Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Cloud-based remote control framework for unmanned surface vehicles [⋆]

Zhao Wang [*] **Shaolong Yang** [*, **] **Xianbo Xiang** [*, **]
Antonio Vasilijević [***] **Nikola Mišković** [***] **Đula Nađ** [***]

[*] *School of Naval Architecture and Ocean Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xbxiang@hust.edu.cn)*
[**] *Hubei Key Laboratory of Naval Architecture and Ocean Engineering Hydrodynamics (HUST), Wuhan 430074, PR China (e-mail: xbxiang@hust.edu.cn)*
[***] *Laboratory for Underwater Systems and Technologies, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia (e-mail: dula.nad@fer.hr)*

Abstract: This paper proposes a cloud-based mission control framework for a fleet of unmanned vehicles. The framework enables easy remote access to fleet operations regardless of operator location. By leveraging cloud-based technologies the framework accomplishes scalable monitoring, remote control, data acquisition and sharing. While the front-end is applicable across mobile robotic systems, the back-end presented in this paper provides integration with the Robot Operating System (ROS); thus, enabling integration of various marine robotic agents based on the same robot framework. The proposed framework operation is demonstrated on the H2OmniX unmanned surface vehicles during trials in Biograd na Moru, Croatia.

Keywords: remote control, distributed control, cloud-based control, Robot Operating System

1. INTRODUCTION

With increasing focus on the exploration and exploitation of ocean resources in recent decades, the marine community has witnessed remarkable growth and development in marine robotics. Unmanned surface vehicles (USV) have become common in marine robotics, executing missions such as source seeking (Junnan Song et al. (2013)), environmental monitoring and ocean sampling (Vasilijević et al. (2017)) in a coordinated way with multi-vehicles (Zhang et al. (2013), Wang and Han (2016)). With the limited range of commonly used communication technologies (WiFi and/or UHF), traditional monitoring and control platforms for marine robots are often co-located with the deployed robotic systems. Assuming constant monitoring is desired, this heavily reduces the operating area of the vehicles. From another perspective, if one of the vehicles is lost, the operator will be unable to recover data from the lost vehicle. Similarly, if the control platform is corrupted the fleet operation will be compromised. With improved connectivity and cellular networks providing bandwidths similar to WiFi, the current technological trend, as seen in Lorencik and Sincak (2013); Saha and Dasgupta (2018), is towards off-loading processing from local platforms onto the cloud.

In this article, the cloud-robotics framework leverages cloud technologies such as cloud computing, cloud storage and cloud visualization to better distribute monitoring and control tasks, thereby removing the bottleneck encountered in traditional approaches. The cloud-computing benefits (high processing capabilities, robust data storage and server distribution/redundancy) allow control commands and data to be transmitted and processed without limitation of distance or demanding hardware requirements (Zhang et al. (2017), Hu et al. (2012)). Once vehicles are equipped with mobile communication hardware and a CPU running the cloud client interface, the proposed distributed cloud architecture can be provided as a common solution for mobile robots. Similar cloud-robotics efforts have been widely applied to unmanned ground vehicles and unmanned aerial vehicles such as DronTrack (Koubaa and Qureshi (2018)), RoboEarth (Riazuelo et al. (2015)) and Rapyuta platform (Mohanarajah et al. (2015a), Mohanarajah et al. (2015b)).In marine robotics applications focus on ships (McGillivary and Zykov (2016); Kaliski (2008)) or shipping related data processing (Yicheng Zheng et al. (2014)). However, when considering coordinated mission control for unmanned surface vehicle fleets applications are scarce.

While the front-end is applicable across heterogeneous vehicle types, the back-end client interface is implemented within the Robot Operating System (ROS) framework (Quigley et al. (2009)). The framework is already well-

[⋆] The research leading to these results was partially funded by the European Union's Horizon 2020 research and innovation programme under the project EUMarinerobots (grant agreement No 731103), Contract CC-MICRO "Cloud-based coordinated mission control of fleet of ASVs", China-Croatia Bilateral Project CC-MARS "China-Croatia Collaboration on MArine Robotic System" , National Natural Science Foundation of China under Grant 51579111 and the Croatian Science Foundation IP-2016-06-2082 CroMarX project.

established in land robotics and continuously gaining popularity in the marine sector. Therefore, this implementation selection enables the use of the proposed cloud-based framework with many different types of robots, among which are the H2OmniX vehicles used for field demonstrations in this paper.

