MATEC Web of Conferences **42**, 01005 (2016)
DOI: 10.1051/matecconf/20164201005
© Owned by the authors, published by EDP Sciences, 2016

# Local and Global Motion Planning for Unmanned Surface Vehicle

Roman Fedorenko[a], Boris Gurenko

*Southern Federal University Sciences, Research Institute of Robotics and Control Processes, 347922 Taganrog, p. Nekrasovsky, 44, Russia*

**Abstract.** The paper shows approach of unmanned surface vehicle motion planning in an environment with obstacles. The structure of the control system hardware, software architecture, based on client-server model with the loose coupling in ROS software environment, is presented. Global planner is designed using the method of Generalized Voronoi Diagrams. Local planner is implemented using the unstable regimes of control to bypass obstacles near USV. The results of simulation, which showed the efficiency of the proposed approaches, are presented.

## 1 Introduction

Field robotics is rapidly expanding into the aquatic domain [1-4]. Unmanned Surface Vehicles (USV) are able to provide long lasting operation in the aquatic domain. Thus, unmanned surface vehicles are considerably useful for tasks like environmental monitoring, wild life tracking, and search & rescue missions.

The scheme of USV usage is as follows. First, remote control station operator constructs USV mission visually on electronic map. The control system performs mapping, obstacle detection, and trajectory planning for movement between mission waypoints avoiding obstacles.

Proposed by the authors in [1-5] automatic control system allows to organize automatic motion of USV along the desired path. This article continues this work considering the motion planning process in an environment with obstacles.

The paper illustrates a structure of control system hardware, software architecture and simulation results.

Authors use ROS environment, the de facto standard in software development for robotics. ROS contains a navigation stack [6], which formed the basis of the software architecture of the USV control system described below. However, to adapt this existing software to the control object – USV – authors within this architecture has developed new motion planning software packages.

Path planning is one of the fundamental problems in robotics. There are a number of fundamentally different approaches applicable in a variety of tasks such as sampling method, the probabilistic method, the search on graphs and the method of generalized Voronoi diagrams. The most frequently used methods are the search on graphs, such as A*. Such a method is implemented in the standard global planner of ROS navigation stack, which will be discussed in more detail below. This planner builds the shortest path to the target point bypassing obstacles. However, in relation to a number of problems, including the USV motion planning in an environment with obstacles, the path length may not be the main criteria in choosing a method of planning. Another important criterion, which is responsible for the reducing of collisions probability, is the distance from the trajectory to obstacles. Therefore, new based on the generalized Voronoi diagrams global path planner software package for ROS was developed.

[a] Corresponding author: frontwise@gmail.com

## 2 Project background

This work is a part of initiative project of Southern Federal University (Russia), aimed to create a fully autonomous USV. Work described in this paper is done on the second stage of the project, which is related to the intellectual motion planning system elements design. At first stage unmanned surface vehicle hardware (shown in Fig.1), navigation and control system able to organize automatic motion along the desired path, as well as control station (shown at Fig. 2) and datalink, was designed and tested. Results of mission performing are shown in Fig. 3. From the control station 8 waypoints were given, as it is shown at Fig. 3. Mission is to go through these points in serial order, as shown by red curve at Fig. 3.

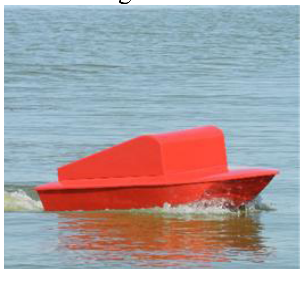
**Figure 1.** USV.

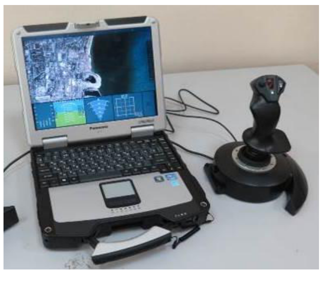
**Figure 2.** Control station.

