Obstacle Tracking for Unmanned Surface Vessels using 3D Point Cloud

Jon Muhovič, Rok Mandeljc, Borja Bovcon, Matej Kristan, Janez Perš

***Abstract*—We present a method for detecting and tracking waterborne obstacles from an unmanned surface vehicle (USV) for the purpose of short-term obstacle avoidance. A stereo camera system provides a point cloud of the scene in front of the vehicle. The water surface is estimated by fitting a plane to the point cloud and outlying points are further processed to find potential obstacles. We propose a new plane fitting algorithm for water surface detection that applies a fast approximate semantic segmentation to filter the point cloud and utilizes an external IMU reading to constrain the plane orientation. A novel histogram-like depth appearance model is proposed to keep track of the identity of the detected obstacles through time and to filter out false detections, which negatively impact vehicle's automatic guidance system. The improved plane fitting algorithm and the temporal verification using depth fingerprints result in notable improvement on the challenging MODD2 dataset, by significantly reducing the amount of false positive detections. The proposed method is able to run in real time on board of a small-sized USV, which was used to acquire the MODD2 dataset as well.**

***Index Terms*—unmanned surface vehicle, obstacle avoidance, 3D fingerprint, visual stereo, obstacle tracking**

I. Introduction

As the ground and aerial autonomous robots become more common, robots that can autonomously navigate the water surface (unmanned surface vehicles, USV) have also become an interesting research topic. Because the environment for such robots is dynamic, the GPS is usually not enough to ensure safe path planning. Accurate detection, classification and tracking of nearby obstacles is needed. The performance of computer vision in the realistic sea conditions is still insufficient for entirely autonomous operation [1]. Unique circumstances on the water surface (reflection, waves, mirroring), as demonstrated by Figure 1 thus require an adaptation of standard procedures and the development of new methods.

The attractiveness of computer vision methods in marine environments is manifold. First, small vessels are still piloted by humans using visual means (e.g. leisure boats, small fishing boats) which proves the usefulness of vision for marine navigation. Second, USVs may operate in environments where human vision offers significant advantages (e.g. close to the shore, in marinas and small waterways). Third, computer vision relies on relatively lightweight opto-electrical sensors (cameras), unlike radar or LIDAR, which require that possibly heavy equipment (a radome [2] or a LIDAR scanning head [3]) is placed high above water surface, destabilizing the small USV and introducing high energy demands. Due to high flexibility regarding the choice of optics and camera placement, computer vision is a very versatile and adaptable means of observing the USV's environment. Merely by selecting proper

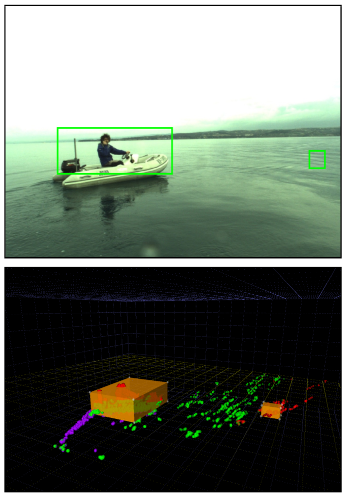

Fig. 1. Illustration of imperfections in the point cloud that cause false positives. Left: one image from the stereo pair. Right: 3D representation of the point cloud with one true detection (inflatable boat) and one false detection (on the right side of the image). Reflection on the water surface resulted in a false match and a group of 3D points appeared above the surface, prominently enough to be detected as a potential obstacle.

optical and mechanical setup (focal length and the baseline), a stereo camera system can be adapted to capture data either in the immediate vicinity of the craft or hundreds of meters away, as shown by [4].

A. Problem statement

Existing computer vision based obstacle detection approaches focus on two main modalities:

- **Pixel data.** Monocular vision [1], [5]–[7] relies on a gray-scale or RGB image sequence from a single camera. The key assumption in this approach is that the obstacles *visually differ from the background*, and that that they can be *segmented from the background* using only the pixel data, exploiting the visual differences. An example of this approach is the work on semantic image segmentation by Kristan et al. [1].
- **Depth data.** The majority of obstacle detection techniques for outdoor environment rely on depth measurements. LIDAR sensors [3], [8] are the most robust method of obtaining depth data and widely used in experimental self-driving vehicles. However, the cost of sensors that are able to generate reasonably dense data (e.g. hundreds of horizontal lines at the time) is prohibitive for small and low-cost USVs. Time-of-flight cameras are robust, but their range is usually too small (generally under

10 m, c.f. [9]) and the outdoor performance in all weather conditions is not assured. Finally, depth data can be obtained from the calibrated pair of relatively low-cost cameras, such as in [4]. The performance of such a system is highly dependent on the presence of textured surfaces to obtain robust stereo matches.

Both the monocular and the stereo approach have their advantages and disadvantages. Monocular approaches cannot distinguish between *an image of an obstacle* and *the obstacle*. However, this becomes less important in waterborne scenarios, since the visual features of ocean surface are far easier to model than (for example) urban driving environment. However, there are still many challenges that are especially pressing in coastal waters, as shown in [1]. Those are mainly related to variation on the water surface due to variable lighting, reflections and general visual cluttering in areas such as marinas. Stereo approaches are in theory a good choice, since the dense point cloud provides all the information needed for successful obstacle avoidance. However, the depth data is never error free, and the stereo approaches usually assume the operation on open sea (no visible shore), while operation closer to the shore reveals problems both with unreliable stereo methods and consequently an increasingly difficult water plane estimation. These problems have not been sufficiently addressed by the related work and thus the possibility of using USVs close to shore (where the obstacle detections and path planning are most crucial) is diminished.

B. Task: short-term obstacle avoidance

The method, presented in this paper, represents sensing subsystem of short-term obstacle avoidance functionality of the USV, which is, in turn, part of the more complex automatic on-board navigation system. *Short-term* refers to the fact that the system is tasked with avoiding obstacles that are in its immediate path (approx 5-10 seconds away), and has to provide adjustments to the planned trajectory in real time to avoid collisions, while preserving trajectory smoothness, which enables precise guidance (e.g. stop-and-go technique is not an option, unlike in autonomous land vehicles).

For the purpose of the sensing subsystem following simplifications on obstacle avoidance are made, as shown in Figure 2: 1) Obstacles move slowly relative to the boat with predictable motion and do not react to USV's presence. 2) Obstacles are either fully visible in the area of interest or their partial visibility allows path planning, which avoids obstacles at their visible end.

Finally, it should be noted that obstacle avoidance is a continuous process. The ideal trajectory, as shown in Figure 2 is not necessarily planned from the single viewpoint, but continuously adjusted and updated as the USV travels through the obstacle field; this allows for the simplifications described above: if the obstacle is moving, it will simply affect the computed trajectory in real time, as its each new position becomes known.

Higher levels of navigation system are responsible for long-term path planning and resolving of situations that cannot be tackled by short-term obstacle avoidance (e.g. an obstacle that has no visible end in sight, adherence to international collision regulations (COLREGs), etc.).

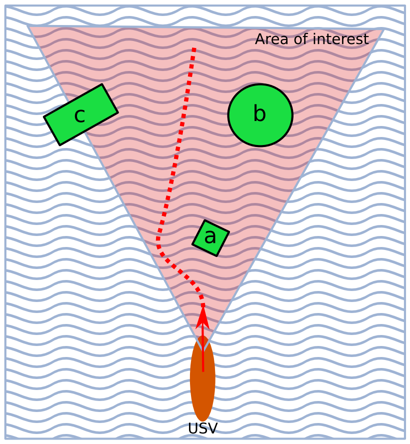

Fig. 2. Illustration (definition) of the short-term obstacle avoidance task, as used in this paper. The area of interest is shown as the shaded triangle. Initial heading of the USV is shown as the red arrow and the ideal trajectory, considering the obstacles "a", "b" and "c" is shown as the dashed line. Note that the obstacle "a" is directly in the path of the USV, therefore the heading needs to be adjusted to pass the obstacle "a" without collision. The obstacle "c" is only partially visible, but considering the area of interest and the task of avoiding it, this is irrelevant. For the sake of simplicity, this illustration assumes that all obstacles are slowly moving and passive (e.g. they will not adapt their behaviour upon detecting the USV).

