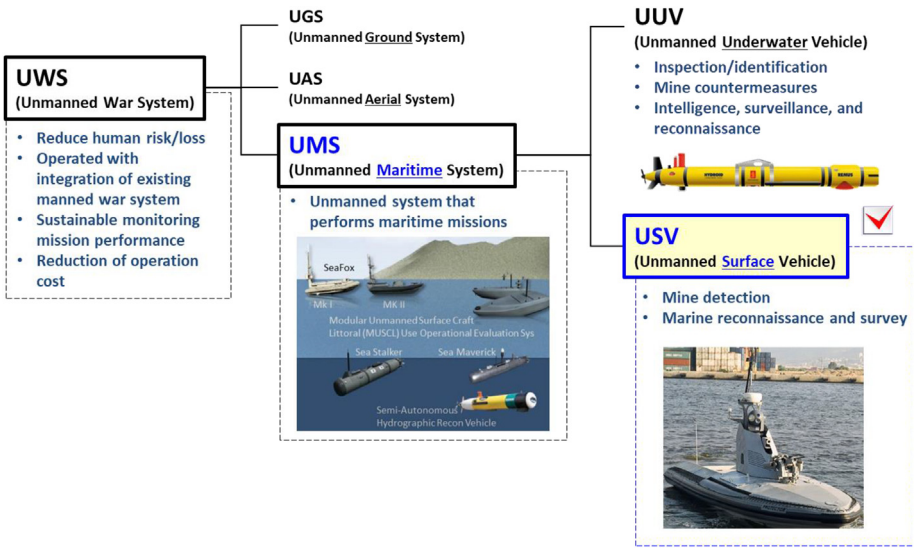

Fig. 1. Classification of UWSs.

according to whether the operation area is underwater or on the sea surface. The main mission of USVs is mine detection, marine reconnaissance, and surveillance. USV is the target of this study.

The autonomous software mounted in a USV is very important for unmanned operation. Because the USV does not carry people on board, the autonomous software must control the process of recognition, judgment, and command. Therefore, testing and validation for high accuracy and robustness are indispensable for the development of the autonomous software mounted in USVs. The main procedure used to develop this software is summarized in Fig. 2.

Firstly, the requirements for the USVs are derived from missions, environmental conditions in the operational area, and previous development. Based on these requirements, several control algorithms are developed to perform missions such as mine detection, autonomous port entry, and path following. The next step is to install the control algorithm in a real USV. Then, tests of the control algorithm are conducted at sea. After the tests, feedback is provided to the developer for the modification of the control algorithms.

This general procedure has three problems. The first problem is that testing the control algorithm is impossible before the hardware of the USV (such as the hull, propeller, and engine) is constructed. The second problem is that testing all cases according to the environmental conditions and mine locations is not sufficient. The third problem is that testing at sea requires considerable time and cost. Therefore, a virtual prototype of the USV and replication of the test environment can solve these problems. The virtual prototype of the USV can test the control algorithm any time and anywhere without restrictions and generate all cases to be tested. Moreover, it does not require much time and cost.

In this study, we propose a virtual prototype of the USV for three scenarios (path following, mine detection, and port entry), which are shown in Fig. 3.

The path following scenario involves following a given path despite disturbances in the motion of the USV by external forces. The mine detection scenario requires finding mines in the image obtained from the side-scan sonar (SSS). The port entry scenario involves entering a port by estimating the position and orientation of the USV from the camera image.

The key to successfully conducting these scenarios is determining how to make the virtual prototype of the USV realistic. The USV itself contains various sensors to obtain data from its surroundings. Data obtained from the sensors must be transferred to the module containing the control algorithms to be analyzed. Based on the analysis results, the USV decides to give a command to control a rudder or a propeller. In other words, the data should be exchanged between the virtual prototype of the USV and the control algorithms through the network interface. Therefore, we adopt the robot operating system (ROS), which is widely used to develop hardware and control algorithms rapidly. The ROS is a framework and a set of tools that provide the functionality of an operating system. ROS provides

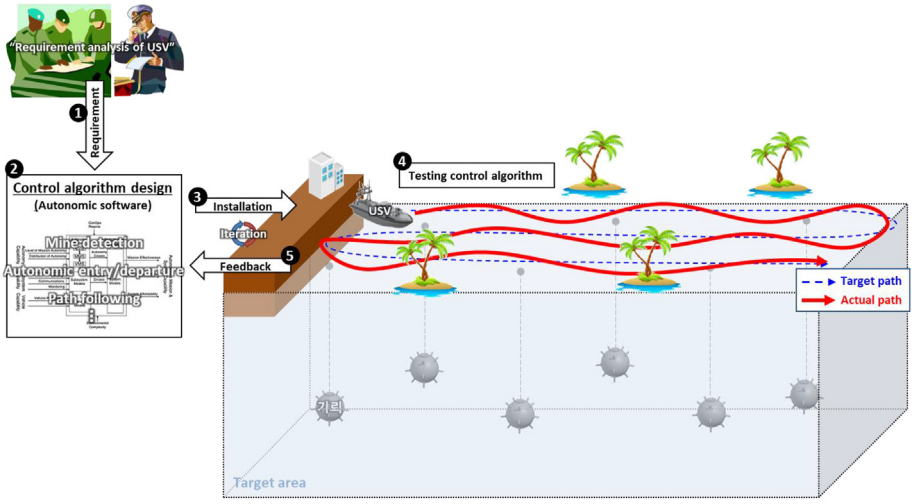

Fig. 2. Development of autonomous software used in USVs.

functionality for hardware, device drivers, communication between processes over multiple machines, and much more [1]. Therefore, the ROS allows us to easily integrate the control algorithms with the virtual prototype of the USV and with the real hardware in the USV (see Fig. 4).

2. Related works

There are plenty of works that performed dynamics modeling and simulation of UMSs for various missions. Heo et al. [2] developed a virtual experimental environment based on modeling and simulation for USVs. The integrated mission planning system was designed with a multi-agent process. The virtual prototype of the USV model was developed, and the mission scenario for the path following was performed. Similarly, Kim et al. [3] studied a simulation-based approach for the highly autonomous multi-agent architecture. In their study, object identification, state estimation, mission planning, and decision were performed for the mine search mission. Although the modeling and simulation of the USVs in the existing studies have been developed at a high-level, including situation awareness and task planning, the virtual prototypes of the environment such as SSS were not presented, and the mission scenarios were limited to mine searching.

Meanwhile, considering an extension to actual hardware, several studies have applied the ROS to integrate the systems of the UMS. Demarco et al. [4] developed an integrated development system for the Yellowfin autonomous underwater vehicle (AUV) that integrates the low-level controller simulation, mission planning, and mission execution processes. The ROS was implemented to interface with several individual communication systems. In the underwater robotics community, the Mission Oriented Operating Suite (MOOS) has been widely used for communication between processes. However, Demarco et al. [4] integrated the ROS with the MOOS by building the MOOS/ROS bridge and focused on the advantage of using the ROS, namely, the flexibility in organizing libraries in the ROS build system. Cashmore et al. [5] constructed a system to find a suitable plan for a single inspection tour of the AUV that optimizes the time required to complete the mission. The mission planning and execution procedure were integrated using the ROS framework. The suggested system was validated through physical trials performed in a large underwater tank. Mendonca et al. [6] developed a multi-robot simulator for water-surface and aerial vehicles, Kelpie. Kelpie is fully compliant with the ROS, which provides standard operating-system services, enabling a distributed computing development framework. Because Kelpie is a node of the ROS, other ROS nodes—such as virtual robots, actuators, and sensors—can interact with Kelpie. Conte et al. [7] developed a multi-agent navigation, guidance, and control (NGC) system structure designed for a low-cost autonomous surface vehicle. The individual agents were implemented using the ROS, which provides the high computational capability and easy interfacing with sensors and actuators. Zhao et al. [8] adopted the ROS as an interface to formulate a HILS (Hardware-In-the-Loop Simulation) for the anti-heave control system of an offshore support vessel. An integrated simulation environment was constructed by implementing the data communication interface of the ROS for the

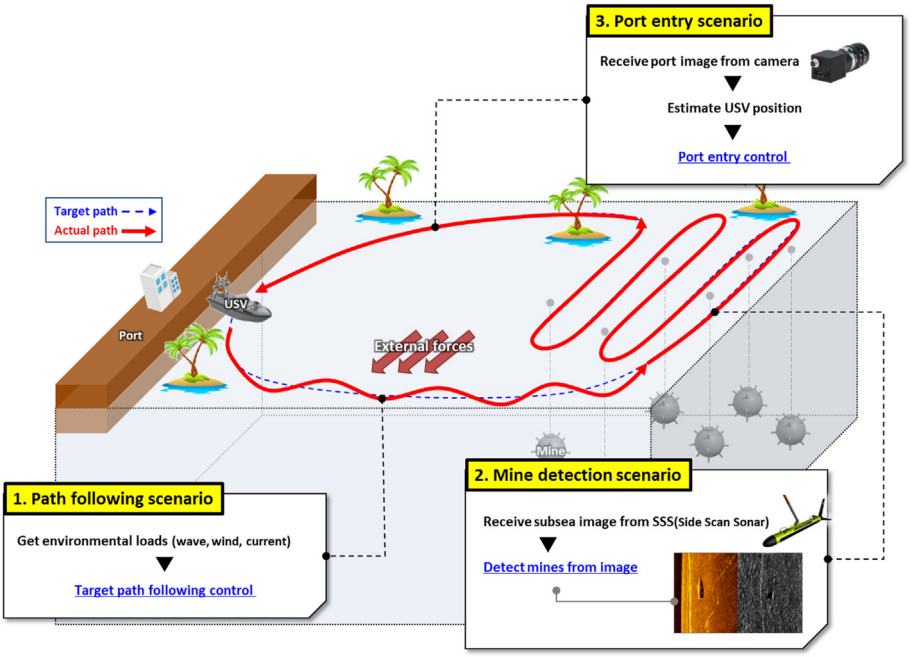

Fig. 3. Three scenarios used for the virtual prototype of the USV.

communication between the control system and the HIL simulator. Recently, the ROS was used in the collaboration system and the framework of an integrated simulation environment for heterogeneous unmanned vehicles by Kim and Lee [9]. In their study, the ROS was integrated with Pixhawk, which is actively used for unmanned vehicles and robot development, for the construction of cooperative system of USVs carrying out illegal fishing vessel capture mission.

