Academia Journal of Scientific Research 3(3): 000-000, April 2019
DOI: 10.15413/ajsr.2018.0133
ISSN 2315-7712

Academia Publishing

Research Paper

Integrated simulation environment for heterogeneous unmanned vehicles using ROS and Pixhawk

Hyeong-Min Kim and Dae-Woo Lee*

Pusan National University, Jangjeon 2-dong Geumjeong-Gu, Busan, Republic of Korea.

*Corresponding author. E-mail: baenggi@pusan.ac.kr.

Accepted 29th March, 2019

ABSTRACT

We proposed a framework for constructing an integrated simulated environment for heterogeneous unmanned vehicles. Systems for cooperation between unmanned vehicles are becoming more important. Unmanned vehicles can carry out missions without a passenger, they are highly stable, and can perform a variety of missions. In addition, due to the difficulty of the recent mission, such as SEAD (Suppression of the Enemy Air Defenses), MUSIC (Manned Unmanned Systems Integration Capability), and Golden-Time in the rescue mission, the collaborative work of multiple Unmanned Vehicles is emphasized. In industry, however, individual vehicles are mainly developed independently, and there is no active growth due to differences in characteristics and barriers to entry. In this research, ROS (Robot Operating System) was proposed as a solution for this problem in the construction of a collaboration system using Pixhawk and an unmanned vehicle integrated simulator based on open source software. We verified the feasibility of the approach through simulation and experiment.

Key words: Unmanned vehicle, cooperation system, open source, Pixhawk, ROS (Robot Operating System).

INTRODUCTION

Since interest in unmanned vehicles has increased, the importance of developing related systems has been emphasized, and many studies on single unmanned systems have been conducted. Unmanned vehicles have an advantage in that they can reduce casualties as compared with manned vehicles, and various missions can be performed, such as search, reconnaissance, and mapping of dangerous areas. Research is actively being done in related fields (Bechtsis et al., 2017; Dongki et al., 2017; Imdoukh et al., 2017; Kim et al., 2016; Mogili and Deepak, 2018; Specht et al., 2017).

However, since the mission complexity of each mission increases, the need for collaborative systems becomes more important. The US Department of Defense uses SEAD (Suppression of the Enemy Air Defenses) to search for enemies, and MUSIC (Manned Unmanned Systems Integration Capability) is used to search for and strike enemies using ground operators, UGVs, and UAVs. In addition, in the case of a rescue mission, there is a "golden time" for rescuing lives, and the search time should be taken into consideration. Thus, the necessity of cooperation among multiple unmanned vehicles rather than a single vehicle is suggested.

Sinisterra et al. (2017) constructed an unmanned surface vehicle system using Pixhawk and Odroid-XU4 and performed a mission based on image processing with a UAVs. However, they used Dronekit instead of ROS. Dronekit only supports the Python language and is less accessible than ROS, which supports C, C ++, and Python. ROS has been applied to various Unmanned Vehicles. For example, multiple UAVs mission (Braga et al., 2017), SLAM mission (López et al., 2017), multi-robot application (Portugal et al., 2019), and multiple UAV operation using crazyflie. Also, these applications have a problem for extending to heterogeneous collaboration system. Most of these researches applied specific hardware dependent

packages available on the ROS wiki, giving up generality. For example, the crazyflie package only applies to the crazyflie platform, and patrolling_simonly applies to the mobile robot. When trying to operate an Unmanned heterogeneous platform, previous studies have difficulty in collaboration due to different communication protocols.

On the other hand, Pixhawkis open source hardware and software, supports UAV, UGV, UUV (Unmanned Underwater Vehicle), USV and other Unmanned Platforms. These platforms equally use the MAVLink protocol. ROS, Robot Operating System, is used widely in the robotics field. It enables communication between multiple processors using nodes, packages, topics, and messages. So ROS can be used for communication of heterogeneous Unmanned Vehicles. In this study, Pixhawk was applied to UAV and USV with Raspberry Pi and MAVROS. Raspberry Pi was used to enable ROS, and MAVROS was one of ROS package that MAVLink was packaged. A collaboration system also can be applied to not only UAV-USV but also other heterogeneous Unmanned Vehicles. Youn et al. (2016) found that the development of small UAVs in Korea is based on open-source products such as Pixhawk and ROS and suggested using them.

In this study, we verify the feasibility of the integrated simulation system by simulating the illegal fishing boat surveillance mission via the USV-UAVs collaboration system using Pixhawk and ROS. USV was operated using the actual hull, and UAV was verified by the simulation program because of the risk of falling. Mission algorithm was made into the ROS package and executed in each UAV. For this purpose, a User-defined topic was constructed using MATLAB / SIMULINK, and the usability was verified through transmission and reception simulation. After simulation, totally 4 cases were executed for verified feasibility of the integrated system.

The USV uses the Pixhawk and Raspberry Pi to configure the hardware of the control system and configure the software using ROS. WIFI repeater was used for ROS communication and Telemetry receiver was used for fail-safety in case the ROS communication is disconnected. In addition, the UAV is composed of four units using Gazebo, and all unmanned vehicles communicate using MAVROS, which is described in then Materials and Methods section. Thereafter is the description of the configuration of a User-defined topic using MATLAB/SIMULINK's Robotics toolbox, mission algorithm construction using C language and made it into the ROS package. Simulation and Experiment were executed to verify the feasibility of the integrated system. Finally, the conclusion of this study is given.

MATERIALS AND METHODS

Unmanned system configuration

USV system configuration

Hardware configuration: In this study, a hobby fishing boat was improved and made to operate unmanned. The rear of the hull is equipped with a rudder for direction control and throttle for thrust control. Table 1 shows the specifications of the boat. Table 2 shows the UAV specifications. Figure 1 shows RTK Here GNSS sensor, which is compatible with Pixhawk 2.1 and was attached to the front of the boat. An RC receiver was used for manual control and attached to the center of the hull. In the rear part of the hull, a WIFI receiver was installed for ROS communication, and a telemetry receiver was used for direct communication with the Pixhawk in case the ROS communication is disconnected. If communication is lost, the ground control system communicates with the Pixhawk in the 433-Mhz band.

Figure 1 also shows an internal system diagram of the USV. The waterproof enclosure contains the main processors (a Pixhawk 2.1 and Raspberry Pi 3b+), which are connected using a USB-to-TTL cable. Power is supplied from a battery to the Pixhawk and Raspberry Pi at a rated voltage of 5V through a power module. The basic WIFI module of the Raspberry Pi has a limited transmission and reception distance, so a separate WIFI receiver was connected to extend it.

The software configuration is centered on ROS communication. We used a bash script in the Raspberry Pi to declare the IP of the Raspberry Pi as the ROS MASTER URI (Uniform Resource Identifier) and set the Raspberry Pi to be the master of the ROS communication.

Figure 2 shows the script contents of the launch file for ROS communication in the USV. fcu_url is for serial communication between Pixhawk and Raspberry Pi and inputs the address. gcs_url declares the IP address and UDP port of the computer that will be used in the ground control system. Mission Planner software was used as the ground control system. Figure 3 shows the Mission Planner screen.

Control system configuration using Pixhawk and Raspberry Pi: Pixhawk was used as the main processor to build the collaboration system. Pixawk includes a basic guide, development guide, firmware, Github, schematics, and a JTAC header, and the related community is also active. The types of hardware include Pixawk 1, 2, and 3, MicroPix, and Pixracer.

Pixhawk's software includes PX4Pro, developed by Dronecode, and Ardupilot, developed by 3DR (Figure 4). Both software support multi-copter, fixed wing, and rover, etc. Ardupilot supports the boat platform through ArduRover firmware and it can set vector thrust for steering performance enhancement through internal parameter modification. ArduRover also used the L1 controller in navigation. This was based in a study by Park et al. (2004). This is different from PX4Pro firmware, so this study used ArduRover firmware on the Boat platform.

Pixhawk2.1 was used as hardware as compared wiht previous versions. Pixhawk 2.1 supports the modular design, vibration/triple IMUs, multiple GPSs, and

Table 1: Boat specifications

List	Specification
Length	1000 mm
Width	350 mm
Height	320 mm
Weight	13 kg
Motor	450-W BLDC Motor

Table 2: UAV specifications.

List	Specification
Payload	0.4 kg
Average flight time	10 ~ 15 minutes
Height	100 mm
Weight	1.282 kg
Maximum velocity	17m/s

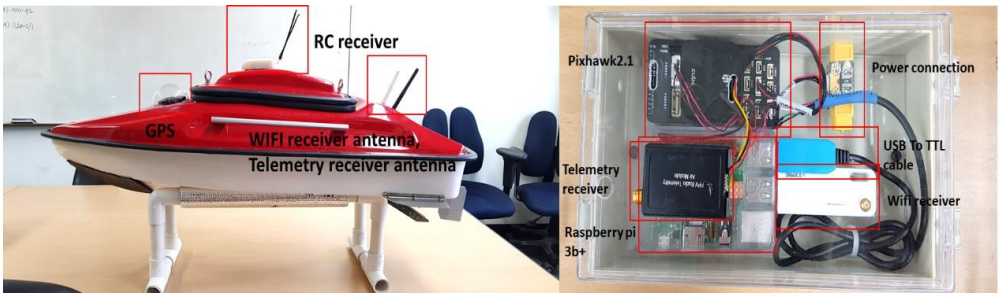


Figure 1: USV system configuration.

centimeter-level RTK GPS. In addition, the Intel Edison port allows high-level operations to be performed, such as image processing in Edison, and then it can send the results to the PixHawk 2.1 to perform collision avoidance. But since the Pixhawk used in this study does not have Intel carrier board, Companion computer was used. The Raspberry Pi used as a companion computer is a Linux-based educational single-board computer developed by the Raspberry Foundation in England. It is cost effective, high performance, and relatively easy accessibility. It has a USB port, GPIO port, and LAN port for connecting a keyboard, mouse, and storage device. A Raspberry Pi 3b+ was used to communicate through serial communication using the Pixhawk and a USB-to-TTL cable. And the WIFI Module of the Raspberry Pi communicates with the ground control system through UDP communication. The system configuration is shown in Figure 5.

