Proceedings of the 2017 IEEE
International Conference on Robotics and Biomimetics
December 5-8, 2017, Macau SAR, China

An Application of a Generalized Architecture to an Autonomous Underwater Vehicle

Changlong Lin, Guoliang Zhang, Jing Li, Yan Chen, Yiwen Zhang
College of Computer Science and Technology
Huaqiao University
Xiamen, China
hurrey@163.com

Yiping Li
State Key Laboratory of Robotics
Shenyang Institute of Automation, CAS
Shenyang, China
lyp@sia.cn

***Abstract*—A common feature of unmanned vehicles is their complexity, which grows apace and provides its own challenges. Frameworks for managing this growing complexity have always been one of the key aspects of designing an unmanned vehicle. In our previous work, a generalized architecture is proposed to not only address the complexity in developing an unmanned vehicle, but also support the algorithm exchange and technology transfer for integrating efforts from different researchers. In this paper, we applied this architecture to the control system of a practical vehicle. In the development of the control system, object oriented method is adopted and the specialization process of element classes from abstract to detail is presented. Then standards for the architecture is discussed. Finally, field experiments are carried out to validate the effectiveness of this architecture.**

Keywords—Autonomous Underwater Vehicle; architecture; generalized

I. INTRODUCTION

Thanks to the research and development activities in the field of Autonomous Underwater Vehicle (AUV) in the past decades, the technologies of AUV have gradually matured and AUVs have found an increasingly wide application in scientific, civilian, and military sectors. However, as the requirement of the capabilities of AUVs raises, the missions for AUVs are getting more and more complicated and in consequence the control systems become much more complex. As a result, system structuring has become inevitably paramount and also the only practical and effective starting point in the development of AUVs [1].

The architecture of a robot defines how the job of generating actions from percepts is organized [2], it is the discipline devoted to the design of highly specific and individual robots from a collection of common software building blocks [3]. That is to say, the architecture forms the backbone of complete control system of an AUV [4]; it is the framework where the following processes are implemented: control laws, errors detection and recovering, path planning, tasks planning and monitoring of the events along the execution of a particular mission [5]. The right choice of the architecture will greatly facilitate the specification, implementation and validation of robotic systems [4].

This work was supported by the National Natural Science Foundation of China under Grant No. 61403150, the State Key Laboratory of Robotics under Grant No. 2012-O02, and the Natural Science Foundation of Fujian Province under Grant No. 2017J01113, 2016J01302, and 2015J01258.

Since Shakey was presented in 1971, a spectrum of architectures have been developed and applied to different unmanned vehicles. It is widely acknowledged that these architectures could be classified into three categories: the deliberative architecture [6, 7], the reactive architecture [8, 9], and the hybrid architecture [10, 11].

The deliberative architecture, also called the hierarchical architecture, adopts a top-down approach [12]. It uses a functional and hierarchical decomposition to divide the system into levels; the higher levels are responsible for the overall mission goals, while the lower levels for solving particular problems to accomplish the mission [13, 14]. It emphasizes the planning process which is strongly based on a world model, thus it has the ability to reason and make predictions; however, it lacks the flexibility to react to dynamic environments.

On the contrary, the reactive architecture, also referred to as the behavioral architecture, follows a bottom-up approach. It consists of behaviors, which run in parallel, continuously reacting to the situation sensed by the perception system. The vehicle's global behavior emerges from the combination of the elemental active behaviors [5]. Although it is well suited for dealing with highly dynamic environments, it fails to achieve non trivial objectives due to the lack of high-level control.

The hybrid architecture can be considered as a combination of the deliberative and the reactive architectures for taking their advantages and minimizing their limitations. Generally it consists of three layers: a deliberative layer for reasoning and planning at the top, a reactive layer for quick response at the bottom, and an intermediate layer to provide the right mix of reactivity and deliberation. It is widely recognized as the most suitable architecture for an unmanned vehicle.