The overall aim of our work is a high quality obstacle detection both from the depth stereo data and the pixel data and late-stage fusion to provide situational awareness for the USV. Figure 1 depicts a key problem we address in our work: the presence of false positives when 3D stereo camera setup is used in marine environment, which severely degrade the performance of path planning and obstacle avoiding algorithms.

II. Related work

Although water surface vehicles receive far less attention than ground and aerial robots, several works have been published that deal with automatic obstacle detection on the water surface. All approaches share several similarities, but place different emphases depending on the particular task (following dynamic obstacles, path planning etc.). This section will present several approaches.

Larson et al. [10], [11] were among the first to apply the approaches used with ground robots on a waterborne platform. Their platform uses a NASA JPL stereo system to acquire depth information. The point cloud is projected onto an obstacle grid. An additional functionality is the monocular approach that first detects the horizon and then estimates the distance to the obstacles under the horizon line by using the distance from the horizon to the obstacle. This is performed by using information from nautical charts.

Huntsberger et al. [4] present the Hammerhead system that uses two pairs of stereo cameras to attain a very large angle of view. The camera images are used to calculate a point cloud, then a plane representing the water surface is fitted into the cloud. The points above the water surface are then projected into a 2D map which is subsequently temporally and spatially filtered, the obstacles are classified and tracked. This system is by design the most similar to the one presented in this paper.

A group of researchers [5] presented their system for detection and tracking of obstacles on the water surface. Their unique approach is using saliency detection in Lab color space along with Harris corner detector and optical flow which produce regions of interest. The disparity for each region of interest is then calculated by using the disparity from the previous time step with normalized cross-correlation.

The same authors improve their approach in their next papers [12], [13] by using temporal information when estimating the depth of an obstacle and using RANSAC to calculate the water surface plane from a subsampled point cloud. The points above the plane are projected into two separate maps, the occupancy grid and the height grid. Both maps are later merged and hexahedrons (cubes) that determine the obstacles are calculated. The detected obstacles are projected into the 2D coordinate system of the left image to simplify the experimental evaluation.

An even further improvement was proposed in [14] by using tracking between neighboring frames. The detected obstacles were tracked in between neighboring frames to find consistent objects. The comparison was performed using the size, location and HSV histogram of the detections. The authors also proposed an image feature based particle filter to handle potential occlusions of tracked objects. This includes a transition model (for coordinate and scale change prediction) and an observation model (HSV and SIFT histograms). The particle filter takes over when the data association fails and stops if/when the obstacle is redetected.

Sinisterra et al. [6], [15] reversed the standard order of actions by first using HSV color segmentation to acquire regions of interest and then calculating their depth using a stereo system. An extended Kalman filter was then used to track the detected objects and estimate their velocity.

Kristan et al. [1] performed a semantic image segmentation using Markov random fields. Their visual model includes weak priors for segmenting the image into the sky, the coast and the sea. The segmentation is then iteratively improved. Regions that violate the visual model are presumed to represent obstacles on the water surface. The approach was further improved in [7] by employing IMU data to modify hyperpriors for the semantic segmentation and additional filtering and validation of the monocular obstacle detections by using both images of the stereo system. They also prepared a fully annotated stereo dataset (MODD2) and made it available for public use.

Cho et al. [16] used the FAST corner detector and the information about the horizon to detect obstacles. The detected points were clustered based on the Euclidean distance and the relative distance to the camera was calculated using the distance to the horizon. To attain a wider camera viewing angle they used a panning module.

Halterman et al. [3] have evaluated LIDAR as a method for detection of obstacles in the vicinity of the boat. They used a Velodyne HDL-64E LIDAR, providing 64 separate lasers that cover an angle of 360°. Because laser sensors typically do not return on the water surface, they could be used to detect and classify obstacles up to 100m away.

Lately, approaches have also emerged that use convolutional neural networks to detect and classify obstacles in images [17], [18].

Asvadi et al. [19] present their method for obstacle detection with 3D LIDAR mounted on a car combined with GPS/IMU data. Their approach uses piecewise RANSAC plane fitting and temporal point cloud matching to extract the static and dynamic obstacles.

An approach to visual tracking was proposed by Fefilatyev et al. in [20]. Their method performs segmentation on color images, then tracks the output regions with a Multiple-Hypothesis Tracking framework. The linear Kalman filter is used to manage each hypothesis and predict the motion of the tracked object.

The most closely related work to ours is the Hammerhead system [4], which uses stereo cameras and water plane estimation to detect obstacles. However, our proposed extensions to that basic concept allow for far better results of obstacle detection. The IMU data allows us to find the water surface plane faster and more reliably, especially in difficult conditions (either in cluttered environments close to the coast or in case of very calm water).

The depth fingerprint features used in our proposed method were previously described by our work in [21], where we presented a concept used in part of the proposed tracking pipeline.

III. Our approach

For operation in real world environment, waterborne obstacles such as buoys, other boats and swimmers have to be detected in order to compute safe path planning for the USV. An accurate estimate of the distance to the obstacle is also required. However, our practical experience indicates that the biggest problem are the false positives. They have an extremely degrading effect on the overall USV performance – they trigger unnecessary emergency stops and adversely affect path planning. The guidance of the USV in straight line becomes extremely difficult in case of false positives, due to interplay between hull dynamics and the low-level propulsion control algorithm.

Implementation-wise, a stereo camera system is used for the analysis of the boat's surroundings. It allows for calculation of the disparity, and by extension, the 3D point cloud of the area in front of the boat. The point cloud is then processed and clusters of points above the surface are identified. The schematics of our approach are shown in Figure 3. As an additional safety measure, the USV is equipped with ultrasound proximity sensor, which initiates emergency stop in case an undetected obstacle (false negative) has appeared in close proximity in front of the USV. This way, it can prevent a damage to both the USV and the obstacle, as long as we operate in the environments, where obstacles are large enough to trigger ultrasound sensor (e.g. marina). The presence of the proximity sensor is significant for the constraints of our approach: we do not need to reach zero false negative rate, but it has to be as small as possible – activation of proximity sensor is still a major event which disrupts the guidance of the boat.

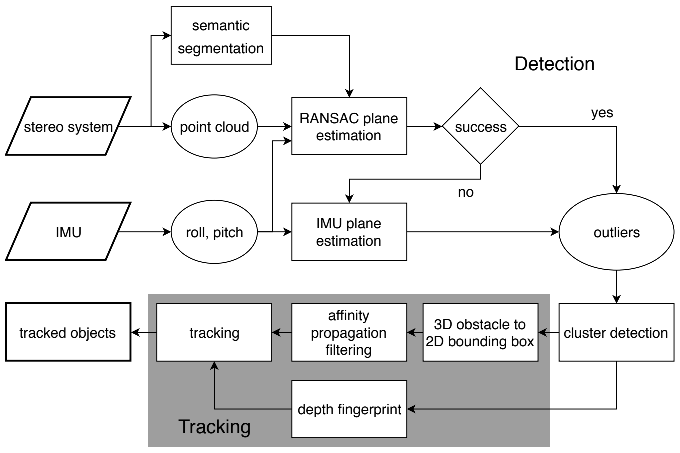

Fig. 3. The processing pipeline of our system. The information is gathered from the stereo camera system and the IMU unit. Then, obstacles are detected and projected to 2D bounding boxes. Temporal verification (tracking) is then performed to filter false detections.