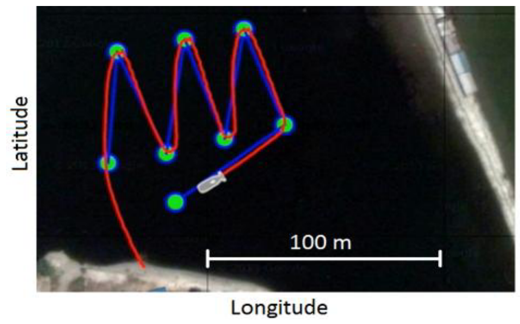

**Figure 3.** Control station while mission performing.

## 3 USV control system structure

### 3.1 USV control system hardware

USV control system hardware structure is shown at Figure 1. Paradigm of division computing part into high (computer) and low (microcontroller) level is used.

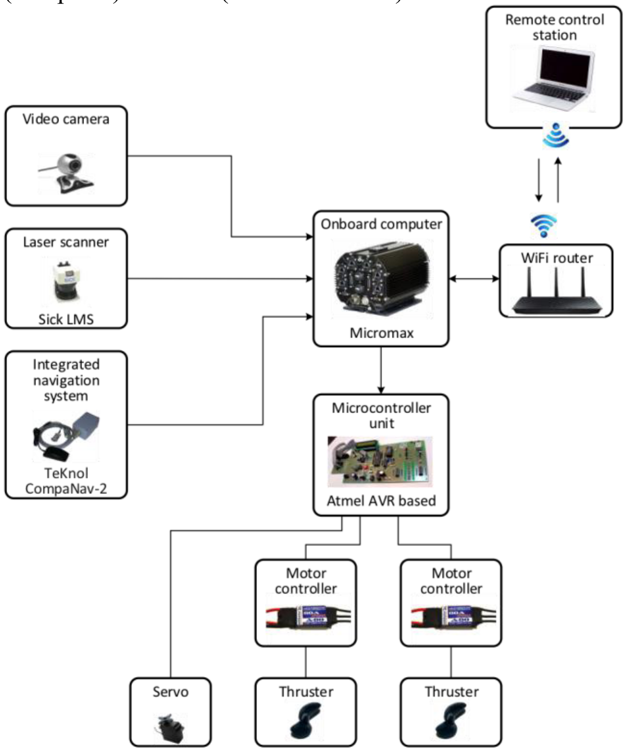

**Figure 4.** USV control system hardware structure diagram.

Microcontroller unit gets data from computer and generates pulse-width modulation (PWM) signal to motors and servos. Onboard computer performs motion planning, calculates required control actions according to control law, runs software part of navigation system, and communicates with remote control station.

### 3.2 Software Architecture

Ubuntu Linux is used as onboard computer operating system. The software is built on the Robot Operating System, which provides client-server interaction with loose coupling of components facilities.

ROS Navigation stack [6] is a basis of control system software architecture (planning part). It was configured for a USV and substantially revised in part of global and local planners.

The architecture of control system software is shown in Figure 2.

The stack uses two obstacles maps (local and global) that are built based on laser scanner and navigation system data by costmap_2d module. Global map is larger and is designed for global path planning; local obstacle map is usually smaller, has higher resolution and is designed to bypass the close-in obstacles.

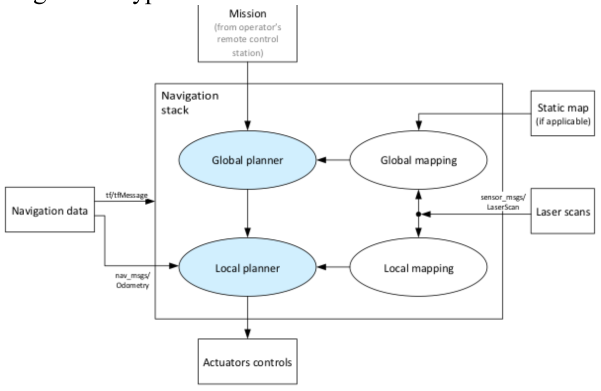

**Figure 5.** Architecture of control system software.

Global map is used by global planner, local – by local (or regulator) respectively. Global planner searches for plan on the global map and sends plan it found (in the form of a sequence of points) to the local planner (regulator). Local planner calculates actuators control inputs to perform out a global plan, and simultaneously takes care of close-in obstacles avoidance, even if it was not provided by global plan (in case of mobile or unknown obstacles).