Since almost all recently developed architectures have been classified as hybrid, grouping of architecture might not be an interesting issue anymore [15]. Our interest already moved to a generalized architecture which is associated with standards and compatibility. This paper presents our latest efforts. We have proposed a generalized architecture for several years, but the validation of it merely remains on the semi-physical experiments, this depresses the persuasiveness of the effectiveness of this architecture. Recently, we have developed the control system of an AUV under this architecture and

978-1-5386-3742-5/17/$31.00 ©

experiments of terrain reconnaissance have been carried out, we expect that this would be the first step toward field application of this architecture.

The rest of this paper is organized as follows. Section II describes our previous work briefly, that is, a generalized architecture for AUVs. Then a control system under this architecture for terrain reconnaissance is developed and the specialization process of element classes from abstract to detail is presented in Section III. Some issues about the standard are discussed in Section IV. Section V demonstrates the field experiments for terrain reconnaissance mission. Finally, conclusions are drawn in section VI.

II. A Generalized Architecture for AUVs

The design of an architecture is much more of an art than a science, thus it is really difficult to evaluate an architecture quantitatively. However, there are still several acknowledged desirable features that a good architecture should meet [5, 16, 17] namely predictability, reactivity, robustness, modularity, extendibility, generality, and standardization. In our opinion, these features could be divided into two aspects: the first three mainly focus on the ability of the vehicle, while the others emphasize the process of development, maintenance and upgrade. Since the proposal of the hybrid architecture, the features of the first aspect has mostly been achieved. On this basis, there is a strong need for a generalized architecture which enables exchange of algorithms across laboratories and technology transfer so as to shorten the development cycle of AUV control systems and to enhance its mission adaptability. This expectation is consistent with the features from the second aspect. Though several institutes and organizations have devoted their effort on this problem and some progress has been made [18-20], it is still far from being satisfactory.

In our previous work, we have also proposed a generalized architecture. We extract each independent module of the system into a node with a common infrastructure, and then organize the nodes hierarchically to obtain the architecture. This section provides a brief description of this architecture, relevant details can be seen in [21, 22].

A. Design of the Autonomic Elements

The working mechanism of each independent module in the system can be generalized as two steps: perception and then decision making. That is to say, each module contains a "perception - decision" closed loop. Thus each module can be extracted into a node called Autonomic Element (AE), which consists of three basic subassemblies: namely perception subassembly, decision subassembly, and database and knowledge subassembly, as shown in Fig. 1.

The perception subassembly firstly collects the information, which is composed of data information, status information, and task progress information, from its resources via sensor. Then data fusion is carried out not only to obtain the best estimation of its status and the environment, but also to check whether a fault occurs. Finally, it passes the latest estimation to world model and knowledge subassembly for record and update; and report to its superior AE when it detects a malfunction or its resource has just recovered from a fault.

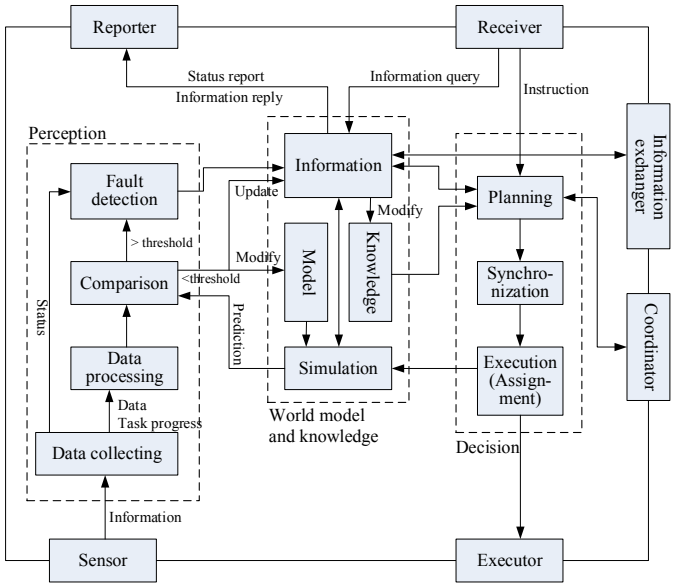

Fig. 1. Details of the Autonomic Element

Based upon the goals received from the superior AE, cooperative instructions from peer AEs, and the best estimation of the state and environment by the perception subassembly, the decision subassembly creates a series of instructions to achieve the goals. After synchronization, these instructions are assigned to its corresponding subordinate AEs or peer AEs for implementation.

