Distributed Simulation Framework for Investigation of Autonomous Underwater Vehicles' Real-Time Behavior

Sergey Melman, Alexander Pavin
Far Eastern Federal University (FEFU)
Vladivostok, Russia
sergeymelman0@gmail.com, pavin@bk.ru

Valery Bobkov
Institute of Automation and Control Processes (IACP FEB RAS)
Vladivostok, Russia
bobkov@iacp.dvo.ru

Alexander Inzartsev
Institute for Marine Technology Problems (IMTP FEB RAS)
Vladivostok, Russia
inzartsev@yahoo.com

***Abstract*—The paper focuses upon framework design with distributed computing for real-time simulation of resource-intensive tasks connected with testing of autonomous underwater vehicles' (AUVs') "intelligent" control algorithms. Algorithm research is carried out by simulation of a wide range of onboard sensors, AUV dynamics, and underwater physics. A distinctive feature of the simulator is the possibility of testing algorithms directly in a selected AUV control system environment. The simulation framework architecture utilizes client-server technology and a centralized database with the possibility of connecting both personal computers and supercomputers as computational nodes. The modular approach utilized provides the possibility to expand the algorithm and functional base by connecting additional modules via plug-in technology. The framework's functional capabilities, modules' interaction diagram under the circumstances of distributed computing, data communications, and storage protocols taking into account specific features of the simulated processes are described. The framework testing results with different computational configurations illustrating the possibility of performing real-time simulations of AUV control processes with heavy computational loads are quoted.**

Keywords—autonomous underwater vehicle; simulation framework; distributed program architecture; hybrid parallelism; real-time

I. INTRODUCTION

Works on engineering of simulators started with the advent of the first working robotic underwater vehicle (UV) solving practical tasks, since field tests of UVs are rather expensive and substitution of field tests by virtual ones has become one of the solutions for acceleration and cost-cutting of program intelligence (control algorithms) engineering of autonomous underwater vehicles (AUVs).

Paper [1] is one of the first papers describing an approach to testing of UVs with the use of an artificial environment, where data arrives at the vehicle sensors from the actual and virtual sensors simultaneously. The authors have developed a technique of testing vehicles by placing them into small testing pools, thus simulating reality. An earlier paper [2] describes a graphics station designed for monitoring and control of remotely operated UVs as well as for use as a simulator, although simulation modeling was not an issue then.

As there are different types of UVs, including tethered, remotely operated, autonomous, special purpose, multi-purpose, and so on, the requirements imposed on designed simulators differ. These requirements can be aimed at operator training, computer testing of operation tasks (missions) in the virtual environment, computer-aided design of the UV and its components, investigation of AUV control methods, and so on. Judging from the literature, there are relatively few works concerning multi-purpose simulators. Use of the virtual environment use for UV trial tests became seriously reported in 1997. Gracanin, Valavanis and Matijasevic [3] proposed the design of multi-purpose software simulating the virtual underwater environment as well as a vehicle with a user-friendly interface. Then topical VRML and Java Web technologies were used for the development of a virtual multi-user testing range. The authors only outlined the general trends and directions of development of the use of simulation modeling. In paper [4], together with a review of modern UV simulators, including both separate vehicles and groups of vehicles, a classification of similar tools was offered. Also, a requirement to have the possibility of including real vehicles with the help of special interfaces in the simulator environment (hardware-in-the-loop, HIL) was laid down. Thetis simulator engineering was described in the paper too. The peculiarity of this simulator implementation lies in the possibility of using the computational capacities of several computers joined by a network for vehicle-sensor modeling. An essential characteristic of the simulator is the possibility of its real-time functioning as part of the operations carried out. So, taking into

This research was supported by Russian Foundation for Basic Research (RFBR) grant No. 15-07-00341 and Russian Science Foundation (RSF) grant No. 14-50-00034.

978-0-933957-43-5 ©2015 MTS

account that modern AUVs can be equipped with many sensor payloads, it is worthwhile to develop multi-purpose simulators on the basis of high-performance designs with the organization of parallel-distributed data processing. Another modern design is the USARSim project [5], which is aimed at precise simulation of the communication environment of acoustic signals between the group of underwater robots and divers for coordination of joint operations. However, this paper does not provide for full universalism, which would allow new possibilities to be added and the set of simulation sensors to be expanded easily. This also applies to several other researches not mentioned in this short review.

