Detecting Water Reflection Symmetries in Point Clouds for Camera Position Calibration in Unmanned Surface Vehicles

L. Qingqing[1,2*], J. Peña Queralta[1*], T. Nguyen Gia[1], Z. Zou[2], H. Tenhunen[3] and T. Westerlund[1]

[1] Department of Future Technologies, University of Turku, Finland
[2] School of Information Science and Technology, Fudan Universtiy, China
[3] Department of Electronics, KTH Royal Institute of Technology, Sweden
Emails: [1]{jopequ, tunggi, tovewe}@utu.fi, [2]{qingqingli16, zhuo}@fudan.edu.cn, [3]hannu@kth.se

***Abstract*—The development of autonomous vehicles has seen considerable advances over the past decade. However, specific challenges remain in the area of autonomous waterborne navigation. Two key aspects in autonomous surface vehicles are sensor calibration and segmentation of water surface. Cameras and other sensors in a car or drone can be installed accurately in a specific position and orientation. In a large vessel, this is not always possible, as sensors might be installed around the vessel or on masts. Taking advantage of the medium in which these vehicles operate, the water plane can be used as a reference for different sensors to calibrate their orientation. This allows more accurate localization of obstacles of objects. State-of-the-art deep learning techniques have been successfully applied for water surface segmentation in open sea. However, in other environments such as small rivers or lakes with still waters, a different approach might enable more accurate water surface estimation. We propose a method to estimate the water plane based on the detection of local symmetry planes that naturally occur when objects are reflected in the water. By using a point cloud generated with a stereo camera, we are able to accurately estimate the water level and, at the same time, calibrate the camera position and improve the localization of obstacles. We assume that an approximate position and orientation of the camera is known with respect to the sea level. We demonstrate the efficiency of our method with data obtained in the Aura river, Finland, with our prototype vessel facing both the riverside and the center of the river.**

***Index Terms*—Camera Calibration; Reflection Symmetry; Symmetries; Stereo Vision; Intel RealSense; Point Cloud; PCL; ROS; Autonomous Vessels; USV; Surface Vehicles; Autonomous Surface Vehicles; Autonomous Vehicles;**

I. Introduction

The past decade has seen a boost in the development of autonomous waterborne vehicles, or unmanned surface vehicles (USV), including autonomous surface vessels [1]–[3] or autonomous ferries [4], [5]. This has happened under the umbrella of a general trend in the development of autonomous systems [6], [7]. In an autonomous vessel, it is essential that cameras and other sensors are able to autonomously calibrate their orientation or even relative positioning [8], [9]. In this work, we focus on the estimation of the water surface plane by detecting local symmetries in a point cloud obtained with a stereo camera. This enables an accurate orientation estimation of cameras installed on-board a vessel, and therefore more accurate obstacle localization. Furthermore, the water plane

Fig. 1. Prototype vessel TIERS1 used as sensing platform for testing.

can be used as reference for collaborative mapping in multi-robot systems [10], or as an orientation reference in formation control algorithms [11], [12].

In a large autonomous vessel, it is often a complex task to accurately install and deploy small sensors such as cameras. For example, a small orientation error in cameras placed on the mast of a very long vessel can cause a significant error in positioning a detected obstacle with respect to the vessel. In addition, a large ship would require the installation of an analogously large number of cameras or sensors, and therefore manual calibration of their position and orientation might be unfeasible. Therefore, it is essential to have algorithms that enable automatic and real-time camera calibration. Even though we focus our work in river and lake operations, similar approaches can be used in calm seas such as those in the Finnish archipelago, where multiple ferries operate. It has been in this archipelago where the world's first autonomous ferry was unveiled [8]. Therefore, when we refer to large vessels we also take into account city ferries that operate on rivers, ships that navigate on lakes and ferries that transport people, vehicles or goods between close islands in an archipelago.