The world model and knowledge subassembly stores all the information about its state and environment. Besides, by combining the latest estimation of the state and environment and the instructions that would be sent to its subordinate AEs, it utilizes the world model to simulate and predict the state at the next time instant. Furthermore, knowledge management is also offered to manage the algorithms and rules needed in the process of perception and decision making. Machine learning methods can be carried out to modify the knowledge.

In addition to the three subassemblies, it has six interfaces: reporter and receiver are used for interaction with its superior AEs, information exchanger and coordinator for peer AEs, and sensor and executor for its resources respectively. (The resources here refer to hardware devices if the AE is at the bottom level; otherwise they refer to adjacent subordinate AEs.) Human operator can also access any AE via information exchanger when debugging the system. Actually, these interfaces can be integrated into one by defining different types of messages. Here, we separate them for clarity.

B. Design of the Generalized Architecture

Although all of the AEs share a common infrastructure, they are filled with different data, algorithms, and rules, hence they play different roles depending on where they reside in the architecture. After the decomposition of the system based upon the mission, we can identify the functionality and content of each AE. Then we develop all the AEs accordingly and obtain the whole control system. Via their “perception-decision” closed-loop, the AEs at the bottom level is responsible for the control and management of its corresponding devices; and then

each higher level AE takes charge of several AEs at the adjacent lower level. Thus the system is constructed by a number of AEs integrated in a hierarchical and nested manner.

We take the terrain reconnaissance mission as an example and build the generalized architecture shown in Fig. 2. Since a single AUV is employed, the architecture contains only one AE in the vehicle level. In this mission, the AUV is about to cruise to and fro alternately along adjacent parallel segments carrying multi-beam sonar mounted at the bottom to scan the terrain in some designated area. Besides, the AUV requires to communicate with the onshore control station to download the mission and report its position and status. Thus there are 3 AEs in the subsystem level: Navigation and control AE, Communication AE and Detection AE, which are under control and supervision of the vehicle AE. The component level consists of 10 AEs which are subordinate to different subsystem AEs. Each of them is responsible for managing one or more sensors or actuators of the AUV. For instance, the Gyro AE is in charge of switching on/off the gyro, setting its parameters, receiving and analyzing its signals to obtain the angles and angle rates of the AUV.

In the architecture, every AE is capable of planning according to its current state and goal. In addition, it can respond quickly to the feedback and modify its actions. The AEs at the bottom level have the shortest spatial and temporal scope but with the most detail. Toward higher levels, the range in space and time increases, accompanied by the decrease in resolution. This limits the computational overload in each AE.

III. Development of the Control System

To address the challenges of maximizing code reuse and simplifying code integration, we use an object-oriented method that separates abstract functionality from the actual implementation. An abstract class highlights the behaviors of a module while hiding its runtime model and implementation details, it can be extended to have several implementations and runtime models to support different applications. A specialized class, which inherits from an abstract class, adapts the generic functionality to the actual physical platform. The advantage of this development process is that it provides various levels of well-defined abstractions of the system. Additionally, it provides generic and flexible interfaces for developing algorithms.

The relationship between abstract classes and specialized classes in the process of control system development is shown in Fig. 3. Element class is the most abstract class, from which we derive the Device Element class, Subsystem Element class, and Vehicle Element class. Furthermore, the Device Element class can be extended to get the Gyro Element class, Doppler Element class, and so on. Similarly, we obtain the Navigation Element class, Communication Element class, and Detection Element class from Subsystem Element class. All the classes mentioned above are more or less abstract, we can derive specialized classes from them according to the mission, the employed AUV, and the devices equipped on the vehicle.

