Mapping Shallow Water Environments using a Semi-Autonomous Multi-Sensor Surface Vehicle

Dominik Kleiser, Alexander Albrecht, Thomas Emter, Angelika Zube, Janko Petereit, Philipp Woock
Fraunhofer Center for Machine Learning
Fraunhofer IOSB, Karlsruhe, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation
Karlsruhe, Germany
[first name].[last name]@iosb.fraunhofer.de

***Abstract*—We show a multi-sensor platform for mapping tasks of shallow water areas like rivers or harbors, both under and above the waterline simultaneously. The autonomy of the platform is realized just by integrating the sensors and the autonomy box which provides, i.a., the motion planning. The sensor data is synchronized purely in software by a newly developed timestamping filter. Maps obtained by the autonomously driven platform are shown as an example for the platform capabilities.**

***Index Terms*—ASV, autonomous surface vehicle, multi-sensor fusion, inspection, mapping**

I. Motivation

Authorities are obliged to keep a directory of reasonably recent maps of underwater areas, e.g., riverbeds or harbor areas. Usually, a fleet of mapping vessels and trained staff to operate the vessels are necessary to accomplish this. However, this is a cost-intensive task and in many cases, the mapping is done much less frequently than it would be advisable or required. This is where (semi-)autonomous surface vessels (ASV) can substantially reduce costs and the need for onboard staff. Only a supervisor will be required to ensure that the ASVs carry out their work properly and assist in difficult scenarios, e.g., when there is a lot of traffic.

Additionally, inspection of structures above water level, i.e., mostly bridges, is of equal importance. This can be achieved in passing by using the same vehicle on a mission where underwater data is captured anyway with only little additional one-time expenses and no additional ongoing expenses for the additional sensor equipment. The data is acquired in addition to the underwater data. As the platform is geo-referenced, the resulting above-water data is also geo-referenced.

II. Existing work

Employing autonomy functionality via ROS [1], [2] is a widespread approach due to the high versatility of the middleware. Of course, autonomous surface vehicles using ROS are not completely new: The very small Minnow vehicle using ROS was presented by Calce et al. [3]. The focus was also on low cost and small size in the ROS-based vessel of Mancini et al. [4]. A recent approach to unmanned surface vehicles for multi-sensor data acquisition using ROS is the catamaran-style WAM-V of Velamala et al. [5]. There are also commercially available small autonomous surface vehicles for hydrographic surveys like the Sonobot [6]. Using ROS for collision avoidance is shown in [7].

III. Solution

We created a platform that is designed to operate together with normal ship traffic, i.e., it is of a larger size to be able to maneuver stably even in the presence of ship wakes, and it can be seen more easily by ship captains. The platform is equipped with an autonomy box which allows to navigate autonomously along a path given by a set of waypoints. The waypoints may also be changed on-the-fly if a mission change should be necessary. On that path, the vehicle perceives the environment through a multitude of sensors to create a complete view of the surroundings. Due to its size, the vehicle is targeted for larger payloads and offers the necessary platform stability for data acquisition.

A. Platform

The platform is based on an inflatable catamaran, which makes it a stable platform that is well suited to capture the environment. It is based on the proven platform »Water Strider« which we used in the Shell Ocean Discovery XPrize finals [8]. The platform is now equipped with two electric motors that may be rotated in common to form the rudder angle. The multisensory equipment of the platform consists of an inexpensive IMU (inertial measurement unit) featuring also GNSS (Xsens MTi-G700), a 3D LiDAR (Velodyne VLP-16), a digital compass (Precision-9), an interferometric side-scan sonar (ITER Systems Bathyswath-2), and stereo cameras. This is depicted in Fig. 1. The platform's ability to maneuver autonomously is realized by employing our algorithm toolbox (ATB) for autonomous mobile robotic systems [9].

B. Mesh Connectivity

In order to establish the connection between the operator at the coast and the Water Strider platform, we setup a wireless mesh network based on the IEEE 802.11s standard [10] and the B.A.T.M.A.N. advanced routing protocol [11]. One notable feature of this setup is that its mechanism is relying on MAC addresses rather than IP addresses, abstracting the mesh to a single layer 2 broadcast domain. Even if the network topology is made of multiple hops, the entire mesh will look like a single layer 3 network from the user's view, thus TCP and UDP connections are not lost even when moving and changing the routing path. This allows us to bridge large distances to the platform without the loss of communication by placing

Fig. 1: Top: Autonomous surface vehicle »Water Strider« where the red square shows where the above-water sensors are placed. Bottom: Detailed view of the laser scanner, GNSS and IMU assembly.