The proposed simulation framework (SF) is the development of previous work by the authors concerning the creation of an SF on the basis of a single computer [6–8]. The SF was designed with orientation to simulation of the hardware–software part of the control system for AUVs developed by the Institute for Marine Technology Problems (IMTP, FEB RAS). A distinctive feature of the previously developed simulators was a wide range of simulated data as well as the possibility to adjust the developed algorithms directly in the control system environment of the selected AUV. Modular architecture made it possible to accumulate the algorithm base and expand the range of simulated sensors, but rather quickly it became clear that the performance of one computer is insufficient for real-time simulation of the whole range of onboard equipment. For example, a simulation task of an underwater-utility-lines inspection process on the basis of side-scan sonar data processing [10] was carried out tens of times slower than real-time.

Equipping AUVs with new sensors makes provisions for opening new possibilities for implementation of AUV intelligent behaviour. The application of multibeam sonars or forward-area sonars (Figs. 1 and 2) implicitly makes it possible to solve tasks on the recognition and classification of objects or obstacles on acoustic images onboard AUVs with subsequent adequate robot reactions to these events. However, increased volumes of information from sensors [11, 13] further aggravate the problem of real-time simulation of the whole range of onboard equipment with the use of the computational capacities of one computer. Besides, a real-time environment is necessary for the connection of AUV onboard equipment to a simulator in HIL mode.

So, it became necessary to develop SF modification with more advanced architecture and use of distributed computing, which, in its turn, requires the development of a more advanced multi-user interface and new functional capabilities.

II. SF Functional Capabilities

Several stages of AUV engineering and testing can be considerably simplified through the application of an SF application, for example, to solve a number of tasks aimed at engineering, exploration, and debugging of robot intelligent algorithms:

- debugging of AUV algorithms and motion control methods;
- debugging of automatic navigation algorithms, including SLAM (Simultaneous Location and Mapping);
- debugging of algorithms of computer vision, identification of objects by side-scan sonars, sector-scan sonars, forward-area sonars, bathymetric side-scan sonars, side-scan sonars with linear frequency modulation, multibeam echosounders, and onboard video cameras for tasks such as mapping, searching for sunken objects, moving along lengthy objects, and surveying of quay piers;
- debugging of intelligent-behaviour algorithms for bypassing obstacles, re-planning missions, and testing of emergency situations.

The above are possible due to simulation of all onboard systems of UVs and the environment in SF. Other functional capabilities allow:

- the use of SF as a simulator for AUV telemetry training in manual mode (maximum approximation to real imitation of an AUV operator workplace terminal);

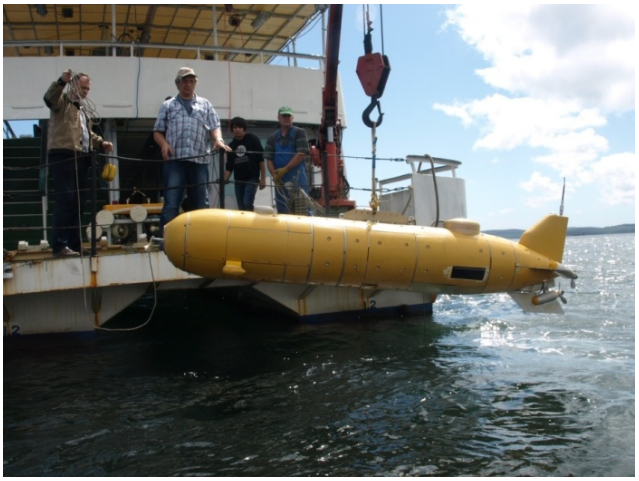

Fig. 1. AUV MMT-2012 with multibeam echo sounder GeoSwath

Fig. 2. AUV Pilgrim. A similar vehicle to be equipped with forward-area sonar

- development, exploration, and specification of AUV missions, taking into account emergencies and circumstances;
- testing of hardware and software as well as onboard software of real UVs in the case of their connection to the virtual environment of the simulation complex in HIL mode.

III. Proposed SF Architecture and Its Functioning

The modular principle on whose basis all functional blocks were designed as separate program dynamics modules (onboard sensors, dynamics, control algorithms, etc.) was the basis of the previous SF version. Within modulation frames depending on the mission, the necessary dynamic modules were automatically connected to SF. This approach made it possible to accumulate the algorithm base without changing the basic application of the program complex. The main and most material weakness of the previous version of SF was a shortage of computational capacity of one computer.

The architecture of the current version of SF is based upon usage of the following approaches and technologies:

- distributed calculations – to provide for real-time calculations;
- client-server model – for centralized complex control;
- use of a database for centralization and unification of data flows;
- plug-in technology;
- hybrid (GPGPU+CPU) parallelism in functional blocks.