In terms of vessel dynamics, roll oscillation in ships is a phenomenon that does not commonly occur in ground vehicles and poses critical challenges to typical navigation and object detection approaches [13]. Roll oscillations significantly affect vision-based localization, mapping and object detection algorithms. Inertial measurement units, and in particular accelerometers and gyroscopes, can be used to

*: These authors contributed equally to this work.

estimate the roll of a ship and the orientation of individual cameras. Nonetheless, this would require either the installation of inertial sensors on each camera separately, or the design of a robust method for estimating the relative orientation of different cameras. Instead, we propose the analysis of stereo images to estimate the orientation of the cameras based on their position with respect to the water surface. Then, the orientation of monocular cameras can be estimated via object detection and matching between different cameras.

The detection of the water surface plane poses different challenges in open sea and on quiet waters such as rivers and lakes. In the former case, the height and movement of waves can have a significant impact on the performance of different methods. In the latter case, the reflective properties of still waters, which act as a mirror, might result in a complex distinction between objects above and below the water surface. However, this reflection can be exploited to find the water surface as a reflection plane in the image or a local symmetry plane in a point cloud obtained from a stereo camera.

The methods we design and develop can be applied to autonomous navigation in environments where water is reflective enough and there are objects nearby that can be detected by cameras on-board the vessel. This is the case of small rivers and lakes such as those where city ferries typically operate. The use of cameras instead of other sensors such as 3D lidars, which are becoming increasingly popular in the development of autonomous ground vehicles [14], [15], has been motivated by the intrinsic limitation of light sensing in water. While water in open seas can provide a limited number of returns for a 3D lidar because of high waves, lidars are practically blind in still waters where laser beams are almost entirely reflected.

The remainder of this paper is organized as follows. Section II covers related works on symmetry plane estimation in discrete sets of points, water surface segmentation with deep learning methods and calibration of cameras in unmanned surface vehicles. In Section III, we describe our algorithm for estimating the water surface plane based on object reflections near the coastline (or riverside). Section IV introduces our prototype vessel TIERS1 and its sensor suite, as well as the data acquisition and pre-processing methods, and delves into the experimental results and their analysis, and we compare the efficiency of our method for individual frames and using previous frames estimations. Finally, Section V concludes the work and outlines future work directions.

II. Related Work

Even though research into autonomous surface vessels has seen a rapid development over the past years, this is still a relatively new research area with a correspondingly limited number of previous research works. In this paper, we are focusing on using stereo vision for estimating the relative position of a camera with respect to the water surface plane. In comparison with the problem of road detection and segmentation in autonomous cars, segmentation of water surface can pose additional challenges because of the reflective and refractive qualities of water. To the authors' best knowledge, symmetry analysis of point cloud data to find the water level surface based on object reflections has not been proposed before. Due to the lack of previous work, in this section we review related works in the areas of (i) vision-based navigation and operation of autonomous vessels in general; (ii) water surface detection and segmentation with vision-based techniques; and (iii) estimation of approximate symmetry planes from discrete point cloud data.

Huntsberger *et al.* presented the first stereo vision-based navigation system for autonomous surface vessels [16]. The authors propose the use of stereo cameras as a low-latency and real-time method for detecting obstacles above the sea level. This is the envisioned application for our system as well. However, Huntsberger *et al.* worked with a relatively small boat and were able to install the camera array in a known position with known orientation. We aim at providing a flexible methodology for camera calibration that will enable the installation of cameras in large vessels without accurate knowledge of their orientation. Therefore, it is essential to provide an accurate method for estimating camera orientation in real time. In addition, it is worth noting that Huntsberger *et al.* already acknowledge in their early work the challenge that specular reflections can pose for cameras.

Bovcon *et al.* have also proposed a stereo-based obstacle detection scheme for USVs [17]. A stereo-view semantic segmentation is shown in their work to outperform state-of-the-art methods that rely on monocular approaches and convolutional neural networks. In terms of water edge detection, Wang *et al.* previously proposed the combination of saliency detection and motion estimation to find obstacles under an estimated water edge [18]. Nonetheless, they assumed a clear distinction between sky and water. The approach of Bovcon *et al.* is more flexible and robust in the water edge detection, but is still mostly limited to open sea. In our work, we do not assume that water surface and sea intersect in the camera horizon as this is often not the case in rivers and lakes.

