REALTIME SIMULATION AND 3D-VISUALISATION OF SURFACE AND UNDERWATER VEHICLES FOR MONITORING AND EVALUATING AUTONOMOUS MISSIONS

Tobias Theuerkauff, Yves Wagner, Frank Wallhoff
Institute for Technical Assistive Systems
Jade University of Applied Sciences
26121 Oldenburg, Germany
E-mail: {tobias.theuerkauff,yves.wagner,frank.wallhoff}@jade-hs.de

KEYWORDS
Simulation, 3D-Visualisation, Real Time, Low-Cost-Monitoring, Unmanned Surface Vehicle (USV), Autonomous Underwater Vehicle (AUV), Remotely Operated Vehicle (ROV), Mission Planning, Maritime

ABSTRACT

The use of autonomous underwater vehicles requires stable and reliable algorithms within the control software. A misdirected vehicle can quickly lead to a costly damage or even loss of the vehicle. In this paper, an application is presented that allows an ongoing underwater mission to be tracked in real-time within a 3D simulation and, in the event of a problem, aborts it to set the vehicles into a safe state. It can communicate directionally with the control software of each vehicle, thus providing the basis of a simulation environment. Furthermore, first implementations for evaluating the control algorithms of autonomous vehicles within the simulation environment will be presented.

INTRODUCTION

The exploration and use of underwater regions in the oceans as well as in inland waters are becoming increasingly important. The use of these regions depends on the research of these areas. Furthermore, the already economically exploited regions, e.g. through the construction of offshore wind turbines, has to be constantly monitored and controlled (Tchakoua et al. 2014) as well as the inspection of underwater pipelines (Xiang et al. 2014). The use of divers to solve the problem is often used successfully today. However, the use of human resources is a costly, time-consuming and dangerous solution. To curb these factors, an increased use of autonomous underwater vehicles is sought. The autonomous underwater missions with independent, or as well as the composite of autonomous operating vehicles is currently increased in the development. One problem with the use of autonomous vehicles under water is the lack of visual contact. In order to make an autonomous movement of the vehicle under water possible, the vehicles must be equipped with various sensors to capture their environment. In addition, the algorithms to control the vehicle must run absolutely reliable and error-free to prevent loss or damage to the vehicles. For the interpretable perception of the environment, the cognitive control algorithms must be able to process and evaluate all sensor data in parallel and in real time. In particular, in the test phase of the control algorithms malfunction can quickly occur. In this work a prototypical software is presented, which visualizes the movements of the vehicles in real time in a 3D environment. The entire mission of the vehicle, or multiple vehicles within a compound operation, can be permanently monitored and analysed with the visualization. In the case of occurring errors within the control algorithms or in case of misinterpretations of the multisensor data, the operator can thus perform a manual and situation-related intervention in the mission execution manually. Thus, the risk of costly damage or the loss of vehicles can be greatly reduced. The system is based on a low-cost implementation and can be applied to various underwater and surface vehicles as well as complete overwater scenarios.
In the further course, the visualization environment is to be further developed into a simulation environment for testing and evaluating underwater vehicle control algorithms. First the visualization as well as the communication of the visualization environment to real and emulated vehicles will be presented. Approaches and first implementations for the simulation of vehicles with emulated sensors are also considered.

RELATED WORKS

At the University of Porto in the LSTS institute was developed a 2D-visualization of subsurface, surface and air vehicles for supporting multiple vehicle operations (Dias et al. 2005). The map basis for the visualization of the vehicles are electronic nautical charts (ENC data), which are used in the seafaring for navigation. Within the map all the different vehicles, which are registered in the network, are shown in the visualization. The state of the position is not visualized in a fluid visualization, but based on the latest received data of the vehicles. The connection to the vehicles takes place via receiving the IMC-messages send by different vehicles. The application can receive messages as well as send messages via a xml-console into the network to communicate with each registered vehicle. Furthermore, it is possible to plan a mission for one or more vehicles

