Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-14 (2022) 77–82

A Communication Interface for Multilayer Cloud Computing Architecture for Low Cost Underwater Vehicles ⋆

Alexandre Cardaillac * **Martin Ludvigsen** *,**,***

* *Department of Marine Technology, Norwegian University of Science and Technology, Trondheim, Norway (e-mail: {alexandre.cardaillac, martin.ludvigsen}@ntnu.no).*

** *Department of Arctic Technology, University Centre in Svalbard, Longyearbyen, Norway*

*** *Department for Arctic and Marine Biology, The Arctic University of Norway, Tromsø, Norway*

Abstract: To enable high computational loads for low cost underwater drones, a cloud based architecture is proposed to take advantage of recent development in machine learning and computer vision. The processing power made available will benefit vehicles with limited onboard processing capacity. The rapid development of cloud computing services have made servers with significant computational resources easier to access. In this paper, a communication interface for cloud based multilayer architecture is proposed to enable real time performance by distributing the workload to networked processing devices. It adopts a publish-subscribe model for efficient communication between the layers. The latency and workload distribution are evaluated to assess the efficiency of the proposed method. An application to semantic segmentation of under-water scenes is also tested to measure the framework capabilities for real-time operation using more resource-demanding tools. The conducted experiments resulted in time and performance gains through offloading the underwater vehicle, and forwarding the computations to the cloud based layer.

Keywords: Cloud computing, Underwater vehicles, Multi-layer architecture, Communication interface, Computer vision

1. INTRODUCTION

Remotely Operated Vehicles (ROVs) allow simple and efficient underwater operations and are mainly used for monitoring, exploration and inspection tasks. To automate the repetitive sub-tasks in subsea inspections, the need for autonomy increases. However, it is more expensive in terms of computing power, and many vehicles can not afford to handle all the processing tasks onboard in real time. The dampening of electromagnetic signals in the ocean requires underwater drones to be tethered to a surface unit for broad band signal transmission. The processing power of the surface unit can enable a first sharing of the workload, however, this does not allow full exploitation of the technological possibilities and modern techniques for underwater vehicles, including, methods for vehicle localization, environment mapping or the use deep learning tools. To overcome this, it is possible to add cloud based solutions. Cloud servers provide high storage capacity and significant computational resources, all with high-bandwidth connections.

This paper presents a communication interface for cloud based multilayer architecture. It enables low cost underwater vehicles with limited computing capacity to have access to more resources in order improve their performance while in operation. The proposed framework is based on a publish-subscribe model for connecting sensor data from a layer to another. Therefore, within each layer it is possible to select the inputs it should receive and the outputs it will provide. In this way, only the data necessary for each layer will be communicated and shared. The number of layers is adaptable and have bi-directional connections with other layers following a top-down model of structure of layers.

This publish-subscribe model is beneficial for real-time robotic applications because they are often event-based. The traditional approach which has a request-response model suffers from a higher latency, mainly because of the polling actions.

2. RELATED WORK

The Internet of Things or IoT, enables the interconnection of devices through internet or other communications networks. It facilitates the exchange of data with other devices or systems inside the network.

In Kamburugamuve et al. (2015), Internet of Things Cloud is proposed. It is a platform that makes it possible to connect IoT devices to cloud services for real-time data processing and control. It is composed of three main layers which all have their own defined tasks: a

⋆ This work was supported by the BugWright2 EU H2020-Project under the Grant agreement No. 871260.

10.1016/j.ifacol.2022.07.586

gateway layer, a publish-subscribe messaging layer and a cloud-based big data processing layer. Its scalable and distributed architecture design allows a large number of robots and devices to connect while maintaining a low-latency messaging system.

In Jiao et al. (2017), a robotic cloud-based framework for Visual SLAM (Simultaneous Localization And Mapping) processing of low-cost agents is developed. It enables real-time rate even if the band-width is limited. It uses WebSocket and HTTP as the communication protocols according to the message size which is in a compressed JSON format. The system is based on two components, the robots and a server which internally runs a cloud-based framework and can concurrently process requests from multiple robots at the same time.