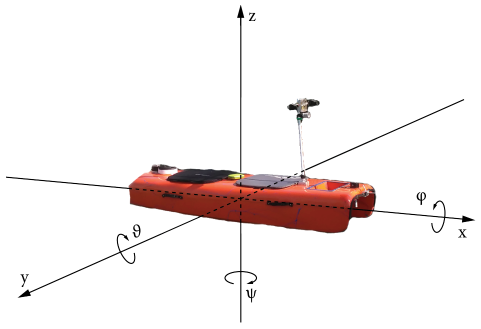

Fig. 4. Coordinate system of our USV. Angles φ, ϑ and ψ denote roll, pitch and yaw, respectively.

Our contributions are: a plane fitting algorithm for water surface detection that is robust to difficult conditions where the point cloud contains few point on the water surface. Such conditions occur when the boat is in the harbor or close to the shore or alternatively, when the water is too calm to provide the necessary texture for the stereo method to work. Additionally, we propose a novel histogram-like depth appearance feature that helps with identifying the same obstacles between neighboring frames and allows discarding non-consistent detections (false positives). Both contributions significantly improve the performance of our USV system, and are computationally inexpensive so as to enable real-time operation.

A. Obstacle detection

To estimate the water surface with a 3D plane, we use RANSAC [22]. Following is an overview of the model parameters and estimation procedure. The coordinate system and direction of angles with respect to the boat hull are shown in Figure 4.

Let Π be a 3D plane with parameters a, b, c and d. All 3D points that are part of the plane satisfy the equation $ax + by + cz + d = 0$. In each iteration of RANSAC, three random non-collinear 3D points $[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]$ are chosen from the point cloud C and used for the construction of a possible solution plane. The parameters of the plane Π are thus determined as follows:

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{p}_2 - \mathbf{p}_1 \\ \mathbf{v}_2 &= \mathbf{p}_3 - \mathbf{p}_1 \\ \mathbf{n} &= \mathbf{v}_1 \times \mathbf{v}_2 \\ d &= -\mathbf{p}_1 \mathbf{n} \\ \Pi &= n_1 x + n_2 y + n_3 z + d. \end{aligned} \tag{1}$$

First, the plane normal $\vec{\mathbf{n}}$ is calculated (its elements equal parameters a, b and c), then one of the chosen points is used to define the remaining free parameter d. The quality of a solution is determined by the fraction of all the points in the cloud C that lie on the plane (i.e. $\delta < \varepsilon$). The distance δ of a point $\mathbf{q} = [x_1, y_1, z_1]^\top$ to the plane Π is calculated as

$$\delta = \frac{ax_1 + by_1 + cz_1}{\sqrt{a^2 + b^2 + c^2}}. \tag{2}$$

The procedure of selecting and evaluating new solutions is repeated until a sufficiently good plane estimate is acquired. If a time limit is imposed (or after a specified number of iterations), the best solution up to this point is returned as the water surface estimate.

In our use case, RANSAC can encounter problems in two ways. When in harbor or looking towards the coast, the fraction of outlying points (i.e. points not lying on the water surface) might exceed the fraction of points on the water. This causes massive errors in plane estimation unless additional, complementary information is used. Another case is very calm water, which does not contain enough texture for the stereo method to work (c.f. Figure 5). This results in a very low number of points in the cloud and can make the plane estimation unreliable. In such cases, the naïve use of RANSAC can be improved by using problem specific knowledge. We describe our improvements in the following sections.

1) Using IMU data: Our first improvement is integration of externally measured pitch and roll in the plane estimation. The space of possible solutions for the plane parameters is significantly reduced by ignoring the solutions that are outside reasonable limits of the actual USV (e.g. excessive roll or

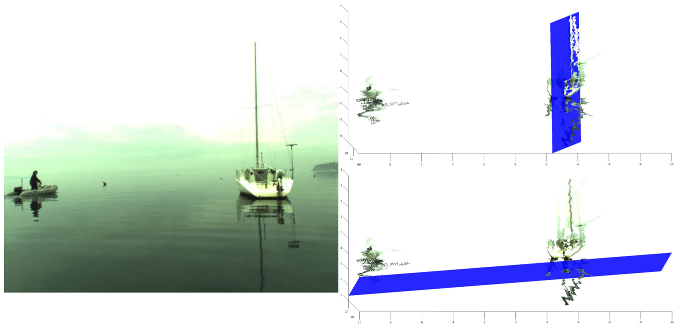
Fig. 5. Problems with plane fitting in case of few points on the water surface. Left: left camera image, top right: RANSAC plane estimation, bottom right: plane estimation with IMU information.

pitch). Additionally, if RANSAC fails to converge, a rough estimate of the water surface can be obtained using solely IMU data.

First we derive the equation for plane estimation directly from IMU measurements. This is achieved by defining two standard 3×3 rotation matrices $\mathbf{R}_x(\varphi)$ and $\mathbf{R}_y(\vartheta)$ and multiplying an upwards pointing normal vector $\mathbf{n} = [0, 0, 1]^\top$ by them as follows:

$$\mathbf{n}' = \mathbf{R}_x(\varphi)\mathbf{R}_x(\vartheta)\mathbf{n}. \tag{3}$$

The elements of $\mathbf{n}' = [n_1, n_2, n_3]^\top$ then equal parameters a, b and c of the plane $\Pi \equiv ax + by + cz + d = 0$. The remaining parameter d is set to the camera's height above the IMU sensor (estimated during calibration).

If we wish to use IMU data to limit the space of possible solutions checked by RANSAC, we also need to derive the inverse formulation (i.e. how to extract roll and pitch angles from a plane equation defined by three points). When we perform multiplication of the rotation matrices ($\mathbf{R}_x(\varphi)$ and $\mathbf{R}_y(\vartheta)$) with the normal vector we get the following expression:

$$\mathbf{n}' = \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\vartheta & 0 & \cos\vartheta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\vartheta \\ -\sin\varphi\cos\vartheta \\ \cos\varphi cos\vartheta \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{4}$$

The roll angle φ can then be derived as follows:

$$\frac{b}{c} = \frac{-\sin\varphi\cos\vartheta}{\cos\varphi cos\vartheta} = \frac{-\sin\varphi}{\cos\varphi} = -\tan\varphi \rightarrow \varphi = \tan^{-1}\frac{-b}{c}. \tag{5}$$

Pitch angle ϑ can be derived in a similar manner:

$$\frac{a}{c} = \frac{\sin\vartheta}{\cos\varphi\cos\vartheta} = \frac{1}{\cos\varphi}\tan\vartheta \rightarrow \vartheta = \tan^{-1}\frac{a\cos\varphi}{c}. \tag{6}$$

This additional information is integrated into RANSAC after the random sampling and before the inlier calculation. By using the IMU data to obtain a rough estimate of the water surface, the plane candidates whose roll or pitch (φ and ϑ respectively) are beyond reasonable limits can be discarded immediately and no further inlier checking is required. If no adequate (i.e. one with a high enough fraction of inliers) solution is obtained by RANSAC, the plane can be estimated directly from the IMU readings.