The paper will introduce the applied cloud-based fleet control architecture in Section 2. Components involved in the system are described in Section 3 while the simulation and pool experiment results are presented in Section 4. Finally, the paper is concluded with the recap of main concepts and planned future work for extending the proposed cloud-based control framework.

2. SYSTEM ARCHITECTURE

The architecture is designed around the usual client/server backbone. The minimum system consists of a cloud server and two clients. One remote client bridges the users/researchers with the vehicle via the cloud-server, while the other interface client ensures direct communication of the server with the vehicles. This way the vehicle and user network are clearly separated. The cloud-server, as the main data handler, implements the web interface for data sending and receiving. Considering that data has to be stored and indexed, a dedicated database is managed by the cloud-server. The remote client, in addition to providing the control interface for users, contains the web modules required to establish communication with the cloud server.

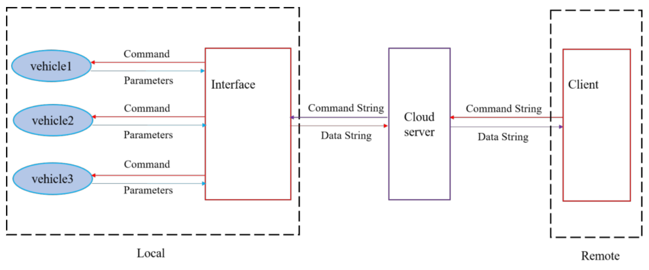


Fig. 1. Centralized client interface architecture: the centralized local interface and vehicles are separated. The centralized interface routes data to the cloud.

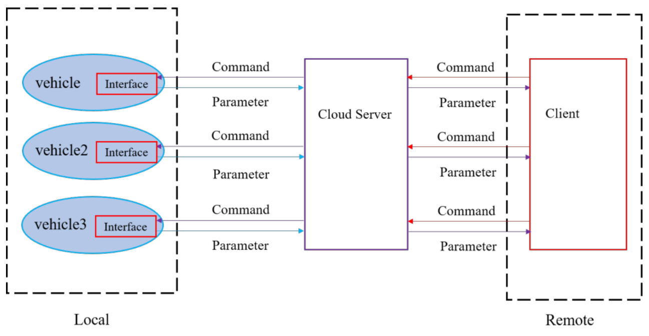


Fig. 2. Local client interface architecture: the client interface is localized on the vehicle and a direct connection to the cloud is possible.

There are two ways the vehicles can connect to the cloud. The first layout, shown in Fig. 1, provides a centralized interface and a star-like topology with the vehicles. In this layout, the control commands for all vehicles are concatenated into a single message to be sent to the cloud server. Similarly, the collected vehicle data and parameters are concatenated into a single message. With this centralized interface, serialized data is transmitted between the vehicles and the remote client. This serialized transmission allows easy data synchronization from multiple sources and is well suited for scenarios where such synchronization is necessary. Since it does not require hardware or software modifications on the vehicles, it is also the first step when introducing cloud-based extensions into a classical centralized mission control system.

The second layout, shown in Fig. 2, entails a local interface on the vehicles. Vehicles establish individual communication links with the cloud. This decentralized architecture does not provide centralized synchronisation, but it reduces the communication load between the interface and the cloud-server, especially with increasing number of vehicles. Decentralization also improves the flexibility concerning the used network layer without affecting the vehicle interface design. This allows for a more heterogeneous approach, as now vehicles with access to mobile or Iridium communication can connect to the cloud from larger distances directly, while other, more localized vehicles, can still utilize a wireless connection with an Internet gateway towards the cloud. Due to apparent benefits and the readiness of the used equipment, this layout was used for the experiments presented in this paper.