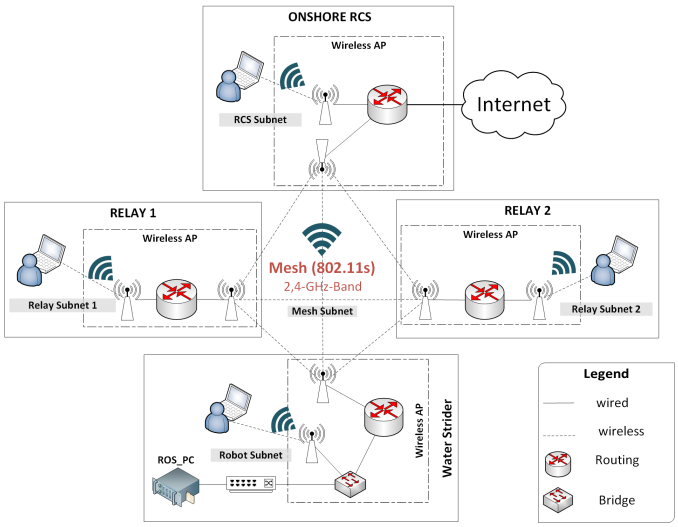


Fig. 2: Network topology.

dedicated relay nodes on the coastline or using other robotic platforms as intermediate mesh nodes. A visualization of the network topology can be seen in Fig. 2.

C. Multi-sensor fusion

1) Fusion architecture: The vehicle employs a versatile fusion filter framework for localization that is able to handle measurements from multiple sources in an asynchronous yet timing-correct way, i.e., all sensor data is processed with the timestamp of arrival in the processing system. This filter framework is based on an extended Kalman filer (EKF) and is able to flexibly handle different sensor configurations comprising an IMU, GNSS, compass, platform specific non-holonomic constraints, and different kinds of odometry in 2D and 3D from, e.g., LiDAR odometry, visual odometry or wheel encoders (in land-based applications). While the measurements of the IMU are incorporated with a strapdown algorithm in the prediction step of the EKF, the GNSS and compass measurements are integrated using correction updates. Odometry measurements like 6DoF LiDAR odometry by scan matching of consecutive pairs of scans being additional relative measurements, i.e., measuring a difference between states, are included in a correct fashion (being relative sensors) via stochastic cloning [12]. The filter framework is not only employed for the Water Strider, but also in a variety of different research platforms, ranging from wheeled platforms to heavy machinery each with its individual sensor setup [9].

2) Timestamp filtering: Typically, the sensors to be integrated do not provide a standardized technique for hardware synchronization and especially on research platforms, it is advantageous to be able to quickly change the sensor setups or add new sensors. Thus, a novel approach for synchronization based on timestamps of arrival without the need for an additional hardware installation has been developed: The clocks of the individual sensors are quite precise but will drift apart without hardware synchronization. Then again, the timestamps

of arrival of the computer processing all sensor data could serve as common reference. However, these timestamps are prone to latency jitter, because a standard PC without a real-time operating system is used. Therefore, a fusion of the precise short-term clock information of the sensors and the timestamps of arrival of the PC is employed. It is also based on a Kalman filter where the clock of the sensor is used for the prediction step and the timestamp of arrival serves for the update step.

Fig. 3 shows a comparison of the deltas of unfiltered and filtered timestamps of arrival of the IMU data with a nominal rate of 100 Hz, i.e., a cycle time of 10 ms. While both mean deltas of 10.0008 ms and 10.0001 ms respectively are very close to the nominal cycle time, the unfiltered deltas has fluctuations (jitter) of several milliseconds. The deltas of the filtered timestamps of arrival are nearly constant and no deviations are visible in the figure.