The mentioned architectural solutions significantly allow the use of previous exploratory work, including the modules of the dynamics and onboard sensors. The client-server part of SF consists of three functional software applications (Fig. 3).

- The server provides connection of clients with the database and supervisor. The supervisor coordinates the work of the whole distributed SF.
- Manager – the parts of the distributed SF providing for data exchange between separate modules and server (supervisor). These applications are located on different computers joined to the network and act as intermediates between separate modules and the server, thus joining the whole complex into a single distributed SF. Besides, managers contain copies of the database data of the last simulation stage.
- Interface – a software application allowing the user to launch mission execution, have control over the simulation process, and carry out online and recorded visualization of the results; it acts as the terminal for manual control in the simulator mode.

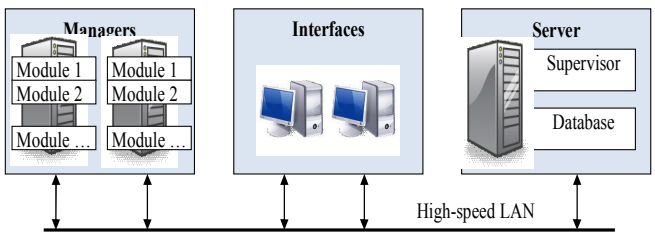

Fig. 3. Distributed simulation framework layout

Distributed SF architecture allows several users to work simultaneously (for example, in the case of the coaching regime for SF working as a simulator for the operator [12]). The interface for each user can be configured on a case-by-case basis. This makes it possible for several operators to simultaneously observe the process of the mission from different computers or to analyze the recorded mission independently of each other.

Software implementation makes it possible to launch applications both on different computers connected by LAN and on a cluster. The user on a stand-alone basis chooses which modules will work as part of the software managers, thus distributing the computational load on nodes.

SF functioning takes place in three stages. Configuration of complex logical structure is carried out in the first stage. It makes it possible to determine the exact working scheme of the complex, that is, which computers will be used with which applications, and then to choose and launch the server, managers, and user interfaces. The user provides the managers with a set of necessary simulation modules. Then, at the second stage, the loading of mission files to the server as well as the loading of all necessary data about the simulation environment and UV are carried out with the help of the user interface. At this stage, an automatic check of the availability of all necessary computation and simulation modules in the SF is carried out, otherwise simulation would be impossible. In the third stage, the simulation process is launched. The user can supervise the simulation process in real-time mode. Underwater situation layouts, video cameras, sensors, onboard instrument readouts, and so on can be viewed on the user monitor. The user determines the configuration of the sensors at his or her own discretion. During the mission, all simulation data are entered into the database and are available for analysis of the results after completion of the mission. The user can look through the results at any point of time as a video recording, moving along the timeline and playing the recording.

Besides three basic functional software applications, SF includes additional software applications for vehicle configuration and creation of the layout of the underwater environment. They are of an auxiliary nature and are kept in a text format that allows them to be edited in any word processor.

A. SF server

The SF server part is implemented in a software application that joins the database and supervisor.

The database performs centralized storage of the simulation results. From the beginning of the simulation in SF, a mission execution timer ("model time") is started as if a real vehicle were working in a real environment. During simulation, the

time is divided into regular intervals (tick marks), where calculation of the AUV's position in the virtual world with the help of vehicle dynamics calculation takes place. So, the simulation starts with a zero tick mark where the initial vector of the AUV state (position, speed, orientation) is known. During each tick mark a new vector of the vehicle state is put into database. Readouts of sensors and onboard instruments can be nominally divided into two types by reference to the data update rate: high-speed and slow. Within the given context, high-speed instruments such as the magnetometer and depth sensor can report their readouts in any period of time. Their readouts are put into database at each simulation tick mark. Slow sensors, which need time to form feedback to receive readouts, are put their data into database at the moment of readouts receiving with the mark of "model time" of their receipt. In this case, we mean the time of response formation which a real sensor would need in actual practice, not the time spent by the computer for simulation. All types of sonars and instruments requiring time for measurement are considered to be among the slow sensors. Besides the sensors' and instruments' readouts, the database keeps information about the environment and the position of dynamic objects of the environment in each period of time. So, two objectives are achieved: storage of the virtual environment and the AUV state in each period of time for further analysis of mission progress as well as for the exchange of information between SF distributed modules.

For unification purposes, all types of data kept in the database are rearranged into a one-dimensional array of bytes called a data packet. Data packets can be of different sizes depending on the type of sensor simulated. The format of these packets is predefined for each sensor and can be decoded by any SF module. Extraction of the necessary information from the database is carried out by means of a driver. The driver provides a universal interface for data exchange between SF modules using the "data identifier" and "time of their receipt", regardless of the type and structure of database. If information has not yet been requested from the slow sensor at the moment of time if the response has not yet been formed, its last readout is taken from the database.