3. SYSTEM COMPONENTS

The proposed demonstration system consists of: a) cloud server, b) remote user client software, c) local vehicle interface, d) controlled vehicle.

3.1 Cloud server

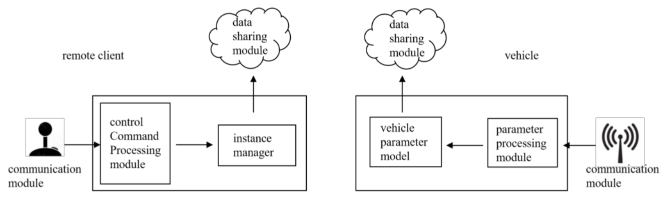


Fig. 3. The cloud-server back-end layout: implementation of the model matching and messaging.

The control system is designed in client/server architecture. Therefore, the cloud-server back-end provides the corresponding `HTTP` backbone enabling the connection between the remote client and the local vehicle interfaces. The cloud-server is accessed through an API providing a set of `HTTP` applications or methods in form of `URIs`. The vehicle subset contains methods for (de)registering new instances and methods for publishing data and subscribing to commands. The remote client subset supports monitoring functions such as retrieval of interfaces and vehicle data. Additionally, control functions are available in the subset for publishing desired commands and control modes to individual vehicle instances.

The cloud-server back-end layout is shown in Fig. 3. The back-end is divided into three parts containing four modules: communication module, control command processing module, interface manager and data sharing module. The communication module exposes the HTTP methods through the uniform resource identifier (URI). The cotrol command preprocessing module is responsible for marshalling of received data and control commands. The provided control missions and attached data are matched with the vehicles through their instance ID in the global management module. Once the received data is analyzed, parsed and matched, the data storage layer stores the information into the data storage module. Finally, the data sharing module provides the methods for remote users and vehicles to extract required data from the storage module for visualization and other purposes.The definition and description of necessary interfaces for remote client and local vehicles are shown in Tab 1

Table 1. Definition of services necessary to interface remote clients and vehicles.

Application URI	**Introduction**
/client/connect_instance	Connect the instance
/client/get_param	Get parameter from server
/client/submit_control	Send control command
/client/get_fbmessage	Get instance control mode
/upper/start_instance	Create a new instance
/upper/finish_instance	Remove created instance
/upper/update_param	Set parameter on server
/upper/get_control	Get control command
/upper/update_fbmessage	Switch the control mode

3.2 Remote Client

The remote client encompassed the graphical user interface with the corresponding back-end model. It includes five fundamental modules as shown in the diagram on Fig. 4. The modules establish communication with the cloud-server, send mission commands to vehicles and monitor the vehicle status through the operations panel.

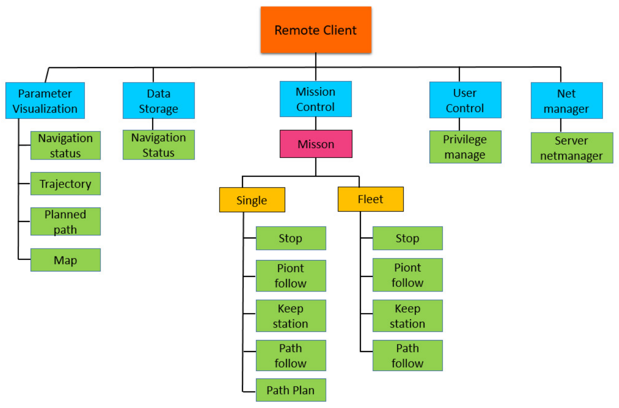


Fig. 4. The tree-like structure of the remote client showing the five fundamental modules and their corresponding submodules.