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

in the network and to send and execute it within the vehicles (Dias et al. 2006).
In the underwater simulator from the IRS Lab the OpenSceneGraph library is used as the basis for visualizing underwater missions. With the open-source UWSim they provide a possibility for simulation and graphical representation of underwater missions. For the control of virtual underwater vehicles and the spread of simulated sensor data, a network-based interface is applied. The underwater terrain model can be configured by the user via a XML file. For this, parameters such as water surface, underwater visibility and particle density in water can be defined. The underwater robot can also be adapted via an XML file. By default, an underwater vehicle with six degrees of freedom (6 DOF) in motion is provided. The software further offers individually configurable sensors that can be linked to the vehicles. It is possible to control several robots within a simulation. The integrated physics engine osgBullet offers the possibility to simulate highly simplified, physical aspects in the underwater world (Prats et al. 2012).

CONCEPT

Submarine vehicles may be connected by a cable to a surface vessel or to an overwater central facility via a cable. In this case we speak about a Remotely Operated Vehicle (ROV). Furthermore, it is possible to connect the vehicle via a wireless connection with a surface vessel or a central office. For this purpose, an Autonomous Underwater Vehicle (AUV) in a Network, especially with larger distances and depths, the acoustic transmission of data is used. The last scenario is a fully autonomous use of the vehicle over a certain time frame up to several days - during which the vehicle has no connection to a base station or an surface vessel. The recorded data of the vehicle will be transmitted to a base station after the mission. The last scenario will not further be considered in this paper.

Within the acoustic communication a bidirectional data transfer between the AUVs are possible. In case of a wired connection, the underwater vehicles always communicate to each other via an USV (Figure 1). When locating and determining underwater vehicles with the use of acoustic modems, the underwater position of the vehicles is calculated in relation to the surface vessel or a base station. This is equipped with a GPS system so the coordinates of the vehicles can be determined within a world coordinate system. In the considered scenarios, the underwater vehicles are in constant contact with each other and with the surface vessel or the base station. Information and sensor- or steering data can be transmitted between the individual vehicles via a high data rate (cable-bound data transmission up to 1 gb/s) or low a data rate (acoustic data transmission with a few kb/s).

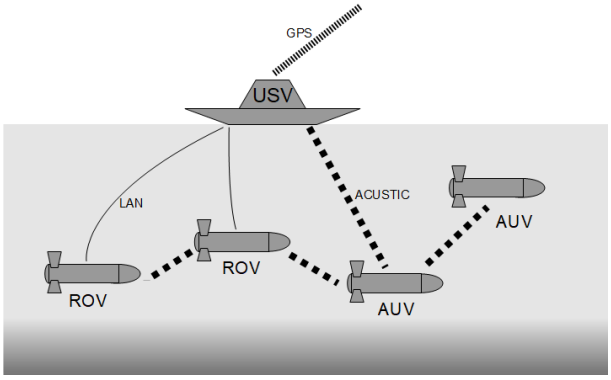


Figure 1: Schematic representation of communication between underwater vehicle and surface vessel

Depending on the transmission rate, simple data such as vehicle ID, position and time stamp up to complex data sets such as camera images can be transmitted. The transmission of data is realized via a uniform transmission protocol. Thereby, each vehicle is able to interpret and evaluate the received data. To do this, the control software of the vehicle must register as an actuator in the communication system. The ongoing communication as well as the generally defined transmission protocol and the registration in the communication network are the base concept of the developed simulations.

The visualization software also registers itself as an actor in the communication network and thus receives all transmitted data of the various vehicles (Figure 2). With every message of a vehicle, we also transmit the ID of the sender, with which the vehicle and the vehicle type can be clearly interpreted. The visualization software can thus automatically emulate and display the corresponding vehicle in the visualization for each vehicle. The software evaluates all received messages and assigns them to the respective emulated vehicle in the visualization. This allows the emulated vehicles to be positioned and oriented within the 3D visualization. Furthermore, all transmitted sensor data of the individual real vehicles can also be visualized on the virtual model. The vehicle status as well as the position, orientation, and sensor values of the vehicles can be permanently monitored by the operator. In the 3D visualization, a georeferenced, digital terrain model (DGM) of the underwater region can be displayed. Thus, by visualizing the position and orientation of the virtual vehicle in combination with the digital terrain model, the operator can detect a hazard, for example, in a foreseeable collision with the terrain. As the visualization software is registered as a real actuator in the communication network of the control software of the vehicles, a bidirectional communication is possible. It is thus able to distribute its own messages within the communication network. In the case of the described hazard scenario, the operator can send a message to the relevant vehicle, to enter into a safe state (e.g., direct arise).