Configuration of UAV system using Gazebo

UAV system configuration

In this study, PX4Pro firmware was used for UAVs, to confirm that both ArduRover and PX4Pro firmware can be used with ROS. Pixhawk also supports ROS and Gazebo, allowing verification of the firmware before actual experimentation. The models supported by Pixhawk include multi-copter drones, generally fixed wings, standard VTOL, Tailsitter VTOL, and Rover. In this study, the multi-copter drone model was used. 3DR Robotics' iris model was used as the UAV model (Figure 6). By using an actual model, more reliable results can be obtained from the simulation results. A total of four UAVs were set up, and the launch file was used to set up the initial position, attitude, UDP communication port with the simulation program, UDP communication port with the ground control system, and packages to be executed together.

Figure 7 shows a part of the launch file and declares the UAVs as a group through the <group ns => command. Then the parameter settings for the UAV are set up. The ID part assigns a unique number to the corresponding UAVs, which causes an error in the simulation execution. The fcu_url part sets the UDP communication port between the Gazebo model and the simulation program, and the following part declares the position, attitude, and so on. Through these settings, two or more multi-UAVs or heterogeneous unmanned vehicles can be used in the Gazebo simulation.

```
<launch>
        <!-- vim: set ft=xml noet : -->
        <!-- example launch script for ArduPilotMega based FCU's -->


        <argname="fcu_url"default="/dev/ttyACM0:57600" />
        <argname="gcs_url"default="" />
        <argname="tgt_system"default="1" />
        <argname="tgt_component"default="1" />
        <argname="log_output"default="screen" />
        <argname="fcu_protocol"default="v2.0" />
        <argname="respawn_mavros"default="false" />


        <includefile="$(find mavros)/launch/node.launch">
        <argname="pluginlists_yaml"value="$(find mavros)/launch/apm_pluginlists.yaml" />
        <argname="config_yaml"value="$(find mavros)/launch/apm_config.yaml" />


        <argname="fcu_url"value="$(argfcu_url)" />
        <argname="gcs_url"value="$(arggcs_url)" />
        <argname="tgt_system"value="$(argtgt_system)" />
        <argname="tgt_component"value="$(argtgt_component)" />
        <argname="log_output"value="$(arglog_output)" />
        <argname="fcu_protocol"value="$(argfcu_protocol)" />
        <argname="respawn_mavros"default="$(argrespawn_mavros)" />
                </include>
        </launch>
```

Figure 2: Script contents of the launch file for ROS communication in the USV (apm.launch).

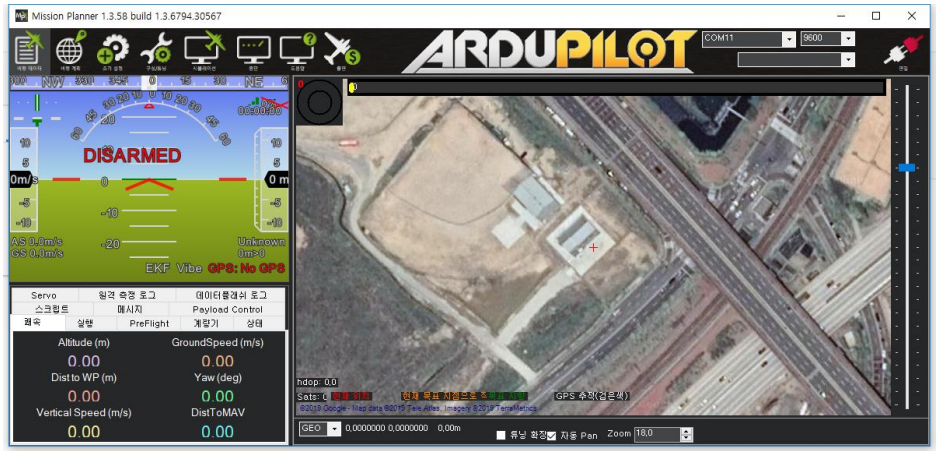


Figure 3: Mission planner.

Figure 8 shows the screen in Gazebo from when the launch file is executed after setting each parameter.

The QGroundControl program was used as the ground control system to control the UAVs (Figure 9). QGroundControl provides a convenient environment for Pixhawk users and includes firmware installation, parameter changes, automatic flight setup, and log recording. Also, it is possible to control multiple platforms with one ground control system.

Cooperative system configuration using ROS

ROS

ROS is an open source meta-operating system that is widely used in robotics. It provides services that can be expected from an operating system, including hardware abstraction, low-level device control, commonly used functionality implementation , inter-process message delivery, and

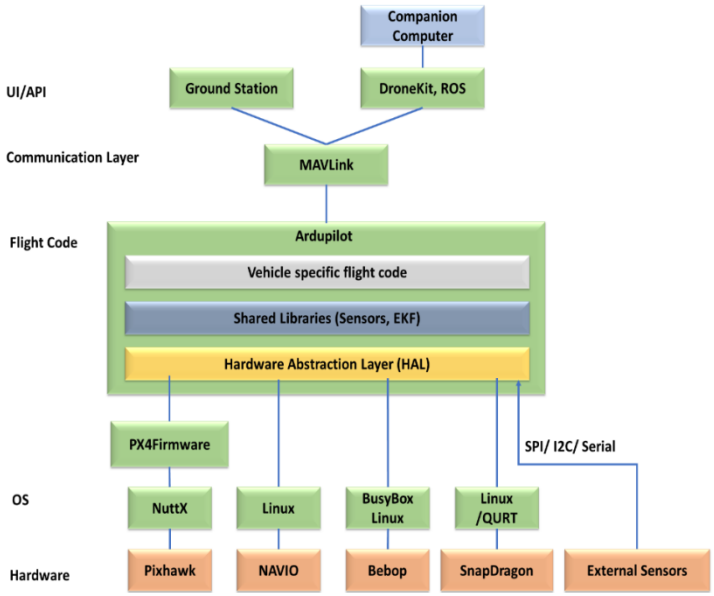


Figure 4: Ardupilot basic structure.

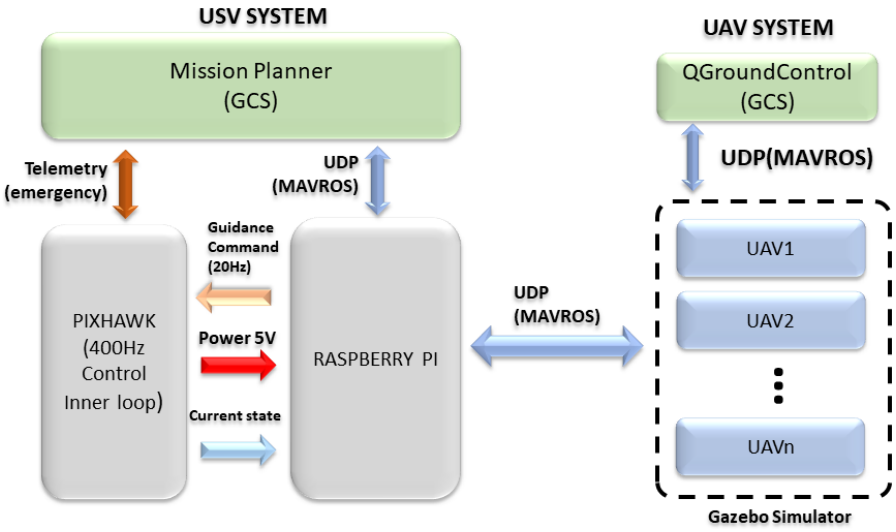


Figure 5: System diagram.

Figure 6: Iris UAV (left: Gazebo; right: real model).

```
<!--UAV1 -->
<groupns="uav1">
<!-- MAVROS and vehicle configs -->
<argname="ID"value="1"/>
<argname="fcu_url"default="udp://:14541@localhost:14581"/>
<!-- PX4 SITL and vehicle spawn -->
<includefile="$(find px4)/launch/single_vehicle_spawn.launch">
<argname="x"value="0"/>
<argname="y"value="0"/>
<argname="z"value="0"/>
<argname="R"value="0"/>
<argname="P"value="0"/>
<argname="Y"value="0"/>
<argname="vehicle"value="$(arg vehicle)"/>
<argname="mavlink_udp_port"value="14561"/>
<argname="ID"value="$(arg ID)"/>
</include>
<!-- MAVROS -->
<includefile="$(find mavros)/launch/px4.launch">
<argname="fcu_url"value="$(argfcu_url)"/>
<argname="gcs_url"value=""/>
<argname="tgt_system"value="$(eval 1 + arg('ID'))"/>
<argname="tgt_component"value="1"/>
</include>
</group>
```

Figure 7: Part of UAV launch file script.

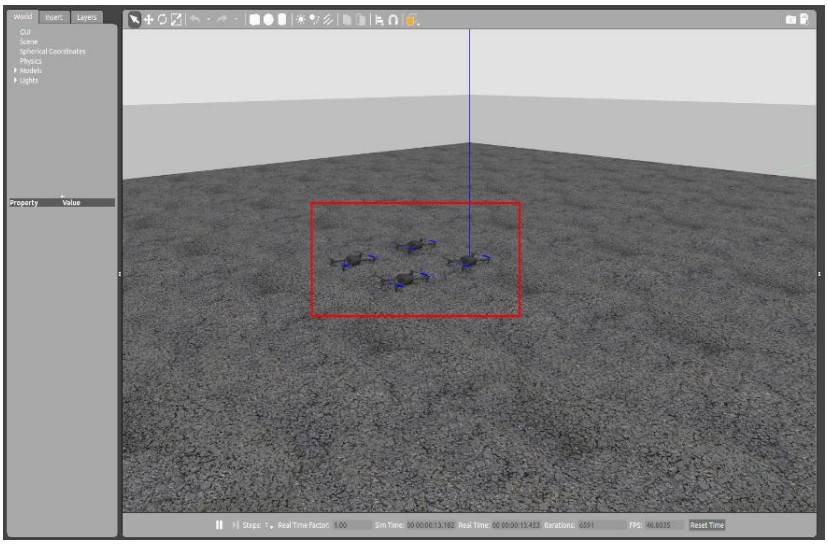


Figure 8: Gazebo with 4 UAVs.

package management. It also provides tools and libraries to obtain, write, and execute code on multiple computers.