The process of specialization from Element class to My Gyro Element class is shown in Fig. 4. In the Element class, only the most general variables and functions are defined. In

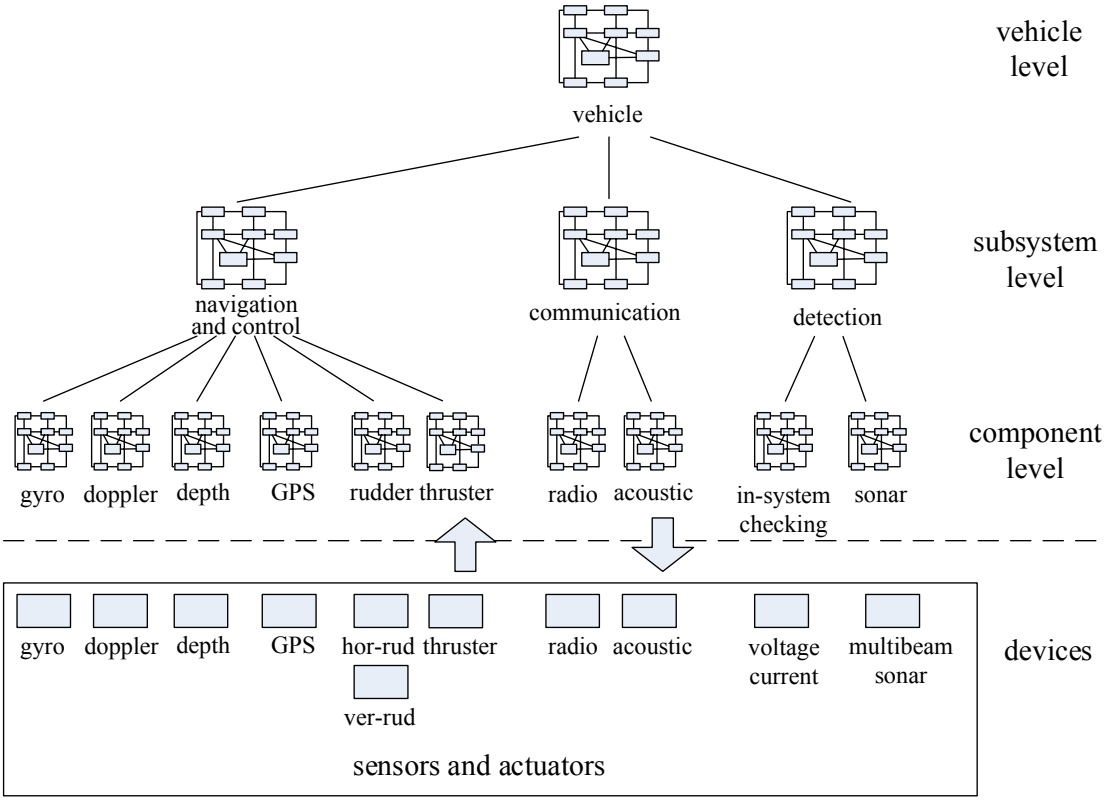

Fig. 2. The generalized architecture for terrain reconnaissance mission

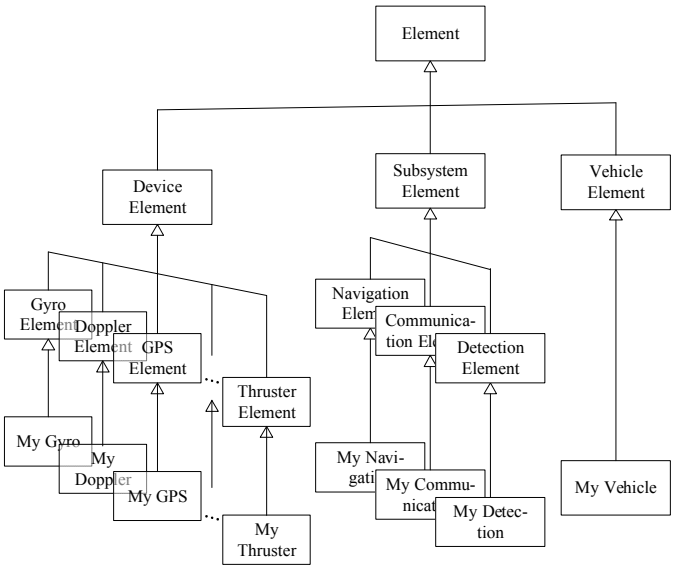

Fig. 3. The relationship between abstract classes

addition to the attributes inherited from the Element class, the Device Element class introduces 3 variables and 2 member functions. Based upon this, the Gyro Element class furtherly adds 2 variables for recording angles and angle rates corresponding to its functionality. Besides, it realizes the member function of message sending and processing for communication with its superior Element. Finally, My Gyro Element class specializes 3 member functions according to the specification of the practical gyro device: data analysis function for extracting the angles and angle rates signals from the output serial string of the device, device switching for turning on/off the device, and plan generation for device error handling.

IV. DISCUSSTIONS

A generalized architecture will not go too far without the support of a series of technical standards which impose some beneficial constrains on it. In the process of the development and application of this architecture, we have paid special attention to the published standards on AUVs or even other types of unmanned vehicles. Unfortunately, the drafting of a standard is such a hard and time consuming job which requires the collaboration of the whole community that currently available literatures are quite few. Some related work include the definition of terminology [23, 24], the specification of interfaces [25, 26] and some technical guide [27-29]. The specification of the interfaces is what we concern most among the standards. It is associated with two problems: message formats and command vocabularies.

Message formats detail how AEs identify and communicate with each other or with human operator. There are 4 types of messages in the control system at present: query for obtaining data and status from peer or subordinate elements, report for informing its superior or peer elements when an event is detected, instruction for task assignment to its subordinate elements, and coordination for the synchronization of peer elements when executing tasks. All the messages begin