In the underwater environment, the IoT, sometimes referred as the Internet of Things Ocean (IoTO) or Internet of Underwater Things (IoUT), is an emerging communication ecosystem to connect underwater agents and gives rise to the concept of Big Marine Data (BMD) because of the increasing volume and availability of data Jahanbakht et al. (2021). It is most of the time used for marine data management Luo et al. (2018); Albaladejo et al. (2010) based on networks of interconnected sensors/devices. These IoT solutions find many applications in marine environment monitoring and protection such as water quality monitoring or coral reef monitoring Xu et al. (2019).

An hybrid use of cloud and edge technologies is proposed in Salhaoui et al. (2020) in order to track the fan mussel population on the seabed in real time. The approach is based on Deep Learning (DL) techniques for image processing techniques such as Convolutional Neural Networks (CNN), which is known to require more computational resources than traditional methods. The solution takes advantage of the resources available thanks to the IoT architecture developed to optimize and improve the vision based method. To achieve this, an Autonomous Underwater Vessel (AUV) is connected to a communication bridge at the surface which is then connected to cloud AI services and a specific platform for AUV operations. The same sort of structures also find applications in autonomous surveillance in marine protected areas Molina Molina et al. (2021).

The proposed framework in this paper enables variable number of processing layers based on the computational load, enabling distributed computing to overcome potential bottle-neck in communication and processing. The tasks of each layer are user defined, allowing a more advanced custom design of the architecture and less requirements to make it possible for the framework to be implemented with a large variety of agents.

3. ARCHITECTURE OVERVIEW

In this section and the followings, all devices/platforms that implement communication interfaces are referred as units or layers in the architecture.

3.1 Communication module design

To maximise its adaptability and scalability, the communication interface is designed to be able to work independently. Therefore, it allows the users to use it with any software or robotic platforms such as ROS Quigley et al. (2009) or DUNE Pinto et al. (2013).

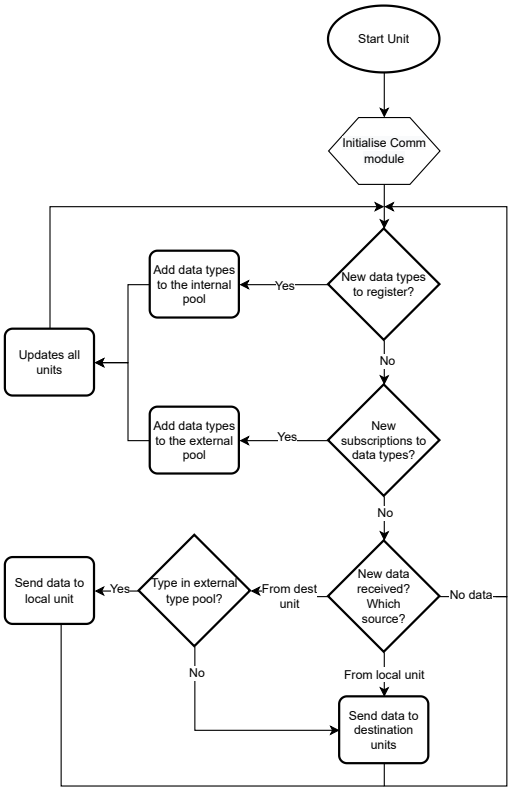

Fig. 1. Flow chart of the communication module.

The architecture adopts a publish-subscribe model. To be able to receive and share data with other units, a processing unit will interact with a communication module in order to register and subscribe to data types. For local units subscribing to a data type, the interface saves the transmitted information in a set of data types called external pool along with the unit ID. The updated information distribution scheme is then communicated to all agents in the network. Similarly, when registering a data type, the local unit will assign information to the new data type, and the interface saves it in an other set called internal pool, and updates the other units. In other words, the external pool contains all the data types a unit produces and provides, whereas the internal pool contains the types that are needed for the local processing.

Registrations and subscriptions can be done at any time, enabling new data types to be created during the operation. When the behavior of the units changes during the operation, new subscriptions and registrations can be added to improve the information flow. It enables dynamic adaptation of the units, data requirements and results.