ROS communicates between processes using packages, nodes, topics, and messages (Table 3). A node is the smallest viable unit of the processor. These nodes communicate based on messages. A package is a tool for running one or more nodes. To execute several nodes at once, the nodes are packaged into one package and executed so that the nodes are executed at one time. A message is an element that is set for communication between nodes in the form of variables such as integer, floating point, and Boolean values.

MAVROS

MAVROS is the ROS package of MAVLink for ROS communication in MAVLink. MAVLink is a lightweight,

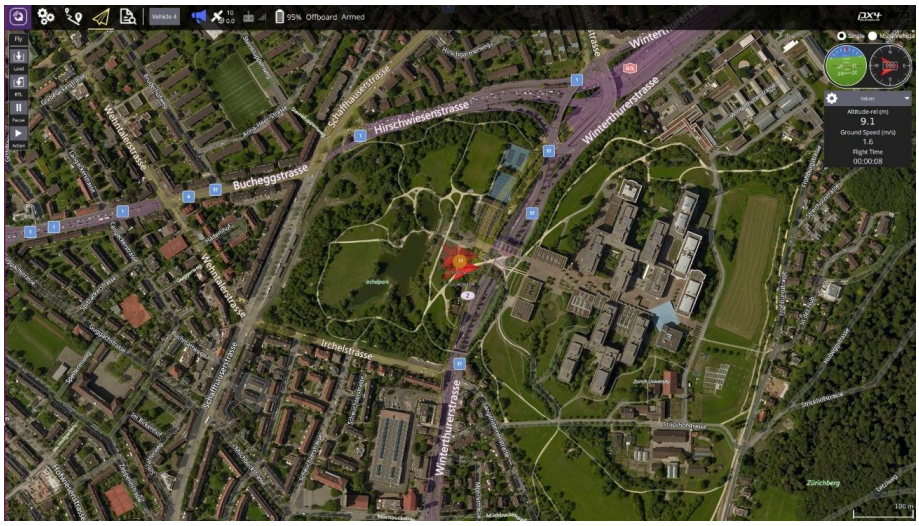


Figure 9: QGroundControl with 4 UAVs.

Table 3: ROS components.

Component	Contents
Node	Minimum available unit
Package	Tools to run one or more nodes
Message	Variables set for communication between nodes

Figure 10: Real-simulation communication.

header-only message marshaling library for small aircraft. MAVLink follows a modern hybrid publish-subscribe structure and point-to-point design pattern and is optimized for applications with highly limited communication bandwidth, eliminating the need for additional frames. The C or C++ reference implementation is optimized for systems with limited RAM and flash-memory resource constraints. The MAVROS package provides communication drivers for various systems using MAVLink and can be used in ground control systems through UDP communications. In this study, we used MAVROS communication among USV and UAVs. MAVROS provides several topics, in which the user sends and receives the topics necessary for system configuration. The main topics used in this study are the /clock, local_position/pose, and set_position/local. This is the part that enables interworking between the real system and simulation.

In Figure 10, the local_position/pose is set to the home point corresponding to the coordinates (0, 0, 0), which is the arming point of the USV for the navigation, and the relative distance based on the NED coordinate system is a topic to represent. Using this, we assume the USV and UAVs in the same space using a USV's home point. In this system,

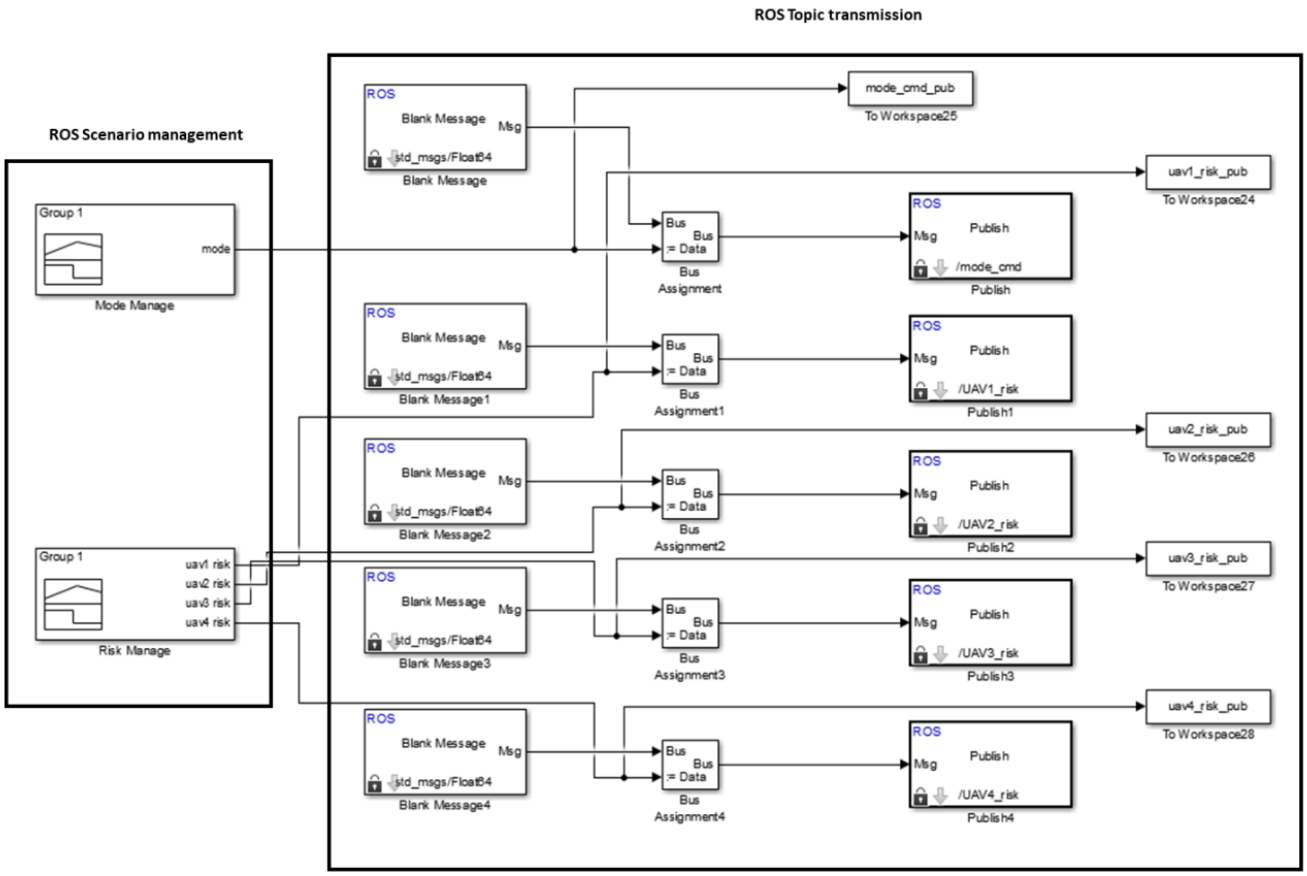


Figure 11: Construction of User defined topic using MATLAB/SIMULINK.

the USV transmits the position data continuously by transmitting the mavros/local_position/pose topic.

Four UAVs were constructed using the Gazebo simulation program. UAVx_mavros/state is used to send a message to arm the UAVs and to switch to OFFBOARD mode to use ROS communication via mavros/set_mode. When these two processes are executed, the UAVs are ready for ROS communication, and the four UAVs also continuously transmit the UAVx_mavros/local_position/pose topic and receive the position data of the USV and UAVs to share the position data. Based on these position data, each UAV receives a position command, which is UAVx_mavros/set_position/local. In addition, the topic for mission performance is set and transmitted/received through the ROS network

RESULTS AND DISCUSSION

In this study, a User defined topic was constructed using MATLAB/SIMULINK in addition to the MAVROS basic topic. The package of each UAV was configured directly using the Catkin tools, and the mission algorithm of each UAV was constructed using the C language.

Construction of user defined topic using MATLAB/SIMULINK

The user-defined topic was constructed using the MATLAB/SIMULINK program in addition to the topic used in MAVROS. MATLAB/SIMULINK supports ROS in Robotics System Toolbox.

Figure 11 shows a User-defined topic using MATLAB/SIMULINK. On the left, it is a block that manages mode and risk signal. It constitutes a mission scenario through time signal control. The Blank Message block is a block that determines the type of the Topic, including std_msgs, geometry_msgs, and so on. The Topic generated through the Publish block is transmitted to the ROS network, and each Unmanned Vehicle performs its mission using these Topics.