State-of-the-art deep learning methods have been applied to enable automatic water-body segmentation. However, to the extent of our knowledge, this has been done mostly with satellite or airborne imagery. Liu *et al.* presented a Segmentation-based Convolutional Neural Network (SLS-CNN) model for ship detection using Synthetic Aperture Radar (SAR) imagery [19]. Other researchers have been applying convolutional neural networks for water-land segmentation from satellite images. Li *et al.* trained a novel deep neural network named DeepUNet for pixel-level sea-land segmentation [20]. In the same direction, Miao *et al.* proposed a restricted receptive field deconvolution network (RRF DeconvNet) [21].

A different approach utilized by Hilsenstein was to use thermographic stereo vision for reconstruction of water waves surface [22]. The author exploits the complex temperature patterns that can be seen in water surfaces under infrared light, as compared to transparent water seen within the visible spectrum. This is a promising approach for open sea applications, but has a more limited impact on navigation in still waters.

Most of the previous works on autonomous vessels or USV

Algorithm 1: Water surface plane estimation in ROS.

Subscribe to:
  $/camera/color/points$;
  $/lidar(0,1)/scan$;
Publish to:
  $/water/surface$;
  $/water/height$;
  $/calibration/camera(yaw, pitch, roll)$;
while *True* **do**
  // Estimate distance to riverside
  $distance\_to\_coastline = \mathbf{F}(lidar\_scan, gnss\_data)$;
  $filtered\_pointcloud = \mathbf{F}(stereo\_pointcloud)$;

  // Estimate water level plane.
  $water\_height =$ **find_symm**$(filtered\_pointcloud)$;
  $roll\_estimate =$
    find_symm$(water\_height, filtered\_pointcloud)$;
  $pitch\_estimate =$ **find_symm**$(filtered\_pointcloud)$;

  // Use compass to obtain the yaw.
  $yaw\_estimate =$ **from_compass**$()$;

  // Combine to obtain camera orientation.
  $camera\_orientation =$
    combine$(water\_line, roll\_estimate, pitch\_estimate)$;

has been focused on open sea navigation. However, in this paper we put an emphasis on the specific characteristics of waterborne navigation in still waters. As quiet waters have good specular properties, we propose to exploit this in order to estimate the water level by finding local symmetries in point cloud data. This is a problem that has been previously studied in the fields of computer vision and computer graphics as a step for modelling objects for several decades [23]. More recently, Li *et al.* proposed a method specifically meant for boundary improvement in models made of discrete point sets [24]. For larger dimensions, such as point clouds that include color or other non-spatial information, Nagar *et al.* provide an algorithm for detecting approximate reflection symmetries [25]. In our work, we take inspiration from works on symmetry detection in order to detect the water surface plane.

III. Water Surface Plane Estimation

In this section, we outline the methodology for estimating the water surface plane based on the stereo camera point cloud. The algorithm we propose utilizes either lidar data or GNSS sensor data to estimate the distance of the boat with respect to the lakeside or river banks, depending on the application scenario. This information is used in order to make initial filtering of the stereo point cloud. The method we propose is described in Algorithm 1, and can be broadly divided into four steps: (i) point cloud filtering, (ii) camera height estimation, (iii) camera roll estimation, and (iv) camera pitch estimation. In this paper, we show results on camera height and roll estimation. The approach we propose can be extended to add the pitch estimation, and we will account for it in future work. In the algorithm, $\mathbf{F}$ denotes an undetermined function.

A. Point cloud filtering

The first step is to pre-process data and apply some filters to the point cloud in order to reduce the number of points and reduce the computational team necessary to run our proposed algorithm. This is essential to enable real-time operation. Taking into account that a considerable part of the point cloud does not appear in the reflection, we use GNSS or lidar data to estimate the distance to the lakeside or river banks. Then, this distance is used to filter points so that we only take into accounts objects near the coastline, which is where reflections occur. Based on this distance, points are filtered in two ways: by absolute distance to the boat, and by vertical distance. If the ship is near the coastline, then only objects with relatively small height are considered, as there would not be enough field of view to perceive the reflection of tall objects. The opposite applies when the ship is far from the water-land boundary, as there might be too much noise around it with objects being too small to be identifiable while big objects have more clear reflections. These phenomena can be observed in Figures 4 and 5. In addition, if the point cloud is very dense, we perform a downsampling operation to speed up the analysis.