When the communication module receives data, it checks the source, the data type and the associated pool. If the data type is part of the internal pool, the data comes from the local unit and therefore needs to be published in order to communicate it to the other units. When the data comes from another unit, there are two possibilities: either the type is in the external pool, in which case the

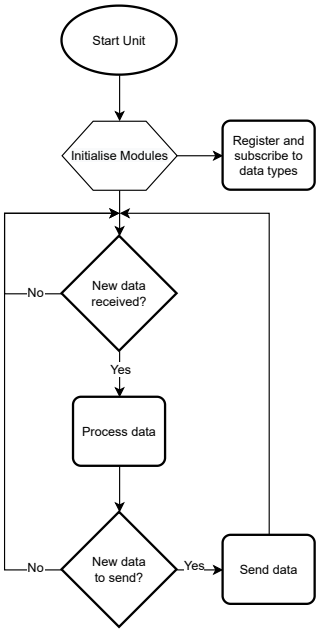

Fig. 2. Flow chart of an unit that interacts with the communication interface.

data is transmitted to the local unit, or, if it is not, it is sent to the next unit, acting like a bridge.

All the above procedures are repeated in a loop according to the received data until the module is stopped. The global design of the communication module is summarised in the flow chart of Figure 1.

A possible interaction between a unit and the communication module is shown in Figure 2. In this example, the registration and subscription procedures of data types are done during the initialisation step. The unit waits to receive data, processes them, and if there are results to send, transmits them to the communication module and starts over.

Making the communication module an independent component enables a higher level of abstraction and ensures reusability. In the example depicted above, the unit only interacts with abstracted functions which makes it impossible for the unit to impact the logical behavior of the module.

The proposed method does not have a limit for the number of layers. However in our application, a minimum of three is required as there must be one layer for the underwater drone, one layer in the cloud and one layer to make the bridge. In Figure 3, the typical architecture, involving three layers is illustrated. To optimally and fully use all the layers, they should all process some data to handle a sub-task in addition to run the communication module. It enables workload distribution, significantly improving the global performance, even for large-scale projects.

3.2 Communication protocols

According to the mission of the drone, the data types and their associated payload of varying sizes will need to be shared between the units. However, communication protocols have size limitations. It is possible to divide the

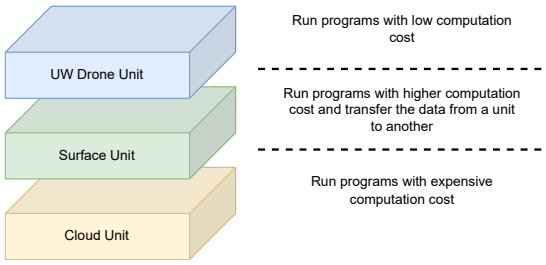

Fig. 3. Framework architecture involving three units.

messages carrying the data into two categories: small messages and large messages. For small messages, WebSockets are used, they are a very good solution for data streaming in real time as they provide full duplex communication channels with low latency. However, they are not well suited to large messages. For larger messages, which correspond to multimedia messages such as, but not limited to, images and sounds, the Real-Time Streaming Protocol (RTSP) is used. This protocol is designed to carry real-time delay-sensitive payloads. It can also stream data from specific sensors of the underwater drone such as sonars.

4. EXPERIMENTS

4.1 Setup

In this section, a series of experiments are described with the main focus on the efficiency and latency of the system. The proposed framework was used in the three-layer architecture presented in Section 3.1. The Blueye Pioneer underwater drone [1] served as UW Unit, containing a single circuit board with a quad core CPU up to 1.2 GHz per core and 4 GB of memory. For the Surface Unit, a laptop was used with an Intel Core i7 vPro with base frequency 1.8 GHz per core and 16 GB of memory. Finally for the Cloud Unit, running on the NTNU IDUN computing cluster Själander et al. (2019), with an Intel Xeon Processor with base frequency 2.2 GHz, 128 GB of memory and an NVIDIA Tesla P100. The underwater drone is tethered to a router to which the laptop is connected over Wi-Fi in a local network. The laptop is also connected to the Internet in order to have access to the cloud server. The implementation of the RTSP is done using the GStreamer framework [2], a pipeline-based multimedia framework that enables streaming workflows.