The parameter visualization module implements visualization extensions for supported navigation and status data objects received from the vehicles. Deserialized vehicle attributes are shown in textual form on the client's user interface. Position and trajectory updates are provided on the map widget through vehicle markers. Based on the vehicle model provided from the cloud-server, the data storage module of the remote client splits the bundled vehicle data into separate parameters and stores them in the local database. This process is similar to the processing done on the cloud-server back-end and provides an additional storage redundancy. This feature is also required if local data availability for offline post-processing is required by the users. Mission commands are defined in the mission control module, these include typical commands such as stop, point follow, keep station and path follow. These commands can be applied to one or more vehicles in parallel. The modular design of the mission control modules allows easy addition of new commands and mission modes. The user control module provides user interface elements for handling arbitration and allowing remote control on vehicles. The module also allows for the existence of a local manager, i.e. vehicle manager in the field. In such cases, remote users can control the vehicles only when the local manager grants permission. This avoids accidents or conflicting commands in cases where the remote user lacks complete situational awareness.

3.3 Local Vehicle Interface

The remote client commands are received on the local vehicle interface through the cloud-server. The interface maintains server communication through a heart-beat mechanism. Even when contact is lost, the vehicle instance on the cloud-server is reserved until connection is reestablished.

The layout of the vehicle interface is shown in in Fig. 5. The interface is divided into four modules. Mission, net manager and user control modules are conceptually similar to the ones in the remote client. The user module manages permissions, the mission module translates generic mission commands into vehicle specific calls and the net manager handles communication and data exchange with the cloud-server. The shared modules are designed in a fashion to be independent of the underlying vehicle architecture. While the mission module has some conceptual knowledge of the vehicle capabilities, it does not handle communication with the vehicle implementation directly. For this purpose, a ROS components module was developed as an intermediary to handle ROS framework peculiarities thereby providing dependency isolation for rest of the modules.

The ROS runtime "graph", running on the vehicle, is a peer-to-peer network of processes that are loosely coupled using the ROS communication infrastructure. A ROS topic implements the concept of asynchronous data streaming, and the ROS service implements the concept of synchronous communication in style of remote procedure calls as detailed in Woodall et al. (2014). The vehicle status and sensor data are received from customized topics through the ROS subscriber submodule. Command execution is implemented through the ROS publisher submodule.

3.4 H2OmniX

Experiments used the H2OmniX unmanned surface vehicles, shown in Fig. 6. The H2OmniX is a versatile omni-directional vehicle designed for operation in shal-

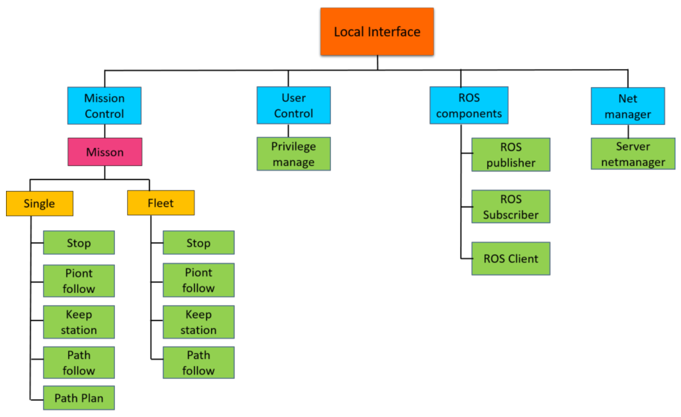


Fig. 5. The tree-like structure of the local vehicle interface showing the four fundamental modules and their corresponding submodules.

low, crowded and/or confined environments where maneuverability is of essence. It supports a wide range of payloads for acoustic localization, underwater monitoring, bathymetry and sensors for environmental data sampling. Due to their versatility, the vehicles have been used in number of different applications from diving support (Miskovic et al. (2015)), oil-spill detection (Vasilijević et al. (2017)) to archaeology (Vasilijević et al. (2015)).

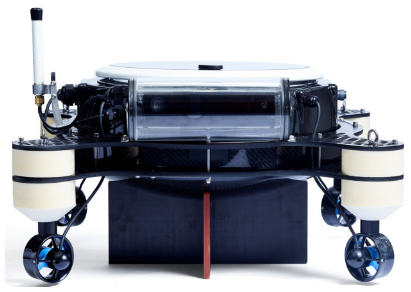


Fig. 6. The H2OmniX unmanned surface vehicle.