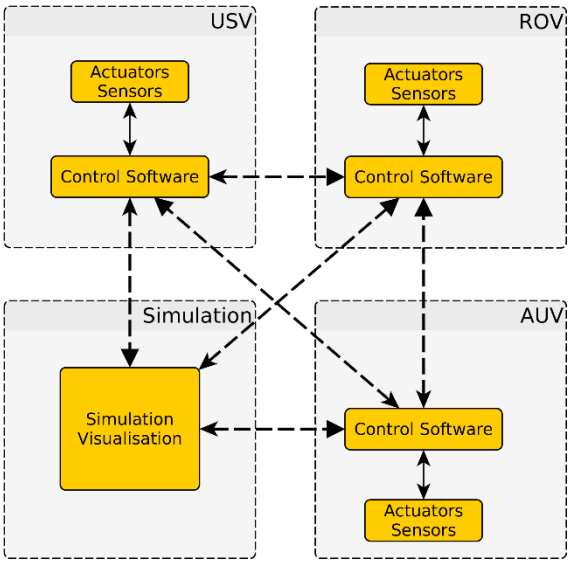


Figure 2: Schematic representation of the communication channels

PROTOTYP COMPOSITION

There are two main components within the development of this monitoring software. The basic hardware components (underwater vehicle and surface vessel) as well as the associated control software have to be considered. The implemented software must accordingly incorporate both components equally. In the experimental environment for the prototype development an underwater vehicle (ROV) was used. As control software the open-source software Dune has been used. The control software is installed on the one hand on the real vehicle to obtain real data. Furthermore, several instances of the control software, without any reference to a real vehicle, were started.

By doing this, several vehicles can be simulated, which are displayed within the visualization. The data of the control software of the real ROV and the data of the simulated ROV are transmitted by the registration in the communication network between all actuators. As a third instance, the visualization environment has been registered as a further actor in the network to receive all transmitted data in the network (Figure 2). For each vehicle, the corresponding, virtual vehicle is automatically added into the visualization. If a real vehicle sends its current position data, it will now be assigned to the corresponding virtual ROV in the visualization. By referring the true coordinates to a basic actuator, which is also displayed in the visualization, since it is registered in the network, the positions of the individual vehicles can be represented in accordance with reality.

Deployed Hardware

Within the project different hardware components were used, which are not based on a project-specific development. BlueRobotic's BlueRov2 underwater vehicle and EvoLogics acoustic communication modems are hardware components purchased for the higher-level project *Development of innovative technologies for autonomous maritime systems* (Entwicklung innovativer Technologien für autonome maritime Systeme - EITAMS).

The underwater vehicle BlueRov2 used in the project is a low-cost vehicle, which in private use is an out-of-the-box vehicle with visual control via PC. But it is also used in research projects for scientific research. It is a small vehicle which is maneuverable only in soft waters. But there it is very maneuverable due to the six specially arranged propellers and can accommodate additional loads and hardware through an expansion set. Additional weight must be compensated by buoyancy, which in turn limits maneuverability. The ROV is equipped with several sensors such as an inertial measuring system, temperature sensors and a camera. The hardware can be addressed freely. The BlueRov2 include a PixHawk controller with integrated IMU, which has its origin in the unmanned aircraft control.