B. Camera height estimation

The next step is to estimate the height of the camera with respect to the water surface. In this paper, we assume that the camera pitch and roll are known beforehand within a certain and small range. This is a practical assumption as these parameters can be roughly estimated when cameras are installed on the ship with errors of a few degrees. We also assume that the camera height is known to be within a certain interval. If this information is not available, the same method can still be applied but the computational complexity and execution time increase significantly.

After filtering, suppose we have a point set $S = \{\mathbf{x}_i\}_{i=1}^{N}$ of size N. We assume that the point set coordinates are adjusted according to the estimated pitch and roll of the camera. Therefore, we suppose that the water surface plane is almost horizontal. In order to follow the coordinate system in which the Intel RealSense's point cloud is defined, we assume that the height is given by coordinate y, z provides the *depth* and x describes the lateral distance with respect to the camera.

Then, given a pre-defined interval (h_l, h_h) where the camera is assumed to be, we choose a step Δh and calculate, for each height $h = h_{min} + k\Delta h$, the reflection of all points such that $\mathbf{x}_{i_y} > h$ and $\mathbf{x}_{i_y} - h < h_{th}$. The constant $h_{th} \in \mathbb{R}^+$ is a threshold defined as a function of the distance to the coastline, and $k \in \{0, \dots, (h_h - h_l)/\Delta h\}$ is an integer that used to sample the interval (h_l, h_h). We estimate the water level height h_w to be defined by: $\mathrm{h}_w = h : h = h_{min} + k\Delta h = \mathrm{argmin}_{k\in\{0,\dots,\frac{h_h - h_l}{\Delta h}\}} \frac{1}{N_k} \sum_{\substack{\mathbf{x}_{i_y} > h \\ \mathbf{x}_{i_y} - h < h_{th}}} \left\| \mathbf{x}_i^R - \mathbf{kdTree}(\mathbf{x}_i^R) \right\|^2$ where $\mathbf{x}_i^R = \mathbf{x} - 2(\mathbf{x}_{i_y} - h)\mathbf{u}_x$ is the reflection of point $\mathbf{x}$ with respect to a horizontal plane perpendicular to $\mathbf{u}_x$ at height h, $\mathbf{u}_x$ is the unit vector that defines the x axis, and N_k is the number

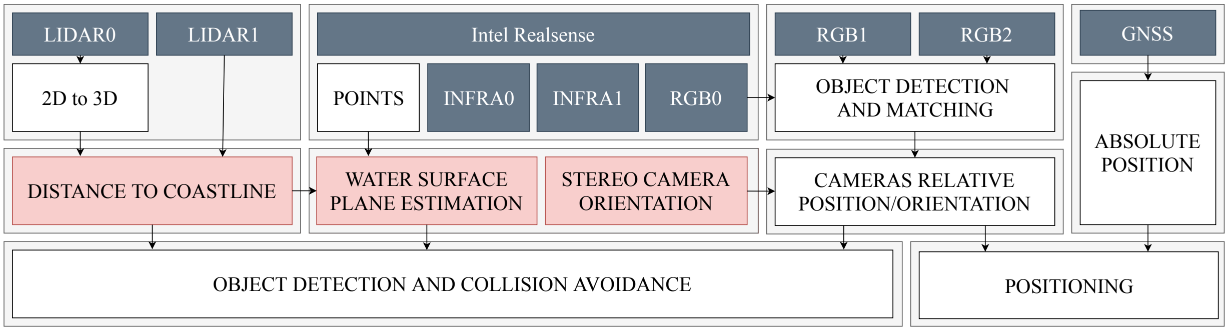

Fig. 2. Sensors installed on-board the TIERS1 prototype vessel and envisioned situational awareness data flow. This paper focuses on the processes marked in red, mainly in water surface estimation and stereo camera orientation.