H2OmniX was developed by the Laboratory for Underwater Systems and Technologies at University of Zagreb Faculty of Electrical Engineering and Computing in Croatia. It is currently produced by H2O-Robotics [1]. H2OmniX is over-actuated which enables horizontal motion in any direction and orientation. It has a diagonal length of $1\,m$ with $0.35\,m$ in height, and it weighs in at $20-30\,kg$, depending on battery type and payload configuration. It is equipped with state of the art 9-axis inertial navigation system and RTK capable GNSS. The main navigation, guidance and control software is operating on the ROS framework and the vehicle is equipped with GSM and WiFi allowing the integration of the proposed cloud-based framework.

4. EXPERIMENTS AND RESULTS

4.1 Simulation

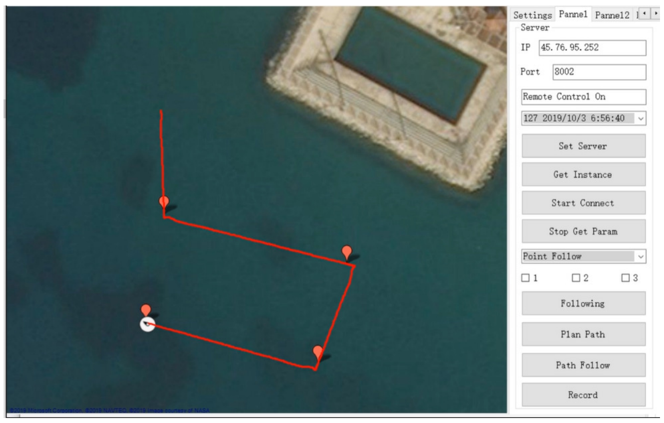
Before real experiments, simulations were performed to validate whether the H2OmniX can execute the proposed missions through the cloud infrastructure. For simulation and pool experiments the provided cloud server was located in Frankfurt, Germany, while the pool experiments were planned for Biograd na Moru, Croatia.

Table 2. Average communication delay of a subset of available `HTTP` methods

`HTTP` method	Average Delay (ms)
connect_instance	37
get_param	104
submit_control	31
update_param	127
get_control	29

First the expected server-client communication delay has been measured. The delay was tested for a subset of `HTTP` methods and the average delays are shown in Table 2. The average delay is low across all of the methods. The `get_param` and `set_param` commands have a higher delay due to larger payload. However, the command setting method delay (e.g. `submit_command`) is small. Since only high-level commands are used, with low-level control loops closed locally on the vehicle, the measured communication delays have minimum impact on control performance. From the perspective of data collection and online processing the delays are still within margins to allow near real-time mission (re)planning.

Secondly, the demonstration missions, including point following and path following, were tested with the simulator. The vehicle simulator is provided in the form of Docker Docker [2] container. Vehicle kinetics, kinematics and sensors are simulated while all other software components, including navigation and control, are identical to those located on the vehicle. The guidance API can be accessed directly from ROS or via `rosbridge` [3] .



IP 45.76.95.252
Port 8002
Remote Control On
127 2019/10/3 6:56:40
Point Follow

Fig. 7. Waypoint guidance simulation: marked target points are sent sequentially to the vehicle during operation.

For simulation, the waypoints are selected by the user through the map widget in the remote client. In addition to single waypoints, the users can provide a set of waypoints and the integrated path-planning module will provide the vehicle with desired path. Single waypoint and path-following simulation results are shown as Fig. 7 and Fig. 8,