To determine the underwater position of the ROV / AUV acoustic modems from EvoLogics GmbH are used. These modems are also used to transmit various data when the cable connection to the ROV is interrupted or deliberately omitted (AUV). But the data transfer rate of max. 31.2 kbps is very low. Every vehicle must be equipped with a USBL modem. The calculation of the position data is relative to the surface vehicle. This is additionally equipped with a GPS system, so relative coordinates can be transformed into absolute coordinates of a world coordinate system.

Deployed Software

The implemented and used software can be divided into three basic areas. The control of the simulated and the real ROV as well as for the control of the AUV a framework is used. This must be able to address the actuators as well as the sensors of the vehicles and distribute the data of the hardware in the system. The visualization of the vehicle in the 3D environment should be real-time capable, so that at any time the actual state of the vehicles is visible. The third area is a communication protocol that can be interpreted by all components.
The basis for the visualization and simulation is the Unreal Game Engine. We used the very powerful game engine from Epic Games Inc. It is open source available and is constantly being developed. Own source code can be implemented as well as the adaptation of the original code which is needed for the own applications. The Unreal Engine contains the typical core elements of a game engine (sound, graphics, network and physics engine) programmed in C++. Using an in-build editor, terrain models, simple geometries and complex visual effects can be embedded in the visualization environment. Furthermore, it is possible to import your own models and terrain data. The spatial reference is

produced via a metric coordinate system, which can be used as the basis for the georeferencing of the models and terrain data. Own source code can be programmed directly in C++ or via a visual scripting system (Blueprints). The connection between visual scripting and direct programming in source code is achieved through Unreal-Specific Classes (UCLASS) and attributes. Due to certain tags, the functions and variables that are programmed in the source code can also be addressed in the graphical interface with visual scripting. In-depth as well as basic functionalities can be implemented in the C++ code and made available for further development in the visual scripting editor. The use of the functionalities thus provided can then also be assigned by users with less programming experience to the objects in the visualization.

For vehicle control and to address the sensors and other hardware components on the vehicles, the framework DUNE is used. This was developed of the University of Porto by the Institute LSTS in Portugal. DUNE is used in various projects, so it is constantly evolving. It is a software that runs on the vehicles and it is able to interact with the vehicle's sensors and actuators as well as communicating the vehicle data over a network. Furthermore, a variety of algorithms and functions are provided, which can be used for navigation, vehicle control and monitoring as well as for maneuver planning and their execution. The framework is CPU architecture and operating system independent. It has been programmed in C ++ and is open source available. The possibility of their own development and adaptation to their own vehicle is thus given.
The basic concept in DUNE is the implementation of so-called tasks. Each time the sensor value of a sensor of the vehicle has to be picked up and distributed in the network, a task is started which packages the respective information into a special message (IMC) and distributes it on the network. Similarly, a DUNE instance can receive IMC messages that were produced and sent by a task from another DUNE instance. Since communication within a DUNE instance is also done by sending and receiving tasks, each DUNE instance is also able to receive and process self-produced messages. Another major aspect of dune is the using of predefined profiles. When starting a DUNE instance, a configuration file of the vehicle and a configuration file for the actions to be executed are transferred as parameters. The vehicle configuration contains all vehicle-specific basic data such as name, size, extent, weight, etc. as well as any information about the available sensors. Each sensor integrated there has in turn its own configuration file with associated sensor-specific information. The configuration file defines, which so-called tasks the started DUNE instance should execute, or which task it has to respond to.
The implementation and working with DUNE is similar to the Robot Operating System (ROS) middleware. One difference is, that ROS is primary developed for land-based roboters while DUNE is primary developed for surface vessels und subsurface vehicles. So in DUNE are many algorithms, sensors and actors already integrated, wich can be used in this project. Furthermore four of five philosophical priciples (peer-to-peer, tool-based, thin, free and open-source) of ROS (Quigley et al. 2009) are comparable to DUNE. Instead of multi-lingual programming, DUNE only supports C++ and Java.