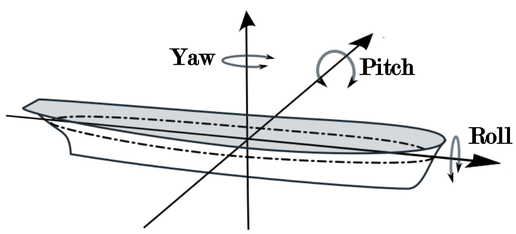

Fig. 3. Illustration of pitch, roll and yaw on a vessel.

of points within the height threshold in order to normalize the error. The value h_{th} is chosen so that only reflections of objects near the water surface are taken into account when the ship is near the coastline, and bigger objects are accounted for otherwise. We only take into account points above the water surface and try to find their reflection in the point cloud. A KD-tree search algorithm is used to find the nearest point in the point cloud to the expected reflection.

C. Camera roll and pitch estimation

Once we have obtained an approximate camera height estimation, we perform the same symmetry analysis on a subset of the point cloud around the symmetry plane in order to adjust the roll and pitch angles. Let $h_c = -h_w$ be the approximate camera height and water level height obtained in the previous step, and $-\phi$ and $-\theta$ the camera roll and pitch angles, respectively. The yaw angle ψ is not taken into account as the water plane is invariant to the camera yaw and it can be estimated with a magnetometer or other sensors. The normal vector of the reflection plane will be defined by

$$\mathbf{n}_r = \cos\phi\cos\theta\,\mathbf{u}_x + \sin\phi\,\mathbf{u}_y + \sin\theta\,\mathbf{u}_z \tag{1}$$

and we assume that the distance between the camera and the water level is the height obtained in the previous step. This assumption holds without incurring in a significant error if the roll and pitch angles are small, as we are assuming, and if the boat is not too far from the coastline, so that the camera horizon is above the water level. This is a logic expectation as we are developing a method for operation in river, lakes or archipelagos, where the distance between the boat and the nearest point of land should never be too large or otherwise there would be no reflections in the water to take into account. Following the same reasoning, we can approximate a point's distance to the plane d_r and its reflection by

$$\mathbf{x}_i^R = \mathbf{x}_i - 2d_r\mathbf{n}_r,\ \ d_r \approx \mathbf{x}_{i_y}\sin\theta\sin\phi \tag{2}$$

We have denoted the pitch and roll angles with a negative sign so that ϕ, θ refer to the angles that define the water plane as seen from the camera. We can then estimate the roll and pitch angles minimizing an error function analogous to that of Eq. III-B, where we would find the argument of the minimum angles within some predefined intervals.

This is a simple approach to calculate the water level, which we have used as a proof of concept for our proposed algorithm. More elaborate methods such as those defined by Nagar *et al.* can be adapted [25]. However, this is out of the scope of this paper and we have focused on demonstrating the possibility of estimating the water surface plane based on water reflections. Future works will focus on the effect of camera pitch and roll on the estimation of the water level plane.

IV. Experiment and Results

In order to test the efficiency of our proposed methods, we have obtained a series of images in the Aura river, Turku, Finland. With our prototype vessel, we compare the performance of our algorithm in two situations. First, when the bow is pointing at the riverside, with clear reflection in the water of walls and trees in the river bank. Second, when the bow is pointing to the center of the river, with only big trees from the river banks being visible and reflected. These two situations are illustrated in Figures 4 and 5, respectively. In the first image, the light source is behind the camera creating a clear view and reflection. In the second image, the light source is partly in front of the camera, and therefore colors are less evident. We have used these two light settings to test the robustness of our methods.

A. Sensing Platform: TIERS1

We have used a small ship model as a platform for gathering data and carrying out experiments. The ship, TIERS1, is a model based on an Aerokits Sea Queen model boat, and has been built at the Turku Intelligent Embedded and Robotic