Simulation of topic transmission/reception

The generated User-defined topics were checked through simulation to see if the transmission/reception is normally performed through the ROS network. The topics were transmitted to and received from the ROS network

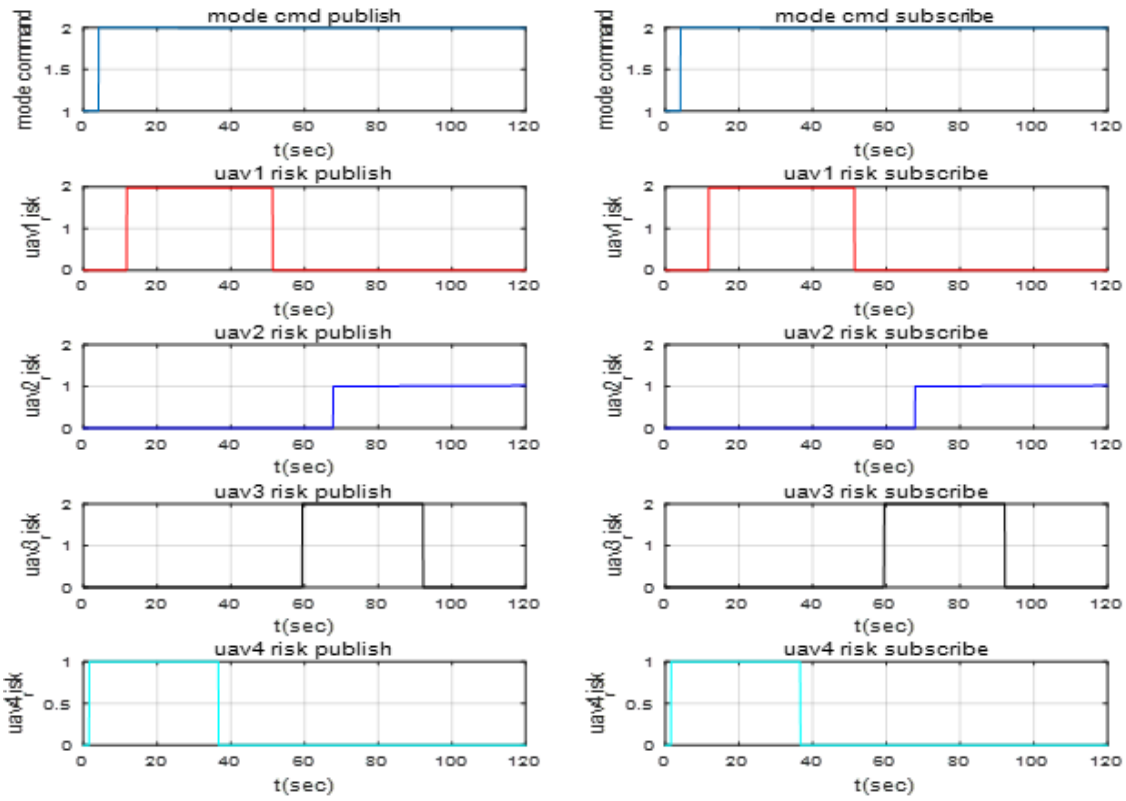


Figure 12: The result of User defined topic transmission/reception simulation.

generated by Raspberry Pi, and the results of the transmitted data were compared with those of the received data. The results are shown in Figure 12. At this time, the publish data represents the transmitted data and the subscribe data represents the received data.

Based on Figure 12, it can be confirmed that the topic transmission/reception is normally performed in the ROS network.

Mission simulation

After confirming that the Topics are normally transmitted and received through the simulation, the mission simulation is performed by setting the mission algorithm and scenario of the UAVs using the ROS package (Figure 13). In the basic MAVROS environment, since no separate mission scenario is set through MAVROS, only vehicle information is transmitted / received through MAVROS. In this study, scenarios were created for each UAV using User defined topic, and the mission according to the scenario set in each UAV was executed.

Algorithm construction

The mission algorithm of this study is shown in Figure14. If the value of mode_cmd is kept at 1, UAVs follow the USV's position and form the basic formation. If the value of the mode_cmd topic is changed to 2, the mode is changed to the mission mode. At this time, the target is captured, and it is determined whether the mission is performed through the occurrence of the risk. If UAVs do not perform a mission, UAVs keep on a basic formation and divide into high-risk or general missions when UAVs perform a mission (Figure 15). When a general mission occurs, the mission is performed by turning the position of the mission to a certain radius. When a high-risk mission occurs, a signal for requesting help is transmitted to other UAVs. At this time, the UAV receiving the help request decides whether or not to collaborate according to whether the UAV is currently performing the mission. Since each UAV has priority in the current mission, if the current mission is being executed even though the help request signal is received, the current mission is performed first. If the mission is not underway, it moves to the position of the UAV that sent the help request signal and performs the collaboration mission. After completing the mission and confirming whether the help request is received from the other UAVs or if the help request signal is not received, the basic form is formed around the USV's position and when the help request signal is received, the collaborative mission is carried out. If a high-risk mission occurs in a large number of UAVs, collaboration is established by a predetermined coupling

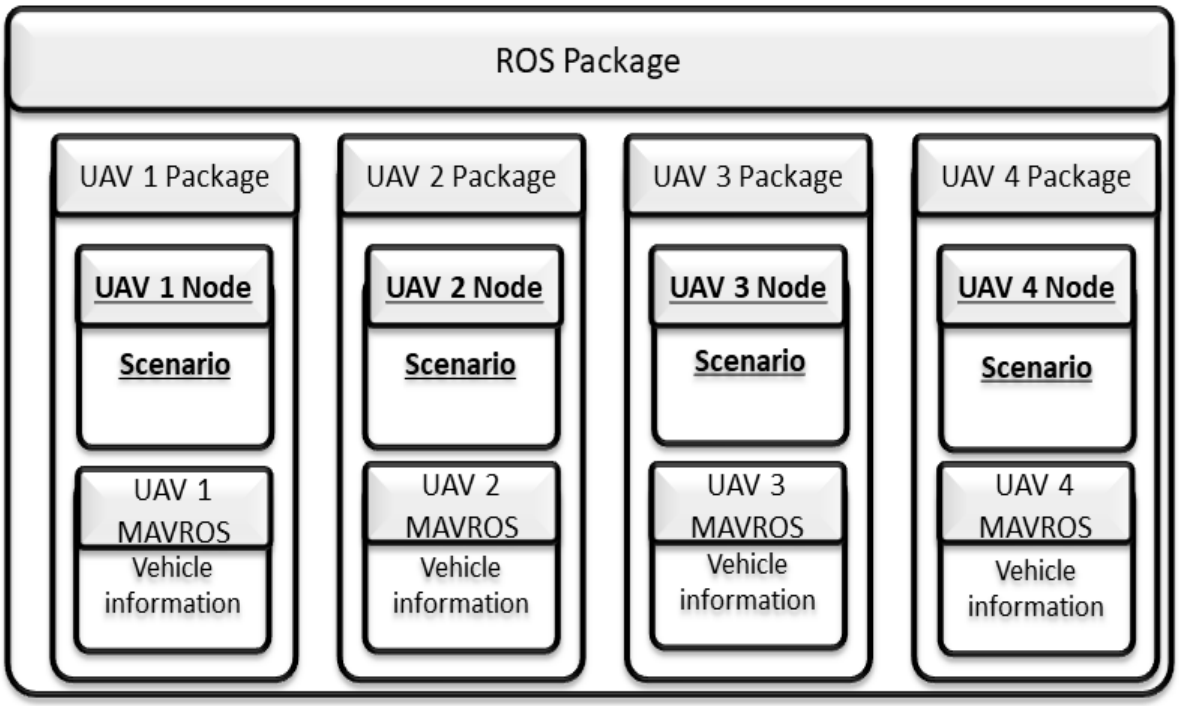


Figure 13: Mission configuration using ROS package.

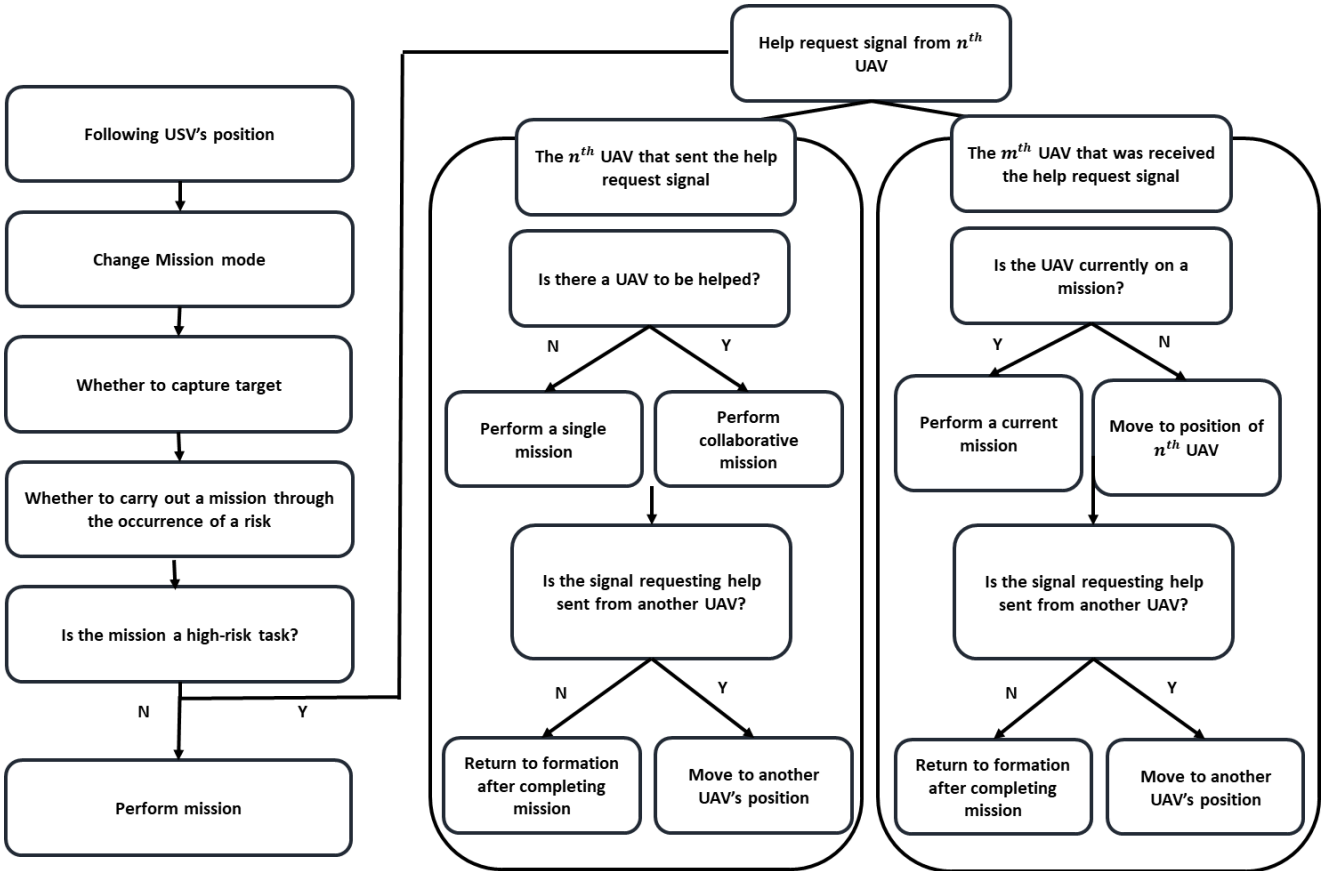


Figure 14: Mission algorithm.