The supervisor coordinates the work of the distributed SF by carrying out the following at each tick mark:

- increasing the timer tick counter and requesting simulation of the repositioning of the AUV and dynamic objects in the environment;
- forming a list of polled sensors and waiting for the end of sensor simulation by all managers at a given tick mark;
- running the AUV intellectual control algorithms and forming the control signals.

In the first stage, the supervisor increases the timer tick counter and polls dynamics modules of the vehicle and environmental dynamic objects. When the data about the change of environment and AUV motion are received, the supervisor sends the data about the changed virtual environment and new position of the robot to all managers. Then the supervisor forms a list of the sensors which are expected to give a response at a given tick mark and requests that the vehicle's sensors and onboard instruments collect the readout at the moment of the given tick mark. Then it delivers these readouts to the managers. The list of requested sensors includes all high-speed sensors and slow sensors which are to send responses to the database. In the final step, the supervisor makes a request to the AUV intelligent control modules. All steps are carried out in sequence, but simulation modules at each of these steps can work independently. So, it is possible to achieve the best performance by distributing the modules of a separate step between different managers during configuration of the SF logical structure.

B. Database and data communications protocol

All data of all simulation stages are kept in a database. All communications and exchanges of simulation results between modules in SF are carried out via the network. As mentioned, all of the data in the database are recorded in the form of a sequence of bytes with the tick mark and information on the module from which they were received. So, the database is a 2D table, whose cells contain byte arrays. The software implementation of such a database distinguishes itself by simplicity and high speed of data extraction from indexed tables. Database data are divided into two types:

- static data, that is, data that are not changed during the whole simulation process: visual models of the environment, AUV, mission, environment configuration data, and so on;
- dynamic data, that is, data produced during the simulation process.

The data communications protocol is based upon the TCP/IP protocol with guaranteed delivery and data integrity. The scheme of data packets formation consists of serializing the data from the sensor with a tick mark of the data receipt time and the sensor mark from which the data were received. Serializing schemes are established for each type of sensor or onboard instrument during the design of the relevant module.

Several types of onboard instruments form a great deal of data (Table 1). For such sensors, the database possesses a mechanism of data storage only for the last point of time and the rest of the readout history is stored on the hard disk drive. So, economy of the short-term memory necessary for the database is achieved. At the same time, the user of the simulated mission has the ability to carry out a visual analysis. With such organization of data storage during one hour of simulation, a short-term memory accumulates about 1.5 Gb of data, which is not a problem for modern computers.

C. Interaction of modules with server

1) Managers

The key role of managers is to act as intermediates between the SF modules and server. During launch, the manager establishes communication with the server, receives the AUV configuration, mission, environment, and vehicle data files from the server. It initializes the necessary plug-in modules according to the configuration files. Besides, the managers provide the modules with the data exchange interface. That is,

the functional modules transfer the simulation results and look for new data through their manager, which is responsible for communications with the server and database and for automatic re-establishment of communications in the case of breakage and for data transfer.

TABLE I. CALCULATION OF DATA VOLUME TRANSFERRED INTO THE DATABASE (1 TICK MARK = 0.01 SEC.)

Data Type	Data Volume			
	Bytes per transfer	*Transfers per second*	*Number of instruments*	*MB/hour*
AUV state vector	48	100	1	16
Side-scan sonar	1 000	5	2	34
High-frequency side-scan sonar	6 250	5	2	225
Side-scan sonar with linear frequency modulation	132 000	2	2	1 800
Multibeam echosounder	–	–	–	800
Colour video cameras (jpg) 640 × 480 × 15fp	~92 160	15	2	10 000

Figure 4 shows the scheme of data exchange in the SF. In the manager, all enquiries to the drivers go through a database cache. If data at a given tick mark have already been taken from the database, then the inquiry through the driver will not be carried out and data will be taken from the cache. Caching is carried out for unloading of the local area network. At each simulation tick mark, dynamic records are deleted.

The manager works as follows. The supervisor forms a list of sensors and modules that will work at the next step and sends a signal about the beginning of the next step to all of the managers. Through the driver, the manager takes this list (one list for all managers) and, checking with the list, chooses those sensors and modules that are synchronized with a given manager by giving them control one by one. After executing a piece of work, the modules transfer the result to the database on the server through the driver. So, if the record in the database is updated, the supervisor considers that the module and corresponding sensor have completed their work and proceeds to the next step.