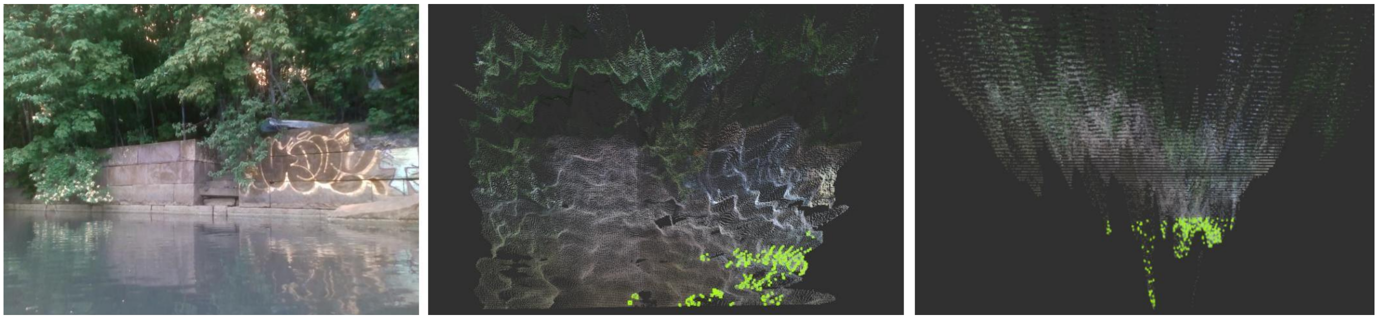

Fig. 4. Prototype looking towards the riverside: RGB view [l]; RGB point cloud from camera point of view [c]; Filtered point cloud from a side plane [r].

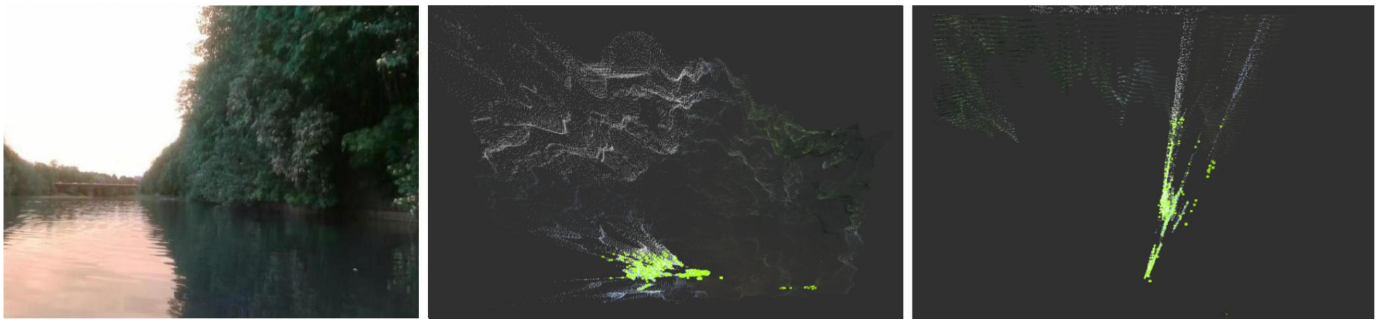

Fig. 5. Prototype looking towards the river center: RGB view [l]; RGB point cloud from camera point of view [c]; Filtered point cloud from a side plane [r].

Systems (TIERS) Group. The ship has a length of 1170mm and a beam of 370mm. The draft is approximately 120mm if loaded. An image of the ship appears in Figure 1.

The following hardware has been installed on-board the ship: (i) 2 x RPLidar A1M8, one of them tilted 9 degrees in order to detect obstacles at the water surface level immediately in front of the boat's bow; (ii) 1 x Intel RealSense D435 stereo camera, with one RGB camera and two IR cameras; (iii) 2 x Logitech c270 USB cameras; (iv) 1 x Ublox M8N GNSS sensor; (v) Intel UPS-GWS01 IoT edge gateway as the main on-board computer, with 4x1.4GHz cores Intel Atom CPU, Intel HD Graphics and 4GB RAM. The Ublox GNSS sensor also has a built-in compass. The computer, lidars and cameras are powered from an 8.6Ah AGM battery. The boat motor, servo motor and radio receiver are powered by a separate 8.4V, 3800mAh NiMH battery. All the sensors are installed on a 380mm mast with a total ship height of 600mm.