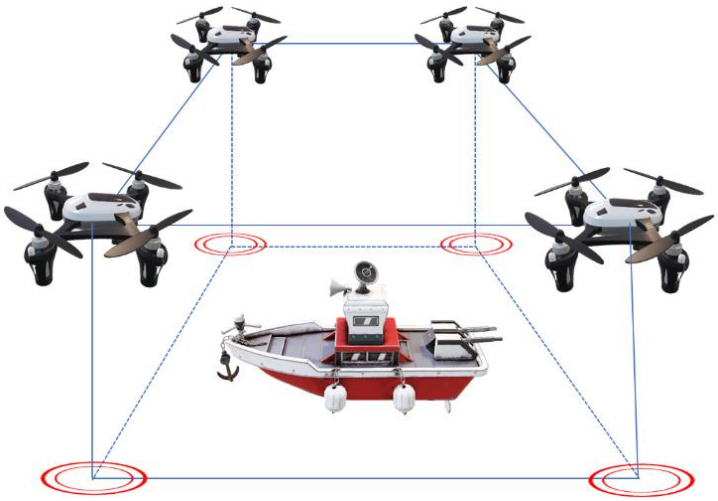


Figure 15: Formation configuration overview.

relationship. This coupling relationship is set to change in accordance with the high-risk mission situation.

Scenario construction

The basic structure of this study is shown in Figure 18. Each UAV is spaced by 5 m in the X and Y directions centered on the USV, and the basic altitude is also set to 10 m.

The purpose of this study was to construct an integrated simulation environment between heterogeneous unmanned vehicles using ROS. Therefore, we verified the feasibility of the system by simulating a simple mission situation rather than constructing and maintaining a formation algorithm or collaboration algorithm. As a result, we used the User-defined topic created to perform the algorithm shown in Figure 14. The position in sailing is transferred to the ROS network as mavros/local_position/pose. Each UAV receives the UAV's position in the ROS network and received the position command in the 20 Hz cycle through the UAVx_mavros/set_position/local topic. The Pixhawk is controlled at a cycle of 400 Hz in order to converge to the received position command. Also, the safety radius was set to prevent collision between UAVs. The safety radius means the distance required to avoid the obstacle or other UAV within a certain distance of the UAV. In this study, collision avoidance through altitude change is performed when the relative distance between UAVs approaches the set safety radius. In this study, the safety radius was set to 1.5 m.

The overview of the formation according to the number of missions is shown in Figure 19. It is assumed that a mission can occur from a minimum of 0 to a maximum of 4.

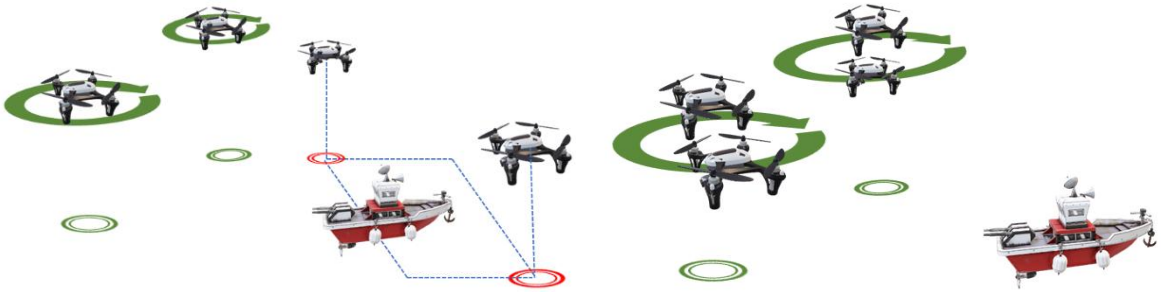
Table 4 shows the parameters used when performing the mission. The mission we tried to simulate in this study is the monitoring of illegal fishing boats. When the target is captured in the surveillance area of the UAV through the Mission_chk variable, the signal is transmitted 1, and when UAV is not captured, the signal of 0 is transmitted. If the target is captured and generates a signal of 1, the target_risk variable is used to determine whether or not to perform the mission. Target_risk's variable setting range is $0 \sim 2$ and if it is 0, it is recognized as a float or a legitimate fishing boat. If it is 1, the UAV captured the illegal fishing boat. If the risk level is 2, the illegal fishing boats are large, and the UAVx_help signal is sent as 1 to request help from other UAVs.

The scenario of this study is set as shown in Figures 20 and 21. In the User-defined topic, we set the timing, the timing of occurrence of risk, and the risk value through the Mode and Risk management block (Figure 17). UAVs are operated, and each UAV performs the set scenarios and missions.

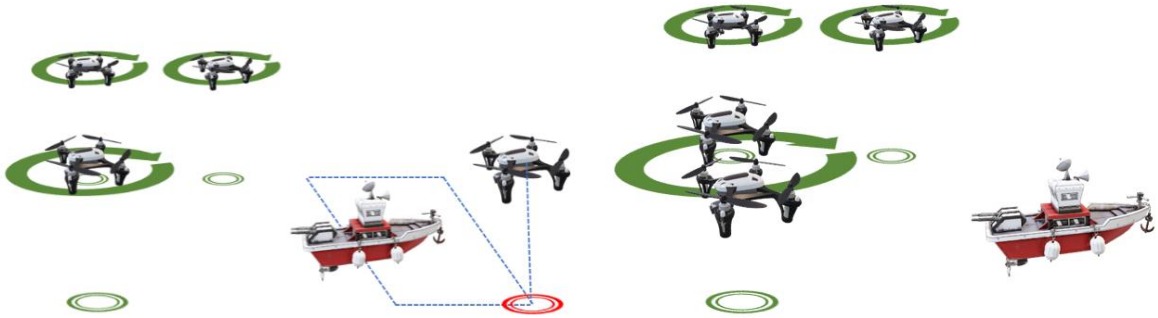
Experiments results

An outline of the experiment is shown in Figure 16. The purpose of this study was to verify the feasibility of the system by a simple position command rather than constructing and verifying separate flight maintenance or collaboration algorithm. In addition, using MATLAB/SIMULINK, /mode_cmd topics were created, and the mode was changed so as to move to a specific position rather than follow the USV's position in pairs in search or rescue situations. Squares of 5-m intervals in the X and Y directions were formed using the GPS location of the USV. Since UDP communication is mainly used in ROS communication, a router and a WIFI repeater are installed as separate equipment for a smoother UDP communication environment. The user sets the route point using Mission Planner and then transmits it to the Pixhawk using ROS communication.

When there is 1 mission (left : general mission / right : high-risk mission)



When there are 2 missions (left : general mission / right : high-risk mission)



When there are 3 missions (left : general mission / right : high-risk mission)

When there are 4 missions

Figure 16: Formation overview for mission number.

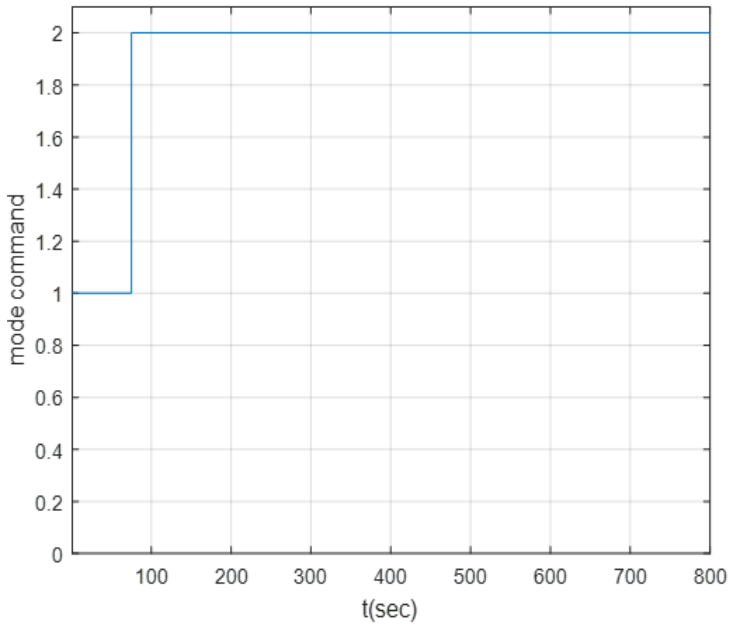


Figure 17: Full scenario mode command settings.