Description of global planner based on Generalized Voronoi Diagrams is given below. As a theoretical framework for the implementation of the local planner unstable control regime approach [7] was used, as also shown further.

## 4 Generalized Voronoi diagrams global planner

### 4.1 Voronoi planner package

In USV motion planning in an environment with obstacles, the path length may not be the main criteria in choosing a method of planning. Another important criterion, which is responsible for the probability of collisions reducing, is the distance from the trajectory to obstacles. Therefore, new based on the Generalized Voronoi Diagrams (GVD) global path planner software package for ROS [8] was developed.

Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells. Thus, the boundaries of Voronoi cells are equidistant from the nearest sites.

Generalized Voronoi Diagrams are built around a set of figures on the plane (instead of the set of points). GVD boundaries are geometrical places of points equidistant from nearest figures. So it can be used to construct the path farthest from obstacles. There are a number of algorithms for constructing GVD and their software implementations. In this paper we used an open source library dynamicvoronoi [9, 10].

Our voronoi_planner package is implemented as a navigation stack plug-in that adheres to the nav_core::BaseGlobalPlanner interface specified in the nav_core package.

Package reads the global obstacles map, converts it and transmits it to the GVD construction library dynamicvoronoi. Further three-stage search is performed: path from the current USV location to GVD search; path from the target point to GVD search; path search on GVD between two GVD points near the start and target position, as shown at Fig. 6.

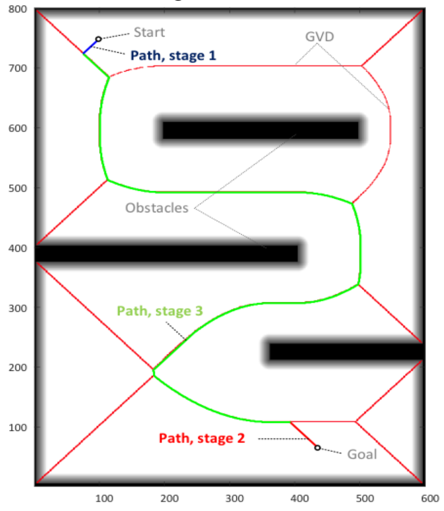

**Figure 6.** Three stages of path search on GVD.

## 4.2 Comparison of A* planner and voronoi_planner

Comparison of the trajectories produced by A* planner and voronoi_planner, given in Fig. 7, 8, shows that the path obtained by GVD planner is most distant from obstacles.

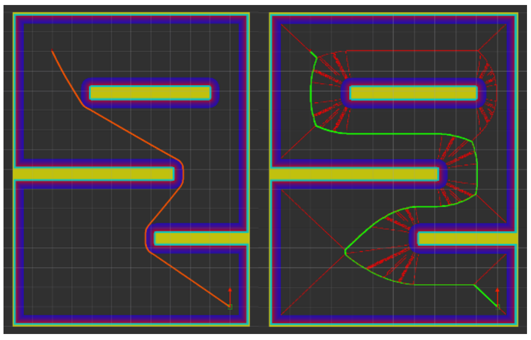

**Figure 7.** Comparison of the trajectories produced by A* planner (left) and voronoi_planner (right).

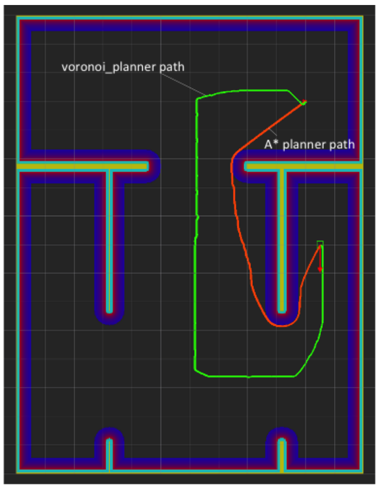

**Figure 8.** Comparison of the trajectories produced planner by A* planner and voronoi_planner, example 2.

## 4.3 Path shrinking and smoothing

Voronoi_planner path may contain too much point and we may want to shrink it in a way that distance between its points in not less than given by deleting intermediate points. Example is given at Fig. 9.