The used communication protocol in DUNE is the Inter-Module Communication (IMC) protocol, which is also developed at the University of Porto. IMC is an XML-based message format, which is divided into the three areas. Each message consists of a header, message and footer part. The header contains the basic message information. These include the timestamp, message ID, source address and the destination address. The message footer contains a checksum for recheck the messages. The actual information is stored in the message part. For the message part the IMC-protocol provides multiple predefined messages. These predefined messages are divided into different subareas, such as navigation, sensor data or mission planning. Each message contains sensor or actuator-specific values such as GPS coordinates, sensor values or status data. The IMC format ensures the common communication interface between the individual components. The IMC format is completely defined in a version based IMC.xml file, and can be extended according to your own requirements. This file is interpreted by DUNE to generate the source code and must therefore be included by all other applications that use IMC messages for communication.

DEVELOPMENT AND IMPLEMENTATION

In the development of the visualization and simulation software, one focus was placed on the real-time capability and extensibility of the application. Furthermore, the implementation of the interfaces for communication with external applications like DUNE played a major role. This ensures that the visualization can also be operated with other applications. In the following, the individual components of the visualization and the interfaces will be explained in detail. In addition, the sending and receiving of messages with the DUNE framework and IMC is explained.

Sending and Receiving Messages in DUNE

The DUNE Framework has a sophisticated concept for message transmission in the network. The transmission protocol is the IMC format. For this purpose, within the DUNE instances so-called Producer task are created, which produce messages and distribute them in the network. In addition, consumer tasks can be created, which listen to all or to specific messages. The message type within an IMC message is always specified for a data set and contains the specific sensor values in addition to the message ID and the name. For example, with the message ‘Acceleration’, the acceleration values are transmitted in the three axis directions. The axis and the associated value, the data type and the unit of measure

are transferred as parameters. The assignment of the message to the corresponding vehicle is defined in the header of the message. Based on these values, the message can be received and evaluated by another listening instance.

Real Time Visualization

For the implementation of the simulation and visualization the Unreal-Game Engine was used. Particularly with the real-time capability of the engine, very good results could be achieved in previous projects. Several environments have been implemented as test scenarios for this project. On the one hand, these are complex areas with georeferenced digital terrain models (DTM) based on real data. On the other hand, smaller experimental pools were modeled, which are also available to the institute for experiments with the real ROV. The georeferenced terrain models cover an area of up to one square kilometer and are based on multibeam echo sounder and electronic nautical chart (EMC) data. The modeled research basins correspond to the size of the real basins with a size of 3m * 2m * 1m and of 10m * 20m * 5m. The modeling of the experimental environment will not be discussed further here; this is described in more detail in (Theuerkauff et al. 2017).

The digital models of the AUV and the ROV were available as CAD models. By preprocessing, they could be converted into a suitable data format (FBX), which can be imported into the Unreal engine. Smaller components such as lamps and sensors can be modeled directly on the vehicle model in the editor of the game engine via the use of simple geometries. Changes to the vehicle components thus require no complete remodeling and import of the vehicle. Furthermore, the components thus added can be changed with little effort, regardless of the main model, in the individual parameters such as orientation at runtime.

A single vehicle always consists of a main vehicle class and a model object, which in turn can contain one or more geometries or complex models. So it is possible to simple substitute or modify the vehicle model. The main vehicle class is an Unreal specific class (UClass), which ensures that functions are programmable that can be reused in the Engine's Blueprint Editor. The base class contains the basic information of a vehicle. Further, vehicle specific data and functionalities are stored as an additional special vehicle data object in the class. Vehicles can be accessed in real time during ongoing visualization. Thus, parameters such as the position and orientation of the virtual vehicle are customizable according to the data received. For this purpose, the position and orientation data are transferred to the respective vehicle class with the virtual vehicle. The corresponding vehicle class can be determined from the source address passed in the IMC message. The actual positioning and orientation of the vehicles in the simulation takes place with the aid of visual scripting (Blueprint). This allows easy customization of the positioning and orientation routines without working directly in C ++ source code. The received position and orientation data are also interpolated to ensure a fluid visualization of the vehicles.