2) Using semantic segmentation: RANSAC-based plane estimation becomes highly unstable in environments that do not contain enough points lying on the water surface. This can occur either in very cluttered environments such as the harbor (c.f. left column of Figure 6) or if the sea is very calm (c.f. upper right of Figure 5). Either a suboptimal solution is found (if we impose hard limit on the execution time), or the estimation would take too long for real-time operation. To address this issue, we apply a fast approximate semantic segmentation [7] to estimate the sea edge and detect obstacles on the water. The result of the segmentation is a mask that denotes the pixels belonging to the water surface. We use the left image of the stereo system as input for the segmentation algorithm. The resulting information is used to limit the point cloud used for sea surface estimation. Let $\mathbf{B}$ be the water region mask where the pixel $\mathbf{B}_i$ has a value of 1 if it belongs to the water and value 0 otherwise. Let the available points at a certain time be represented by an array $\mathbf{C} = \{c_i\}_{i=1:M}$ where $c_i = [x_i, y_i, z_i, u_i, v_i]$ is a vector of the 3D point coordinates and corresponding image coordinates in the left image. We define a subset of the point cloud $\tilde{\mathbf{C}}$ by

$$c_i \in \tilde{\mathbf{C}} \Leftrightarrow \mathbf{B}_{u_i,v_i} = 1; i = 1 : M. \tag{7}$$

The new point cloud only includes points that likely lie on the water surface. Since $\tilde{\mathbf{C}}$ contains significantly less outliers, the convergence speed of the RANSAC algorithm is substantially improved (c.f. V-B). This also improves the quality of the plane estimation and by extension, the obstacle detection. In the remainder of the paper, the standard RANSAC plane estimation (e.g. as used in [13]) and RANSAC plane estimation using segmentation information will be referred to as E_R and E_S respectively. Estimating the plane directly from IMU measurements will be denoted with E_I. The effect of segmentation on the contents of the point cloud is depicted in Fig. 6.

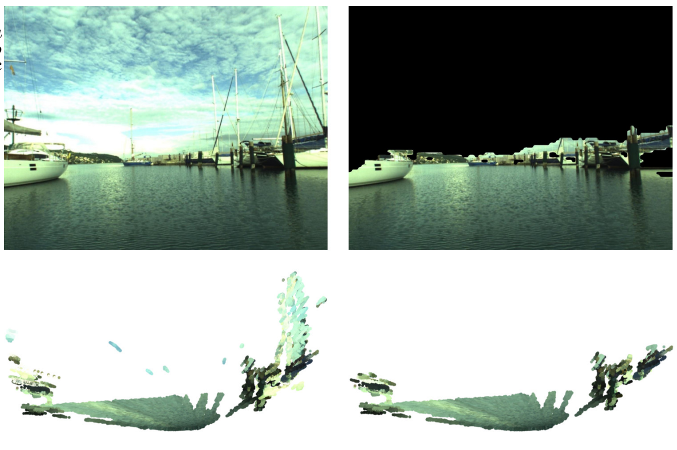
Fig. 6. Illustration of the use of semantic image segmentation. First row: left camera image without and with the segmentation mask (black pixels are non-water). Second row: respective point clouds. A reduction in sea surface outliers and a reduction in overall number of points can be observed.

3) Obstacle detection: After a water surface estimate is obtained, the points for which $\delta > \varepsilon$ holds are considered to lie above the water surface, and are denoted as $\mathbf{C_{above}}$.

$$c_i \in \mathbf{C_{above}} \Leftrightarrow \delta_i > \varepsilon; i = 1:M. \tag{8}$$

All other points are discarded, as they are certainly not part of any obstacles. Parameter ε determines the minimal height of obstacles we are trying to detect, and, conversely, the maximum height of the waves our system is able to ignore. Note that the used values for all parameters are specified in Table I. Then, the algorithm proceeds as follows.

- A three-dimensional histogram $\mathbf{H^p}$ of 3D point positions in front of the USV is defined. It covers the range from x_{min} to x_{max}, y_{min} to y_{max} and z_{min} to z_{max}. Histogram bins $h^p_{x,y,z}$ are cubes with dimensions $\Delta x = \Delta y = \Delta z$.
- The set of points $\mathbf{C_{above}}$ is scanned and histogram H^p is filled, based on point positions. Note that from this point on, H^p actually represents point density and can be processed as any other discrete three-dimensional image.
- The 3D convolution operation with $3 \times 3 \times 3$ box kernel (containing all ones) is run on $\mathbf{H_p}$ to smooth the noise and mitigate the effect of spurious detections.
- A threshold T_H is applied to histogram $\mathbf{H^p}$, generating binary occupancy map $\mathbf{H^b}$ using the following rule:

$$h^b_{x,y,z} = \begin{cases} 1 & h^p_{x,y,z} > T_H \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

  Most of the cells of $\mathbf{H^b}$ are set to 0; those who are not, represent possible fragments of the objects to be detected.
- The connected components ([23]) are then extracted, representing the obstacle candidates. Their main axes are determined using PCA.

The rotation of an obstacle is available to the USV's low level control algorithm, however, for the purpose of evaluation, we define a single obstacle as $\mathbf{o} = [\mathbf{o}_c, \mathbf{o}_e]$, where $\mathbf{o}_c = [c_x, c_y, c_z]^\top$ is the center of the obstacle and $\mathbf{o}_e = [e_x, e_y, e_z]^\top$ are the width, height and depth of the obstacle.

B. Depth fingerprint

In order to establish consistent positions of detected obstacles, we need to be able to connect those that represent the same physical object. To this end, we propose a histogram-like depth feature descriptor (that we call a *depth fingerprint*) for appearance verification. It employs the depth information gained from the point cloud to extract the depth distribution of points that comprise an obstacle. The approach is bears a slight resemblance to [24], but it does not use clustering and does not require imposing a limit on the number of obstacles.

The purpose of the depth fingerprint is twofold: it serves as a descriptive feature that allows us to confidently make the connections between separate detections of the same obstacle in different frames. Secondly, false detections on water may appear in roughly the same location, but their depth fingerprint is typically inconsistent, which makes it easy to detect and discard them.

To enable the extraction of depth information, the obstacle detector has to be modified to store the entire set of 3D points $\mathbf{o}_p$ that make up each obstacle instead of just the

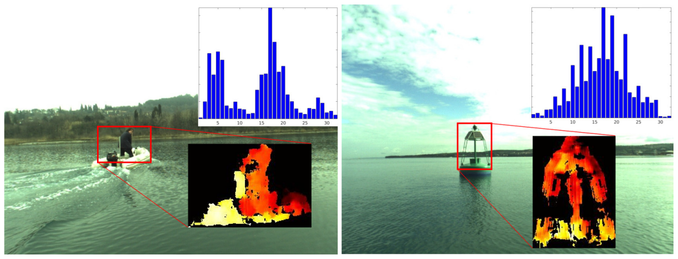
Fig. 7. Illustration of depth fingerprinting. Left: inflatable boat with the boat operator visible, right: a buoy at the entrance to the Port of Koper. Inlaid are the closeups of the cropped point cloud of the detected obstacles. Reddish hues represent distant points, while the yellow ones represent the closer ones. Depth fingerprints - the relative object depth histograms of the obstacles are shown in the upper right corners.

position and extent of the obstacle as described in III-A3. Since we focus only on the local depth distribution, the mean of the points positions $\mu(\mathbf{o}_p)$ is subtracted from each point. A 1D histogram is then calculated from the depth of the points comprising each obstacle (x axis in the coordinate system of the USV). Since the size of the depth fingerprint is independent of the obstacle size, comparisons between objects of different sizes are possible. Examples of detected obstacles and their respective depth fingerprints are shown in Figure 7. As the features are only used to compare detected objects in consecutive frames, long-term stability is not required. The same holds for scale invariance, since the size is not expected to vary significantly between neighboring frames. In fact, the objects with large scale variation in such short time interval should not be matched – it is very likely the case of two different but similarly shaped objects being observed.

C. Temporal consistency verification

The point cloud recovered from the stereo system is locally sparse. Regions that are connected in the real world might break into smaller obstacles during the obstacle detection. On many occasions, false stereo matches occur due to repeated vertical structures such as multiple boat masts in the marina, as seen in Figure 11. Finally, the surface of the water may cause false positives as well (c.f. Figure 1). These are pressing problems, since they distort the true location of an obstacle and deteriorate path planning. To filter out these inconsistencies, temporal consistency verification must be included, to track only objects that are consistently appearing in multiple sequential frames.