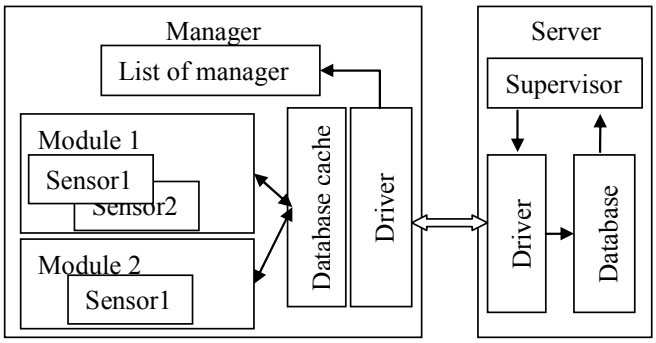

Fig.4. Scheme of data exchange between SF modules

The list is performed in sequence, so all computer resources are used by a single module for more efficient parallelism within the frames of this module.

2) Interfaces

All users' work with SF is carried out through a client application providing a user interface. To eliminate contradictory instructions from different users, only one client can control the simulation, while the rest can connect only in the view mode of the simulation results. The user launches the interface application, connects to the server, and chooses the configured mission, and then all data files necessary for mission simulation are automatically loaded onto the server. Then, if the current SF configuration complies with the mission requirements, that is, if all SF components and modules required by the mission are present, the user can initiate the process of mission simulation. With the help of the user interface, one can view data on the current mission, AUV configuration, and onboard sensors configuration. A view of the underwater situation is located in the main window of the user interface. There are two modes of monitoring of the underwater situation:

- free operation: in this mode, the user can move freely along the underwater space;
- AUV following mode: a virtual camera follows the UV and is always targeted at it; the angle can be selected by the user.

There are six types of widgets designed for viewing sensors' simulation data: an indicator (needle and digital), a time-based historical graph, a widget for side-scan sonar, a widget for sector-scan sonar, and a widget for photographs and videos.

During mission execution or analysis, the user chooses which robot sensors will be viewed, places widgets at the screen, and sets the display parameters, for example, minimum and maximum values, widget sizes, and color.

Figure 5 represents a possible user interface configuration. Information about the project tree and several project parameters are located to the left. An overall view of the underwater situation and views from two inboard video cameras are located to the right. Also side-scan sonars images for both sides are located at the bottom, and text information about the camera parameters and AUV position and orientation are located in the bottom right part of the window.

D. Functional modules

1) Network exchange module (driver)

The driver provides functions for data exchange between modules and data storage, transmitting data via the network, and addressing the database. Communication is established via TCP/IP protocol. Data are communicated in asynchronous mode, that is, the transmitting side does not wait for the end of communication. The driver works as follows. The module transmits the ID of the simulated sensor and serialized data to the driver for data communication. The driver transmits this information to the supervisor via the network. The supervisor attaches a tick mark to the sensor data packet and places it into