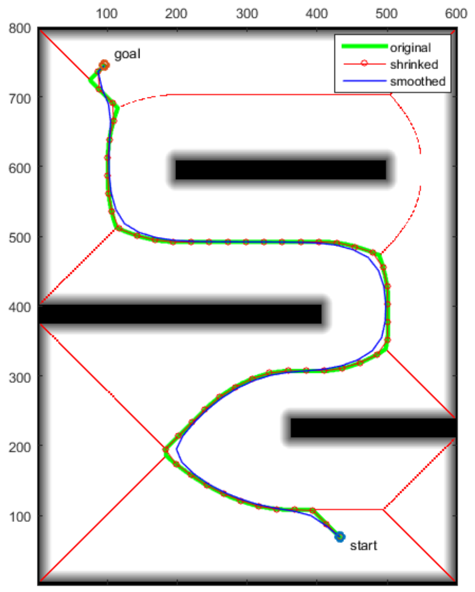

**Figure 9.** Example of path shrinking and smoothing.

In some cases, a path generated by voronoi_planner has some disadvantages, because it is not smooth and has corners, and it will force the USV to move really slowly around the corners. A much better path is generated by smoothing original voronoi_planner path. Smoothing algorithm optimize original path by two criteria by minimizing these two expressions [11]:

$$norm(P_i\text{-}S_i) \rightarrow min;$$
$$norm(S_i\text{-}S_{i+1}) \rightarrow min,$$

where $P$ – is the original path, $S$ – is new smoothed path.

The first expression minimizes the distance between the original point and the smooth point, and the second minimizes the distance between two consecutive smooth points. Obviously, these two criteria are in conflict with each other, so we minimize both with some weights.

We use gradient descent to optimize these expressions. The expression for the first objective $norm(P_i\text{-}S_i) \rightarrow min$ is following:

$$S_i = S_i + \alpha(P_i\text{-}S_i),$$

where $\alpha$ – is weight of data coefficient.

Note that $S$ is initially set equal to $P$.

The expression for the second objective is:

$$S_i = S_i + \beta(S_{i-1} + S_{i+1} - 2S_i),$$

where $\beta$ – is weight of smooth coefficient.

Example of path shrinking and smoothing is given at Fig. 9.

### 4.4 Performance of voronoi_planner

Performance is, of course, voronoi_planner drawback in comparison to A* planners. However, the voronoi_planner performance evaluation data, provided in Table 1 and Fig. 10, shows that with proper choice of map size and resolution this planner may be used for the global planning problem in real-time.

The measurement was carried by following commands: timestamp of the target point setting acquisition ("rostopic echo /move_base/goal/header/stamp" command) and plan reception timestamp acquisition ("rostopic echo /move_base/VoronoiPlanner/plan/header/stamp" command).

**Table 1.** The voronoi_planner plan computation time for maps of various sizes.

| Map Size | The average plan computation time, s |
|---|---|
| 1200 X 900 | 2,8 |
| 800 X 600 | 0,8 |
| 600 X 450 | 0,5 |

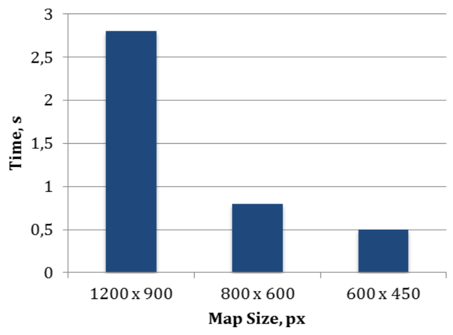

**Figure 10.** The voronoi_planner plan computation time for maps of various sizes

## 5 Local planner

In this paper we propose an approach of implementation of USV tactical control level (local planner) with the involvement of a third Lyapunov's theorem (theorem on instability) that allows forming of control actions in real time, with the weakening of the requirements for the strategic level and the minimum requirements to sensors. This approach has been proposed and studied by professor V.Kh. Pshikhopov [7]. In summary, the proposed approach is as follows. We introduce an additional parameter β, which is formed as follows:

$$\beta = \sum_j \left|R_c^j\text{-}R\right| + \sum_j (R_c^j\text{-}R) \qquad (1)$$

where j – is a number of the nearest point of one or several obstacles within range of sensors, $R_c^j$ – the distance to the j-th point of the obstacles, $R$ – the minimum critical distance to the obstacle (distance at which obstacle avoidance is activated).