We propose a tracking-by-detection algorithm that connects consistently detected obstacles between sequential frames. To track the detections between sequential frames, we need to connect ones representing the same object as well as detect non-matched objects and newly seen ones. Let D_i and D_j be two sets of detected obstacles. The Hungarian method [25] is used to calculate the most probable matches between the sets. The method requires a matrix of distances between the elements of both sets. The distances are calculated using a cost function that returns the cost of assigning one object to another.

The tracking algorithm uses two global lists of obstacles. The list T_l contains currently tracked obstacles and P_l contains consistently appearing but not yet confirmed obstacles. For each available frame F_i the detected obstacles are represented by the list D_i. To achieve tracking, consistently appearing obstacles must be recognized and connected between neighboring frames and false detections must be ignored. The proposed algorithm is not limited by a fixed number of obstacles but tracks all detections that appear consistently enough.

When an obstacle is first detected, it is added to P_l. If it is continuously assigned an existing object in P_l for the next N frames, it is moved to T_l. If not, it is removed from P_l. If an obstacle in T_l is not detected in frame F_i, it is not immediately discarded. Instead, its lost counter is incremented and only if it remains unassigned for M sequential frames it is considered lost and removed from T_l. This prevents tracked objects from being prematurely discarded due to short periods of missing detections (due to occlusion or stereo method errors). Elements of D_i that were not assigned to either an element from T_l or an element from P_l are considered new potential obstacles and are added to P_l.

For the algorithm to return relevant results, a suitable cost function for the assignment must be chosen. A naïve approach would be to use the Euclidean distance and try to match detections that are close together. However, this discards information about the shape of the obstacle and can cause problems when a false detection (c.f. Figure 1) appears consistently.

To determine whether two detections are of the same object, we use a combination of Euclidean distance (L^2 norm) and depth fingerprint similarity (calculated as Hellinger distance). The probability of a new detection $\mathbf{x}$ being equal to a known object detection $\mathbf{o}$ is defined as follows:

$$p(\mathbf{x}|\mathbf{o}) = \mathrm{e}^{-\lambda||\mathbf{x}_c-\mathbf{o}_c||_2}\mathrm{e}^{-H(\mathbf{x}_f,\mathbf{o}_f)}, \tag{10}$$

where first factor denotes the probability of spatial consistency, and the second one the probability of visual consistency. λ is the parameter of the exponential distribution and affects the relative influence of the both factors. The Euclidean norm is calculated from the centers of mass $\mathbf{x}_c$ and $\mathbf{o}_c$ and the depth fingerprint similarity $H(\mathbf{x}, \mathbf{o})$ is calculated from the obstacles' respective depth fingerprints $\mathbf{x}_f$ and $\mathbf{o}_f$. The Hellinger distance $H(\mathbf{p}, \mathbf{q})$ is defined as:

$$H(\mathbf{p}, \mathbf{q}) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{\mathbf{p}_i} - \sqrt{\mathbf{q}_i})^2}, \tag{11}$$

where p and q are depth fingerprints and k is the number of histogram bins.

Finally, the logarithm of Eq. (10), $\log(p(\mathbf{x}|\mathbf{o}))$ is used as a cost function for identity assignment. This way, our approach includes both geometrical (spatial) constraints (e.g. object in the next frame should be close to its previously detected position) and quasi-visual similarity constraint (e.g. object in the next frame should be similar to its previous detection).

D. Obstacle defragmentation

The obstacle detection method sometimes fragments large obstacles (large ships, sailboats, piers) into many 3D obstacles. This can happen due to local sparsity of the point cloud caused by the lighting conditions on the water surface, or by repeatable structures in the images. When projected back into the 2D image, these detections can overlap (because of the projection to 2D image plane) and cause errors. We used unsupervised clustering to address such problems, specifically the affinity propagation algorithm [26]. The algorithm clusters a set of data points based on a distance metric. It detects representative elements called exemplars and clusters the data points based on the distance to the nearest exemplar. The number of exemplars is determined automatically. In our case, data points are detection bounding boxes and the distance metric is the negative squared distance between bounding box centers. After the exemplars are detected, the bounding box with the largest area in each cluster is kept, others are discarded. This process immediately follows the detection and its results are fed into the tracking phase. The result of AP filtering is shown in Figure 8.

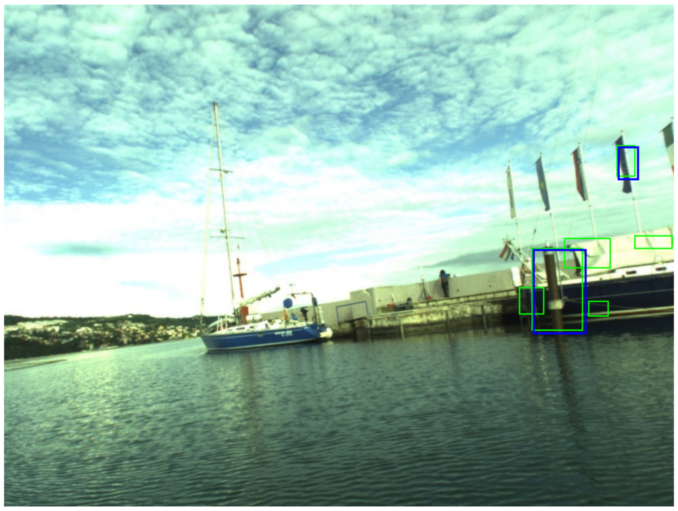
Fig. 8. Obstacle detections before and after affinity propagation clustering. Initial detections are shown in green, while the clustered detections are represented in blue.

IV. Implementation details

This chapter contains details regarding the hardware and software attributes of our USV platform. Section IV-A lists the sensors present on the USV platform and their properties. In Section IV-B we provide information on the software implementation of our system, while Section IV-C details the concrete parameters values used in our method.

A. Hardware and image acquisition

Our USV (Figure 9) has two cameras of the type Vrmagic VRmS-14, that are affixed to the main mast, approximately $70\,\mathrm{cm}$ above the water surface. The distance between the cameras is $35\,\mathrm{cm}$ and their resolution 1280x960 pixels. The cameras use $3.5\,\mathrm{mm}$ lenses. A fast bus connects the cameras with the control unit that ensures the pixel synchronization of the cameras. USB 2.0 is used to transfer the image data to the CPU (Intel i7-4770), which allows the capture of 10

TABLE I
PARAMETERS OF OBSTACLE DETECTION PHASE.

Parameter	Value
ε	$0.2\,\mathrm{m}$
x_{min}	$1\,\mathrm{m}$
x_{max}	$19\,\mathrm{m}$
y_{min}	$-10\,\mathrm{m}$
y_{max}	$10\,\mathrm{m}$
z_{min}	$-2\,\mathrm{m}$
z_{max}	$2\,\mathrm{m}$
$\Delta x = \Delta y = \Delta z$	$0.25\,\mathrm{m}$
T_H	20

images per second. These parameters allow a high-quality point cloud acquisition up to $20\,\mathrm{m}$. The cameras are calibrated using the OpenCV camera calibration toolbox and a target pattern with asymmetrical circles. A target of size 1m is required to calibrate the stereo system geometry. Our platform is additionally equipped with a high-precision GPS receiver using real-time-kinematics (RTK). This way we are able to accurately estimate the boat position down to $10\,\mathrm{cm}$. An IMU unit is also used to provide the roll, pitch and yaw information of the USV hull.

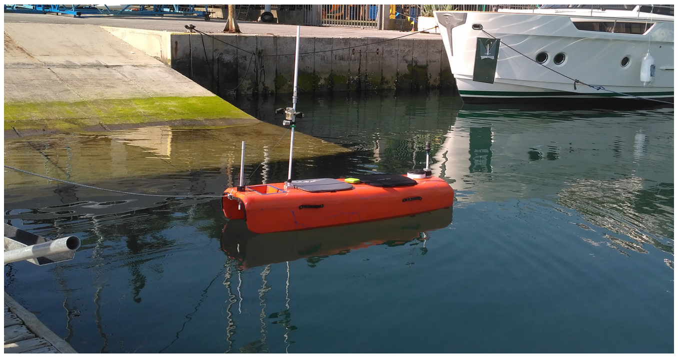
Fig. 9. Our USV system in the Koper marina.