Connection to the Control Software

The connection of the visualization to other applications takes place via a UDP connection. The interface depends on a DUNE instance which is running in front of the simulation software, so that the simulation only communicates with the DUNE instance, which in turn distributes the data into the network or forwards received data to the simulation. Through this network interface, the visualization software is able to receive messages from the control network as well as to distribute new messages into the control network (Figure 3).

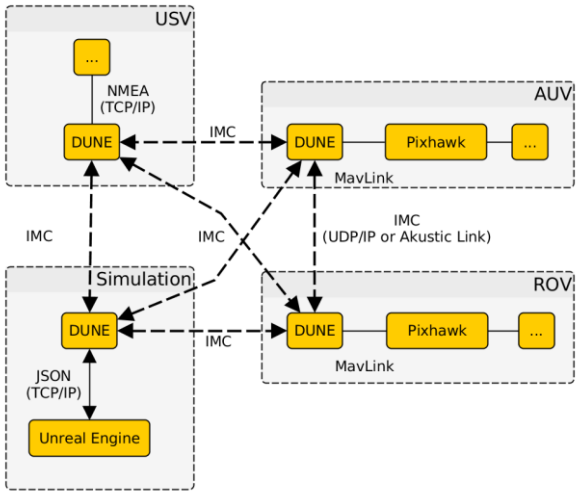


Figure 3: schematic representation of the implemented connection of the simulation environment

So that the DUNE instance serving as simulation interface is not displayed as an additional vehicle in the network, the profile of the DUNE instance has to be set into a special simulation mode. The profile can set at startup via the transmitted configuration file as previously described. With this implemented *UnrealSimulation* profile the interface DUNE instance will receive all communication data and send them to the simulation environment. Furthermore, with the defined profile the DUNE instance can receive messages from the simulation environment to distribute them in the vehicle network.

To do that, the DUNE instance converts all received data into a JSON object, and then forwards it to the visualization environment. On the other side the DUNE instance listen on an UDP interface for messages from the simulation environment. The UDP socket created in the simulation environment constantly listens to the messages of the DUNE instance on a specified port. The in the simulation environment received messages are converted back to the IMC format via a wrapper class. For this purpose, the IMC configuration file (IMC.xml) is imported when the visualization is started and saved in a suitable data structure. The message structure of the incoming JSON messages is determined by the message

name from the IMC data structure. Thus, all messages can automatically be converted from the JSON format back to the IMC structure. Newly defined message structures in the IMC file are automatically converted within the visualization component without further adjustments.

Real Time Simulation

At the virtual underwater vehicles and surface vessels in the simulation, various sensors are connected. At present, an ROV has a depth sensor, an acoustic modem, several laser distance sensors and a camera. The AUV is equipped with an acoustic modem and a GPS system. The sensor data are simulated according to the environment. For this it is possible to set various parameters of the environment within the visualization. These include ambient light, water temperature and turbidity of the water by particles. The influence on the measurement data of the sensors currently consists only in changing the range of the laser distance detection. The sensor data are transmitted in real time to the control software for further processing using the methodology described in chapter *Connection to the Control Software*. Thereby they directly influencing the further behavior of the vehicle.

TEST SCENARIO AND RESULTS

The test environment initially consists of a small test basin with the extension of 2m * 3m * 1m [L, B, T]. The used test vehicle is a BlueROV2 from BlueRobotics Inc. Since the acoustic position determination was not fully implemented when the paper was finished, the position data has been emulated in GPS format. The orientation data are recorded by an inertial measurement unit (IMU) installed in the ROV. The control of the ROV was done manually via an XBox controller. All data are distributed in the network by the DUNE instance of the BlueRov2 and can thus be recorded by the visualization environment, which is also registered via another DUNE instance in the network. Furthermore we integrated up to ten more virtual ROVs and USVs to the simulation. Each vehicle was controlled by an own DUNE instance. By registration the single instances in the same network it is possible to send steering data to each vehicle as same as to send sensordata and messages from a vehicle to each other vehicle in the network. Each virtual vehicle was equipped with an simulated IMU and up to three simulated laser distance measurement systems. The framerate in the running visualization was stable with about 50fps by using an Intel Core i7-6820HQ CPU with 2.7GHz, 40GB RAM and a NVIDIA Quadro M2000M graphiccard. Another test scenario is planned and will be run in the near future. The test environment will be a basin with the size of 20m * 10m * 5m [L, B, H]. The integration of the acoustic modems into the DUNE framework will be done then, so the position data from the underwater vehicles does not need to be emulated anymore for the real vehicles.