Obviously, if all the inequalities $R_c^j > R$ are true, parameter $\beta$ value is zero and consequently parameter $\beta$ is not equal to zero if at least one of the inequalities $R_c^j > R$ is violated. In which case the value is always positive. In the latter case the value of $\beta$ is always positive.

The equations of a closed-loop system, as shown in [1], is as follows

$$T_1T_2\ddot{\Psi}_{tr} + (T_1\text{+}T_2)\dot{\Psi}_{tr} + \Psi_{tr} = 0$$

Stability of USV motion along the path is provided by a positive definite matrixes $T_1$ and $T_2$, which are the parameters of the control law presented in [1].

Hence if one of conditions $R_c^j > R$ is false, one or both matrixes $T_1$ and $T_2$ must be negative definite.

We assume that $T_1 = T_2 = diag\ (t_1, t_2)$, where $t_1, t_2$ – are some functional parameters, and the elements of the matrixes $T_1$ and $T_2$ are set by the following function

$$t_i = f(x) = \begin{cases} t_0 = const, & \beta = 0 \\ -\dfrac{1}{\beta}, & \beta \neq 0 \end{cases} \qquad (2)$$

where t0 specifies the character of movement in the obstacle-free zone.

Thus, forming the matrixes $T_1$ and $T_2$ elements according to (3), we provide a stable motion of USV along the manifold $\Psi_{tr}$ (stable motion along given path), with the exception of areas in which at least one of conditions $R_c^j > R$ is false.

In case of violation of conditions $R_c^j > R$, matrixes $T_1$ and $T_2$ elements are set in accordance with the second expression of (3), and USV goes into an unstable movement till zeroing of the parameter $\beta$, i.e. the exit of USV into free of obstacles zone.

As prof. Pshihopov shown [7], this approach has the following limitations:

– targeted blocking of USV motion by the obstacles or other objects, when the object cannot exit the unstable motion;

– if the obstacles are of a rather complex shape, such as a labyrinth, global planner should be used;

– control objective may not be achieved if the available power of the object does not match the dynamics of non-stationary obstacles.

The main advantages of this approach are the simplicity of its implementation and that obstacle avoidance in a non-formalized environment does not require the construction of paths in the area of the obstacles that in some cases is not always possible in real time.

## 6 Simulation results

To confirm the efficiency of the proposed solutions computer USV simulator was designed. Simulator allows to model USV motion in an environment with obstacles, to set actuators controls, receive simulated navigation and laser scanner data [5].
Fig. 11 shows a three-dimensional scene of USV exit out from bay simulation and corresponding map obtained through the simulated laser scanner, and the planned USV path.

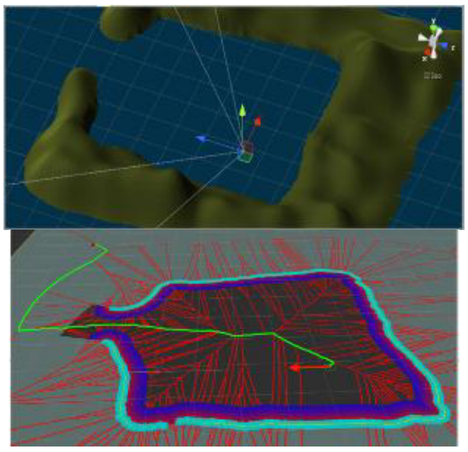

**Figure 11.** USV exit out from bay simulation

Local planner simulation results are shown at Fig. 12. Motion of USV at global path with obstacles that were not previously known is considered. The task of the local planner is to make a local path, excluding collision with obstacles and return to the global trajectory.