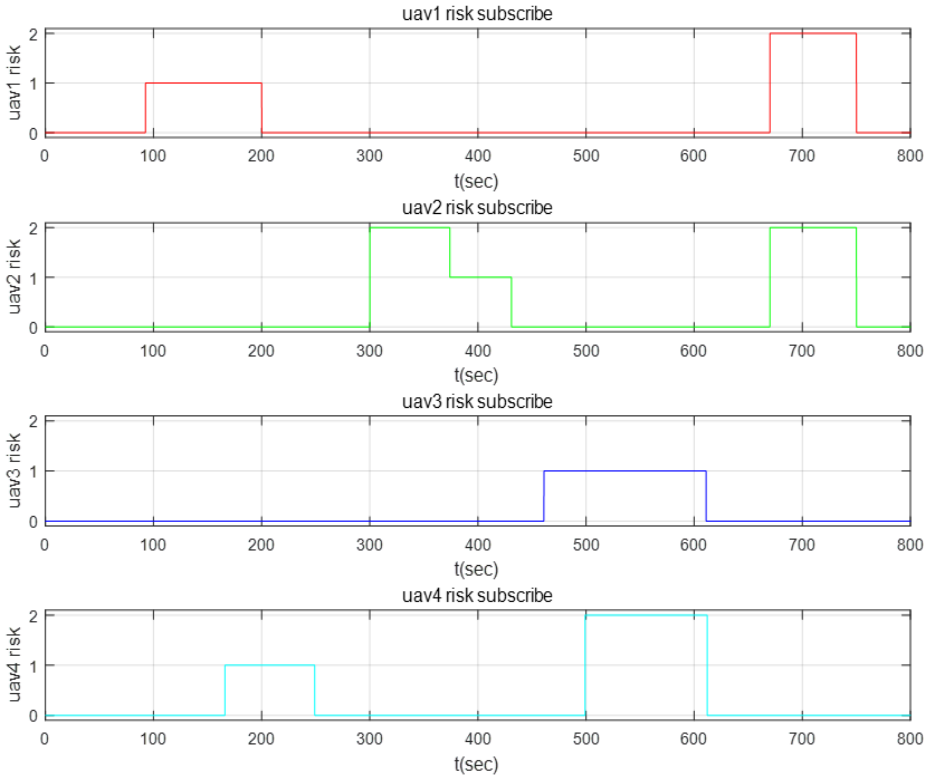


Figure 18: UAVs risk setting of all scenarios.

In the navigation of the route point, the USV position is transmitted at a frequency of 2 Hz using the mavros/local_position/pose topic. Each UAV receives the position of the USV and the position of the UAVs and transmits the position command at 20 Hz through the UAVx_mavros/set_position/local topic based on it. The Pixhawk controls the transmitted position command at a cycle of 400 Hz. Four scenarios were conducted to prove

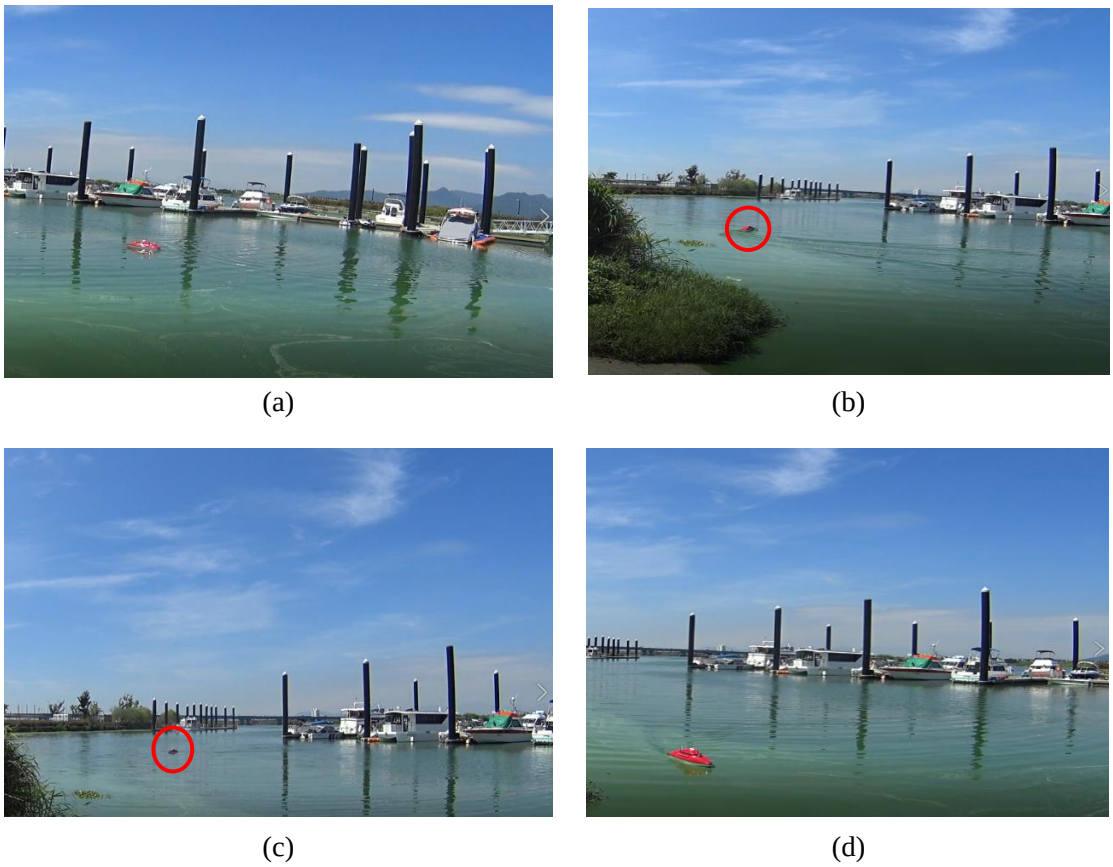


Figure 19: Sailing experiment ((a) : Sailing start (b): During mission (c): During another mission (d) : Return).

Table 4: Mission usage parameters.

Parameter	Contents	Value
Mission_chk	Target detection	0 or 1
Target_risk	The risk of detected targets	0 ~ 2
UAVx_help	Whether to request for help from a UAV	0 or 1

that heterogeneous communication is possible using ROS communication.

Scenario 1: Following the USV's position, which constitutes the basic formation, general missions occurs in one UAV for about 90 s. Because it is not a high-risk mission, it does not request for help from another UAV, performs its mission alone, and returns to its basic formation again after 200 s to change its risk level to 0. In addition, the general mission occurs in about 170 s from UAV 4, and the mission also performs alone. In 250 s, the risk level changes to 0, and the UAV 4 returns to the basic formation and follows the USV's position. Figure 21 shows the trajectory for the case and confirms that it is performing the mission according to the risk. Figure 22 shows the risk graph between the UAVs of the corresponding scenarios during the single mission out of the total scenarios.

Scenario 2: In this case, a risk level 2 occurs in the UAV 2 from 300s, and a high-risk mission occurs. Since the high-risk mission has occurred in the UAV 2, a request signal is sent to the UAV to request help. The other UAVs receive the help signal, and the UAV 1, UAV 3, UAV 4 move to the UAV 2's position and perform the collaborative mission. Since then, the risk of UAV 2 has changed to 1 in the vicinity of 370s, and it has been converted into a general mission, not a high-risk mission. Therefore, UAV 2 does not need help by switching the value of the help request signal to 0. UAV 1, 3, 4 do not ask for help from the UAV 2, and they do not need help from other UAVs, so they return to the basic formation and follow the USV's position. Figures 23 and 24 show the resulting trajectory and risk graph for that case. At this time,

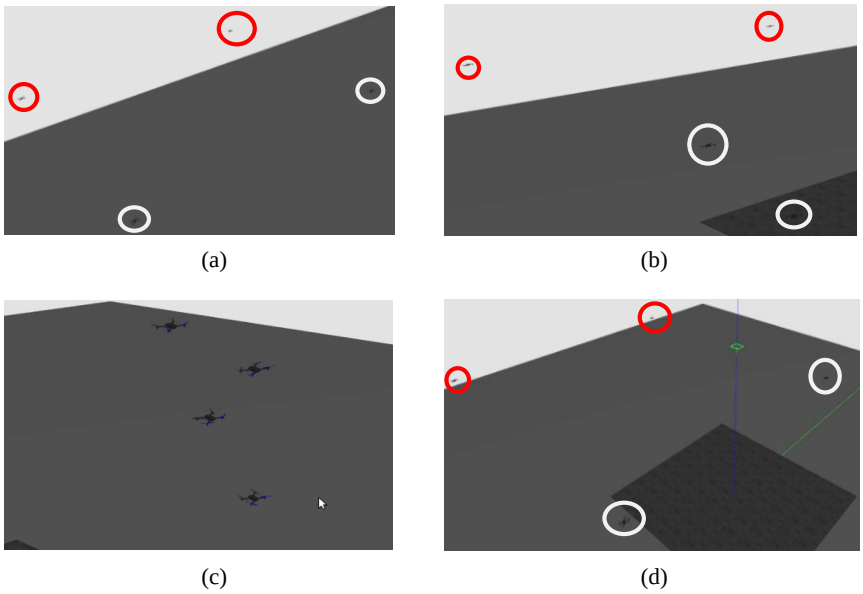


Figure 20: Gazebo simulation ((a: Flight start (b): During mission (c): During another mission (d): Return).