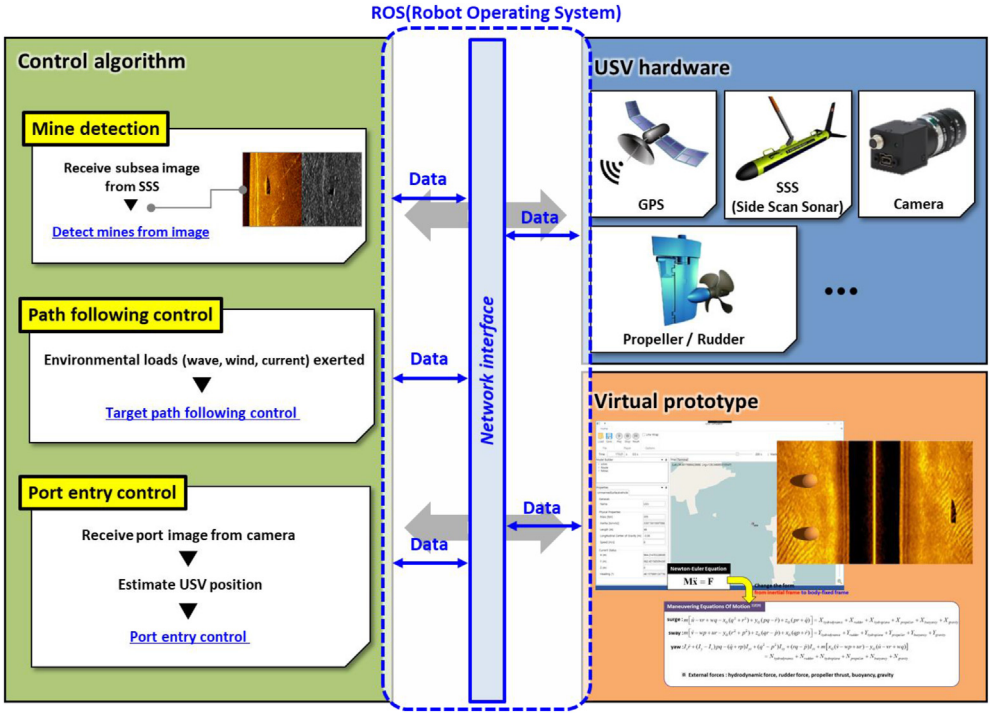

Fig. 4. Integration of the control algorithm and the USV by using the ROS.

The previous works implementing the ROS to the UMS have been mostly applied to the actual experiments to validate the on-board control algorithms and the integrated systems. However, since it is difficult to test the actual USVs with the autonomic software, the virtual prototypes of the environment and USV are still necessary. This study defined the mission scenarios that USVs perform into three; mine detection, path following, and port entry scenarios. The simulation management and control algorithms for the USV were developed, and the virtual prototypes of the environment and the sensors of USV, such as SSS and camera, were constructed. The ROS was implemented for the communication between modules, with the actual sensors and actuators of the USV, and with additional control algorithms for other missions. Finally, the integrated system framework was constructed to validate the control algorithms for the simulation and actual operation.

3. System configuration

3.1. Overall configuration

The integrated system, which includes the simulation management, virtual prototype, and the control algorithm, can be divided into several modules according to their functions, as shown in Fig. 5.

The scenario management and display modules help construct the simulation environment and the scenario and manage the whole simulation. The virtual prototype represents the sensors and actuators of the USV. The control algorithm is mounted on the autonomous software of the USV and gives a command to the USV to execute the assigned mission. Each module interacts with the others by transferring data through the ROS interface.

3.2. Scenarios

In each scenario, the corresponding modules transmit and receive data for communication.

3.2.1. Path following

In the path following scenario, the USV follows the target path under the given environmental conditions. For this purpose, the path following control algorithm mounted on the USV gives commands to the propeller and the rudder. Fig. 6 shows the configuration of the active modules in this scenario.

The motion analysis module is the virtual prototype of the USV. It includes the global positioning system (GPS), which is the sensor, and the propeller and rudder, which are the actuators. The scenario management module gives the information of the USV, the scenario, and the environmental conditions (such as the target path, wave, and current) to the motion analysis module. Then, the virtual prototype of the USV transmits the current position to the path following control module in the autonomous software of the USV. As a result of the algorithm, the control command is transferred to the virtual propeller and the rudder of the USV. The

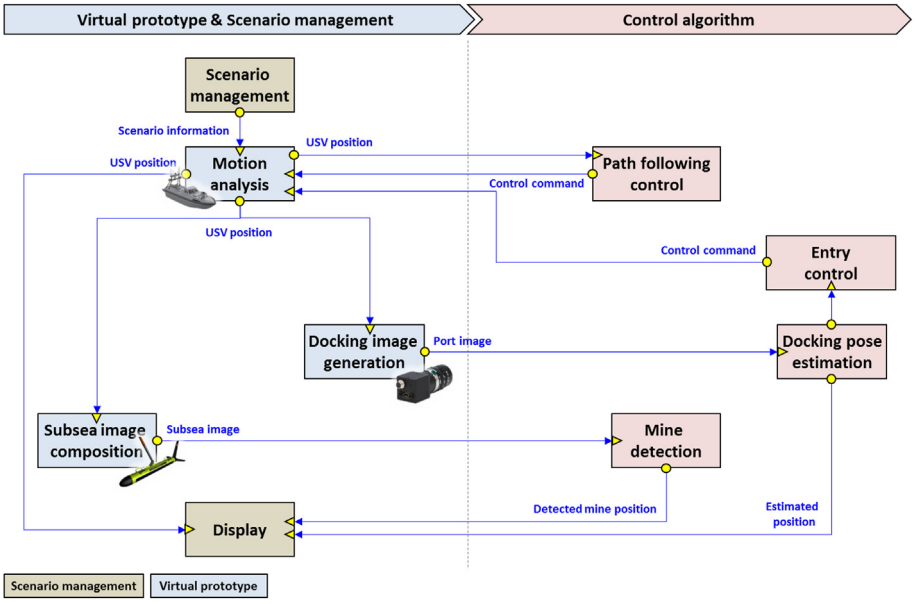

Fig. 5. The overall configuration of the virtual prototype, scenario management, and control algorithm.

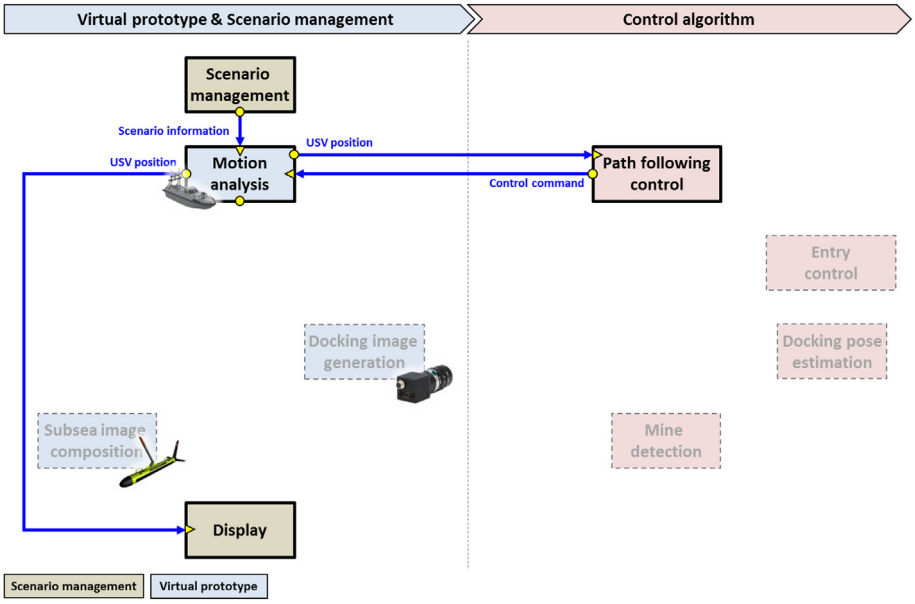

Fig. 6. The configuration of the modules in the path following scenario.

actual path that the USV follows is displayed in the display module so that the user can see the result.

3.2.2. Mine detection

Mine detection using SSS is one of the major missions that USV performs. The SSS is a sonar system that uses sound propagation to detect objects on the seabed. In the mine detection scenario, the SSS is towed by the USV and generates the underwater image that contains the mine. The modules and the transferred data in the mine detection scenario are presented in Fig. 7.

In this scenario, the virtual prototype of the SSS—the subsea image composition module—is included. The scenario management module provides the information of the mines that are scattered in the mission area to the subsea image composition module. Then, it generates the subsea image based on this information and the position of the USV. The subsea image is then transferred to the mine detection module, which is the control algorithm. The mine detection module detects the mines in the received image, and this information is transmitted to the display module.

3.2.3. Port entry

When the USV enters a port, detailed control is required because the location given by the GPS is not sufficiently accurate for port entry and contains errors. In the port entry scenario, the camera mounted on the USV is used to recognize the exact distance from the port. The entry control algorithm then gives a command to the propeller and the rudder to make the USV stop at the target point. Fig. 8 shows the configuration of the modules used in the port entry scenario.

The docking image generation module is the virtual prototype of the camera on the USV. It receives information about the port from the scenario management module and the position of the USV from the motion analysis module. Then, it generates the virtual image of the port seen by the camera. Based on the docking image, the control algorithms estimate the relative position of the USV and the port and control the USV the target point in the port.

3.3. Virtual prototypes

The control algorithms in the autonomous software of the USV are essential because they – instead of people on board – give commands to the USV. Therefore, testing and validation of the developed control algorithms with actual hardware are required before the installation of the control algorithms.

3.3.1. Scenario management

The scenario management and display modules manage the information of the virtual scenario for the simulation. They construct the virtual environment of the mission area, execute the simulation, and display the result.

The scenario management module creates the USV units and assigns specific missions to them. The main information that is necessary to create the scenarios is shown in Table 1.

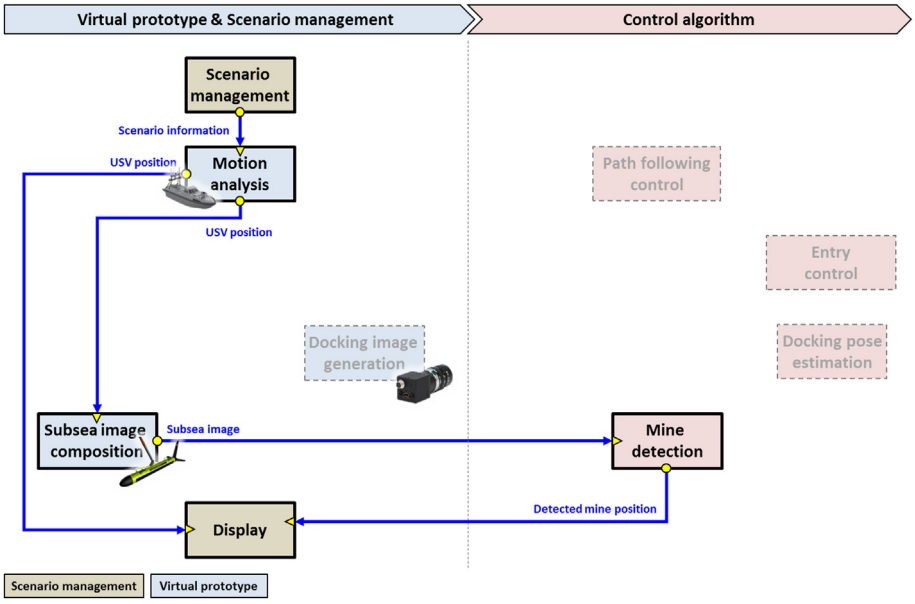

Fig. 7. The configuration of the modules in the mine detection scenario.

For all scenarios, the USV and the mission area should be defined. The USV can be created with given physical and rudder properties. In the mine detection scenario, several USVs can operate together along different paths, as shown in Fig. 9. The mission area is defined as a region that contains mines. For a given number of mines, the mines are located randomly in the mission area. The target path that the USV follows is generated automatically in a mine detection scenario with a given heading angle and the search area of the SSS. In other scenarios, the user can add the waypoints of the target path. In the path following scenario, the environmental conditions due to the current can be considered. When the simulation is executed, the above information is transmitted to the other modules.

The scenario management module also manages the execution of the simulation. It can communicate with the other modules through the ROS interface. Thus, the module can start, pause, and stop the simulation, sending the signal to the other modules. The simulation execution information includes the simulation time and the time step.