[1] h2o-robotics.com

[2] www.docker.com
[3] robotwebtools.org

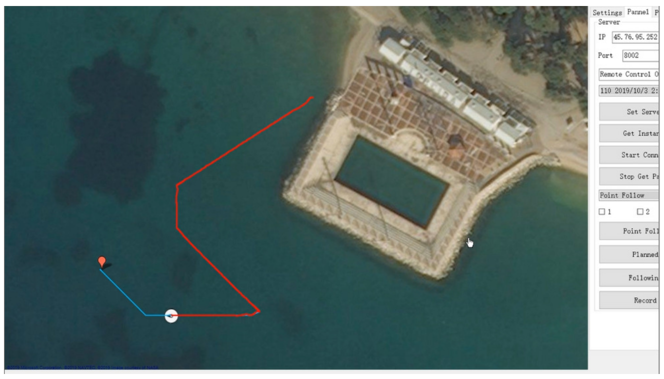


Fig. 8. Path following simulation: the blue line shows the remainder of the planned path generated with a modified A* algorithm (Wang and Xiang (2018)).

respectively. The markers (shown as red icons) represent the selected target points the vehicle should visit. Figure 8 shows a planned path based on a set of waypoints with completed section shown in red and remaining path section shown in blue.

These simulations focused on demonstrate functionality rather than tracking performance. Although the example simulation scenario is shown for a single vehicle, the simulations can be easily scaled to multiple vehicles to demonstrate fleet operation.

4.2 Pool experiments

The real-life experiments were performed at Biograd na Moru, in the controlled pool environment. Experiments included line following, station keeping and path following controllers available on the two vehicles and described in Vasilijević et al. (2017) and Miskovic et al. (2015). During the experiments, the vehicle velocity limit was $0.5\,m/s$. Disturbances in the pool are negligible but the control algorithms can compensate disturbances expected in the open-sea.

Figures show the display of the remote client (on the left) and corresponding frames extracted from the video recording (on the right). Depending on mission type, either both vehicles were operated or one of them was used as an obstacle for the path-planning algorithm. In case of the path-following experiment, shown in Fig. 9, the desired path around the obstacle is shown in blue. It can be noted that the path passes next to the obstacle, i.e. the drifting second vehicle.

During waypoint guidance, shown in Fig. 10, two diagonal points, relative to current vehicle positions, were selected as target points. All vehicle information, including position estimates, is stored in the cloud and can be easily retrieved. This feature was used to review the trajectories generated during this experiment by both vehicles. The retrieved latitude and longitude coordinates, seen in Fig. 11, are replayed in the map module directly from the cloud after the experiment.

The vehicle information is stored in MongoDB[4] and is extracted through Robo3T[5], a data visualization tool for MongoDB. In addition to position estimates, the information stored in the database includes vehicle model parameters, sensor measurements and other navigation, guidance and control signals. These can easily be accessed by any user with corresponding credentials.

[4] mongodb.com
[5] robomongo.org

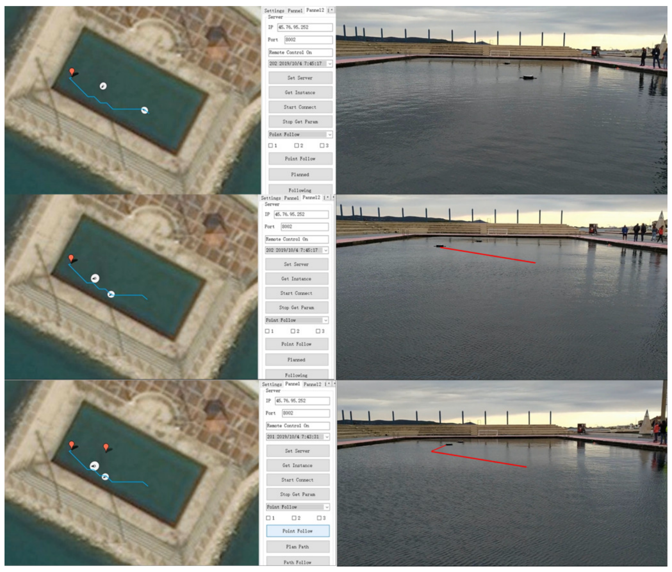


Fig. 9. Path following test: (left) the remote client display, (right) frames from scene recording.