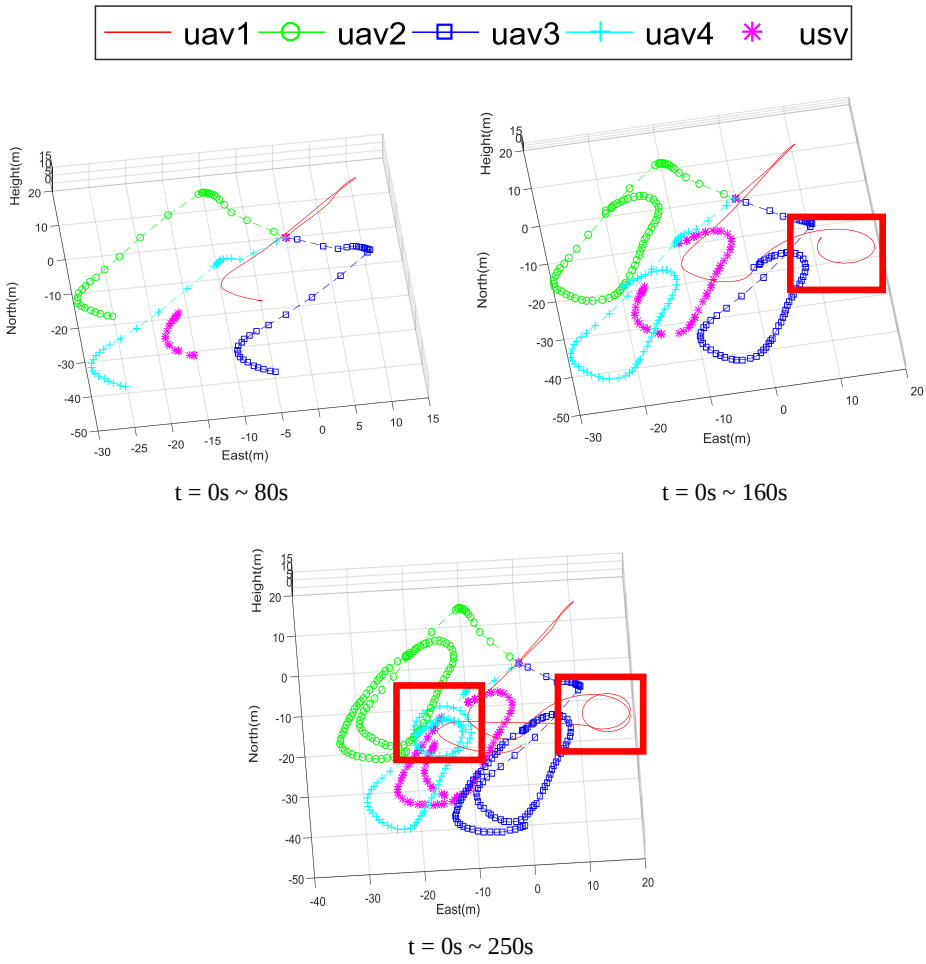


Figure 21: Trajectory when performing single mission.

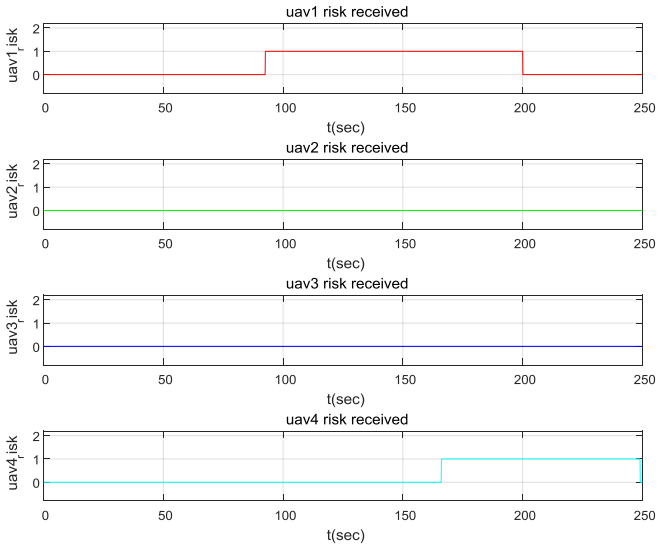
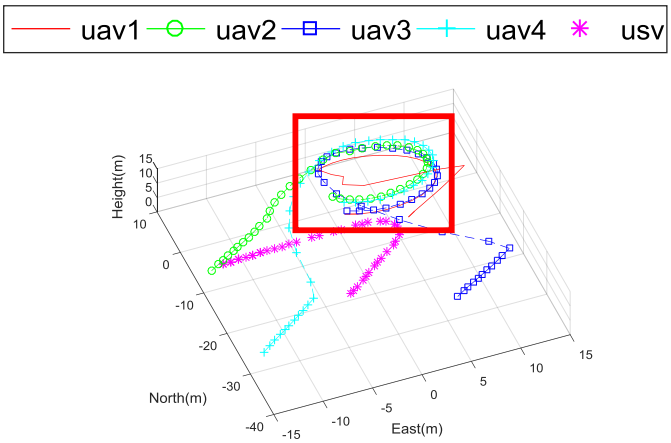
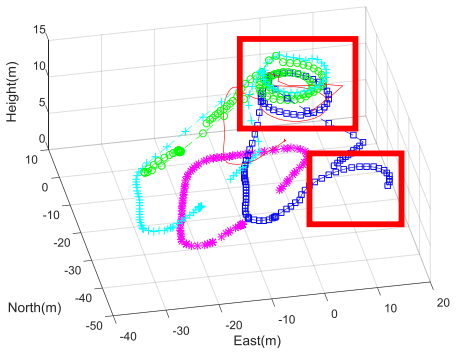


Figure 22: A graph of the risk between UAVs when performing single mission.



t = 280s ~ 360s



t = 280s ~ 500s

Figure 23: The trajectory in case of a mission change from high-risk mission to general mission.

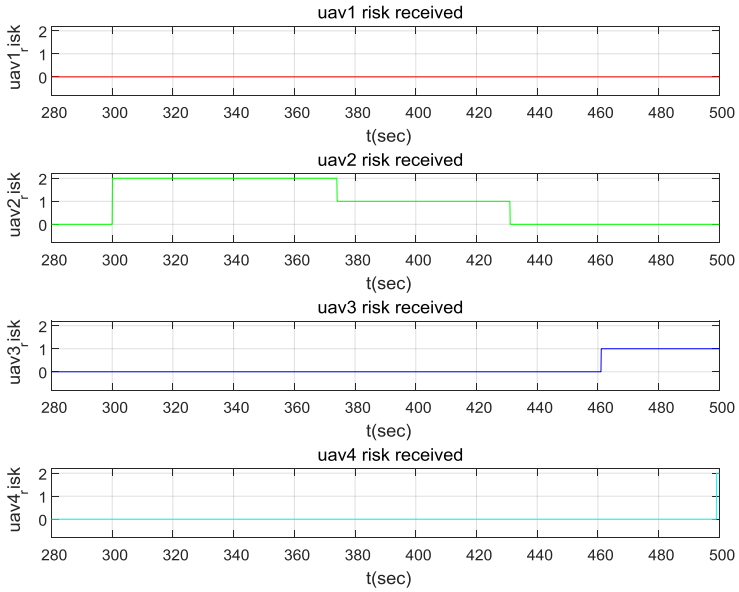


Figure 24: A graph of the risk between UAVs when performing mission change case.

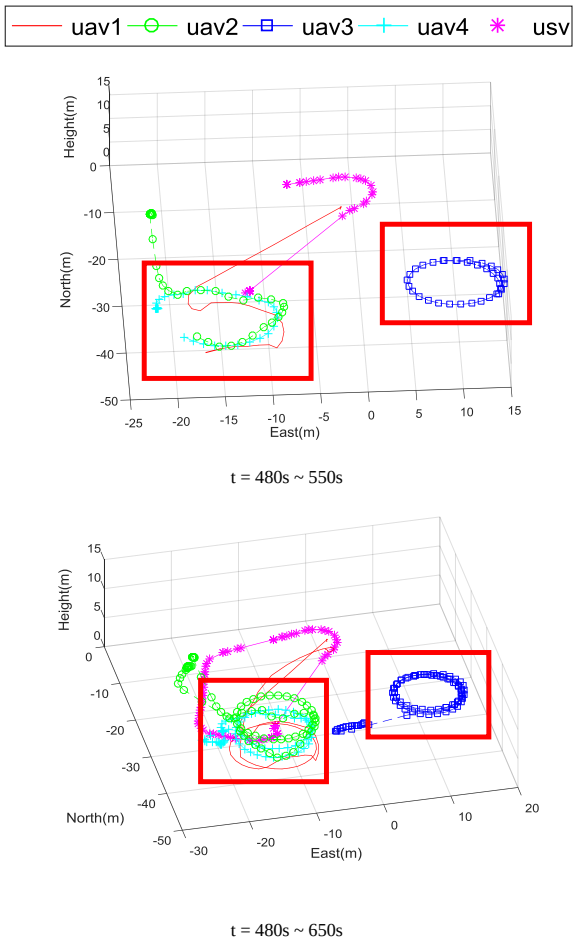


Figure 25: The trajectory in case when occurs high-risk mission during performing mission.