B. Implementation

The system is implemented in C++ using the OpenCV library. The point cloud calculation is performed using the block matching algorithm. In our case, block matching is an exceptional tradeoff between quality and execution speed. After tuning, it provides good results for highly differing visual conditions encountered on the water surface. Other authors have also preferred block matching to other stereo methods [13], [27]. A memory buffer is used to reduce the amount of data copying and enable real-time performance on a desktop-class CPU (Intel i7-4770), which is used in our USV.

C. Parameters

Parameters of our method were established heuristically, by running the algorithms on real world data, acquired through several years and many vehicle missions. The parameters of the obstacle detection phase are shown in Table I.

When calculating the assignment cost, we ignored pairs of detections more than $1\,\mathrm{m}$ apart, as well as those whose depth histogram difference scored higher than 0.2 (Hellinger distance defines lower values as more similar). The buffer size for promoting potential objects to tracked objects and the number of frames we wait before declaring a tracked object lost also needed to be defined. For the purpose of our experiments, at the frame rate of 10 frames per second, both values were set to 5. This means that a detection has to be detected five times before we start tracking it and a tracked object is allowed to be unassigned for a maximum of five frames before it is removed.

V. EVALUATION

The evaluation was performed in the coordinate system of the images, which allows comparison with methods that do not use a calibrated system of cameras. The vertices of the cuboids that represent obstacles are projected back into the image and the obstacle is represented by a bounding box. Ground truth annotations were pre-processed to conform to our system's capabilities. Namely, only annotations that are present in both stereo images were considered as that allows the disparity computation. Normalized cross-correlation on the image patches defined by the annotated bounding boxes was used to estimate the obstacle depth. Since the stereo images are rectified, the displacement of an obstacle on the x axis represents the disparity and knowing the stereo system baseline allows us to calculate the approximate depth of an obstacle. Then, annotations of obstacles further away than the range of our system (approx. $19\,\mathrm{m}$) were ignored for the purpose of evaluation. Because the point cloud data is unavailable past a certain distance from the USV, they cannot be detected. For the evaluation we followed a protocol similar to the one described in [1]. A minimum overlap between the ground truth and our reported detection also has to be defined. We use a value of 0.3. The overlap is calculated as the intersection over union. We report the following metrics that illustrate the performance of our approach: the numbers of true positives, false positives and false negatives, the F-score and the average number of false positives per sequence frame (aFP). We prepared ten different versions of the method for evaluating the performance of our proposed improvements.

- v_1: E_I
- v_2: E_R
- v_3: E_S
- v_4: E_I + tracking with Euclidean distance
- v_5: E_I + tracking with depth fingerprints
- v_6: E_R + tracking with Euclidean distance
- v_7: E_S + tracking with Euclidean distance
- v_8: E_R + tracking with depth fingerprints
- v_9: E_S + tracking with depth fingerprints
- v_{10}: $v9$ + affinity propagation clustering

We performed all experiments by pre-calculating plane estimation and obstacle detection with both versions of RANSAC, then additionally performing the tracking of detected obstacles. The plane estimations for different approaches were thus identical (i.e. all methods that rely on E_R have the same raw obstacle detections).

A. Dataset

The MODD2 dataset (Marine Obstacle Detection Dataset v2) was used for the evaluation. This dataset was recorded on

TABLE II
PERFORMANCE OF THE DIFFERENT STAGES OF THE PROPOSED METHOD ON THE MODD2 DATASET.

	obstacles					no obstacles	
method	TP	FP	FN	F-score	aFP	FP	aFP
v_1	1234	9358	524	0.380	1.025	5287	1.493
v_2	1270	5455	488	0.544	0.490	250	0.066
v_3	**1366**	5504	392	0.567	0.495	262	0.070
v_4	1196	4304	562	0.480	0.463	2259	0.609
v_5	1161	1200	597	0.635	0.128	492	0.089
v_6	1234	2414	505	0.642	0.234	91	0.023
v_7	1342	2285	**397**	0.671	0.220	94	0.023
v_8	1238	986	501	0.707	0.106	35	0.010
v_9	1326	798	413	0.725	0.096	31	0.007
v_{10}	1292	**477**	447	**0.744**	**0.053**	**31**	**0.007**

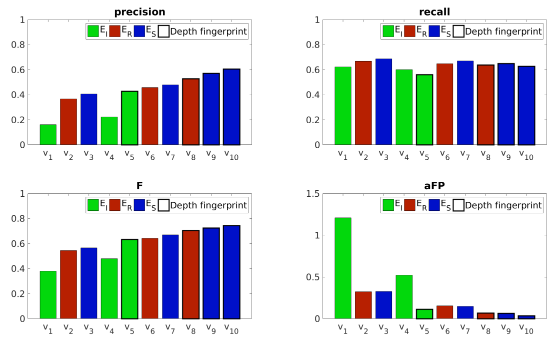

Fig. 10. Graph of results for different versions of our method. Outlined bars indicate depth fingerprint was used in that case.

our platform. It consists of 28 sequences that include images from both cameras of the stereo system as well as GPS and IMU data. Obstacles on the water surface and points that define the horizon have been manually annotated for all images. A larger part of the proposed algorithm was developed before the establishment of the dataset so it represents an unrelated test set. The dataset was released for public use as part of [7][1]. For the evaluation of our method the dataset was split into two subsets, based on the presence of detectable objects (i.e. annotations of objects within the range of our stereo system - 20m).

B. Results

Table II shows the results over all sequences for both subsets of our dataset. For the sequences with no detectable obstacles we only report the number of false positives and the average of false positives as there cannot be any true positives or false negatives. For the subset containing obstacles we report true positives, false positives, false negatives, F-score and the average number of false positives per frame. We can observe a large drop in the average number of false positives when we use the semantic segmentation and depth fingerprints. This is consistent with our goal to significantly reduce the number of false positives, as suggested by the USV operators. A simultaneous increase in the F-score can also be observed. Because of the improved sea surface estimation, the average number of false negatives (i.e. annotated obstacles that were not detected/tracked) is also somewhat reduced. The improvement in scores stems from two factors: the improved sea surface estimation provides better obstacle detections and the use of depth fingerprints improves false positive filtering. Finally, the affinity propagation clustering additionally filters out fragments of large obstacles that could normally show up as false positives but are in fact not. The clear effect of this can be seen in obstacles such as the sailboat in Figure 8.

Limiting the point cloud solely to points with a high probability of lying on the water surface has a strong effect on the speed of convergence of the RANSAC algorithm. Given the reduced number of outliers, a sufficiently good solution is found approximately twice as fast than the previous approach using the whole point cloud. The numbers are shown in Table III. The first column shows the average number of trials in the RANSAC algorithm while the second column shows the average time needed (over all frames of the dataset).

[1]The dataset is available for public download at http://www.vicos.si/User:Borjab

TABLE III
RANSAC CONVERGENCE SPEED.

	$\mu(N)$	$\mu(t)$ [s]
E_R	295	0.5040
E_S	**125**	**0.2180**

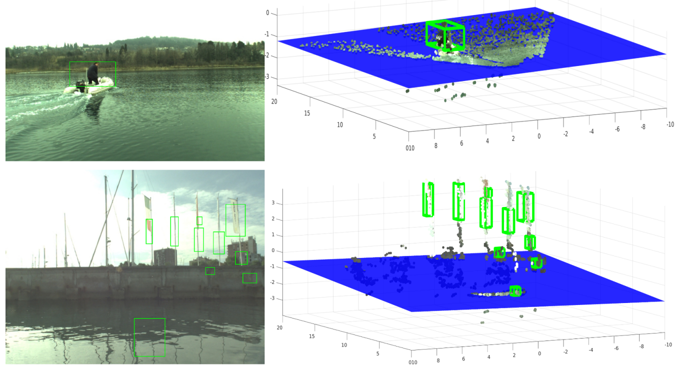

Fig. 11. Example of properly working obstacle tracking (top), and the failure in the presence of large objects (e.g. piers, large ships, etc. The presented approach is unable to properly detect the obstacles that are larger than the stereo system's field of view. Instead, they become fragmented.