Figure 4: Visualization of a real ROV; left: Simulation environment; right: Real environment

CONCLUSION AND OUTLOCK

The visualization environment presented here provides the basis implementation for a real-time simulation environment of underwater vehicles and surface vessels. It is able to visualize virtual and real ROVs and AUVs in a 3D environment. The application communicates bidirectionally with the control units of each vehicles in the network. The simulated vehicles can thus be moved via the control software like a real ROV. The design concept is to handle virtual vehicles in the simulation environment from the control software like real vehicles. So it is possible to use the control software equally for the simulation as well as for the real vehicles without further adjustments (Figure 5). Furthermore it is possible to use these software for other vehicletypes wich are using the DUNE framework for controlling.

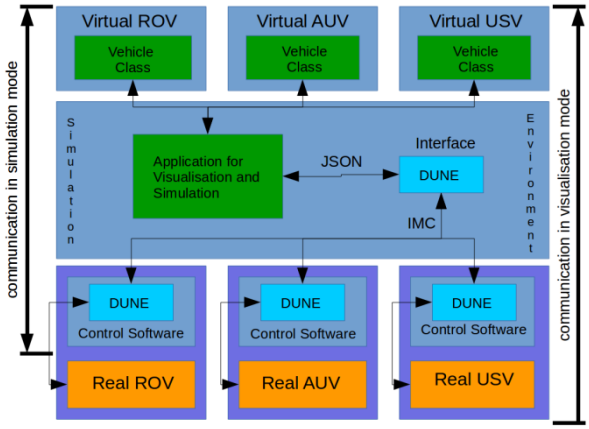


Figure 5: Principle of visualisation and simulation mode. The real vehicles (orange boxes) are only required in visualization mode

At present, the virtual vehicles receive only the coordinates and orientation data of the real vehicle. In the first implementation, the visualization thus always corresponds to the state of the real vehicles. In the further course, the vehicles in the simulation environment are to be expanded with the sensors of the real ROV. The sensors can then be used in the simulation environment e.g. to determine the distance to underwater obstacles and send these data to the control software via the communication interface. This data can than be incorporate into the control logic to generate new control commands for the vehicle. First simple virtual sensors, such as laser based distance sensors, implemented in a previous work, already integrated into the simulation

software presented in this paper. Until now these sensors are not based on the real vehicle sensors and could only deliver fictive data.

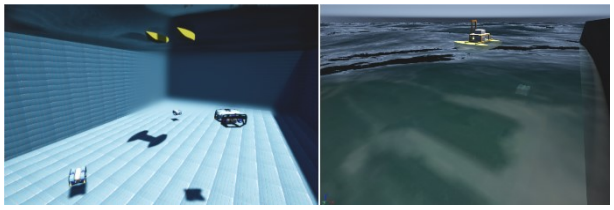


Figure 6: Different simulation scenes; left: Indoor basin; right: open water

An important step in the development of the simulation environment is an investigation into the extent to which water specific data, such as flow models, have an influence into the simulation environment for testing the control algorithms. It is conceivable that the control algorithms can initially also be evaluated by simple, randomly generated drift vectors. Furthermore, the simulation of underwater sensors within the simulation environment will be improved.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the received grants from the internal research and development funds from Jade University of Applied Sciences.
Furthermore, the work became possible by grants of Lower Saxony's Ministry for Science and Culture (Nds. Ministerium für Wissenschaft und Kultur) in the funding scheme VW-Vorab Science for sustainable development.