Lastly, the display module shows the current conditions on a two-dimensional map for convenience. As shown in Fig. 10, the USV, mission area, mines, and target path of the scenario are displayed on the map. During the simulation, the display module receives the current condition of the USV, the subsea image generated by the SSS, and the detected mines. With this information, the user can see the result of the simulation and evaluate the performance of the control algorithms.

The graphical user interface (GUI) of the scenario management and display modules is presented in Fig. 10. The user interface consists of five parts: the menu, progress bar, model builder, property view, and visualization.

3.3.2. *Motion analysis*

The motion analysis module represents the sensors and the actuators of the USV itself. The sensor (the GPS) provides the location of the USV, and the actuators, propeller, and rudder control the motion. In the engineering model, the motion of the USV is based on the maneuvering equations of motion in the horizontal plane, which have three degrees of freedom (DOF). The maneuvering equations of motion suggested by Gertler and Hagen [10] are formulated as in Fig. 11. In this study, the equations for the surge, sway, and yaw motion used in the six-DOF maneuvering equations of motion of Gertler and Hagen [10] are implemented. The motion of the surface ship can be represented by simplifying the six-DOF equations, reducing three-DOF (roll, pitch, and heave) from six-DOF [11].

The motion analysis module calculates the position and the heading angle of the USV based on the physical properties received by the scenario management module and the control command from the path following control module. The obtained position and the heading angle are transmitted to the other modules of virtual prototypes and the control algorithms.

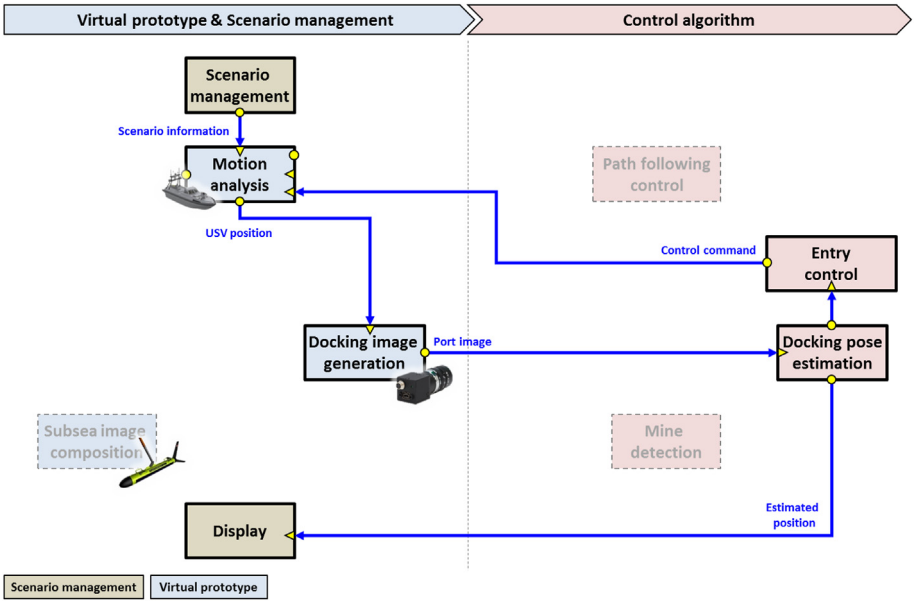

Fig. 8. The configuration of the modules used in the port entry scenario.

Table 1. Information for scenario management.

Object	Property
USV	Physical properties (inertia, length, mass, search radius) Rudder properties (initial angle, limit)
Mission area	Region, number of mines, port information (location, angle)
Path	Waypoints
Mine	Position, angle
Environment	Current property (speed, heading)

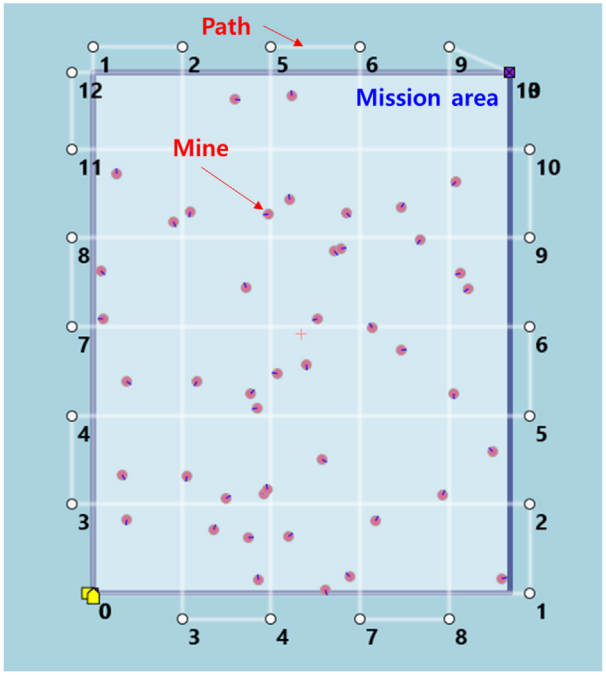

Fig. 9. An example of the mine detection scenario using two USVs.

3.3.3. Subsea image composition

The subsea image composition module aims to synthesize the subsea image detected by the SSS, which is mounted on the USV, and transmit the image to the mine detection module in real-time. The subsea image is generated according to the relative position and direction of the SSS and the mines, which are received from the scenario management module.

The SSS towed by the USV uses a wide-angle to emit sound frequencies at right angles to detect mines on the sea bed. The SSS emits acoustic signals and then receives the returning echoes when the signals arrive at the sea bed. When the SSS is reset, it emits sound again, and the cycle continues. The acoustic signals are converted to electrical signals and displayed on a monitor in real-time. The subsea image shown on the monitor is adjusted to different color schemes based on the subsea appearance. For example, if mines are mounted close to the SSS, they will provide fast-return echoes and produce stronger reflected acoustic signals than the surrounding flat sea bed. In this instance, the raised

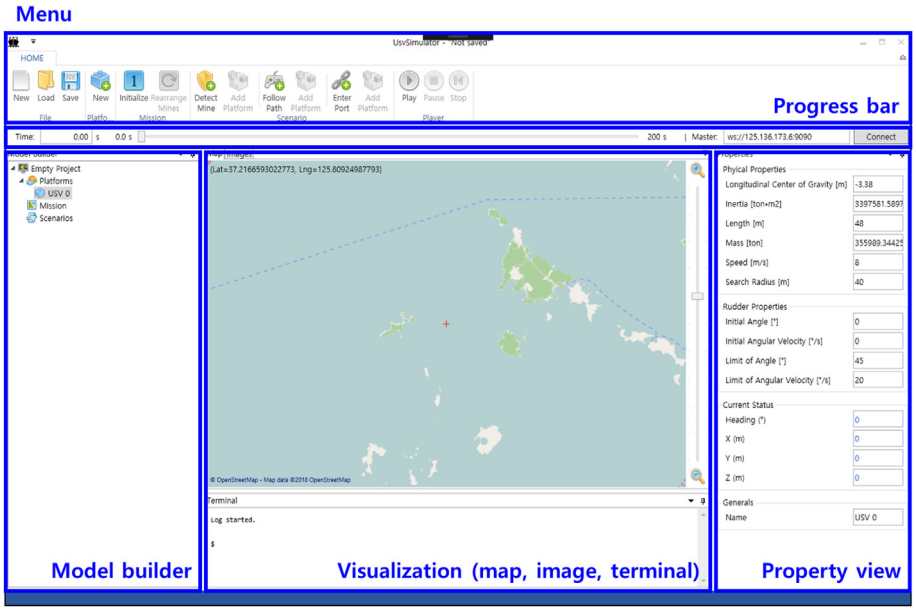

Fig. 10. GUI of the scenario management and display modules.

body (mines) will be easily detected as a brighter color in the image compared with the surrounding flat sea bed.

The right side of Fig. 12 shows an image in which the acoustic signals are propagated out of both sides of the SSS. When synthesizing the subsea image with mines, we applied a hybrid technique to synthesize the highlighted and shadow regions that can imitate the effect of the virtual mines on the real subsea background image. The synthesis of the subsea image is performed in three stages: firstly, when the SSS emits signals on a mine (Fig. 12 ①), the mine is projected on the seabed. Secondly, the shape of the mine in the section view is assumed to be a circle (Fig. 12 ②). Based on the information including the relative position and direction θ of the SSS and the mine, the distance of the SSS to the seabed H, the position and dimensions of the mine (length L and diameter D), and the positions of the highlighted region (A−B) and the shadow region (B−C) on the original subsea image can be calculated. Thirdly, to represent the highlighted and shadow region of the mine on the original subsea image, the brightness of the pixels of the highlighted region and the shadow region should be adjusted (Fig. 12 ③). Because the pixel value ranges from 0 to 255, where zero is black, and 255 is white, the pixel value can be increased in the shadow region and decreased in the highlighted region.

Consequently, as shown in Fig. 13, the subsea image with the virtual mines is synthesized.

3.3.4. Docking image generation

The docking image generation module is designed to produce a virtual monocular camera image while the USV enters the port. In the port

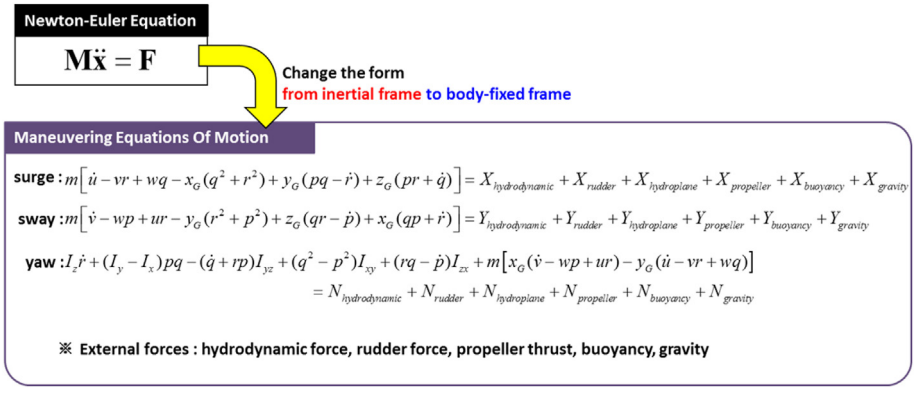

Fig. 11. Three-DOF maneuvering equations of motion.