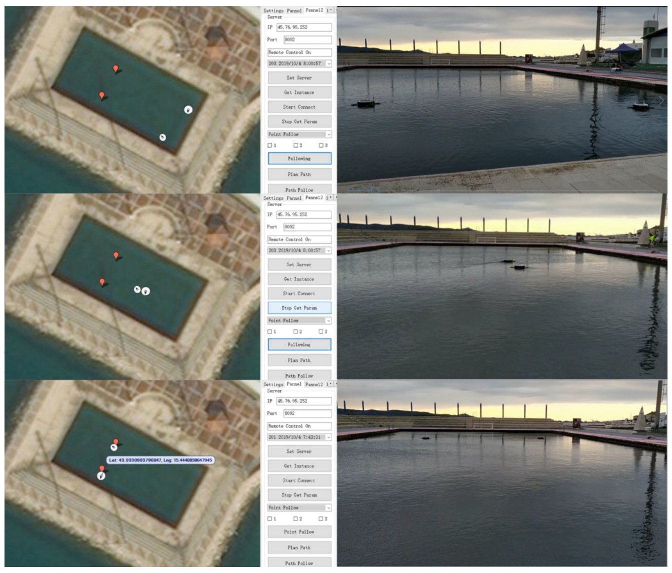


Fig. 10. Waypoint guidance test: (left) the remote client display, (right) frames from scene recording.

5. CONCLUSIONS

The paper presented a cloud-based framework for mission planning and control of unmanned surface vehicles. The selected system architecture was presented with individual component details. The preliminary results show that the proposed system is feasible for coordinated control of multiple H2OmniX vehicle or, generally, any unmanned surface vehicle satisfying the current software/hardware requirements for the framework.

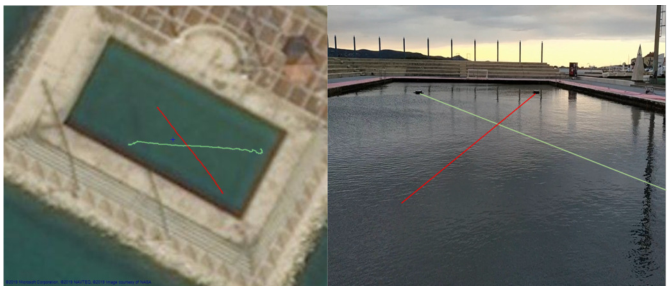


Fig. 11. The replayed trajectories of the two vehicles as shown in the map widget of the remote client.

However, for long term sea-trials there are still technical concerns that need addressing. Firstly, in the proposed framework, the control is sensitive to the communication delay between vehicles and the cloud-server. During measurements the communication delay was shown to be tolerable, but these expected values are still limited to the same region, i.e. to Europe since both the server and clients where within 1200 km. The measurements were also performed for a cloud-server in Wuhan, China with resulting delays increasing close to 1 second. These delays could still be considered tolerable in guidance commands, but can easily become problematic in critical situation or at higher vehicle speeds. Therefore, limitations are to be expected when crossing regional boundaries in which case only supervision level commands can be provided rather than sending direct guidance commands.

Secondly, the experiments were carried out in controlled conditions with only two vehicles. This was already expanded to three vehicles and operations outside of the pool but due to space limitations there results are left for future publications. However, next steps would need to include more vehicles and cloud-based formation management. Additionally, mission complexity should be increased with specific application, e.g. towards distributed environmental monitoring. These next steps also need to expand towards a more heterogeneous setup by including different vehicles, and combining them with different vehicle types (e.g. unmanned aerial vehicles).

REFERENCES

Hu, G., Tay, W.P., and Wen, Y. (2012). Cloud robotics: architecture, challenges and applications. *IEEE Network*, 26(3), 21–28. doi:10.1109/MNET.2012.6201212.

Junnan Song, Gupta, S., Hare, J., and Zhou, S. (2013). Adaptive cleaning of oil spills by autonomous vehicles under partial information. In *2013 OCEANS - San Diego*, 1–5. doi:10.23919/OCEANS.2013.6741246.