For all the experiments, each layer has the same communication module but the internal software may differ from one experiment to another.

4.2 Latency Evaluation

To measure the latency of small messages from one unit to another, the average Round Trip Time (RTT) was calculated for each message. This was done in three independent scenarios with message payloads of varying sizes. Each scenario corresponds to a possible route, i.e. from the UW Unit to the Surface Unit (1), from the Surface Unit to the Cloud Unit (2) and from the UW Unit to the Cloud Unit (3). The results are shown in Figure 4. The tested size

[1] Blueye, https://www.blueyerobotics.com/
[2] GStreamer, https://gstreamer.freedesktop.org/

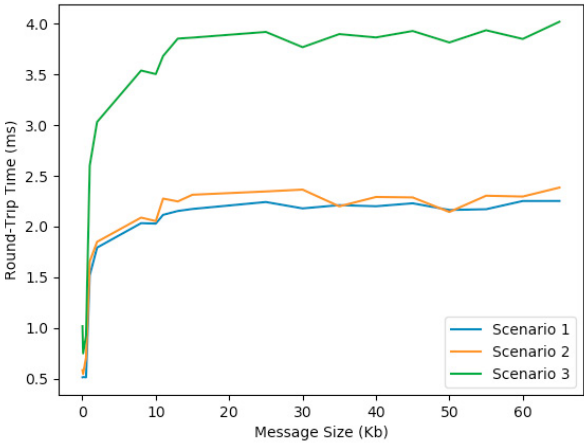

Fig. 4. Round Trip Time of messages of different sizes in three scenarios. Scenario 1: messages between the UW Unit and the Surface Unit. Scenario 2: messages between the Surface Unit and the Cloud Unit. Scenario 3: messages between the UW Unit and the Cloud Unit.

of the messages goes up to 65 Kb as larger messages are considered as media messages.

As expected the third scenario takes twice as long on average as the time of the first and second scenario. It also means that the time needed for a message to change network, i.e. from the local network to the cloud network and vice versa, is extremely low and therefore does not need to be taken into account. It is also possible to observe that the latency in scenario two is on average slightly higher than in scenario one but it is only a matter of tenth of a millisecond. Globally, the latency evaluation shows the viability of the WebSockets in this context for small messages.

The delay for large messages was tested by transmitting video streams from the underwater unit to the surface unit. Four video streams were tested with different combinations of resolution and frame per second (FPS). In Table 1, it is possible to observe the average measured delay and frame per second of video streams from a unit to another using different combinations of resolution and frame per second.

Table 1. Evaluation of communication with video payload

ID	Resolution	Video FPS	Measured FPS	Delay (ms)
1	1280 × 480	30.0	29.77	42.2
2	1280 × 480	60.0	57.12	45.2
3	2560 × 720	30.0	29.64	45.5
4	2560 × 720	60.0	56.52	47.3

The Video FPS column corresponds to the video streaming source FPS, therefore is the expected FPS, whereas the Measured FPS column corresponds to the FPS at the other end of the streaming pipeline. To ensure a minimum delay, some frames might be dropped because of potential latency in the streaming pipeline on one of the end. To increase the quality of the stream, it is possible to introduce controlled latency, however the global delay of the stream will be increased. In some applications, this does not represent an issue, or can be easily compensated. This is can be done by adapting the GStreamer pipeline parameters.

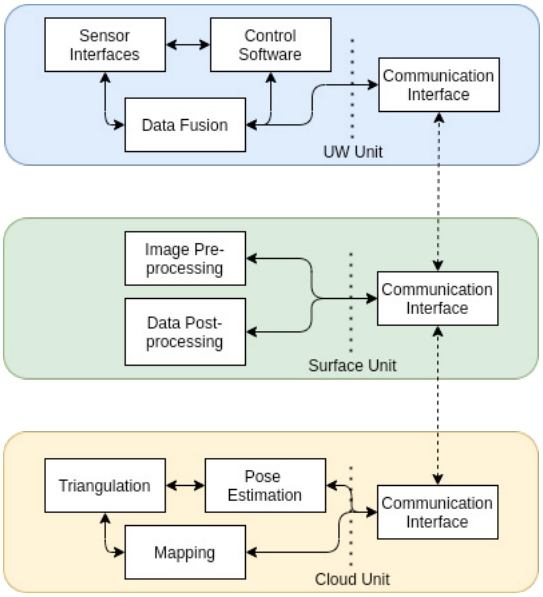

Fig. 5. Software setup for a light Visual SLAM to test the framework. It contains three layers.