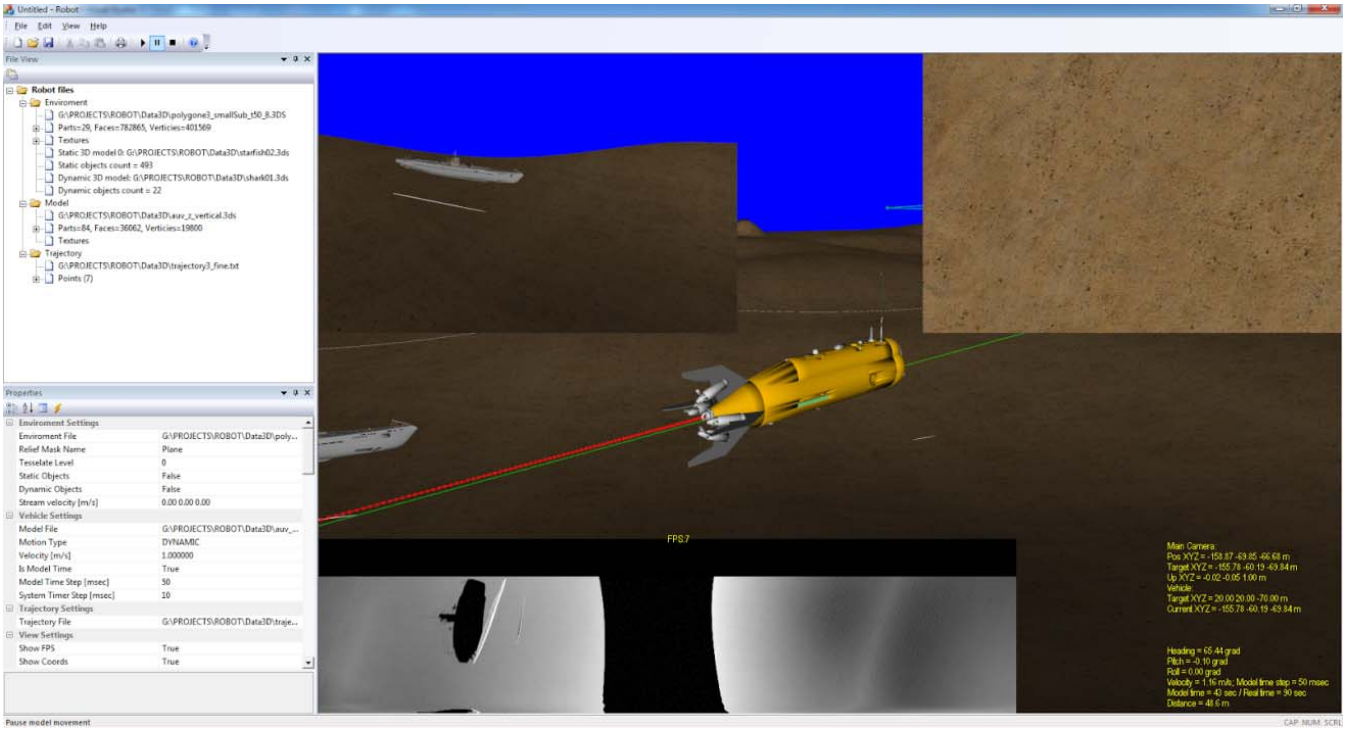

Fig.5. User interface

the database. To receive data from the database, on the basis of the data ID, the module receives the packet of data saved in the database during the previous step.

2) Video cameras and photo cameras

The module implementation of video or photo cameras requests from the database an environment model, the AUV position in absolute coordinates, and the photo camera position onboard the robot in the related coordinate system. Information about device parameters from the AUV configuration is requested as well. This is done as a matter of setting a virtual sensor window in absolute coordinates, after which rendering of the underwater situation takes place. If necessary, in order to create a more realistic situation filters are applied to imitate noise, optical distortions, hot spots, and so on. Images are placed into the database after the application of the filters.

3) Sonars

Simulation of sonars is detailed in article [6]. The algorithm described was implemented in the form of modules for three types of sonars: single-beam distance gauge, side-scan sonar, and sector-scan sonar. Let us analyze the implementation of the sonar module on the example of a single-beam distance gauge. The module requests from the database an environment model, the AUV position with respect to the world coordinates, and the sonar position onboard the robot in the bound coordinate system. Information about sensor parameters from the AUV configuration is requested as well. This is done for setting a virtual server window in the world coordinates; then, rendering of the underwater situation and recalculation Z-buffer into response string takes place. The nearest non-zero response is chosen from the response string and transformed into distance. Then these data are communicated into the database through the driver. The side-scan sonar module sends the whole response string to the database. The side-sector sonar, besides calculating the position in the world coordinates, performs a turn of the sensor virtual window. The angle of the turn depends on the time that has passed since the beginning of the mission.

4) Navigation sensors

Depth sensors, accelerometers, compasses, and logs are simulated for simplicity on the basis of data about robot position and navigation in the medium, which is exactly known from the data on the AUV motion. To create more realistic situations, distortions or noises can be introduced into the readings of these devices.

5) Communication medium simulation

To imitate a communication medium, a scheme is used that supports both the simplified simulation of signal propagation in aqueous media and a physically more precise and realistic simulation. In each period of time the transmitting side leaves a data packet in the database that must be transmitted together with the precise position and parameters of the transmitting antenna (known from the data on the AUV configuration and dynamics). The receiving side, knowing the position of the source, environment data, and the parameters of the receiving set, introduces noise and distortion peculiar to receiving-transmitting devices and propagation media into the initial data. As a result, one may simulate both simple attenuations of acoustic signals in underwater modes as well as more complicated transmission devices. In the simplest case, if the distance between the transmitting and receiving antennas exceeds the specified limit the signal is considered lost. In complicated cases, a similar scheme allows not only the

parameters of the modems, but AUV traveling speed, terrain configuration, temperature, salinity, water currents, and the simulation of the partial loss of signal and peculiar miscommunication of information to be taken into account. Such an approach provides the possibility of debugging high-level algorithms of data transfer between the modems.