The on-board computer runs Ubuntu 16.04 and ROS Kinetic, which is used as a framework for interfacing with all sensors. We use Intel's *realsense_camera*, Slamtec's *rplidar* and *nmea_navsat_driver* ROS packages. On top of that, we run our own ROS package. As one of the lidars is tilted 9 degrees towards the ship bow, we apply a transformation to publish 3D point cloud data from the 2D lidar data. We use this information and the Realsense's point cloud to estimate the waterline and water surface level with PCL, which are then published with their own respective topics. Figures 4 and 5 show the stereo-based point cloud (center) and a side view of the filtered point cloud (right).

ROS (Robot Operating system) is an open source operating system for robots, which provides a publish-subscribe communication framework that can quickly build distributed computing systems [26]. The goal of ROS is to provide algorithm reuse support for robot research and development. PCL (Point Cloud Library) is a cross-platform open source C++ library, which implements common algorithms and data structures of point clouds [27]. It can realize point cloud acquisition, filtering, segmentation, registration, retrieval, feature extraction, recognition, tracking, surface reconstruction, visualization and so on. If OpenCV is the crystallization of 2D information acquisition and processing, PCL has the same position in 3D information acquisition and processing.

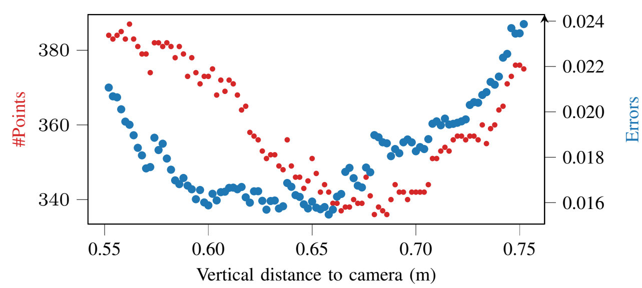

Fig. 6. Water plane height estimation error for data in Figure 4.

B. Analysis of Results

We have applied the water height estimation in (III-B) to the point cloud data included in Figures 4 and 5. The computation of the error according to (III-B) is shown in Figures 6 and 7. The water height is estimated towards the minimum of the

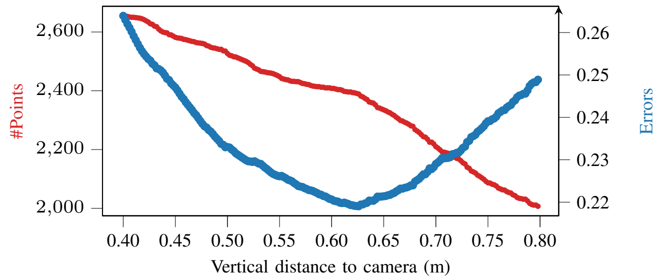

Fig. 7. Water plane height estimation error for data in Figure 5.

blue curve, and therefore we obtain a height of 0.6 to 0.65 m in both cases, with more noise in the first case.

In the situation where the ship is close to the river bank, we have used a height threshold $h_{th} = 0.5$ m as reflections are more clear for objects close to the water level. In the case of the ship bow pointing towards the center of the river, we have used $h_{th=3}$ m to account for taller trees and their reflections. Analogously, we have used absolute threshold distances of 5 and 8m, respectively, taking into account the distance to the river bank estimated from the lidar data. These constants have an impact on the number of points that are used to calculate the error. The number of points for each potential height analyzed is shown in red in the graphs. We have added this information as the method would not be reliable if the number of points found within the predefined height thresholds is too low.

The original size of the point cloud in the first situation was of 175985 points. From these, after filtering and downsampling we utilized a total of 9009 points to estimate the water height. In the second scenario, the original number of points was 183806 and the final number after filtering was 11164.

V. Conclusion and Future Work

We presented a methodology for estimating water surface plane in an autonomous boat via point cloud analysis. The point cloud data was generated with a stereo camera and an algorithm for finding local symmetry planes was applied. These symmetry planes are caused by the specular properties of still waters, such as those of lakes or rivers. An automated and accurate calibration of cameras enables a correspondingly accurate measurement of obstacles. Therefore, the situational awareness and orientation of individual cameras can have a significant impact on the performance of navigation and collision avoidance algorithms in autonomous vessels.

We have tested our proposed algorithm in the Aura river, Turku, Finland. Data has been gathered with a customized small model boat. Different settings have been analyzed, with the boat both pointing to the riverside and the center of the river. In both cases, our algorithm has been able to estimate the camera height. In future work, we will integrate this system into an obstacle detection and avoidance scheme, and study further the estimation of camera pitch and roll.