C. Error analysis

The remaining number of false positives and false negatives warrants some analysis into what could be done to remedy them in the future. Note that our main goal was to reduce the number of false positives as they significantly impact the performance of our system. This is often overlooked metric in autonomous vehicles. In autonomous driving it affects the comfort of car occupants, but in autonomous vessels, it affects the ability of the vessel to operate, as guidance systems have difficulty coping with effects of large number of emergency stops combined with hull hydro dynamics – the vessel becomes difficult to guide accurately.

It is worth noting, that the proposed method causes the dramatic fall of false positives as shown in Table II, which may be the difference between the USV that is still able of autonomous navigation (albeit with many unnecessary stops) and the one that cannot navigate at all.

1) False positives: Note that in videos without obstacles, the number of false positives per frame is radically smaller than in videos with true obstacles. There are two main contributing factors to this effect. First is the absence of annotations of coastal features (large piers for example). Although the USV path during dataset acquisition was not close to these obstacles, and the observation area in front of the USV is limited, they still may influence the result – the false stereo matches on the repeatable features in marina result in false stereo detection that may appear much closer than they are in reality. These false detections could be perhaps handled by inexpensive LIDAR (with small number of beams, insufficient to reconstruct the scene, but could be used for verification). The second factor is plane surface estimation, which is dependent on a large enough number of points lying on the water surface. The method can thus have problems with either a very cluttered surroundings (e.g. a harbor) or very calm water (the water which is visually uniform). Fallback to IMU reduces the number of these problems, but it is still possible to find situations (early morning mist for example) where the inability to find enough corresponding points on the surface causes frequent invocation of IMU-based plane estimation, which is less accurate (as shown), and therefore causes false positives by mis-detection of some points on the water as obstacles. The partial remedy for that is the detection of breakdown point (e.g. by evaluation the number of pixels that have any stereo correspondence calculated) and subsequent self-declaration of the USV that it is unable to navigate safely in current conditions.

D. False negatives

Our method reduces the number of false positives by imposing initial delay in confirming that the clump of stereo detections is indeed an obstacle. This, on the other hand produces several frames of 'missed' obstacles. While these delays show up as false negatives, they do not influence safety (recall that guidance of the USV depends on whole sequence of observations, not only single frame).

Another reason for false negatives is also the discrepancy between the size of the annotated ground truth and detections our system produces. For instance, our system can sometimes only detect a smaller part of the annotated obstacle and consequently the IOU is too low, which results in a false negative, even that the obstacle is detected, but not accurately enough. This could be dealt with lower IOU in evaluation, and consequently by large safety margins in guiding the USV around the obstacles (but then, some narrow passages could be impossible to solve).

Since the depth of annotations of obstacles is calculated instead of measured, this can sometimes (due to minute errors in annotations) lead to incorrect depth estimations. The ground truth thus may includes obstacles that (due to fixed stereo baseline) cannot be detected by our system.

There are also a few of other techniques that could further improve the results. Using monocular approach, one could pre-train the convolutional neural network (CNN) to detect the most frequent types of marine obstacles (boats, ships, sailboars, standard shape buoys, etc.). Our future work will also deal with fusing these approaches with the proposed stereo method.

VI. Conclusion

We presented a method for detecting obstacles on the water surface by using stereo data and an algorithm that allows tracking of consistently detected obstacles. Additionally, we employed the results of a semantic scene segmentation method to limit the point cloud used for sea surface estimation. By reducing the number of outlying points, the convergence speed of the algorithm is increased by about a factor of 2. The quality of the sea surface estimation was thus also improved. We also developed a feature descriptor based on depth data which is used for matching detections of waterborne obstacles. Comparison of these features improved the quality of the temporal tracking of obstacles. We evaluated the method on the MODD2 dataset and shown the quantitative improvements caused by our proposed approach. We showed that our approach significantly improves the overall quality of tracking the obstacles, mainly by filtering out false positives while not significantly impacting the number of missed obstacles. In further work the problem of detecting large objects (e.g. coast and piers) will be addressed. Currently these obstacles cannot be addressed by the short-term obstacle avoidance alone, as they demand high level reasoning, and, possibly, additional information (e.g. map). They also violate our assumption of the water surface as the most dominant part of the point cloud. This problem is shown in Figure 11 and illustrates the limits of the presented approach.

Acknowledgements

This work has been supported by the Slovenian Research Agency grants J2-8175 and P2-0095 and by the GOSTOP programme, contract no C3330-16-529000, co-financed by Slovenia and the EU under ERDF. Visual recordings and other sensor data have been acquired onboard of the USV owned and operated by Harpha Sea, d.o.o, Koper and Robotina d.o.o, Koper.

Appendix

In Table IV we present separate results for each sequence of the MODD2 dataset.

References

[1] M. Kristan, V. S. Kenk, S. Kovačič, and J. Perš, "Fast image-based obstacle detection from unmanned surface vehicles," IEEE transactions on cybernetics, vol. 46, no. 3, pp. 641–654, 2016.

[2] C. Almeida, T. Franco, H. Ferreira, A. Martins, R. Santos, J. M. Almeida, J. Carvalho, and E. Silva, "Radar based collision detection developments on usv roaz ii," in Oceans 2009-Europe. IEEE, 2009, pp. 1–6.

TABLE IV
PER-SEQUENCE RESULTS FOR DIFFERENT STAGES OF THE PROPOSED METHOD.

		v_6		v_7		v_8		v_9		v_{10}	
#	obstacles	F	aFP	F	aFP	F	aFP	F	aFP	F	aFP
1	n	/	0.062	/	0.071	/	**0.013**	/	0.027	/	0.027
2	y	0.765	0.130	0.785	0.121	0.830	0.102	0.823	0.100	**0.886**	**0.062**
3	n	/	0	/	0	/	0	/	0	/	**0**
4	y	0.931	0.025	**0.939**	0.017	0.928	0.017	0.928	0.017	0.928	**0.017**
5	n	/	0	/	0	/	0	/	0	/	**0**
6	y	0.519	0.024	**0.550**	0.022	0.526	0.019	0.400	0.019	0.400	**0.019**
7	y	0.229	0.283	0.468	0.255	0.240	0.255	**0.514**	0.150	0.508	**0.081**
8	n	/	0	/	0	/	0	/	0	/	**0**
9	n	/	0	/	0	/	0	/	0	/	**0**
10	n	/	0	/	0	/	0	/	0	/	**0**
11	n	/	0	/	0	/	0	/	0	/	**0**
12	n	/	0	/	0	/	0	/	0	/	**0**
13	n	/	0	/	0	/	0	/	0	/	**0**
14	y	0.906	0.034	0.906	0.034	0.937	0	0.937	0	**0.937**	**0**
15	y	0.835	0.065	0.868	0.038	0.856	0.053	0.879	0.031	**0.879**	**0.031**
16	n	/	0.163	/	0.200	/	0.032	/	0.048	/	**0.011**
17	n	/	0.113	/	0.109	/	0.064	/	0.051	/	**0.051**
18	y	0.870	0.080	0.883	0.066	0.922	0.030	0.954	0.003	**0.954**	**0.003**
19	y	0.059	1.152	0.061	1.121	0.099	0.618	**0.102**	0.593	0.100	**0.351**
20	y	0.904	0.029	0.907	0.030	0.924	0.019	0.927	0.020	**0.933**	**0.014**
21	y	0.905	0.029	0.906	0.029	0.929	0.021	0.929	0.021	**0.929**	**0.021**
22	y	0.712	0.123	0.760	0.115	0.763	0.079	0.807	0.076	**0.827**	**0.060**
23	y	0.798	0.062	0.815	0.059	0.861	0.032	0.864	0.036	**0.927**	**0.009**
24	n	/	0.076	/	0.072	/	0.027	/	0	/	**0**
25	y	0.822	0.009	**0.822**	**0.009**	0.716	0.009	0.743	0.012	0.743	0.012
26	y	0.220	0.674	0.221	0.668	0.800	0.028	**0.800**	0.028	0.780	**0.017**
27	y	0.155	0.862	0.176	0.729	0.273	0.377	0.270	0.385	**0.431**	**0.138**
28	n	/	0.335	/	0.359	/	0.151	/	0.155	/	**0.055**