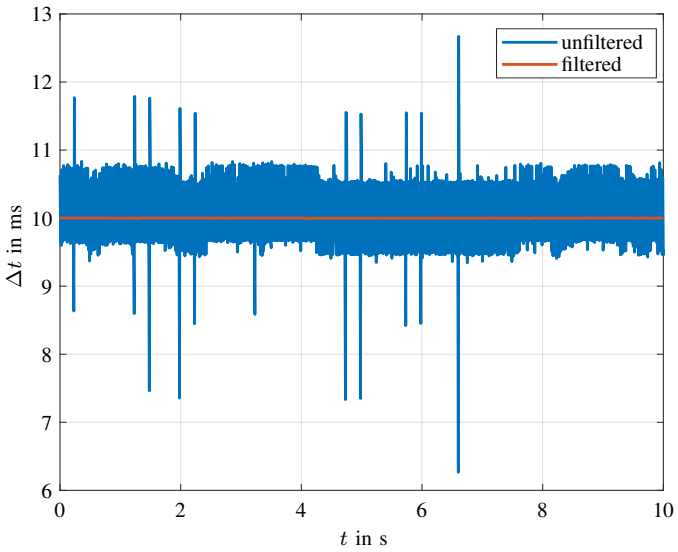


Fig. 3: Timestamp of arrival deltas of the IMU of the Xsens MTi-G700 with a nominal rate of 100 Hz.

D. Motion planning

The platform provides two different methods for motion planning: A naïve waypoint tracking control system (see Fig. 7), as it is already widely available in ship autopilots, and a more advanced path-following control system (see Fig. 9), allowing for precise navigation in tight areas that can be easily extended with obstacle avoidance capabilities.

The waypoint tracking control system takes a list of GNSS coordinates as an input. With a rate of 2 Hz the system calculates the heading difference between the ships current position and the next waypoint solving the inverse geodesic problem. Based on that, a course change command is sent to the rudder controller over NMEA 2000. When the ship enters the waypoint's radius of 5 m, the waypoint is marked as reached and the next waypoint is targeted. The vessel's speed is kept at a constant speed between waypoints and reduced linearly if the ship is in close distance to the waypoint.

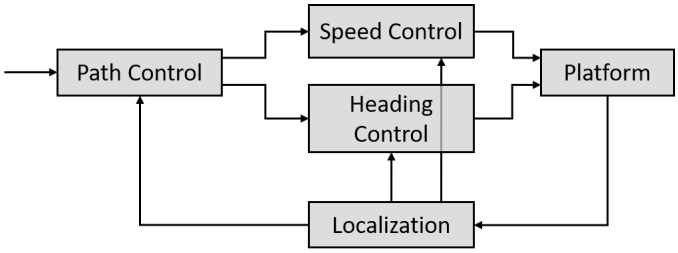


Fig. 4: Control scheme.

The path-following control system also plans the platforms motion through a given sequence of waypoints. Howevever, graph search in a state-time lattice structure is applied [13] to compute feasible paths based on the platform kinematics. If an obstacle detection is available, the planned paths are collision-free considering the platform length and width. During planning, not only the next but multiple waypoints are included in the planned path in order to achieve an optimal motion sequence. The planning is done online with a replanning rate of about 10 Hz. This allows fast reactions to unforeseen platform behavior (e.g., due to drift) or collision situations.

For ease of simplicity, we will focus in the following sections on the more advanced path-following control system.

E. Control

The platform is actuated by commanding the motors' number of revolution and the rudder angle, which rotates the frame of both motors simultaneously. In order to follow a given path, a control scheme with multiple layers is realized (see Fig. 4).

On the top level, a path controller controls the platform motion along the planned path. The path controller computes the desired platform speed and orientation based on the current platform pose determined by the localization filter by means of a nonlinear adaptive line-of-sight (LOS) control law [14]. Several constraints, like minimum and maximum translational speed, rotational speed, and maximum acceleration, are considered.

The desired platform speed is controlled by an underlying PI-controller. The actuating variables are the numbers of rotation of the two electric motors. The current platform speed is determined by the localization filter.

For heading control, a simple proportional controller is implemented. This controller computes the rudder angle according to the desired yaw angle and the current yaw angle identified by the localization filter.