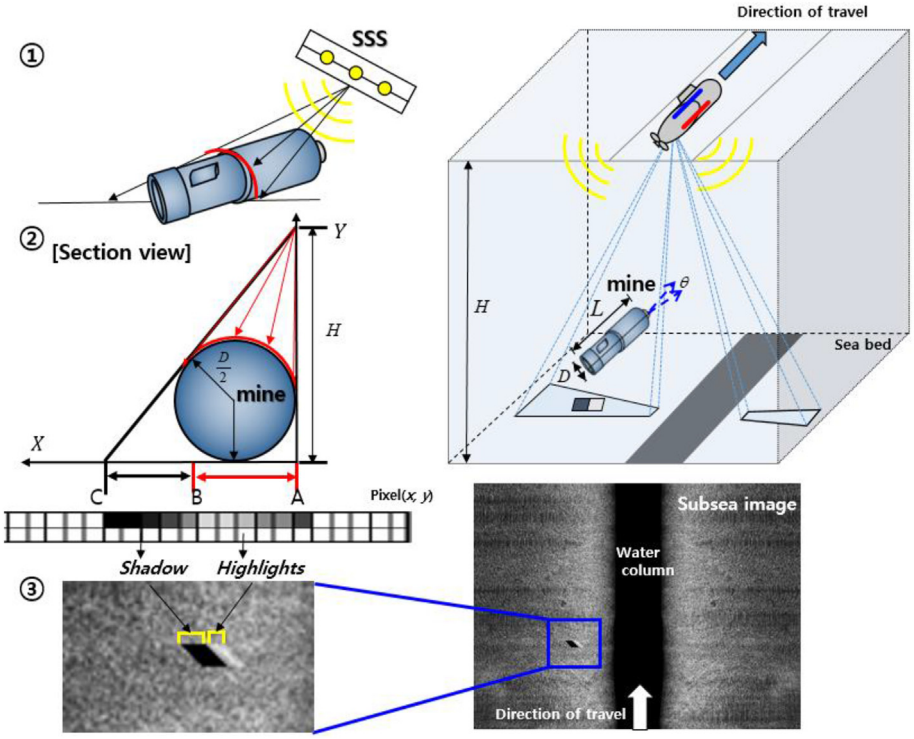

Fig. 12. Scheme of subsea image synthesis procedure.

entry scenario, as the USV uses the monocular camera image to estimate its relative position from the docking station, generating an accurate and realistic virtual docking image is critical. The docking image generation module produces a virtual image from the perspective of a camera mounted on the USV. To produce a realistic docking image, we used a docking image database of the actual docking site. It means that we recorded a set of images in advance before the USV was docking and used this database to make virtual docking images during the simulation. Fig. 14 shows the flowchart of the docking image generation module; as stated, docking images are collected in advance (offline) and used as source images to generate virtual perspective docking images for use during the simulation (online).

When creating the docking image database, instead of using a conventional mono-vision camera, we used a 360-degree camera to capture the scene of the port. While the docking simulation progresses, the position and attitude of the USV constantly change while the USV approaches the docking destination. One advantage of using a 360-degree camera is that we can easily extract a two-dimensional (2D) rectangular image (monocular camera image) with a certain attitude (any specific roll, pitch, and yaw angular position) at the designated USV position where the 360-degree image is taken. The 360-degree image is obtained in an

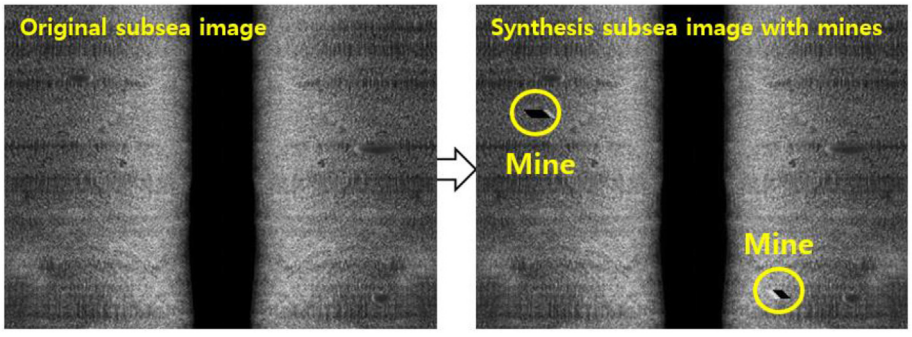

Fig. 13. Example of the synthesis of virtual mines in the original subsea image.

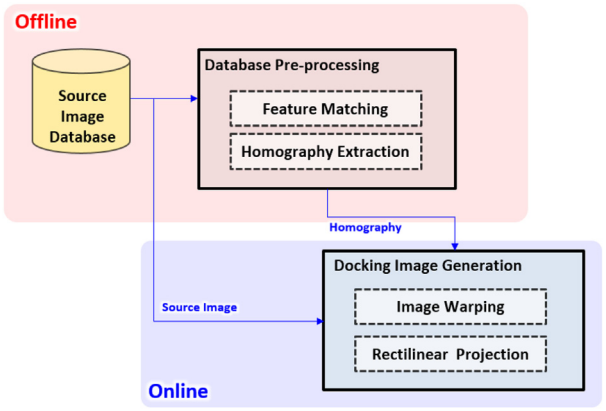

Fig. 14. Flowchart of docking image generation module.

equirectangular image form, which has 360° horizontal and vertical field of view (FOV). After the attitude and FOV of the virtual monocular camera have been defined, we can extract a 2D image of a specific perspective by applying a rectilinear projection to the source image. Fig. 15 shows an example of the 360-degree camera image. The left panel shows a 360-degree image and two extracted images with two different perspectives. The right panel presents a top view of the USV and the FOVs that correspond to the two extracted images.

Although the 360-degree camera can generate an infinite number of docking images with varying attitudes, to manage a perspective shift due to the positional change of the USV, a single source image is not sufficient. Therefore, we sampled a number of 360-degree source images in the docking region, which was defined in a 20 m × 20 m area. A total of 121 source images were recorded uniformly over the docking region, with 2 m intervals. Fig. 16 shows a schematic description of the sampling process of the source image. To collect the source image, we recorded a video using the 360-degree camera while the USV sequentially tracked the assigned path (a total of 11 paths). Since the vehicle has navigational sensors, we can reversely calculate the frame number of the video when the vehicle is located at specific sampling points. By extracting the image that corresponds to the designated sampling point, we can build a database of 360-degree source images.

During the USV's approach to the docking station, the perspective shift due to its positional change can be accounted for by switching the source image. The current source image is selected as long as the vehicle is located within the boundary of the sampling point region (2 m by 2 m area). When the vehicle moves to another sampling point region, another source image is used. However, since the source image dataset is sampled at 2 m intervals, the unnatural and sudden image changes occur when the vehicle crosses the boundary of the sampling region. To deal with this tendency, we adopted an image warping technique [12]. Image warping is a digital image processing method that can distort or transform image shape. For example, image warping can distort a partial image by assigning a new location to any detected feature points. A new homography matrix that relates the location of two sets of feature points can be calculated, and image warping can be processed by using the calculated homography matrix.

In our application, the image warping technique is used to generate middle perspectives within the sampling point region. Fig. 17 shows an example of image warping to match the perspectives at the boundary of the sampling region, which can be achieved by the following procedures. Firstly, we extract feature points from two neighboring source images using Speed-Up Robust Features (SURF) extraction [13]. Then, we match the feature sets using the Kanade–Lucas–Tomasi feature-tracking algorithm [14]. When a matched feature set is found, we can calculate the pixel difference of each feature

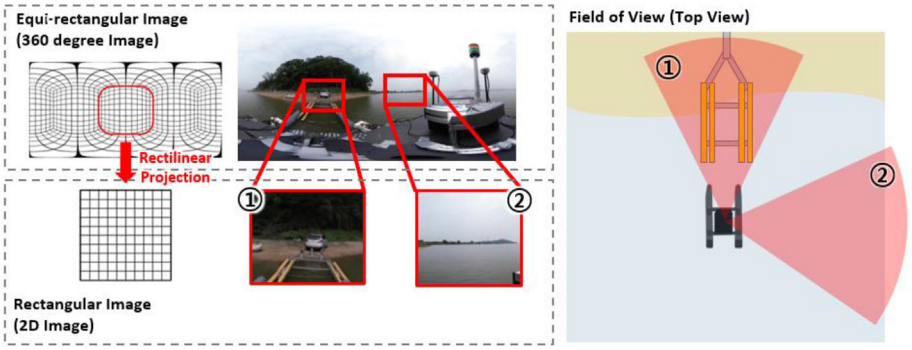

Fig. 15. Example of the use of the 360-degree image in the docking image generation module.

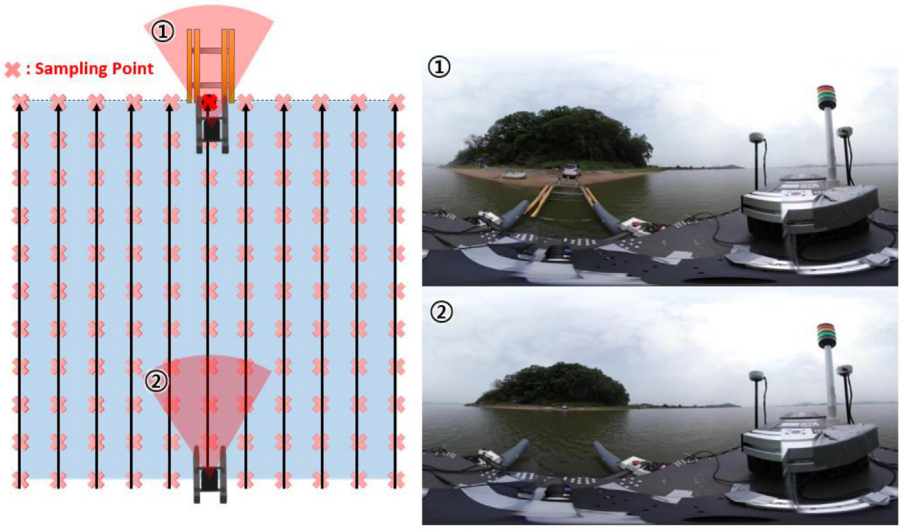

Fig. 16. Schematic description of the sampling process of the source image.

set. The homography matrix can be calculated, and image warping can be processed using the matrix using this information. By applying the interpolation method to the pixel difference matrix, we can generate a homography matrix of any virtual point within the sampling region, and a virtual image at that specific position can be obtained. Subsequently, the virtual docking image generation module can provide a smooth source image transition when crossing the boundary of the sample region and a finer perspective change inside a specific sample region.

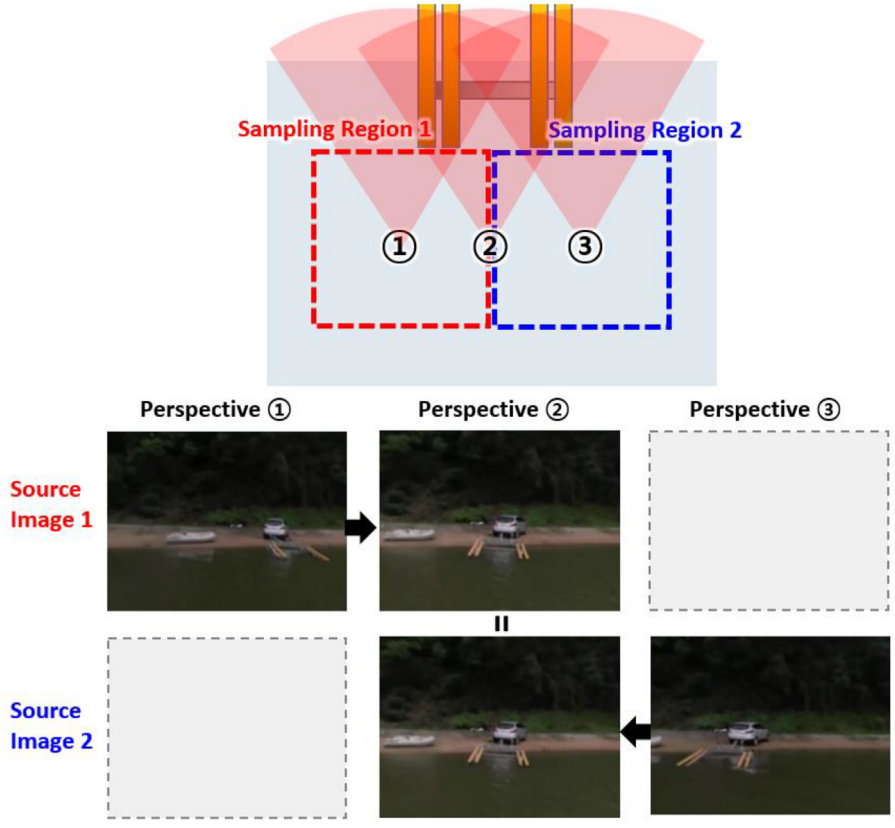

Fig. 17. Image warping usage to match perspectives at the boundary of the sampling region.