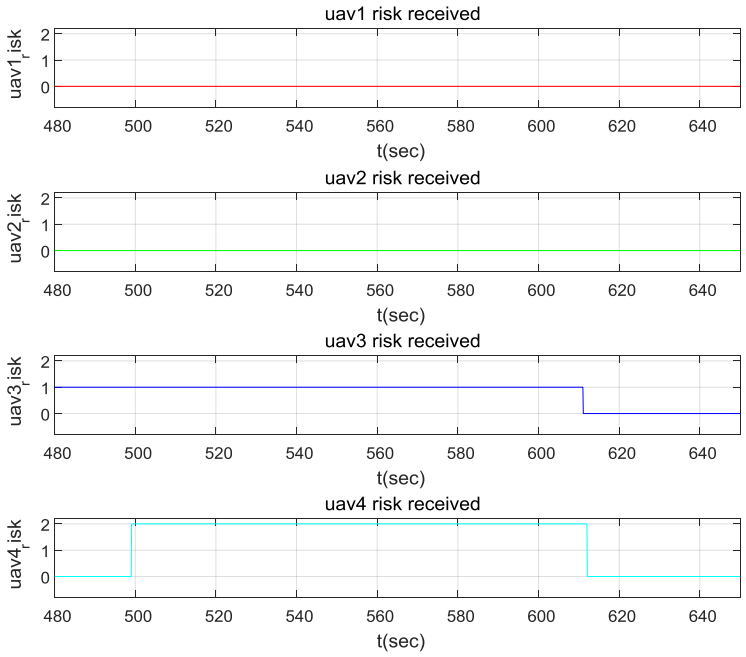
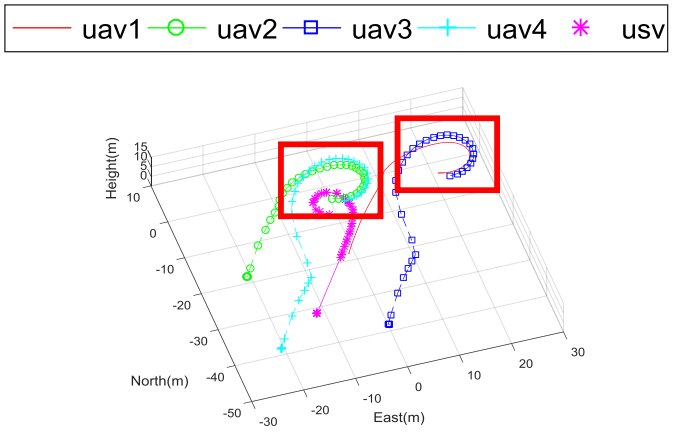
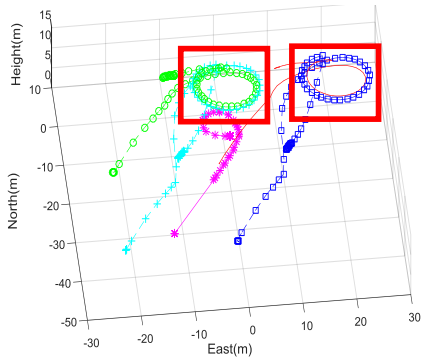


Figure 26: A graph of the risk between UAVs when a high-risk mission occurs during performing mission.



t = 650 ~ 720s



t = 650 ~ 800s

Figure 27: The trajectory in case of high-risk missions.

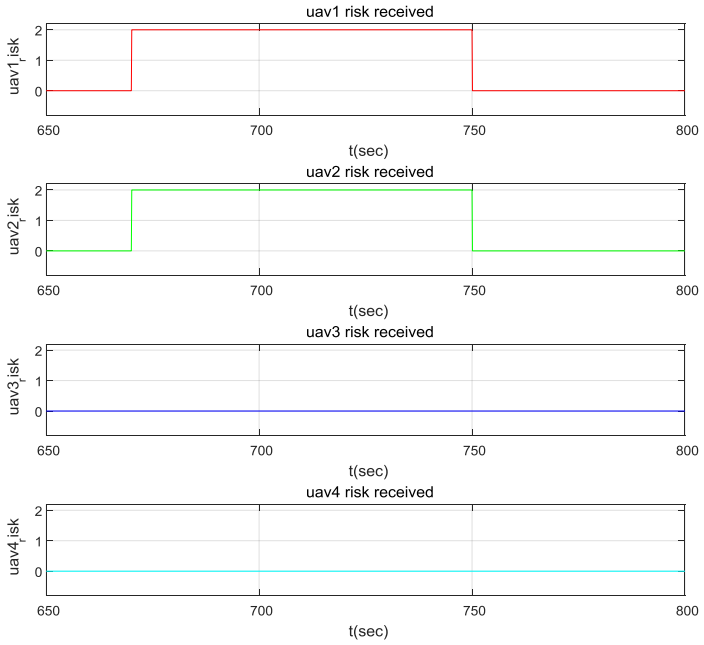


Figure 28: A graph of the risk between UAVs when performing high-risk missions.

it can be confirmed that each UAV has changed altitude for collision avoidance.

Scenario 3: In this case, a general mission of risk 1 occurs near 460s in the UAV 3, and a high-risk mission of the risk 2 occurs in the UAV 4 from 500s during the mission. At this time, the UAV 3 is continuously performing the mission being performed because the UAV is in the process of performing the mission, and the UAV 1 and UAV 2 that are not performing the mission move to the UAV 4 and perform the mission together. In the vicinity of 620s, the risk of UAV 3 and 4 are changed to 0, so they return to the basic formation and follow the USV's position. Figures 25 and 26 show the results. At this time, UAV 1, 2, and 4 can be confirmed to have changed altitude to avoid the collision.

Scenario 4: In this case, the high-risk mission of risk 2 occurs in 670s in the UAV 1 and the UAV 2. The UAV 1 and UAV 2 request help through the help request signal, and the UAV 3 and UAV 4 receive the UAV and UAV 2 help requests, respectively. To perform the mission together, the basic coupling relationship is set as UAV 1 – UAV 2 and UAV 3 – UAV 4, but in this case, UAV 1 and UAV 2 require help, so the coupling relationship is changed so that UAV 1 – UAV 3 and UAV 2 – UAV 4 are coupled. Figures 27 and 28 show the trajectory and risk graph of each UAV, respectively.

Conclusion

In this study, the integrated simulation environment of heterogeneous unmanned vehicles was constructed using ROS, and the feasibility of the collaboration framework was verified using the system and simulation program using Pixhawk and Raspberry Pi.

The hardware used in this study was Pixhawk 2.1 and Raspberry pi 3b+, and ArduRover, PX4Pro, and ROS were used as software. The UAV system was configured using the Gazebo simulator, and the ROS package was used to switch to the OFFBOARD mode, which enables automatic ROS communication instead of manual mode switching. We also used the MAVROS package to construct a collaboration framework for USV and UAVs, and the mission algorithm was constructed as a ROS package using catkin tools. To generate a topic and confirm that it is applicable, a mode_cmd, target_risk, and UAVx_help topic was created using MATLAB/SIMULINK and a simulation to verify whether the mission is performed normally according to totally 4 scenarios or not was conducted.

The simulation runs ROS MASTER in Raspberry pi of the USV system, and after creating the ROS network, 4 Gazebo's UAVs were connected to the ROS network at each node. Each of the UAVs is formed using the position of the USV and the position between the UAVs. When switching to the mission mode, the target acquisition signal was generated. When the mission was performed, the signal changed as 1. At this time, the system was configured to request help from other UAVs that were not carrying out missions when a high-risk target was captured and classified as a high-risk target.

In this study, the realization of the ROS network in the real environment was verified through the construction of

the integrated simulation environment and the topic transmission/reception between the USV and the simulator UAVs, and a simulator was used for the UAVs with high risk such as a crash. To verify the feasibility of the ROS network, only simple position commands were used to verify the simulation environment, not the previously studied formation or collaboration algorithms. Through this experiment, it can be confirmed that the ROS is used to design a certain formation and execute the mission with just a few commands. Also, since it is possible to transmit not only the position command but also the values such as the speed and the acceleration as command values, it is considered that the development period of the collaboration system between the heterogeneous Unmanned Vehicles will be further shortened using the already studied formation or collaboration algorithm.

As a future work, the UAV system will conduct verification using actual UAV, not simulations, and more efficient collision avoidance will be possible that the collision avoidance through altitude change through the previously studied collision avoidance algorithm.

The framework presented in this study can be used as a solution to vertical development and high entry barriers in the ground, marine and aviation industries

REFERENCES

Bechtsis D, Moisiadis V, Tsolakis N, Bochtis D, Vlachos D (2017). Scheduling and control of unmanned ground vehicles for precision farming: A real-time navigation tool. In CEUR Workshop Proceedings, pp. 180-187.

Braga RG, Da Silva RC, Ramos AC, Mora-Camino F (2017). UAV swarm control strategies: A case study for leak detection. In 2017 18th International Conference on Advanced Robotics (ICAR), pp. 173-178. IEEE..

Dongki H, Hyeok R, Jaeeun K, Bumjin P (2017). "Current Status of Technology for Autonomous Cooperative Operation of Multiple-Heterogeneous Unmanned Vehicles." Proceeding of the Korean Society for Aeronautical and Space Sciences(KSAS) conference.

Imdoukh A, Shaker A, Al-Toukhy A, Kablaoui D, El-Abd M (2017). Semi-autonomous indoor firefighting UAV. In 2017 18th International Conference on Advanced Robotics (ICAR) (pp. 310-315). IEEE.

Kim G, Lee S, Park H (2016). Design and Development of Integrated Video Transmission Receiver for Aerial Photograph Using Drone. Proceedings of Symposium of the Korean Institute of communications and Information Sciences. pp 130-131.

López E, García S, Barea R, Bergasa L, Molinos E, Arroyo R, Pardo S (2017). A multi-sensorial simultaneous localization and mapping (SLAM) system for low-cost micro aerial vehicles in GPS-denied environments. Sensors, 17(4): 802.

Mogili UR, Deepak BBVL (2018). Review on application of drone systems in precision agriculture. Procedia Comput. Sci. 133: 502-509.

Park S, Deyst J, How J (2004). A new nonlinear guidance logic for trajectory tracking. In AIAA guidance, navigation, and control conference and exhibit p. 4900.

Portugal D, Iocchi L, Farinelli A (2019). A ROS-Based Framework for Simulation and Benchmarking of Multi-robot Patrolling Algorithms. In Robot Operating System (ROS) Springer, Cham. pp. 3-28.

Sinisterra A, Dhanak M, Kouvaras N (2017). A USV platform for surface autonomy. In OCEANS 2017-Anchorage IEEE, pp. 1-8.

Specht C, Świtalski E, Specht M (2017). Application of an autonomous/unmanned survey vessel (asv/usv) in bathymetric measurements. Polish Marit. Res. 24(3): 36-44.

St-Onge D, Varadharajan VS, Li G, Svogor I, Beltrame G (2017). ROS and Buzz: consensus-based behaviors for heterogeneous teams. arXiv preprint arXiv:1710.08843.

Youn W, Cho IH, Kim T-S (2016). Development Status of Open-architecture based Flight Control/Mission Computer for Small UAV. Curr. Ind. Technol. Trends Aerosp. 14(2): 116-123.