F. Accessibility

The distinctive feature of our platform is its flexibility. Due to our modular software architecture (see Fig. 5), modifications and extensions of the platform's capabilities are easy. Marine systems have been traditionally created using standards defined by the National Marine Electronics Association (NMEA). Another, recent effort called Signal K [15] aims to bring a totally open data format for marine data exchange and is built on modern technologies used in web applications such as

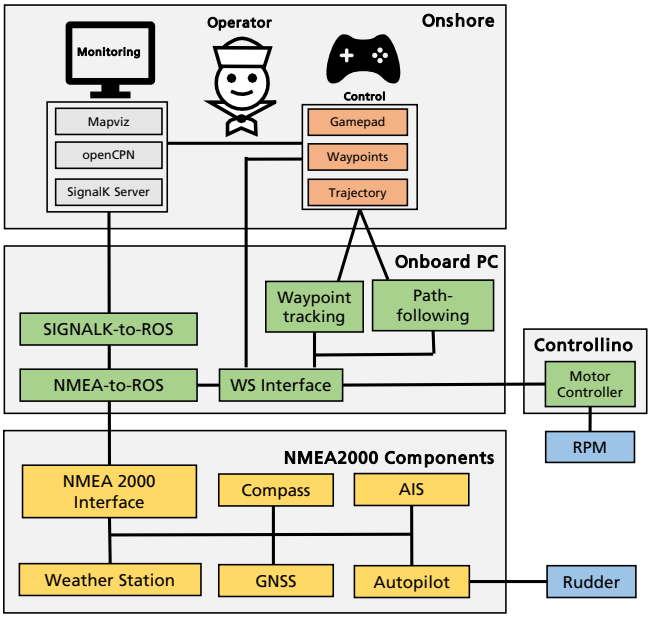


Fig. 5: Schematic overview of the vehicle components and their interaction. Colors have the following meaning: Blue denotes actuators, yellow denotes NMEA components, green denotes ROS components, orange denotes inputs and grey denotes 3rd-party components.

HTTP, JSON and WebSockets. We implemented interfaces to bridge between the ROS framework and both NMEA 2000 and Signal K, which means that our autonomy capabilities can be added to basically any vehicle that has rudder and motor accessible via one of the mentioned protocols. This also allows us to easily integrate any sensor connected to the ship's communication bus in our software. Additionally, the platform can be monitored either with graphical interfaces provided by the ROS framework, i.e., rqt, rviz and mapviz [16], or by using using open technologies like the web-based *signalk-server-node* [17] (see Fig. 6) or OpenCPN [18].

IV. Operation and Results

A. Mission Planning

The vessel is given a mission area and after generation of a trajectory covering the area, the vehicle follows that trajectory, avoiding detected obstacles in the way.

In Fig. 7, a bird's eye view of the mission points and the driven trajectory in the naïve waypoint motion planning is given. Also shown is the map comprising a point cloud of the installments on shore.

B. Control resilience

In the scenario shown in Fig. 9 , one of the motors was disabled from time to time in the marked area. The vehicle trajectory remained stable as the vehicle control continued seamlessly using the remaining motor.

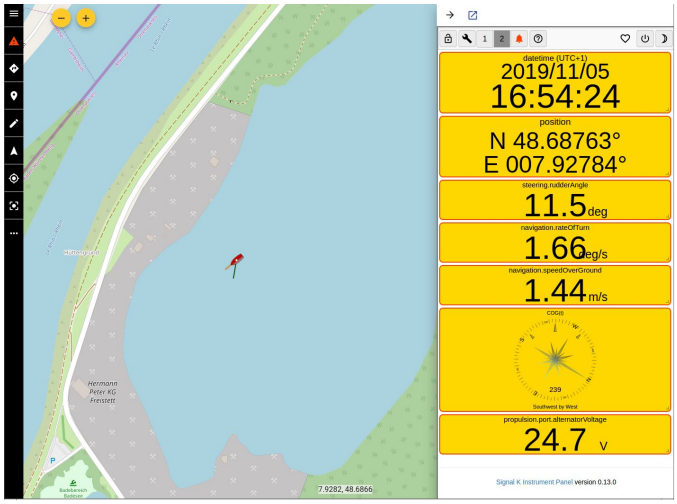


Fig. 6: Screenshot of the visualization via SignalK showing the vehicle's internals.

Fig. 7: Visualization of autonomous waypoint navigation within mapviz/ROS: The current position and heading of the vessel is given by the yellow boat icon. The red line represents the vehicle trajectory. The waypoints are displayed as a point with a radius. Once inside the radius, the waypoint is considered reached. The color convention for the waypoints is as follows: Reached waypoints are depicted in turquoise, the currently targeted WP is depicted in light green, and the remaining waypoints are depicted in white. The LiDAR point cloud is shown in magenta. Map data: © 2020 Google, CNES / Airbus, GeoBasis-DE/BKG, GeoContent, Maxar Technologies