3.4. Control algorithms

In this study, we developed a number of control algorithms to accomplish the missions successfully. For validation purposes, the control algorithm is currently connected to the virtual prototype modules through the ROS. However, when the validation process is over, the developed control algorithm can be directly connected to actual USV sensors and actuators.

3.4.1. Path following and entry control

The control module controls the USV actuators (engine rpm (revolutions per minute), rudder angle). This module receives the three-DOF position and velocity variables from the motion analysis module. The control module generates the desired actuator command to achieve the goal of the current mission scenario. Because each mission has a different purpose, the control strategy varies as well. Fig. 18 shows a block diagram of the USV control process. In the port entry scenario, the purpose of the control is to approach the docking station safely without collisions. For that purpose, the desired speed of the vehicle is limited to 2 knots, and the desired rpm is applied according to the stage of docking. For the path following and mine detection scenarios, the primary goal is to follow a predetermined path, with a few tracking errors as possible.

As illustrated in Fig. 18, the control module is composed of a guidance part and a controller part. According to the mission scenario, two different controllers are designed. However, for the guidance part, the same guidance law is used for all three scenarios. The vector field guidance method for a linear path, suggested by Jantapremjit and Wilson [15]; was used for the guidance. This method calculates the desired course angle for the USV based on the direction of the path and the lateral distance of the USV from the path (see Fig. 19). If the lateral distance is large, then the desired course angle χ is defined as χ^{∞}; as the vehicle approaches the path, the cross-track error y is decreased, as is χ_{VFG}, which is the additional course angle deviation needed to make the USV approach the path. The course angle of the vehicle is defined by Eq. (1).

$$\chi^{d}(y) = \chi_{VFG} + \chi_{path}$$
$$= \chi^{\infty}\frac{2}{\pi}tan^{-1}(ky) + \chi_{path} \quad (1)$$

For the port entry scenario, a target path is defined as a virtual linear path parallel to the longitudinal direction of the docking station. For path following and mine detection missions, a piecewise linear path made by the current- and next-waypoint set is used as the target path.

After the desired course angle is calculated from the guidance part, the controllers create the desired command for each actuator. For speed dynamics, the engine rpm is controlled using a PID (Proportional Integral Derivation Control) controller that adjusts the vehicle's speed to the desired speed. For steering dynamics, to manage unpredictable external environmental loads, we adopted a reinforcement-learning-based controller.

The reinforcement-learning-based controller can adaptively learn about the environmental load and its effect on the vehicle's maneuvering motion. Among various reinforcement-learning algorithms, we chose the deep deterministic policy gradient (DDPG)) algorithm to calculate the optimal rudder command of the vehicle because it is a policy-based method and can therefore handle a continuous action space. If the action space is discrete, like the one addressed with the value-based reinforcement-learning method (e.g., Q-learning), only some rudder angle candidates can be selected, which will result in a chattering phenomenon. To handle the USV control problem using the DDPG algorithm, we defined a Markov decision process (MDP), a

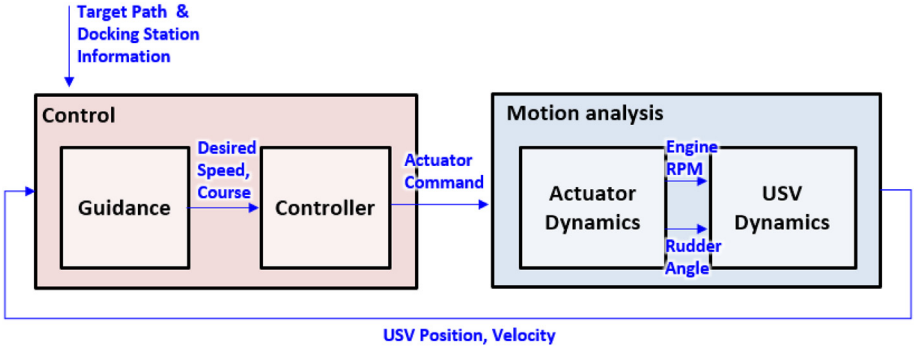

Fig. 18. Flow chart of the USV motion control system.

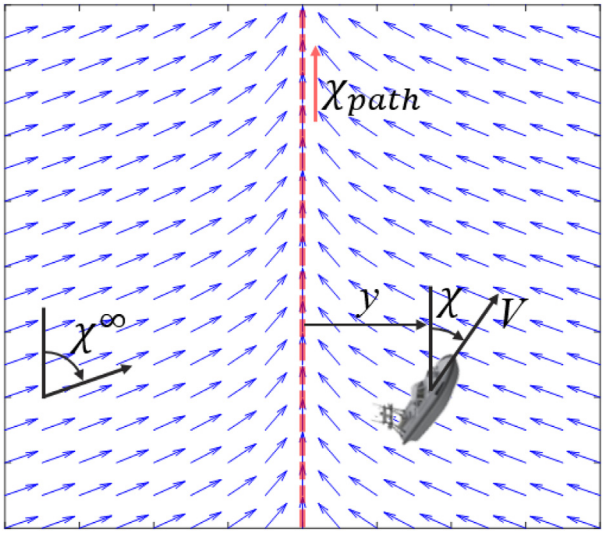

Fig. 19. Vector field guidance for linear path following.

decision-making model for the reinforcement-learning problem. The state variables of the MDP are composed of the course angle error and its derivative, the cross-track error and its derivative, the environmental load information, and the rudder angle and its derivative. For the action variables of the MDP, the rudder command in a continuous action space is used. For the reward function, we defined an exponential-based reward function that has its maximum value when the course angle and cross-track error are equal to zero. A number of path following simulations were conducted to train the reinforcement-learning-based controller. During the training, the USV learned the relationship between a conducted action and its effect on the performance of path following. By repeatedly evaluating and reinforcing the controller, the reinforcement-learning-based controller for steering dynamics can be trained. A detailed explanation about the proposed reinforcement-learning based control algorithm is described in the study of Woo et al. [16].

3.4.2. Mine detection

The mine detection module was designed to identify mine-like objects (MLOs) in the mine detection scenario. This module analyzes the underwater image produced by the subsea image composition module (3.1.2X). To detect MLOs in the subsea image, an object-detection algorithm AdaBoost, proposed by Viola and Jones [17]; is used. Although this algorithm was developed to detect a human face in an image, it can be applied to detect any object of interest with distinctively shaped features. One of the advantages of the AdaBoost algorithm is that it can significantly reduce the calculation time by selecting an important feature among a large feature set; thus, it can be applied to a real-time image.

Additionally, Sawas [18] applied this algorithm to detect underwater mines, so its applicability and effectiveness in underwater mine detection have already been verified. To detect MLOs using the AdaBoost algorithm, we divided the process into two stages. In the first stage, we extracted the image patterns of MLOs and non-mine-like objects (NMLOs) using the supervised learning method and used it to design the MLO classifier. This stage should be conducted in advance of the mine detection simulation (offline). When the classifier is constructed, we can detect any MLO included in the image in real-time (online) by applying the classifier to the subsea image.

To construct a classifier that can detect MLOs in a subsea image, we followed four sub-stages, as illustrated in Fig. 20. The first step to construct the classifier is collecting image data of MLOs and NMLOs. In a real-world problem, the underwater image collected from an SSS or a synthetic-aperture sonar can be used to construct an image database. In our case, an underwater image produced by the subsea image composition module is used to construct the subsea image database. Fig. 21 shows some examples of the subsea and template images used to build the classifier. A total of 5000 underwater images were used in the subsea image database, and 10,000 MLO and 20,000 NMLO templates were extracted to train the MLO classifier.

When the image template database (MLO and NMLO template images) is collected, the Haar-like features of all of the images must be extracted. The Haar-like features are defined by using a mask composed of black and white regions. When applying the mask to an image, its scale may vary,

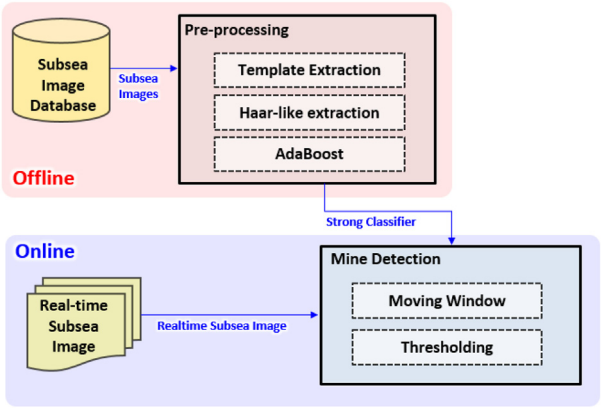

Fig. 20. Flowchart of mine detection module process.

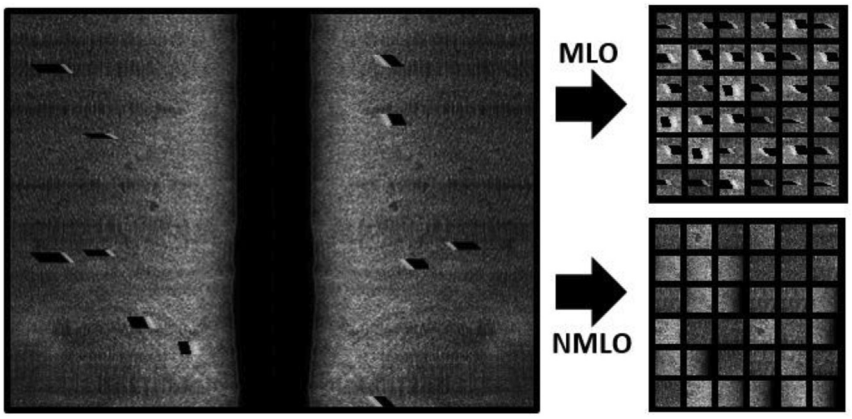

Fig. 21. Process of extracting MLO and NMLO templates from the subsea image.

and its position is random, as shown in Fig. 22. The Haar-like feature can be calculated by using the intensity sum of the white and black regions. When applying four different types of Haar-like-feature masks on a template image (with a size of 24 pixels by 24 pixels), over 160,000 Haar-like features are defined.

After the calculation of the Haar-like features, a weak classifier is designed using these features to classify the MLOs and NMLOs based on a single Haar-like feature. As it uses only one Haar-like feature, the classifier has relatively poor classification performance but a simple structure. As mentioned, over 160,000 Haar-like features can be calculated on the images; over 160,000 weak classifiers can be designed. The weak classifier uses a threshold value to classify the object. The threshold value for each Haar-like feature can be obtained from the probability distribution of the MLO and NMLO images. By constructing a normal-distribution model for each Haar-like feature value of the image set, the optimal threshold that can classify two probability distributions can be calculated.