The delay here corresponds to the Glass-to-Glass (G2G) measurement, which can be referred as the time it takes for a visible event to go from the glass of a camera to the glass of a display Bachhuber et al. (2017). The measurements were done by aiming the camera at a laptop screen setup to have the same rate as the camera, i.e. 30/60 Hz, which is displaying the video stream. The measured delays are all very close to each other, of about 45ms, although comparing Video 1 and Video 2 allows to spot a difference of 5.1ms which may have an impact on the application over time. Overall, the results show the protocol scale well with the resolution and FPS of the video.

4.3 Efficiency Evaluation

To test how efficient the method is or how it enables more efficient solutions, a special software setup is implemented to match a possible real-world scenario. The Figure 5 represents this setup which is distributed over the three units. It corresponds to a light visual localisation and mapping framework implemented using the OpenCV library OpenCV (2015). The control of the drone is based on PIDs, and two Inertial Measurement Units (IMUs) are used to compute the attitude of the drone based on the filter developed in Madgwick (2010) which is then used to restore the scale of the visual odometry. Each unit has its own set of tasks so that the workload can be shared. In this experiment, the video stream has a resolution of 1080 × 720 and 30 FPS and continuously ran for about 6 minutes.

Four independent configurations were designed to test and assess the efficiency of the proposed architecture:

- Configuration 1: Everything is running in the Underwater Unit - no communication needed, 0 shared data types.

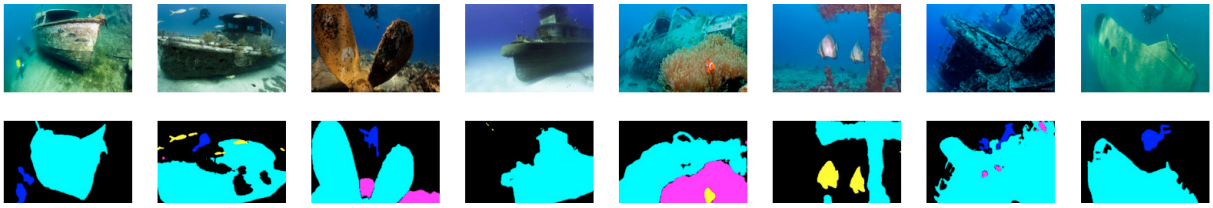

Fig. 6. Set of sample images from SUIM data set Islam et al. (2020), the original images are on the top row and the corresponding pixel-annotations after the SUIM-NET predictions are on the bottom row.

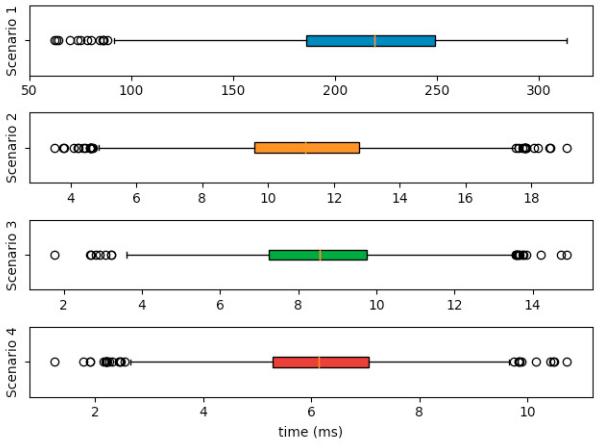

Fig. 7. Processing time benchmark of the developed framework with a three-layers architecture in four scenarios displayed as box-and-whisker diagrams.