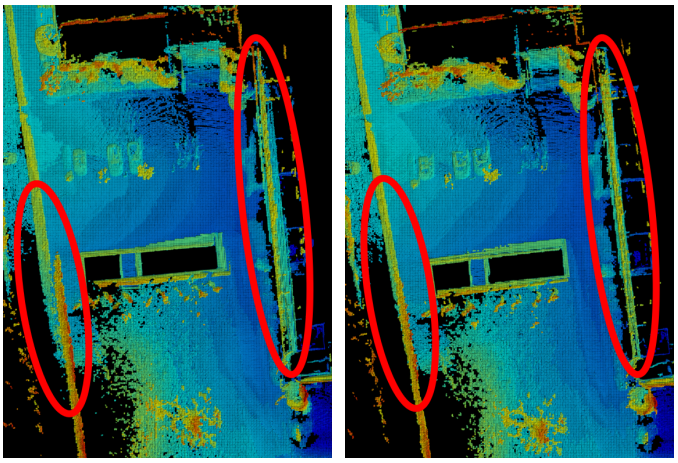
C. Localization and Mapping

In Table I, the standard deviations of the timestamps of arrival of different sensors are listed for comparison. It can be seen that the timestamp filtering leads to improvements of several orders of magnitude and the filtered deltas have standard deviations of only a few microseconds.

For the data set used for evaluation, the maximum measured deviation without filtering was less than 5 ms and with a maximum measured platform speed of 4 m/s, results in a potential error of less than ±2 cm for a position estimate. The

TABLE I: Standard deviation σ in µs.

		unfiltered	filtered
IMU	@ 100 Hz	589.2	1.6
GNSS	@ 4 Hz	2121.8	7.4
LiDAR	@ 347.22 Hz	26.8	0.3



(a) (b)

Fig. 8: Mapping result without (a) and with timestamp filtering (b). The example is taken from a land-based robot to better illustrate the benefits of the filtering. The relevant areas are marked with red ellipses. After timestamp filtering, the resulting map has much less consistency errors. The height of the map is color coded from blue (low) via green to red (high).

maximum rate of rotation measured by the IMU was below 40 °/s, resulting in a potential angular error of less than ±0.2°. I.e., the influence of the time stamp deviation on the individual localization estimates is very small. However, the filtering of the timestamps of arrival still has a positive effect on the fusion for localization and mapping, since small errors in the position have a greater effect on the 3D-LiDAR measurements. For example, an angular error of 0.2° at a distance of 50 m results in a deviation of about 17 cm. In addition, even small deviations can have a greater effect on mapping over time. As can be seen in Fig. 8 with a data set of a land vehicle on our institute campus in some areas, which are marked by red ellipses, the accuracy of the mapping can be increased by timestamp filtering.

The LiDAR data shown in Fig. 9 depicts the detail which is in the raw data. In many cases, such a level of 3D detail is not needed and a 2.5D grid map is sufficient. In Fig. 10 a grid height map of a port facility is shown which is generated by accumulating and fusing the localized 3D point clouds into grid cells. For height mapping, the *grid_map* package is utilized [19]. The grid cells have a resolution of 0.5 m. The altitude is coded by color from low (blue) to high elevation (brown). Two detailed views (enlarged by a factor of three) are also shown. The shown height map of the port facility was generated by an autonomous drive of approximately 600 m.