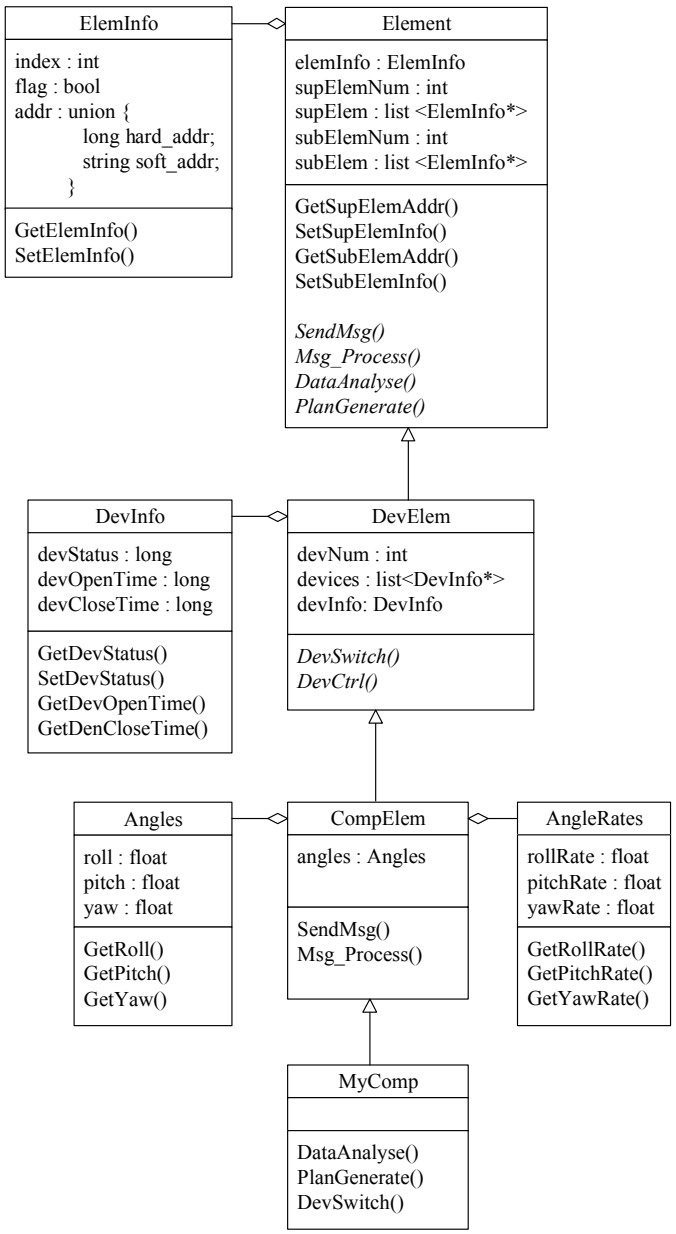

Fig. 4. The process of specialization from Element class to My Gyro Element class

with a header that indicates the source, destination, type, and length of the message followed by the message payload which is rigidly defined according to its type.

A command vocabulary is the set of named actions or tasks that an element can perform. Each element in the control system has its own unique command vocabulary corresponding to its distinctive capability. Command vocabularies determines the instructions an AE would receive from its superior and what it can assign to its subordinate; hence it specifies the duty of the AEs and moreover the vertical partition of the system, namely the border between layers. Referred to the work in NIST [26], we have tried to define the command vocabularies for our architecture. However, it is a provisional result which requires further discussion and modification.

V. EXPERIMENTS

The AUV employed in these experiments is shown in fig. 5. It is about 2.5 meters long, 20 centimeters in diameter, and weighs about 60 kilograms in the air. The vehicle is capable of reaching a depth of 100 meters and cruising at a velocity of 5 knots. It is propelled and controlled by a thruster and a pair of rudders mounted on the stern. Autonomous navigation is aided by a suite of instruments in the waterproof hull, and terrain information is gathered by a multi-beam sonar fitted on the bottom of the AUV.

After the development of the control system, tests and experiments from single AE to the whole AUV control system are successively carried out. Firstly, AEs in the component level are tested to see whether they can gather and process sensor signals or instruct the actuators effectively. Besides their capabilities to detect the faults of devices and to inform their superior AEs are also checked. Then subsystem tasks, each of which involves a subsystem AE and some or all of its subordinate AEs, are employed to verify the functionalities of these subsystems. Take the navigation and control subsystem as an example, tasks of cruising at a designated heading, depth or altitude, waypoint navigation, precise tracking, and so on are used to examine the maneuverability of the AUV. Finally, field experiments of the AUV for terrain reconnaissance have been carried out to validate the effectiveness of this generalized architecture. On receiving the mission via wireless communication, the AUV cruises to the vicinity of the designated area. Then it switches on the sonar and voyages along a lawn mowing trajectory to collect the terrain information. After the area has been completely scanned, the AUV returns to its starting point for recovery. Fig. 6 shows the trajectory of AUV in the horizontal plane in one of the experiments, and the map of the designated area is depicted in Fig. 7.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have applied a generalized architecture we proposed previously to the control system of a practical vehicle. The control system is developed based on the object oriented method; and the specialization process of Element classes from abstract to concrete has been presented. Some issues about the standard have been discussed, and field experiments have been carried out to validate the effectiveness of this architecture.

Due to the fact that this is the first time we apply the architecture to a practical AUV, we choose a relatively simple mission in our experiments. However, the control system could be easily adapted for a more complex mission because of the generality of the architecture.

Besides, this architecture is suitable for all unmanned vehicles. Due to the constraint of our experiment condition, we mainly focus on AUVs. The differences between AUVs and the others unmanned vehicles during the development of control systems are the problems we need to consider in future.