REFERENCES

Dias P. S., Paulo Sousa and Gomes, Rui M. F. and Pinto, José and Gonçalves, Gil Manuel and de Sousa, João Borges and Pereira, Fernando Lobo. 2006. "Mission Planning and Specification in the Neptus Framework". *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.,* 3220-3225

Dias P. S. and S. L. Fraga and R. M. F. Gomes and G. M. Goncalves and F. L. Pereira and J. Pinto and J. B. Sousa. 2005. "Neptus - a framework to support multiple vehicle operation". *Europe Oceans 2005* Vol. 2 963-968.

M. Prats and J. Pérez and J. J. Fernández and P. J. Sanz. 2012. "An open source tool for simulation and supervision of underwater intervention missions." *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2577-2582

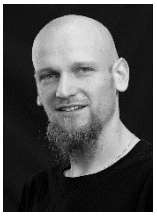
Quigley M. , B Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng. 2009. "ROS: an open-source Robot Operating System.". *ICRA workshop on open source software*, Vol. 3 (May)

Theuerkauff T.; T. Werner; F. Wallhoff; T. Brinkhoff., 2017 „3D-Visualisierung von Über- und Unterwasserfahrzeugen zur Evaluation von Steuerungsalgorithmen mithilfe eine Game-Engine.". *Go-3D 2017 Mit 3D Richtung Maritim 4.0*, Fraunhoferverlag , 135-164

Tchakoua, Pierre, Wamkeue, René, Ouhrouche, Mohand, Slaoui-Hasnaoui, Fouad, Tameghe, Tommy Andy, Ekemb, Gabriel. 2014. "Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges." *Energies*, Vol 7, 2595-2630

Xiang X., B. Jouvencel, O. Parodi. 2010. "Coordinated Formation Control of Multiple Autonomous Underwater Vehicles for Pipeline Inspection".*International Journal of Advanced Robotic Systems,* SAGE Publications Vol. 7, No. 1, 75 – 84

AUTHOR BIOGRAPHIES



Tobias Theuerkauff was born in Leer, in Germany and lives in Oldenburg. He graduated in 2011 at the Jade University of Applied Sciences in Oldenburg as Master of Science in Geodesy and Geoinformatics. Since 2011 he has been working as a research assistant at the Jade University. Until 2011, he worked at the Institute of Photogrammetry and Geoinformatics (IAPG), where he specialized in the fields of virtual and augmented reality. Since 2017 he works at the Institute for Technical Assistance Systems (ITAS) at Jade University of Applied Sciences. There he focuses on the field of artificial intelligence in combination with 3d visualization. His e-mail address is: `tobias.theuerkauff@jade-hs.de` and his Web-page can be found at `https://www.jade-hs.de/team/tobias-theuerkauff`

Yves Wagner was born in Salzwedel, Germany and moved to Oldenburg. In 2014 he graduated as Bachelor of Science in the study course assistive technologies at Jade University for Applied Science in Oldenburg. After that he started as research assistant at the Fraunhofer Institute for Digital Media Technologies (IDMT). Since 2014 he is research assistant at the Institute for Technical Assistance Systems (ITAS) at the Jade University of Applied Science in Oldenburg. His e-mail address is: `yves.wagner@jade-hs.de`

Frank Wallhoff was born in Rheinhausen, now Duisburg in Germany and studied Electrical Engineering in Duisburg where he received his diploma degree in 2000. Hereafter he started as a research assistant at Duisburg University and changed to the Technical University of Munich in 2002 where he was promoted as Dr.-Ing. in 2006 in the area of face detection and recognition. In 2010 he became professor for Assistive Technologies at Jade University of Applied Sciences at the campus in Oldenburg. Since then he is the leader of the group Interactive Systems with focus on artificial intelligence at the Institute for Technical Assistance Systems (ITAS). He is also the director of a Fraunhofer Center of Transfer. His e-mail address is: `frank.wallhoff@jade-hs.de` and his Web-page ca be found at `https://www.jade-hs.de/team/frank-wallhoff`