- Configuration 2: Control processing and data fusion are done in the Underwater Unit, the rest is done in the Surface Unit - 3 shared data types.
- Configuration 3: The Underwater Unit does the same as in configuration 2. Only data pre-processing and post-processing are done in the Surface Unit, the rest is done in the Cloud Unit, see Figure 5 - 6 shared data types.
- Configuration 4: Same as configuration 3, but the code of the Cloud Unit is optimised to use the available resources in the cloud infrastructure. Additional data processing are done for solution optimisation - 6 shared data types.

The results for each configuration is displayed in Figure 7. The time measured corresponds to the time needed to complete a full loop of the algorithm, including the data communication. In each case, the box-and-whisker diagram is calculated from a sample of 3500 loops. It is clear that the UW Unit alone does not perform well, it can only complete approximately three loops in one second, which makes it not suitable for this type of operation. With the addition of the Surface Unit, the performances are significantly improved and can allow good real-time performances. The third scenario performs similarly but when the resources of the cloud server are used, the algorithm results are similar but it is considerably faster. This gives room to add more features or replace some of the existing features by more computationally expensive ones in order to improve the results. The fourth scenario which includes these changes and is still faster than the other scenarios including better algorithm results.

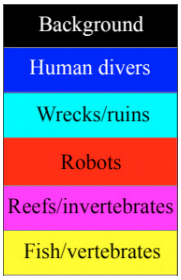

Fig. 8. Corresponding colors to the categories of objects classified by SUIM-NET.

4.4 Application to real time semantic segmentation of underwater scenes

Modern autonomous technologies take advantage of deep learning techniques at an increasing rate, and it is useful to enable real time implementation of the proposed framework. For this, SUIM-Net and the associated data set Islam et al. (2020) were used. It consists of a fully-convolutional encoder-decoder model trained for semantic segmentation of underwater images. It allows to detect five categories of objects: human divers, wrecks and ruins, robots and instruments, reefs and invertebrates, fish and vertebrates. This deep learning solution was chosen because, firstly, it requires computational resources that not all the platforms have, and secondly, because it can be used in multiple applications in the field of underwater robotic vision which makes it a tool to improve the performance and autonomy of an underwater vehicle. It can be integrated into the complete robotic system in order to solve or contribute to tasks such as visual tracking, scene understanding and autonomous exploration. These tasks have to be able to run the model with low-latency. The lower the latency, the more images will be processed and therefore more data will be available, enabling potentially a deeper and more accurate analysis for a given task.

To test the proposed cloud based architecture, the same three layers were deployed and the sub set of the data set used for model testing was used to simulate a real-time feed of images coming from the drone through the surface unit in order to be inferred by the artificial neural network located in the cloud unit. For comparison, the model also ran independently in the surface unit, reducing the global architecture to a two-layers structure.

Although the execution speed are different according to the unit, the result for a same given image remains the same. With the cloud unit, it was possible to run the model with on average 24.23 frames per second which makes it possible to run it in real time and use it for visual aid and/or visual-guided navigation. However, on the surface

unit, it reached 2.5 frame per second on average which is about 10 times less than the previous test. This highlight the use of the previous setup and how important it can be, especially because drone operators do not often have a laptop with a graphical card at hand. A sample of images and their corresponding semantic segmentation predicted by SUIM-NET is displayed in Figure 6. The color code is presented in Figure 8.

5. DISCUSSION

The proposed communication interface for a multilayer structure showed satisfying results for real-time operations of underwater vehicles. The implementation of the communication interface makes it easy to use and adapt, and enables the user to add middleware. It enables quick deployment of additional layers for independent processing and further workload sharing.

However, the current implementation only allows a two-links top-down structure of layers which means that each layer can only be connected to the previous and next layers if they exist. It has the benefits of incremental functionalities, i.e. as the data moves down or top, it produces new results on top of the previous ones. It also allows a clear understanding of the structure and the dependencies of each unit. The main drawback of this type of structure is that it may introduce additional delays for each layer. If the ultimate functionality is located in the last layer, it might take more time, especially if the results need to be sent back to the first layer. Fortunately, these delays can be distributed and dispersed according to the design and locations of the functionalities in the layers.

While this method enables low-latency communications for messages of variable size, which are important for real time operations, it involves risks for the multimedia payloads. Indeed, the protocol used for this type of payload, RTSP, relies on UDP and can lead to packet loss. If important packets are lost, it may affect significantly the behaviour of the implemented algorithms.