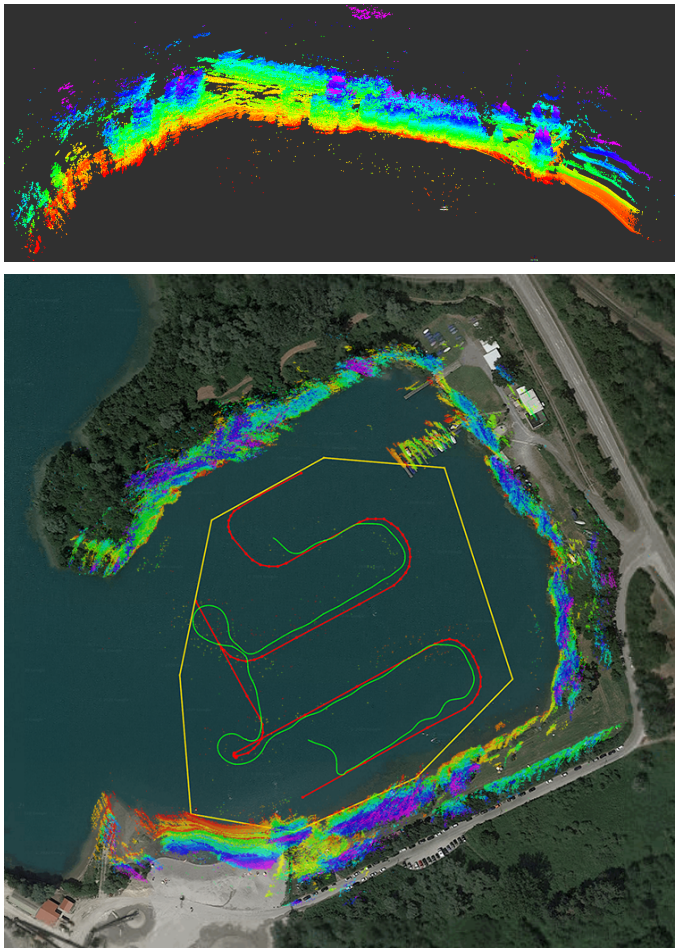


Fig. 9: Top: Slightly oblique bird's eye view of a point cloud from the shoreline with trees starting close behind the shoreline. Bottom: The mission area is depicted as yellow polygon yielding the planned trajectory (depicted in red). The actual taken trajectory is shown in green while the map in form of a height-colored point cloud is shown as an overlay. The taken trajectory does not start right at the beginning, because we needed to do a mode switch. Satellite base imagery is imported from Google Maps. Map data: © 2020 Google, GeoBasis-DE/BKG, GeoContent, Maxar Technologies

V. Outlook

For autonomous missions, obstacle avoidance is essential. At the moment, LiDAR measurements of the water surface may produce fake obstacles which the vehicle then would try to avoid. Hence, we are planning to integrate a radar sensor for a more reliable obstacle detection as it should provide a clearer separation between a true obstacle and the water surface. Since obstacle avoidance has already been employed on our land-based platforms, the integration should be straightforward.