References

[1] T. Statheros *et al.* Autonomous ship collision avoidance navigation concepts, technologies and techniques. *Journal of Navigation*, 61(1):129–142, 2008.

[2] B. E. Jose *et al.* Optimisation of autonomous ship manoeuvres applying ant colony optimisation metaheuristic. *Expert Systems with Applications*, 39(11):10120 – 10139, 2012.

[3] A. Bassam *et al.* Experimental testing and simulations of an autonomous, self-propulsion 2 and self-measuring tanker ship model. *Ocean Engineering*, 2019.

[4] D. Henkel *et al.* Towards autonomous data ferry route design through reinforcement learning. In *2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6. IEEE, 2008.

[5] J. Leikas *et al.* Ethical framework for designing autonomous intelligent systems. *Journal of Open Innovation: Technology, Market, and Complexity*, 5(1):18, 2019.

[6] D. J. Fagnant *et al.* Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[7] O. Levander. Autonomous ships on the high seas. *IEEE Spectrum*, 54(2):26–31, 2017.

[8] Rolls-Royce *et al.* Finferries' falco world's first fully autonomous ferry. *Finnferries*, 2018.

[9] J. Villa Escusol *et al.* Autonomous and collaborative offshore robotics. In *Proceedings of the 2nd Annual SMACC Research Seminar 2017*, pages 57–59. Tampere University of Technology, 2017.

[10] J. Peña Queralta *et al.* Collaborative mapping with ioe-based heterogeneous vehicles for enhanced situational awareness. In *IEEE SAS*, 2019.

[11] J. Peña Queralta *et al.* Communication-free and index-free distributed formation control algorithm for multi-robot systems. *Procedia Computer Science*, 2019.

[12] C. McCord *et al.* Distributed Progressive Formation Control for Multi-Agent Systems: 2D and 3D deployment of UAVs in ROS/Gazebo with RotorS. In *2019 European Conference on Mobile Robots (ECMR)*, 2019.

[13] R. A. Ibrahim *et al.* Modeling of ship roll dynamics and its coupling with heave and pitch. *Mathematical Problems in Engineering*, 2010.

[14] J. Zhang *et al.* Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE, 2015.

[15] B. Li *et al.* Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

[16] T. Huntsberger *et al.* Stereo vision–based navigation for autonomous surface vessels. *Journal of Field Robotics*, 28(1):3–18, 2011.

[17] B. Bovcon *et al.* Obstacle detection for usvs by joint stereo-view semantic segmentation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5807–5812, Oct 2018.

[18] H. Wang *et al.* Improvement in real-time obstacle detection system for usv. In *2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1317–1322, Dec 2012.

[19] Y. Liu *et al.* Sar ship detection using sea-land segmentation-based convolutional neural network. In *2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP)*, May 2017.

[20] R. Li *et al.* Deepunet: A deep fully convolutional network for pixel-level sea-land segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(11):3954–3962, Nov 2018.

[21] Z. Miao *et al.* Automatic water-body segmentation from high-resolution satellite images via deep networks. *IEEE Geoscience and Remote Sensing Letters*, 15(4):602–606, April 2018.

[22] V. Hilsenstein. Surface reconstruction of water waves using thermographic stereo imaging. In *Image and Vision Computing New Zealand*, volume 2. Citeseer, 2005.

[23] H. Schultz. Retrieving shape information from multiple images of a specular surface. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):195–201, 1994.

[24] L. Ming *et al.* Detecting approximate symmetries of discrete point subsets. *Computer-Aided Design*, 40(1):76–93, 2008.

[25] R. Nagar *et al.* Detecting approximate reflection symmetry in a point set using optimization on manifold. *IEEE Transactions on Signal Processing*, 67(6):1582–1595, 2019.

[26] M. Quigley *et al.* Ros: an open-source robot operating system. In *ICRA workshop on open source software*. Kobe, Japan, 2009.

[27] R. B. Rusu *et al.* 3D is here: Point cloud library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, May 2011.