Since the number of weak classifiers is over 160,000, using them to detect MLOs is neither efficient nor effective. Although some of the weak classifiers have a high-level performance, the weak classification is constructed based on a randomly placed and scaled mask on the image template, making most of them have poor classification ability. Therefore, weak classifiers with high performance must be selected and combined to build a new classifier. For this purpose, the adaptive boosting (AdaBoost) algorithm is used. Boosting employs a series of processes that combine less-effective classifiers to construct a more effective classifier. In the AdaBoost algorithm, a strong classifier is constructed by calculating a weighted sum

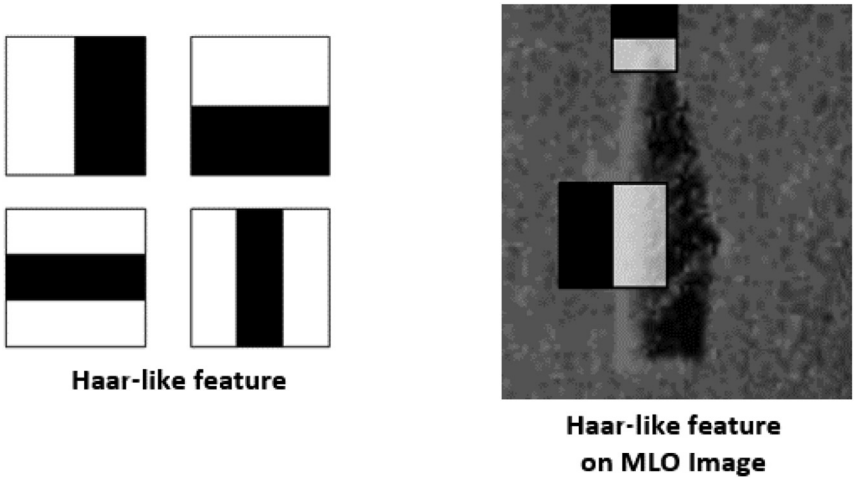

Fig. 22. MLO detection using Haar-like features.

of weak classifiers. To build strong classifiers, the relevant features that are effective in distinguishing MLOs from NMLOs should be obtained. In addition, the weight for each weak classifier must be determined. As the training stage proceeds, the AdaBoost algorithm iteratively evaluates and reinforces the strong classifier. In this work, we selected 100 weak classifiers from 160,000 classifiers to build a final strong classifier.

After the final strong classifier is trained (offline), it can be applied to detect mines in the real-time mine detection simulation. Because the constructed classifier is defined in the template image, which has a size of 24 pixels by 24 pixels, a moving window with an identical size can be defined to apply the classifier. As shown in Fig. 23, if the value of the strong classifier on the moving window is higher than the pre-defined threshold, we determine that there is a mine at the location of the moving window. The bottom row of Fig. 23 shows some of the mine detection results obtained during the simulation.

3.4.3. Docking pose estimation

The docking pose estimation module predicts the position and attitude (pose) of the USV based on the docking image that is produced by the docking image generation module. In this module, we adopted a method called PoseNet, proposed by Kendall et al. [19]; to estimate the USV pose (see Fig. 24). PoseNet is a machine-learning-based localization method that can estimate a six-DOF pose in real-time. The method uses supervised learning to train a convolutional neural network (CNN), which can estimate a pose corresponding to the image input to the network. Recently, a number of deep-neural-network-based models have been designed for the classification problem, such as Alex Net [20], Res Net [21], or GoogLeNet [22]. In PoseNet, the structure of GoogLeNet is slightly modified. Because GoogLeNet is designed for the classification problem, instead of using a softmax layer, the output layer is directly connected to the fully connected layer. Thus, PoseNet can deal with the regression problem instead of the classification problem.

To use PoseNet to estimate the USV pose, we must collect a dataset of docking images and the corresponding pose information. A number of docking simulations with varying initial pose conditions were conducted while recording the docking image and pose. A total of 230,000 labeled images were used to train the CNN. After the supervised learning process, we conducted a docking simulation for validation. According to the result, the estimation error tends to be bounded, with root-mean-square errors of 0.24 m for the lateral position, 0.19 m for the longitudinal position, and 0.64° for the heading angle. Fig. 25 shows snapshots of virtual docking images, as well as the top view of the actual USV position and the corresponding estimated position at the same time.

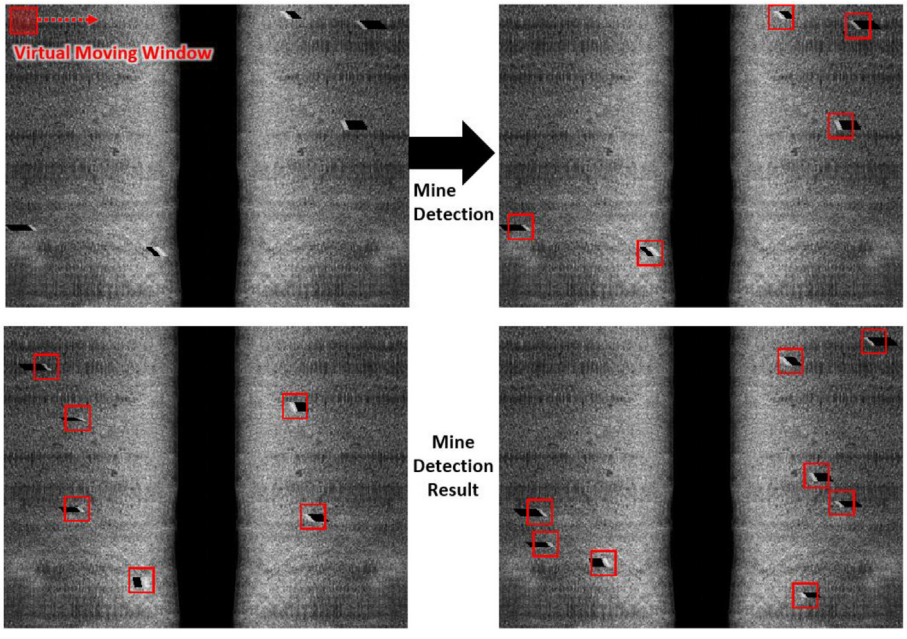

Fig. 23. Mine detection result obtained using the trained strong classifier.

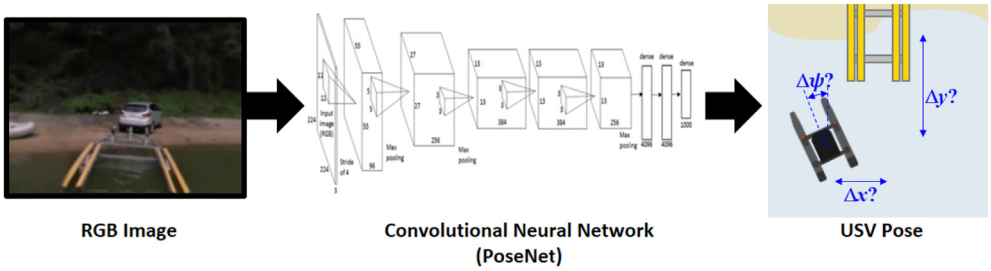

Fig. 24. Concept of PoseNet for USV localization.

4. Integrated simulation environment

To construct the simulation environment of the USV, the integration of the modules introduced above is necessary; this was achieved by implementing the ROS. Generally, communication methods such as high-level architecture and TCP/IP (Transmission Control Protocol/Internet Protocol) are used for data communication between multiple modules. However, in such cases, program conversion or additional interfaces are necessary to communicate with the hardware. On the other hand, the ROS includes a hardware platform, and various hardware units, such as motors and sensors, are supported by the ROS. Therefore, it is possible to achieve integration not only between the modules but also between the modules and other hardware, without program conversion or additional interfaces.

The ROS is a communication-based program containing the core of the message communication between nodes. In the ROS, the minimum execution unit is the node, which is a program that controls one function. The master acts as a namespace that manages multiple nodes. In the past, programs in the robotics field were created with a single frame from the drivers of the sensors and the actuators to provide sensing, recognition, and operation. However, to reuse robot software, it is necessary to divide the program into small parts according to the purpose of each processor. In the ROS, this process is called node packaging, and the nodes, which are divided as the

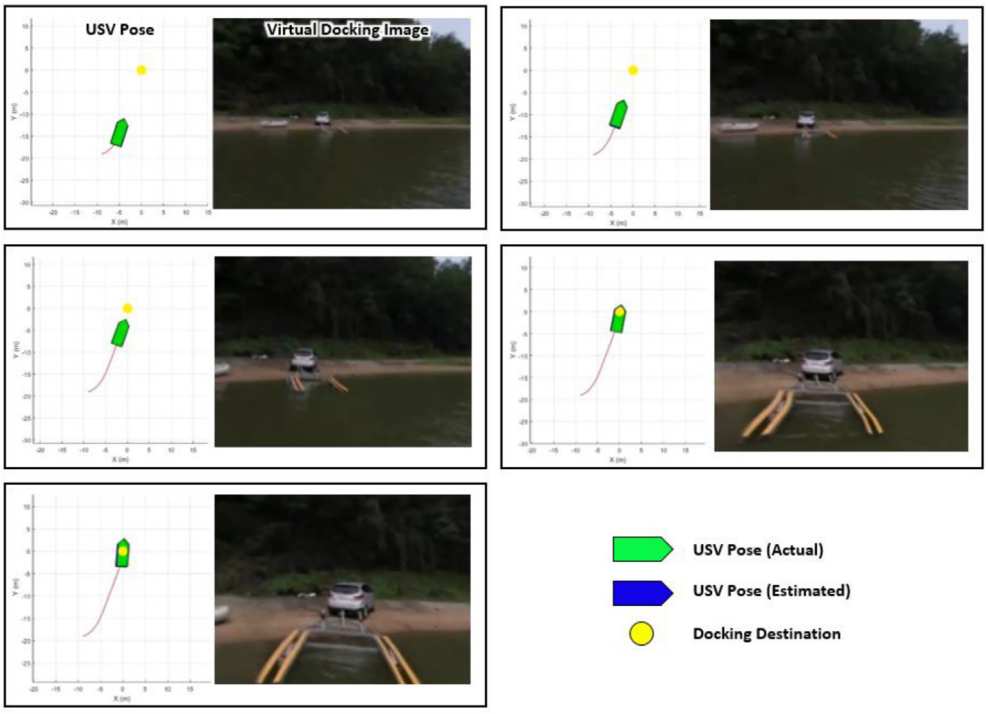

Fig. 25. Docking simulation result of PoseNet-based USV localization.

minimum execution units, transmit and receive data by TCP/IP message communication.

The modules of the virtual prototype and the control algorithm were developed as the ROS node. Therefore, each module is an independently executable program that sends and receives the necessary data via the ROS interface. Table 2 shows the information of the transmitted data. The operating system that the ROS officially supports is Ubuntu. However, in many cases, some nodes were developed in other operating systems, although the master of the ROS is operating in Ubuntu. The modules used for scenario management, the virtual prototype, and the control algorithm were developed in Windows and written in C# or MATLAB. The ROS master and each node developed in other operating systems can communicate using the ROS interface by using Rosbridge, which is a ROS package that helps non-ROS programs use ROS functions.

5. Application

Based on the developed modules of the virtual prototypes and the control algorithms, USV simulations for three scenarios – path following, mine detection, and port entry – were performed. The physical properties of the USV are assumed to be as in Table 3.

5.1. Path following

In the path following scenario, the USV follows the target path under the influence of environmental loads by learning. The path following control algorithm calculates the target rudder angle required to follow the given path as a result of learning. Fig. 26 shows the modules and the transmitted data in this scenario.