6) AUV dynamics simulation

The robot's motion simulation module calculates the vehicle's dynamics under certain controlling variables of the propeller system. For realistic simulation of each vehicle, a unique module is designed. It implements the dynamics of a given vehicle taking into account its design features and set of propellers [7, 8], although there are universal mathematical models of UVs [9].

E. Configuration files

Simulation of AUV behaviour in the virtual world requires awareness of the geometry and object properties, robot sensors, and interaction of the world and the AUV submerged in the virtual world. So, the simulation complex must be properly adjusted. Adjustment is carried out in XML text files containing the structure of the objects engaged in simulation of their properties. The structure means the hierarchic interaction of objects such as bottom features with stones and underwater structures and telemetering buoys attached to the structure. The features mean the geometric form defined by the polygonal network, geometric properties, color, texture, reflectivity, and temperature, if necessary. Properties of the environment are defined as well: temperature, salinity, currents, and so on.

The virtual environment (topography, environmental objects, their properties, etc.), vehicle (geometrical model, set of onboard sensors and their positions), and sensors (performance characteristics) require configuration. Models of these objects (environment, vehicle, sensors) are described plausibly by a hierarchical tree-type structure. The "simulation project" is at the top hierarchical level. It is a part of the description necessary for unification at the next level of equivalent nodes: "environment", "vehicle", and "mission". An alternative hierarchical version was analyzed, with the "mission" node at the top level. That is, the AUV "mission" is developed for a specific "vehicle" and "environment". But such an approach does not allow the development of missions that are not synchronized with a certain vehicle and area. A bottom survey with photographing and mapping with the use of side-scan sonar can serve as an example. In such a way, a developed mission can be applied to any area and any vehicle that is potentially capable of performing this mission, that is, that possesses a photographic camera and side-scan sonar.

IV. SF TESTING

For the purposes of performance evaluation of the distributed approach, an experiment with installation of two SFs on two computers with equivalent characteristics connected by a 100 MB network was carried out. The computer parameters were Intel i5-760 @ 2.8 GHz NVIDIA GTX-470. The server, user interface, and manager were launched on the first computer. The manager was launched on the second computer. Performance of distributed SF was compared with the performance of the single-processor version of SF. The input data were the virtual scene, one AUV equipped with two side-scan sonar channels and activated noise-masking model, a five-beam echo-sounder system, and one color video camera. The simulation time-step was 0.01 s, and the virtual scene refinement was ~800,000 triangles. Each of the simulated sensors carried out full scene rendering. The video camera took 15 photographs per second; the range sensor and side-scan sonar signals were processed 5 times per second. The computational load was distributed between the managers as follows: the first computer simulated the video camera, the portside side-scan sonar, and two-beam echo-sounder system operation, while the second one simulated starboard the side-scan sonar and three-beam echo-sounder system operation. As a result of the computational experiments it was proved that in the single-processor version of SF, simulation of 100 seconds of mission took 263 seconds of computer time, which does not correspond to real-time operation mode. The distributed SF on two computers took 164 seconds. So, a twofold increase of SF performance increased the computational speed by ~1.6 times (efficiency = 62%), but it failed to provide real-time operation mode.

The second experiment was carried out with a configuration consisting of one computer and a giant-powered SMH11 computer with eight computation nodes. The configuration of one SMH11 node is 2 × CPU Xeon L5506, 2.13 GHz, 2 × NVIDIA Tesla M2050 GPU computing processors with 3 GB, 32 Gb RAM, and Linux (Gentoo)+x-server, and a 1 Gb/s network. The server and user interface were launched on the computer. Three nodes were engaged on the cluster (one manager per node) with load distribution was launched on each node: the first node had a video camera and two side-scan sonars, the second node had a three-beam echo-sounder system, and the third node had a two-beam echo-sounder system. It took approximately 80 seconds to complete the mission. This corresponds to the real-time operation mode. In the case of use of all eight computation nodes, with one sensor per node, computational time decreased by up to 40 seconds. With this version of SF, most of the time is spent on data exchange via the network.

V. CONCLUSION

SF testing on computational complexes with different configurations validated the accepted architectural concepts and gave encouraging results in real-time mode implementation during simulation of the operation of AUV onboard sensors. Further development of the complexes is expected to be as follows:

- optimization of the complex architectural and network structure to decrease the time taken by data exchange;
- development of simulation methods for long-range acoustic sensors;
- development of approaches to SF integration into a modified, event-oriented software environment of AUV control;
- debugging of intelligent-behaviour algorithms with the use of advanced features.

A series of papers devoted to the solution of the aforementioned problems is expected to be published.