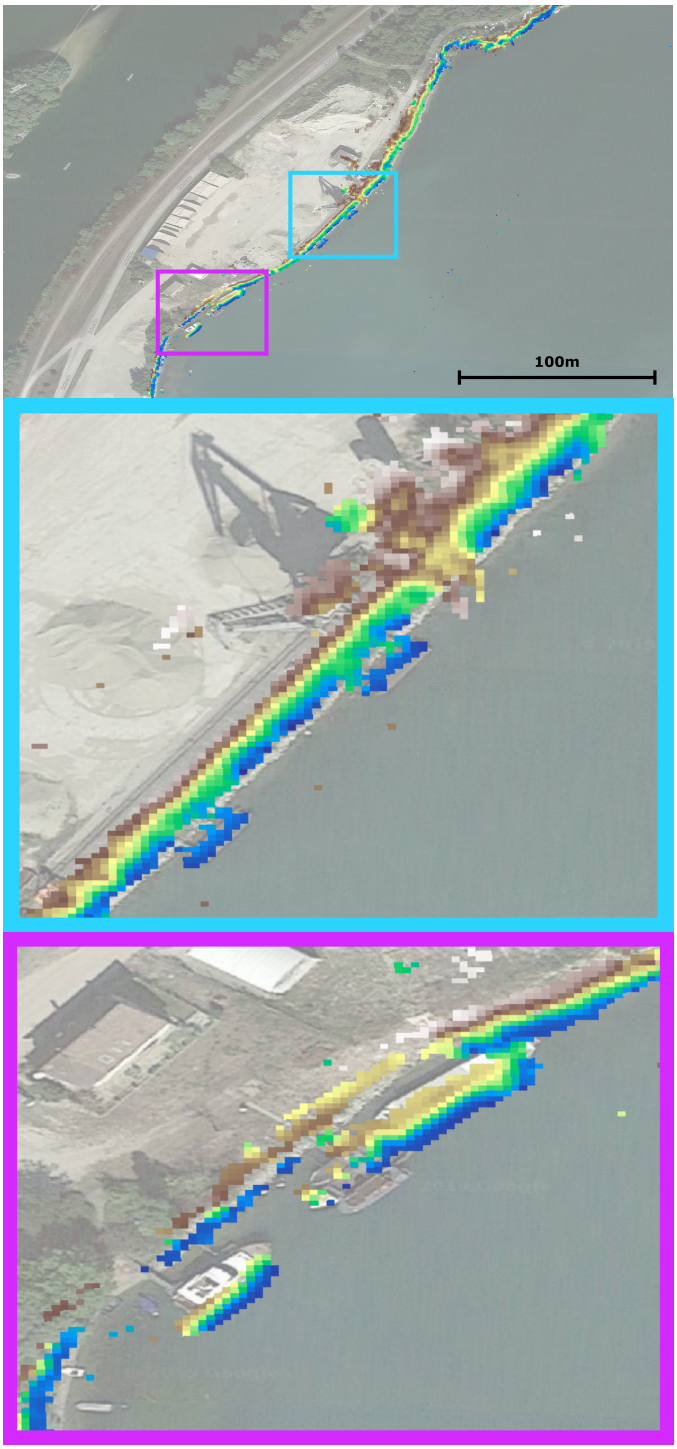


Fig. 10: Height map of a port facility generated by an autonomous drive of approximately 600 m. The height map is quantized into a resolution of 0.5 m×0.5 m. The altitude is coded by color from low (blue) to high elevation (brown). The two details are enlarged by a factor of three. Satellite base imagery is imported from Google Maps. Map data: © 2020 Google, CNES / Airbus, GeoBasis-DE/BKG, GeoContent, Maxar Technologies.

References

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[2] "ROS.org: The Robot Operating System ," 2020. [Online]. Available: https://www.ros.org

[3] A. Calce, P. M. Forooshani, A. Speers, K. Watters, T. Young, and M. R. Jenkin, "Autonomous aquatic agents." in *ICAART (1)*, 2013, pp. 372–375.

[4] A. Mancini, E. Frontoni, and P. Zingaretti, "Development of a low-cost unmanned surface vehicle for digital survey," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.

[5] S. S. Velamala, D. Patil, and X. Ming, "Development of ros-based gui for control of an autonomous surface vehicle," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2017, pp. 628–633.

[6] K. Kebkal, O. Kebkal, I. Glushko, T. T., R. Bannasch, M. Komar, and S. Yakovlev, "SONOBOT - an autonomous unmanned surface vehicle for hydrographic surveys, hydroacoustic communication and positioning in tasks of underwater acoustic surveillance and monitoring," June 2014.

[7] P. B. Andrzej Stateczny, "Universal autonomous control and management system for multipurpose unmanned surface vessel," *Polish Maritime Research*, vol. 26, no. 1(101)1(101), pp. 30–39, Jan 2019.

[8] XPRIZE Foundation, "Shell Ocean Discovery XPrize: Team Arggonauts," 2019. [Online]. Available: https://oceandiscovery.xprize.org/prizes/ocean-discovery/teams/arggonauts___fraunhofer_iosb

[9] T. Emter, C. Frese, A. Zube, and J. Petereit, "Algorithm Toolbox for Autonomous Mobile Robotic Systems," *ATZ Offhighway*, vol. 10, no. 3, pp. 48–53, 2017.

[10] R. C. Carrano, L. C. S. Magalhães, D. C. M. Saade, and C. V. N. Albuquerque, "Ieee 802.11s multihop mac: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 13, no. 1, pp. 52–67, First 2011.

[11] "B.A.T.M.A.N. (better approach to mobile ad-hoc networking)." [Online]. Available: https://www.open-mesh.org/projects/open-mesh/wiki

[12] T. Emter, A. Schirg, P. Woock, and J. Petereit, "Stochastic Cloning for Robust Fusion of Multiple Relative and Absolute Measurements," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[13] J. Petereit, T. Emter, and C. Frey, "Mobile robot motion planning in multi-resolution lattices with hybrid dimensionality," in *IFAC Intelligent Autonomous Vehicles Symposium*, 2013, pp. 158 – 163.

[14] T. I. Fossen, K. Y. Pettersen, and R. Galeazzi, "Line-of-sight path following for dubins paths with adaptive sideslip compensation of drift forces," in *IEEE Transactions on Control Systems Technology*, 2015, pp. 820–827.

[15] "Signal K." [Online]. Available: http://signalk.org

[16] Southwest Research Institute, "Mapviz: A modular ROS visualization tool for 2D data," 2020. [Online]. Available: https://swri-robotics.github.io/mapviz/

[17] "signalk-server-node." [Online]. Available: https://github.com/SignalK/signalk-server-node

[18] "OpenCPN Chart Plotter Navigation." [Online]. Available: https://opencpn.org

[19] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525