The target path and environmental conditions are set as follows. The target path is created by linearly connecting the waypoints in the order input in the scenario management module (see Fig. 27). When the USV arrives at the boundary at each waypoint, it is assumed to have reached the waypoint and follows the path to the next waypoint. The current is considered as an environmental load.

The results were compared with that obtained without environmental loads. The actual path of the USV is shown as a blue line in the display module. As shown in Fig. 28, the control algorithm is successfully verified in the given path and environment. In the case with current, the USV controls the rudder angle to the opposite direction to follow the target path.

Table 2. Transmitted data between modules.

Scenario	From	To	Data
Initialize	Scenario management	Motion analysis	Number of USV
			Physical properties of USV
			Simulation time step
			External forces
		Path following control	Number of USV
			Physical properties of USV
			Target path
		Subsea image composition	Mine properties
		Entry control	Physical properties of USV
			Simulation time step
			Port location
Path following	Scenario management	Motion analysis	Simulation time
	Motion analysis	Path following control	USV position
	Path following control	Motion analysis	Control command
	Motion analysis	Display	USV position (controlled)
Mine detection	Scenario management	Motion analysis	Simulation time
	Motion analysis	Subsea image composition	USV position
	Subsea image composition	Mine detection	Subsea image
	Mine detection	Display	Detected mines
			Detected subsea image
Port entry	Scenario management	Motion analysis	Simulation time
	Motion analysis	Docking image generation	USV position (actual)
	Docking image generation	Docking pose estimation	Docking image
	Docking pose estimation	Entry control	USV position (estimated)
	Entry control	Motion analysis	Control command
	Motion analysis	Display	USV position (controlled)

Table 3. Physical properties of the USV.

Mass [ton]	1.1971	Speed [m/s]	8
Inertia (Izz) [ton*m^2]	31,583	Search radius [m]	40
Length [m]	7	Rudder angle limit [°]	45
COG (X) [m]	3.5	Rudder angular velocity limit [°/s]	20

5.2. Mine detection

In the mine detection scenario, the SSS towed by the USV generates the subsea image, and the control algorithm in the autonomous software detects mines in the image. The information of the detected mines is then transmitted to the display module. The modules used for the scenario are presented in Fig. 29. Similarly, the path following module controls the USV during the mine detection scenario.

Fig. 30 shows the input data for the search area, including mines and the target path of the USV. The mines can be arranged randomly in the given area according to their number. As shown in Fig. 30, 10 mines are assumed to exist in the search area. During mine detection, the USV moves on a straight line to generate the subsea image correctly. Therefore, in the mine detection scenario, the target path of the USV is generated automatically based on the search area, the initial heading angle of the USV, and the search radius of the SSS. In this scenario, we assumed two USVs operating together in two directions and crossing at right angles to detect the mines accurately.

The result is shown in Fig. 31. As the USV moves along the target path, the virtual prototype of the SSS generates the subsea image, including the virtual mines. Subsequently, the display module shows the mines detected by the control algorithm, together with the actual mines arranged by the user. As shown in Fig. 31, the USV detects every mine correctly. The average distance error between the actual and the detected mines is 2.36 m, with a maximum value under 5 m. In the figure, some mines are overlapped because they are detected by the two USVs simultaneously. In some cases, if the mine is located close to the path, or the angle of the mine is perpendicular to the path direction, the mine cannot be detected. In those cases, the other USV that crosses at right angles can detect the mine.

5.3. Port entry

After the whole mission, the USV controls itself to return to the port. For detailed control, the camera mounted on the USV sends the image of the port seen by the USV. Then, the autonomous software in the USV estimates its position and sends the control command to enter the port to the propeller and the rudder. The modules required for the port entry scenario are shown in Fig. 32. Unlike the path following scenario, both the propeller and the rudder of the USV are controlled.

As shown in Fig. 33, the docking area is defined as a square of 20 m in length near the port. When the USV arrives at the docking area during the mission, the port entry scenario begins, and the USV is controlled to enter the port. The location and the heading angle of the port are defined in the scenario management module at the initial stage.

The generated docking image and the actual path of the USV are shown in Fig. 34. The docking image generation module synthesizes the docking image seen by the virtual camera with the given position and heading angle of the USV. Then, the docking pose estimation module estimates the position of the USV by analyzing the image. The port entry control module sends the control command to the

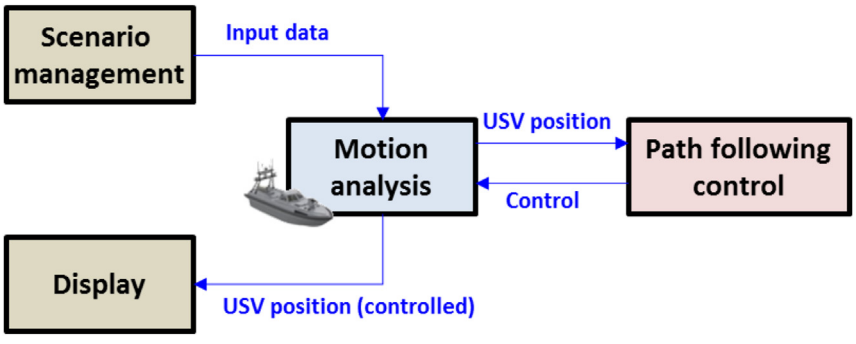

Fig. 26. The modules and the transmitted data for the path following scenario.

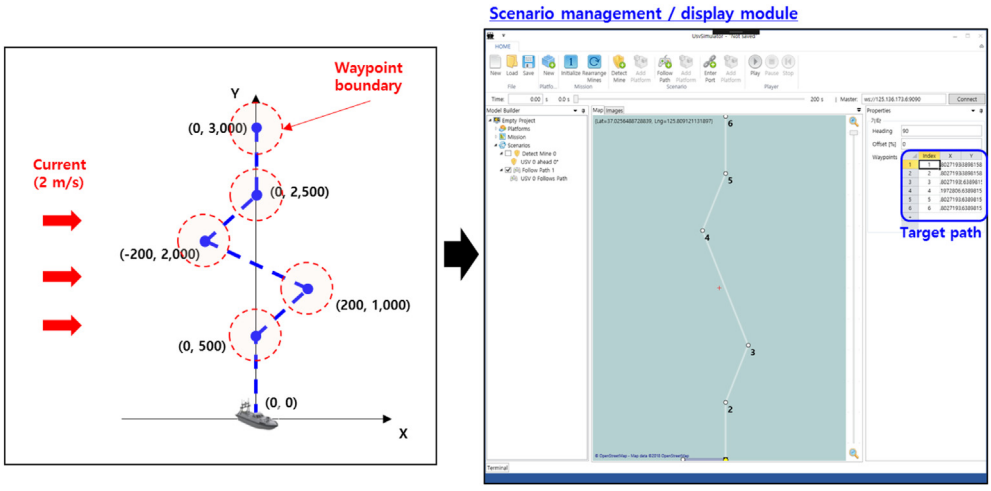

Fig. 27. The target path and the environmental conditions in the path following scenario.

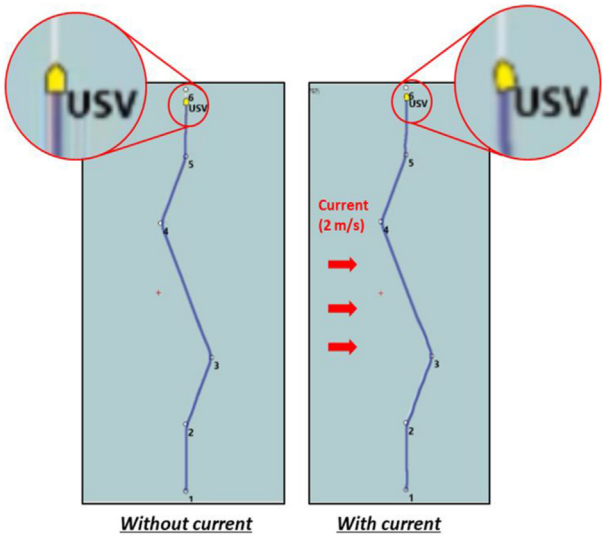

Fig. 28. The result of the path following scenario.

propeller and the rudder of the USV based on the estimated position. As shown in Fig. 34, the USV successfully enters the port using the virtual camera.

6. Conclusions and future work

In this study, a simulation environment was constructed to verify and validate the autonomous software of the USV. For this purpose, virtual prototypes representing the sensors and actuators of the USV and the ocean environment are proposed. Specifically, the subsea image generated by the SSS was composed using the virtual prototypes. The control algorithms in the autonomous software were developed according to the mission scenario of the USV. For the path following control, a reinforcement-learning controller was adopted. In the mine detection algorithm, the subsea image composed of the virtual prototypes was analyzed using the AdaBoost algorithm. Lastly, for the port entry scenario, the position of the USV was estimated from the virtual docking image.

For the communication between the virtual prototype and the autonomous software, an integrated simulation environment was constructed using the data communication interface of the ROS. With this simulation environment, the virtual prototype can be easily replaced by actual hardware, which is supported by the ROS. With actual hardware, a hardware-in-the-loop simulation can be performed with the scenario management module and the control algorithms.

Three mission scenarios were considered for the USV: the path following, mine detection, and port entry scenarios. The suggested simulation

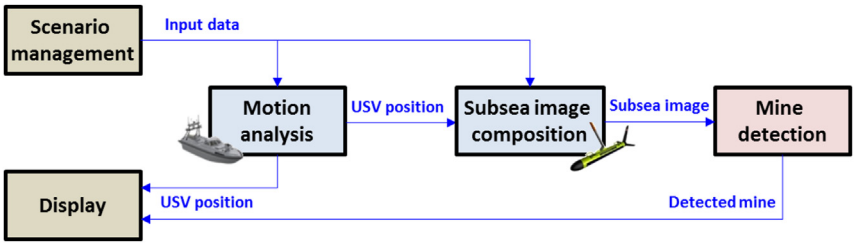

Fig. 29. The modules and the transmitted data for the mine detection scenario.

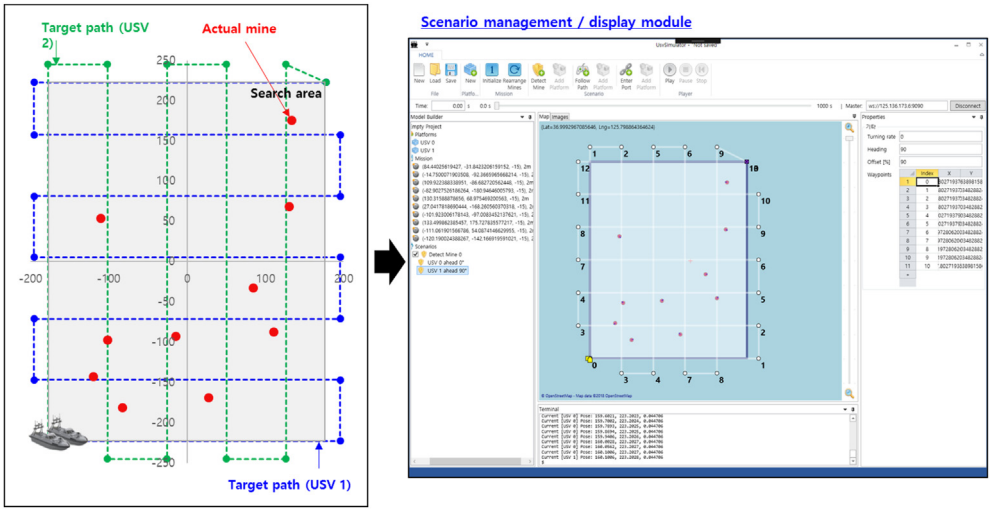

Fig. 30. The search area and the target path in the mine detection scenario.