Kaliski, B. (2008). Multi-tenant Cloud Computing: From Cruise Liners to Container Ships. In *2008 Third Asia-Pacific Trusted Infrastructure Technologies Conference.*

Koubaa, A. and Qureshi, B. (2018). Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet. *IEEE Access*, 6, 13810–13824.

Lorencik, D. and Sincak, P. (2013). Cloud robotics: Current trends and possible use as a service. In *2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 85–88.

McGillivary, P.A. and Zykov, V. (2016). Ship-based cloud computing for advancing oceanographic research capabilities. In *OCEANS 2016 MTS/IEEE Monterey*, 1–7. doi:10.1109/OCEANS.2016.7761339.

Miskovic, N., Nad, D., and Rendulic, I. (2015). Tracking divers: An autonomous marine surface vehicle to increase diver safety. *IEEE Robotics & Automation Magazine*, 22(3), 72–84. doi:10.1109/MRA.2015.2448851.

Mohanarajah, G., Hunziker, D., D'Andrea, R., and Waibel, M. (2015a). Rapyuta: A Cloud Robotics Platform. *IEEE Transactions on Automation Science and Engineering*, 12(2), 481–493.

Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., and Waibel, M. (2015b). Cloud-Based Collaborative 3D Mapping in Real-Time With Low-Cost Robots. *IEEE Transactions on Automation Science and Engineering*, 12(2), 423–431. doi:10.1109/TASE.2015.2408456.

Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software.*

Riazuelo, L., Tenorth, M., Di Marco, D., Salas, M., Gálvez-López, D., Mösenlechner, L., Kunze, L., Beetz, M., Tardós, J.D., Montano, L., and Montiel, J.M.M. (2015). RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach. *IEEE Transactions on Automation Science and Engineering*, 12(2), 432–443. doi:10.1109/TASE.2014.2377791.

Saha, O. and Dasgupta, P. (2018). A Comprehensive Survey of Recent Trends in Cloud Robotics Architectures and Applications. *Robotics*, 7(3).

Vasilijević, A., Buxton, B., Sharvit, J., Stilinovic, N., Nad, D., Miskovic, N., Planer, D., Hale, J., and Vukic, Z. (2015). An ASV for coastal underwater archaeology: The Pladypos survey of Caesarea Maritima, Israel. In *OCEANS 2015-Genova*, 1–7. IEEE.

Vasilijević, A., Nađ, Đ., Mandić, F., Mišković, N., and Vukić, Z. (2017). Coordinated navigation of surface and underwater marine robotic vehicles for ocean sampling and environmental monitoring. *IEEE/ASME Transactions on Mechatronics*, 22(3), 1174–1184.

Wang, Y. and Han, Q. (2016). Network-Based Fault Detection Filter and Controller Coordinated Design for Unmanned Surface Vehicles in Network Environments. *IEEE Transactions on Industrial Informatics*, 12(5), 1753–1765. doi:10.1109/TII.2016.2526648.

Wang, Z. and Xiang, X. (2018). Improved Astar Algorithm for Path Planning of Marine Robot. In *2018 37th Chinese Control Conference (CCC)*, 5410–5414.

Woodall, W., Liebhardt, M., Stonier, D., and Binney, J. (2014). ROS Topics: Capabilities [ROS Topics]. *IEEE Robotics Automation Magazine*, 21(4), 14–15.

Yicheng Zheng, Feng Deng, Qingmeng Zhu, and Yong Deng (2014). Cloud storage and search for mass spatio-temporal data through proxmox ve and elasticsearch cluster. In *2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems*, 470–474.

Zhang, L., Huaxi, Zhang, H.Y., Fang, Z., Xiang, X., Huchard, M., and Zapata, R. (2017). Towards An Architecture-Centric Approach to Manage Variability of Cloud Robotics.

Zhang, Q., Lapierre, L., and Xiang, X. (2013). Distributed Control of Coordinated Path Tracking for Networked Nonholonomic Mobile Vehicles. *IEEE Transactions on Industrial Informatics*, 9(1), 472–484.