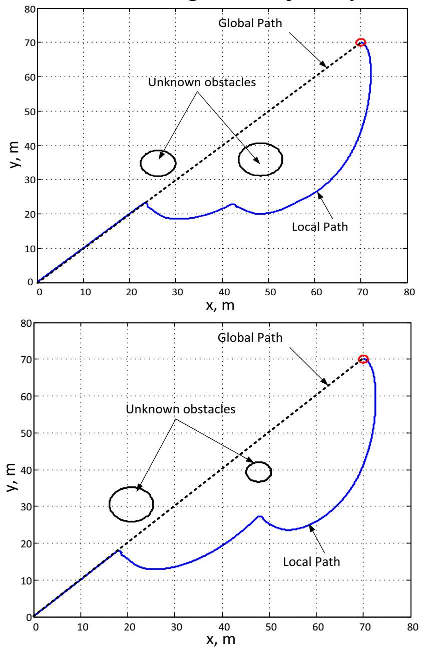

**Figure 12.** Local planner simulation

Simulator usage helped to confirm the efficiency of the proposed solutions and move on to the preparation of full-scale tests.

## 7 Conclusions and further work

The proposed modification of described software architecture components (global and local planners) in accordance with the simulation results allow the use of the proposed solutions for the future field experiments with a USV.
Our development plans include:

– experiments with proposed planning system conducting;

– using gyro stabilized platform for LIDAR or developing data filter algorithm to work in pitching;

– LIDAR, sonar and computer vision data fusion to enable safe navigation in real environment with static and dynamic obstacles;

– conducting further experiments with external disturbances (wind and water current) to test and adjust disturbances estimator and confirm the operability of the control system in different conditions.

## Acknowledgment

This work was supported by the Russian Ministry of Education and Science, research No114041540005 “Theory and methods of position-trajectory control of maritime robotic systems in extreme conditions and under uncertainty environment”, grant from the Southern Federal University “Theory and methods for energy-efficient control of distributed systems of generation, transmission and consumption of electricity” and Grant of the president of Russian Federation for state support of the Russian leading scientific school SSc-3437.2014.10.

## References

1. Boris Gurenko, Roman Fedorenko, Anatoly Nazarkin “Autonomous surface vehicle control system”, Applied Mechanics and Materials Journal.- 2014 (ISSN: 1660-9336). - Vol. 704. - p. 277.
2. E. Pinto, F. Marques, R. Mendonça, A. Lourenço, P. Santana, and J. Barata An Autonomous Surface-Aerial Marsupial Robotic Team for Riverine Environmental Monitoring: Benefiting from Coordinated Aerial, Underwater, and Surface Level Perception . In Proc. of the IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO). IEEE Press, 2014
3. Pshikhopov, V.Kh., Medvedev, M., Gaiduk, A., Belyaev, V., Fedorenko, R., Krukhmalev, V. // Position-trajectory control system for robot on base of airship. 2013 Proceedings of the IEEE Conference on Decision and Control, Pp. 3590-3595.
4. Pshikhopov, V. K., Medvedev, M. Y., Gaiduk, A. R., & Gurenko, B. V. (2013, October). Control system design for autonomous underwater vehicle. In 2013

Latin American Robotics Symposium and Competition.

5. B. Gurenko, R. Fedorenko, M. Beresnev, R. Saprykin, "Development of Simulator for Intelligent Autonomous Underwater Vehicle", Applied Mechanics and Materials, Vols. 799-800, pp. 1001-1005, Oct. 2015
6. Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, Kurt Konolige The Office Marathon: Robust Navigation in an Indoor Office Environment", International Conference on Robotics and Automation, 2010
7. V. Pshikhopov, M. Medvedev, V. Krukhmalev, V. Shevchenko, "Base Algorithms of the Direct Adaptive Position-Path Control for Mobile Objects Positioning", Applied Mechanics and Materials, Vol. 763, pp. 110-119, May. 2015
8. voronoi_planner - ROS Wiki Robot Operating System URL: wiki.ros.org/voronoi_planner (accessed: 20.09.2015).
9. B. Lau, C. Sprunk and W. Burgard "Improved Updating of Euclidean Distance Maps and Voronoi Diagrams", IEEE Intl. Conf. on Intelligent Robots and Systems (IROS). - Taipei, Taiwan. - 2010. - pp. 281 – 286.
10. dynamicvoronoi - ROS Wiki Robot Operating System URL: wiki.ros.org/dynamicvoronoi (accessed: 20.09.2015).
11. Artificial Intelligence for Robotics URL: https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373 (accessed: 20.09.2015).