Although the framework was experimented using a local computer and with cloud computing, it is possible to imagine alternatives, such as edge computing with different sensors/agents setups. For example an underwater agent connected to a buoy which is itself connected to a surface vessel. Also, despite the fact that the proposed method is used for underwater applications, it can be used for other categories of devices/robots such as ground vehicles.

6. CONCLUSION AND FUTURE WORK

In this paper, a communication interface for cloud based multilayer architecture was developed. It is based on a publish-subscribe model with a topic system to enable clear and effective communication between the units. The access to other units also enables work load distribution according to the resources available. With this architecture, low cost underwater agents with reduced computational capabilities are able to operate in real-time using advanced tools which require important resources.

Future work includes study of how this architecture can scale to cooperative missions involving multiple underwater drones and how the work load can be further distributed to reduce the software latency. This corresponds also to a study of how the layers can be inter-connected, adopting a more complex distributed structure.

REFERENCES

Albaladejo, C., Sanchez, P., Iborra, A., Soto, F., López, J., and Torres, R. (2010). Wireless sensor networks for oceanographic monitoring: A systematic review. *Sensors (Basel, Switzerland)*, 10, 6948–68. doi: 10.3390/s100706948.

Bachhuber, C., Steinbach, E., Freundl, M., and Reisslein, M. (2017). On the minimization of glass-to-glass and glass-to-algorithm delay in video communication. *IEEE Transactions on Multimedia*, PP, 1–1. doi: 10.1109/TMM.2017.2726189.

Islam, M.J., Edge, C., Xiao, Y., Luo, P., Mehtaz, M., Morse, C., Enan, S.S., and Sattar, J. (2020). Semantic segmentation of underwater imagery: Dataset and benchmark.

Jahanbakht, M., Xiang, W., Hanzo, L., and Rahimi Azghadi, M. (2021). Internet of underwater things and big marine data analytics—a comprehensive survey. *IEEE Communications Surveys Tutorials*, 23(2), 904–956. doi:10.1109/COMST.2021.3053118.

Jiao, J., Yun, P., and Liu, M. (2017). A cloud-based visual slam framework for low-cost agents. 471–484. doi: 10.1007/978-3-319-68345-4$_4$2.

Kamburugamuve, S., Christiansen, L., and Fox, G. (2015). A framework for real time processing of sensor data in the cloud. *Journal of Sensors*, 2015, 1–11. doi: 10.1155/2015/468047.

Luo, H., Wu, K., Ruby, R., Liang, Y., Guo, Z., and Ni, L.M. (2018). Software-defined architectures and technologies for underwater wireless sensor networks: A survey. *IEEE Communications Surveys Tutorials*, 20(4), 2855–2888. doi: 10.1109/COMST.2018.2842060.

Madgwick, S. (2010). An efficient orientation filter for inertial and inertial / magnetic sensor arrays.

Molina Molina, J., Salhaoui, M., González, A., and Arioua, M. (2021). Autonomous marine robot based on ai recognition for permanent surveillance in marine protected areas. *Sensors*, 21, 2664. doi:10.3390/s21082664.

OpenCV (2015). Open source computer vision library.

Pinto, J., Dias, P.S., Martins, R., Fortuna, J., Marques, E., and Sousa, J. (2013). The lsts toolchain for networked vehicle systems. In *2013 MTS/IEEE OCEANS - Bergen*, 1–9. doi:10.1109/OCEANS-Bergen.2013.6608148.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. volume 3.

Salhaoui, M., Molina Molina, J., González, A., Arioua, M., and Ortiz, F. (2020). Autonomous underwater monitoring system for detecting life on the seabed by means of computer vision cloud services. *Remote Sensing*, 12, 1981. doi:10.3390/rs12121981.

Själander, M., Jahre, M., Tufte, G., and Reissmann, N. (2019). EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure.

Xu, G., Shi, Y., Sun, X., and Shen, W. (2019). Internet of things in marine environment monitoring: A review. *Sensors*, 19(7). doi:10.3390/s19071711. URL `https://www.mdpi.com/1424-8220/19/7/1711`.