Fig. 5. The employed AUV

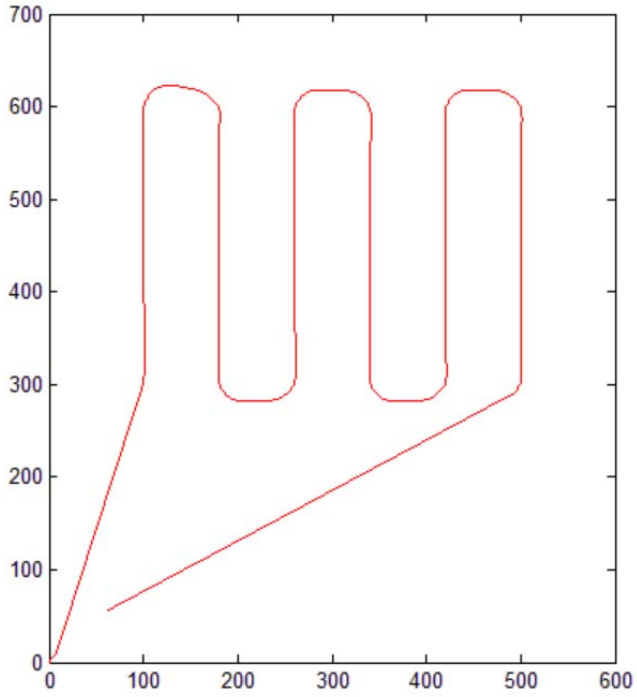

Fig. 6. The trajectory in the horizontal plane

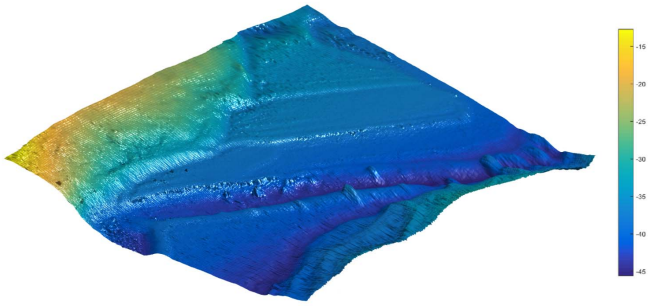

Fig. 7. The terrain of the designated area

REFERENCES

[1] H. Tianfield, "Structuring of large-scale complex hybrid systems: From illustrative analysis toward modelization," J. Intell. Rob. Syst., vol. 30, no. 2, pp. 179-208, February 2001.

[2] S. Russel, and E. Norvig, "Artificial Intelligence: A Modern Approach," Applied Mech Materials, vol. 9, no. 2 ,pp. 215–218, 2002.

[3] R. Arkin, Behavior-based Robotics, MIT Press, 1998.

[4] E. Coste-Maniere and R. Simmons, “Architecture, the backbone of robotic systems,” in 2000 Proc. IEEE. Int. Conf. Rob. Autom., pp. 67-72.

[5] P. Ridao, J. Batlle, J. Amat, and G. N. Roberts, “Recent trends in control architectures for autonomous underwater vehicles,” Int. J. Syst. Sci., vol. 30, no. 9, pp. 1033-1056, September 1999.

[6] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, “A deliberative architecture for AUV control,” in 2008 Proc. IEEE. Int. Conf. Rob. Autom., pp. 1049-1054.

[7] D. Yoerger, A. Bradley, and B. Walden, “System testing of the autonomous Benthic explorer,” in 1994 Proc. IARP 2nd Workshop on: Mobile Robots for Subsea Environments, pp. 159-170.

[8] J. Rosenblatt, “DAMN: a distributed architecture for mobile navigation,” J. Exp. Theor. Artif. Intell., vol. 9, no. 2-3, April 1997.

[9] J. Bellingham, T. Consi, R. Beaton, and W. Hall, “Keeping layered control simple,” in 1990 Proc. Symposium Autonomous Underwater Vehicle Technology, pp. 3-8.