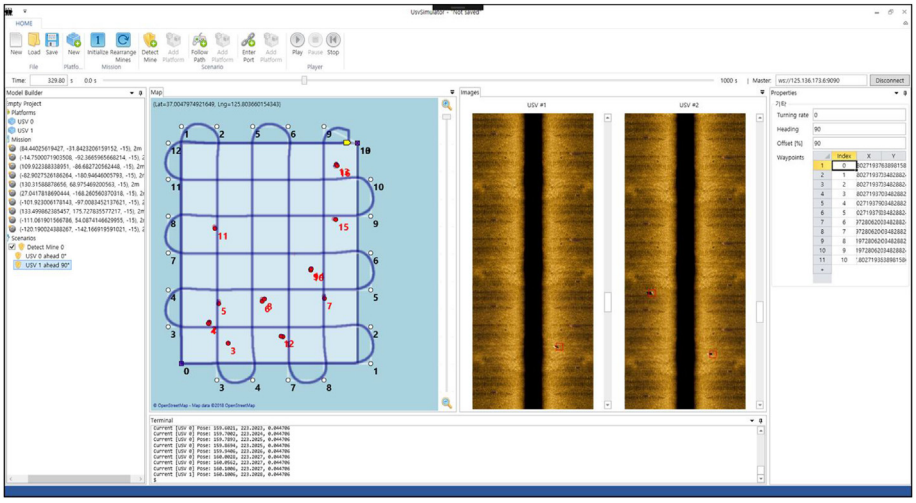

Fig. 31. The result of the mine detection scenario.

environment was applied to the three scenarios to check its applicability. In the path following scenario, the USV follows the target path under unexpected environmental loads as a result of reinforcement learning. In this case, it is not necessary to adjust the control coefficients as in the PID controller. In the mine detection scenario, the virtual subsea image is transmitted to the mine detection algorithm, which detects mines in the image. As a result, ten actual mines located randomly in the search area were all detected by two USVs crossing right angles. Lastly, in the port

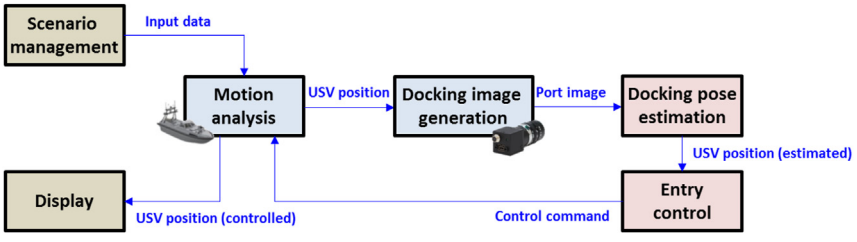

Fig. 32. The modules and the transmitted data for the port entry scenario.

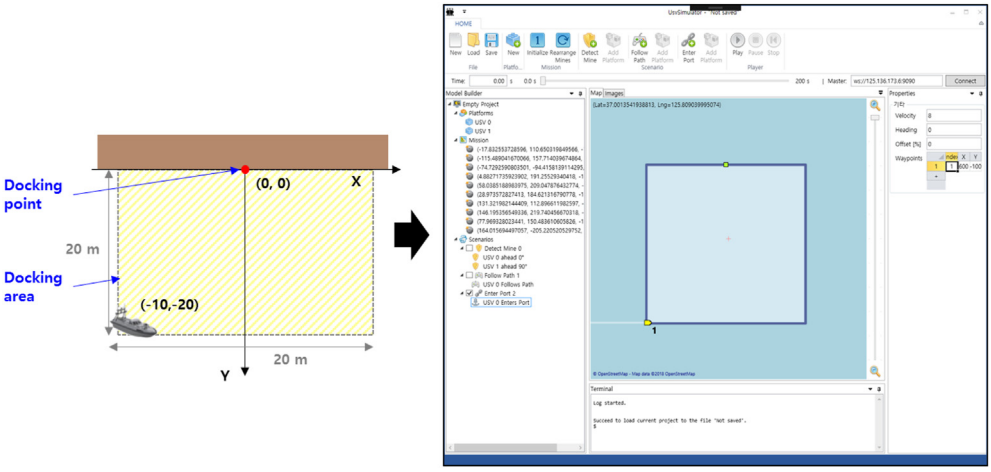

Fig. 33. The port and the docking area in the port entry scenario.

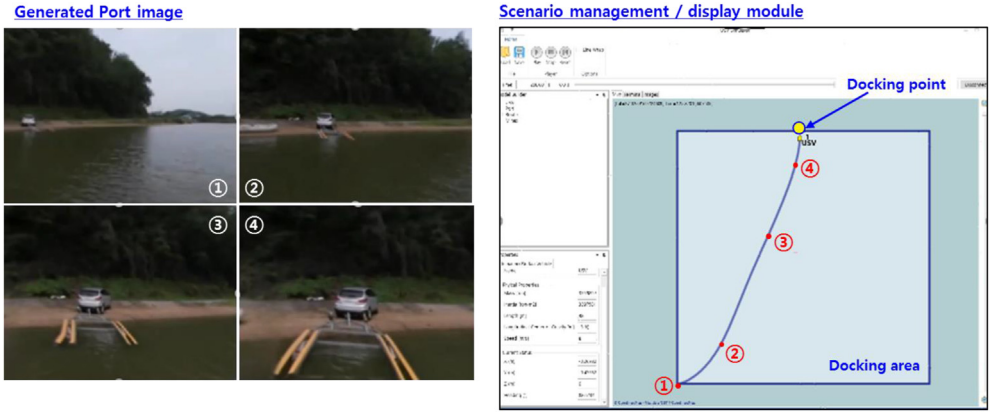

Fig. 34. The result of the port entry scenario.

entry scenario, the position of the USV is estimated correctly, and the USV is controlled to enter the port automatically. Here, the docking image is generated using the actual port image from a 360-degree camera. The result shows that the proposed method can be effectively used to test and develop the control algorithms of the autonomous software of the USV. In future work, the modules of the control algorithms and scenario management can be integrated with the actual hardware, which is supported by the ROS. Then, an experiment using actual USVs can be conducted in the ocean environment to validate the control algorithms.

Acknowledgments

This study is an extension of our previous studies [23–28] and was partially supported by.

(a) Agency for Defense Development, Republic of Korea (UD160009DD).

(b) Research Institute of Marine Systems Engineering of Seoul National University, Republic of Korea.

References

[1] Ademovic A. An introduction to robot operating system: the ultimate robot application framework. Toptal; 2015.
[2] Heo J, Hwang KC, Kwon Y. Remote operation SW for USV: Part II. Simulation development. World J Eng Technol 2018; 6:816–24.
[3] Kim JH, Kim JH, You YJ, Chi SD. Simulation-based effectiveness analysis of mission planning for autonomous Unmanned Surface Vehicles (USVs). Int J Eng Res Technol 2018; 11(5):715–24.
[4] Demarco K, West ME, Collins TR. An implementation of ROS on the Yellowfin autonomous underwater vehicle (AUV). In: Proceedings of 2011 IEEE Oceans Conference; 2011.
[5] Cashmore M, Fox M, Larkworthy T, Long D, Magazzeni D. AUV mission control via temporal planning. In: Proceedings of 2014 IEEE International Conference on Robotics and Automation; 2014.
[6] Mendonca R, Santana P, Marques F, Lourenco A, Silva J, Barata J. Kelpie: a ROS-based multi-robot simulator for

water surface and aerial vehicles. In: Proceedings of 2013 IEEE International Conference on Systems, Man, and Cybernetics; 2013.
[7] Conte G, Scaradozzi D, Sorbi L, Panebianco L, Mannocchi D. ROS multi-agent structure for autonomous surface vehicles. In: Proceedings of 2015 IEEE Oceans Conference; 2015.
[8] Zhao L, Roh MI, Ham SH. Performance analysis of an active heave compensation system on an offshore supply vessel using the hardware-in-the-loop simulation. J Mar Sci Technol 2018;26(5):678−91.
[9] Kim HM, Lee DW. Integrated simulation environment for heterogeneous unmanned vehicles using ROS and Pixhawk. Acad J Sci Res 2019;3(3):1−14.
[10] Gertler M, Hagen GR. Standard equations of motion for submarine simulation. Bethesda, Maryland, USA: David W. Taylor Naval Ship Research and Development Center; 1967.
[11] Abkowitz MA. Stability and motion control of ocean vehicles. M.I.T Press; 1967.
[12] Wolberg G. Digital image warping. IEEE Computer Society Press; 1996.
[13] Bay H, Tuytelaars T, Gool LV. SURF: speeded up robust features. In: Proceedings of the European Conference on Computer Vision; 2006.
[14] Tomasi C, Kanade T. Detection and tracking of point features. Int J Comput Vis 1991;9:137−54.
[15] Jantapremjit P, Wilson PA. Guidance-control based path following for homing and docking using an autonomous underwater vehicle. In: Proceedings of 2008 IEEE Oceans Conference; 2008.
[16] Woo JH, Yu CW, Kim NW. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Eng 2019;183:155−66.
[17] Viola P, Jones MJ. Robust real-time face detection. Int J Comput Vis 2004;57:137−54.
[18] Sawas J. Automatic target recognition in sonar imagery using a cascade of boosted classifiers. Heriot-Watt University; 2015.
[19] Kendall A, Grimes M, Cipolla R. PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In: Proceedings of 2015 IEEE International Conference on Computer Vision; 2015. p. 2938−46.
[20] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012.
[21] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition; 2016.
[22] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition; 2015.
[23] Lee HW, Roh MI, Ham SH, Yu CW. A study of data transmitting and receiving technique based on ROS (Robot Operation System) for integrated simulation of USV (Unmanned Surface Vehicle). In: Proceedings of 2017 Annual Winter Conference, the Society for Computational Design and Engineering; 2017. p. 556.
[24] Lee HW, Roh MI, Ham SH, Zhao L, Ha S, Kim NW, et al. Development environment of autonomic softwares for USV (Unmanned Surface Vehicle) based on ROS. In: Proceedings of 2017 Annual Autumn Conference, the Society of Naval Architects of Korea; 2017. p. 374.
[25] Lee HW, Roh MI, Ham SH, Zhao L, Kim NW, Ha S, et al. A study on integrated simulation method for mine detection mission of USV. Kor J Comput Design Eng 2017c;22(3): 306−16.
[26] Lee HW, Roh MI, Ham SH, Zhao L, Kim NW, Woo JH, et al. A study of integrated simulation method for the design of USV (Unmanned Surface Vehicle). In: Proceedings of 2017 Conference on Marine Robot Technology 2017; 2017.
[27] Lee HW, Roh MI, Ham SH, Zhao L, Kim NW, Woo JH, et al. Integrated simulation of control algorithms and the virtual prototype of USV based on ROS. In: Proceedings of the Society of Naval Architects of Korea; 2017.
[28] Lee HW, Roh MI, Zhao L, Ham SH, Kim NW, Yu CW. Integrated development environment of autonomic software for USV (unmanned surface vehicle) based on ROS (robot operating system). In: Proceedings of 2017 International Conference on Computer Applications in Shipbuilding; 2017.