[3] R. Halterman and M. Bruch, "Velodyne hdl-64e lidar for unmanned surface vehicle obstacle detection," in SPIE Defense, Security, and Sensing. International Society for Optics and Photonics, 2010, pp. 76 920D–76 920D.

[4] T. Huntsberger, H. Aghazarian, A. Howard, and D. C. Trotz, "Stereo vision–based navigation for autonomous surface vessels," Journal of Field Robotics, vol. 28, no. 1, pp. 3–18, 2011.

[5] H. Wang, Z. Wei, S. Wang, C. S. Ow, K. T. Ho, and B. Feng, "A vision-based obstacle detection system for unmanned surface vehicle," in Robotics, Automation and Mechatronics (RAM), 2011 IEEE Conference on. IEEE, 2011, pp. 364–369.

[6] A. J. Sinisterra, M. R. Dhanak, and K. von Ellenrieder, "Stereo vision-based target tracking system for an usv," in Oceans-St. John's, 2014. IEEE, 2014, pp. 1–7.

[7] B. Bovcon, R. Mandeljc, J. Perš, and M. Kristan, "Stereo obstacle detection for unmanned surface vehicles by imu-assisted semantic segmentation," arXiv preprint arXiv:1802.07956, 2018.

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.

[9] D. Piatti and F. Rinaudo, "Sr-4000 and camcube3. 0 time of flight (tof) cameras: Tests and comparison," Remote Sensing, vol. 4, no. 4, pp. 1069–1089, 2012.

[10] J. Larson, M. Bruch, and J. Ebken, "Autonomous navigation and obstacle avoidance for unmanned surface vehicles," in Defense and security symposium. International Society for Optics and Photonics, 2006, pp. 623 007–623 007.

[11] J. Larson, M. Bruch, R. Halterman, J. Rogers, and R. Webster, "Advances in autonomous obstacle avoidance for unmanned surface vehicles," DTIC Document, Tech. Rep., 2007.

[12] H. Wang, Z. Wei, C. S. Ow, K. T. Ho, B. Feng, and J. Huang, "Improvement in real-time obstacle detection system for usv," in Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on. IEEE, 2012, pp. 1317–1322.

[13] H. Wang and Z. Wei, "Stereovision based obstacle detection system for unmanned surface vehicle," in Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on. IEEE, 2013, pp. 917–921.

[14] H. Wang, X. Mou, W. Mou, S. Yuan, S. Ulun, S. Yang, and B.-S. Shin, "Vision based long range object detection and tracking for unmanned surface vehicle," in Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2015 IEEE 7th International Conference on. IEEE, 2015, pp. 101–105.

[15] A. J. Sinisterra, M. R. Dhanak, and K. Von Ellenrieder, "Stereovision-based target tracking system for usv operations," Ocean Engineering, vol. 133, pp. 197–214, 2017.

[16] Y. Cho, J. Park, M. Kang, and J. Kim, "Autonomous detection and tracking of a surface ship using onboard monocular vision," 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 26–31, 2015.

[17] F. Bousetouane and B. Morris, "Fast cnn surveillance pipeline for fine-grained vessel classification and detection in maritime scenarios," in Advanced Video and Signal Based Surveillance (AVSS 2016). IEEE, 2016, pp. 242–248.

[18] J. Yang, Y. Xiao, Z. Fang, N. Zhang, L. Wang, and T. Li, "An object detection and tracking system for unmanned surface vehicles," in Target and Background Signatures III, vol. 10432. International Society for Optics and Photonics, 2017, p. 104320R.

[19] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3d lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," Robotics and Autonomous Systems, vol. 83, pp. 299–311, 2016.

[20] S. Fefilatyev, D. Goldgof, M. Shreve, and C. Lembke, "Detection and tracking of ships in open sea with rapidly moving buoy-mounted camera system," Ocean Engineering, vol. 54, pp. 1–12, 2012.

[21] J. Muhovic, R. Mandeljc, J. Perš, and B. Bovcon, "Depth fingerprinting for obstacle tracking using 3d point cloud," in Proceedings of the 23rd Computer Vision Winter Workshop. CVWW 2018, 2018.

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in Readings in computer vision. Elsevier, 1987, pp. 726–740.

[23] R. M. Haralick and L. G. Shapiro, "Connected components labeling," Computer and robot vision, vol. 1, pp. 28–48, 1992.

[24] S. Hannuna, M. Camplani, J. Hall, M. Mirmehdi, D. Damen, T. Burghardt, A. Paiement, and L. Tao, "Ds-kcf: a real-time tracker for rgb-d data," Journal of Real-Time Image Processing, pp. 1–20, 2016.

[25] H. W. Kuhn, "The hungarian method for the assignment problem," Naval Research Logistics Quarterly, vol. 2, no. 1, pp. 83—-97, 1955.

[26] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," science, vol. 315, no. 5814, pp. 972–976, 2007.

[27] B.-S. Shin, X. Mou, W. Mou, and H. Wang, "Vision-based navigation of an unmanned surface vehicle with object detection and tracking abilities," Machine Vision and Applications, pp. 1–18, 2017.

Jon Muhovič received his M.Sc. degree from the Faculty of Computer Science and Informatics at University of Ljubljana in 2017. He is presently working as a researcher at the Laboratory for Machine Intelligence, Faculty of Electrical Engineering, and the ViCoS Laboratory, Faculty of Computer and Information Science, both at the University of Ljubljana. His research interests include computer vision, obstacle detection and visual tracking.

Rok Mandeljc received his Ph.D. from the Faculty of Electrical Engineering at University of Ljubljana in 2015. He is presently working as a researcher at the Machine Vision Laboratory, Faculty of Electrical Engineering, and the ViCoS Laboratory, Faculty of Computer and Information Science, both at the University of Ljubljana. His research interests include computer vision, with emphasis on various aspects and applications of object detection and recognition.

Borja Bovcon received his M.Sc. degree from the Faculty of Mathematics and Physics at University of Ljubljana in 2017. He is currently working as a researcher at the ViCoS Laboratory, Faculty of Computer and Information Science, University of Ljubljana. His research interests are computer vision, obstacle detection and autonomous systems.

Matej Kristan received the Ph.D. degree from the Faculty of Electrical Engineering, University of Ljubljana, in 2008. He is an associate professor at the ViCoS Laboratory at the Faculty of Computer and Information Science, University of Ljubljana. His research interests include probabilistic methods for computer vision with focus on visual tracking, dynamic models, online learning, object detection, and vision for mobile robotics. He is a member of the IEEE.

Janez Perš received B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering at the Faculty of Electrical Engineering (FE), University of Ljubljana, in 1998, 2001, and 2004, respectively. He is currently an assistant professor at the Laboratory for Machine Intelligence, Faculty of Electrical Engineering, University of Ljubljana. His research interests lie in image-sequence processing, object tracking, human-motion analysis, dynamic-motion-based biometry, and in autonomous and distributed systems.