ACKNOWLEDGMENT

This work is part of integrated project "Modern technologies and technical means of monitoring the state of marine ecosystems and marine biological resources" [13]. The reported study was partially supported by the Russian Foundation for Basic Research (RFBR) under research project No. 15-07-00341. Also the work is supported by Grant of Russian Science Foundation (project No. 14-50-00034) concerning statement of the problem and carrying out the computational experiments.

REFERENCES

[1] Kuroda, Y., Aramaki, K., Ura T. "AUV Test Using Real/Virtual Synthetic World", Proceedings of IEEE Conference on Autonomous Underwater Vehicle Technology. 3–6 June, Monterey, CA (1996). [Online]. Available: http://ieeexplore.ieee.org/iel3/3780/11039/00532436.pdf, accessed 8 Dec. 2014

[2] Choi, S.K., Yuh J. "Design of Advanced Underwater Robotic Vehicle and Graphic Workstation", Proceedings of IEEE International Conference on Robotics and Automation, 2–6 May, Atlanta, GA (1993). [Online]. Available: http://ieeexplore.ieee.org/iel2/904/7227/00292131.pdf, accessed 8 Dec. 2014

[3] Gracanin, D., Valavanis K.P., Matijasevic M. "Virtual Environment Testbed for Autonomous Underwater Vehicles", Proceedings of 8th International Conference on Advanced Robotics, ICAR'97, 7–9 July, Monterey, CA (1998). [Online]. Available: http://ieeexplore.ieee.org/iel3/4880/13463/00620272.pdf, accessed 8 Dec. 2014

[4] Parodi, O., Lapierre, L., Jouvencel, B. "Hardware-in-The-Loop Simulators for Multi-Vehicles Scenarios: Survey on Existing Solutions and Proposal of a New Architecture", in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, Sept. 2009, St Louis, MO, USA (2009).

[5] Sehgal, A., Cernea, D. "A Multi-AUV Missions Simulation Framework for the USARSim Robotics Simulator", Proceedings of 18th Mediterranean Conference on Control & Automation: MED), 23–25 June 2010, pp. 1188–1193 (2010).

[6] Bobkov, V.A., Morozov, M.A. "Sonar Devices Simulation by Computer Graphics", Underwater Investigations and Robotics, Vol. 1, No. 13, 47–51 (2012).

[7] Bobkov, V.A., Borisov, Yu.S., Inzartsev, A.V., Mel'man, S.V. "Simulation Program Complex for Studying Motion Control Methods for Autonomous Underwater Vehicles", Programming and Computer Software, Vol. 34, No. 5, 257–266 (2008).

[8] Bobkov, V.A., Morozov, M.A., Bagnitskiy, A.V., Inzartsev, A.V., Pavin, A.M., Shcherbatyuk, A.F., Tufanov, I.E. "Imitation Simulation Complex for Survey Autonomous Underwater Vehicle", Electronic Magazine "Science Visualization", IV quarter, Vol. 5, No. 4, 47–70 (2013).

[9] Fossen, T.I. "Guidance and Control of Ocean Vehicles", John Wiley & Sons, p. 494 (1994).

[10] Bagnitsky, A., Inzartsev, A., Pavin, A., Mel'man, S., Morozov, M. "Side Scan Sonar Using for Underwater Cables & Pipelines Tracking by Means of AUV", Proceedings of Underwater Technology (UT), 2011 IEEE Symposium on and 2011 Workshop on Scientific Use of Submarine Cables and Related Technologies (SSC), 5–8 Apr. 2011, Tokyo, Japan (2011).

[11] Tahri, O., Araujo, H., Chaumette, F., Mezouar, Y. "Robust Image-Based Visual Servoing Using Invariant Visual Information", Robotics and Autonomous Systems, Vol. 61, No. 12, 1588–1600 (2013).

[12] Inzartsev, A.V., Sidorenko, A.V., Senin, R.A., Matviyenko, Yu.V. "Comprehensive AUV Software Debugging Based on Simulation Modelling Complex", Underwater Investigation and Robotics, Vol. 1, No. 7, 9–14 (2009).

[13] Kulchin, Yu.N., Mayor, A.Yu., Proschenko, D.Yu., Zhizhchenko, A.Yu., Golik, S.S., Babiy, M.Yu., Mirochnik, A.G. "Modification of a new polymer photorecording material based on PMMA doped with 2,2-difluoro-4-(9-antracyl)-6-methyl-1,3,2-dioxaborine by ultrashort pulses", Quantum Electronics, 2015, 45 (5), 477–481 DOI: 10.1070/QE2015v045n05ABEH015764