[10] S. M. Smith, “An approach to intelligent distributed control for autonomous underwater vehicles,” in 1994 Proc. Symposium on Autonomous Underwater Vehicle Technology, pp. 105-111.

[11] J. Yuh, Underwater Robotic Vehicles: Design and control. Albuquerque. NM: TSI Press, 1995.

[12] J. Albus, R. Lumia, and H. McCain, “Hierarchical control of intelligent machines applied to space station telerobots,” Trans. on Aerospace Electron. Syst., vol. 24, pp. 535-541, September 1988.

[13] E. Coste-Maniere, H. Wang, and A. Peuch, “Control architectures: What’s going on?,” Proc. of the Int. Program Development in Undersea Robotics & Intelligent Control: A Joint U.S./Portugal Workshop, Portugal, pp. 54-60, March 1995.

[14] K. Ganesan, S.M. Smith, K. White, and T. Flanigan, “A pragmatic software architecture for UUVs,” Proc. of the I996 Symp. on Autonomous Underwater Vehicle Technology Canada, vol. 2, pp. 209-215, June 1996.

[15] H. Choi, and J. Sur, “Issues in software architectures for intelligent underwater robots,” Robot Intell. Tech. App. 2, Springer International Publishing, pp. 831-839, 2014.

[16] A. Oreback and H. Christensen, “Evaluation of architectures for mobile robotics,” Auton. Robots., vol. 14., no.1, pp.33-49, January 2003.

[17] W. Li, C. Henrik, and O. Anders, “Architecture and its implementation for robots to navigate in unknown indoor environments,” Chin. J. Mech. Eng. Engl. Ed., vol. 18, no. 3, pp. 366-370, September 2005.

[18] J. S. Albus, "4-D/RCS reference model architecture for unmanned ground vehicles," in 2000 Proc. IEEE. Int. Conf. Rob. Autom., pp. 3260-3265.

[19] M. Ricard and S. Kolitz, “The ADEPT framework for intelligent autonomy,” Belgium: Von Karman Institute, 2002.

[20] S. Whitsitt and J. Sprinkle, "Message modeling for the joint architecture for unmanned systems (JAUS)," in 2011 Proc. IEEE. Int. Conf. Eng. Comp. Sys., pp. 251-259.

[21] C. Lin, X. Feng, Y. Li, and K. Liu, “Toward a generalized architecture for unmanned underwater vehicles,” in 2011 Proc. IEEE. Int. Conf. Rob. Autom., pp. 2368-2373.

[22] C. Lin, and Y. Li, "Study on mission reachability problem for multiple AUVs based on Object-oriented Petri Net," in 2015 Proc. IEEE. Int. Conf. Cyber Tech. Autom. Ctrl. Intell. Syst., pp. 1259-1264.

[23] H. Huang, “Terminology for specifying the autonomy levels for unmanned systems Version 1.0,” NIST Special Publication 1011, National Institute of Standards and Technology, Gaithersburg, Md, USA, 2004.

[24] ASTM Standard F2541, “Standard Guide for Unmanned Undersea Vehicles (UUV) Autonomy and Control,” ASTM International, West Conshohocken, PA, 2006

[25] S. Rowe, and C. Wagner, "An introduction to the Joint Architecture for Unmanned Systems (JAUS)," January 2008.

[26] US Department of Commerce, NIST, “4D/RCS Version 2.0: a reference model architecture for unmanned vehicle systems,” NIST Interagency/Internal Report, 2008.

[27] ASTM Standard F2545, “Standard Guide for Unmanned Undersea Vehicle (UUV) Physical Payload Interface,” ASTM International, West Conshohocken, PA, 2007.

[28] ASTM Standard F2594, “Standard Guide for Unmanned Undersea Vehicle (UUV) Communications,” ASTM International, West Conshohocken, PA, 2007.

[29] ASTM Standard F2595, “Standard Guide for Unmanned Undersea Vehicle (UUV) Sensor Data Formats,” ASTM International, West Conshohocken, PA, 2006.