Efficient LiDAR-based In-water Obstacle Detection and Segmentation by Autonomous Surface Vehicles in Aquatic Environments

Mingi Jeong and Alberto Quattrini Li

Abstract— **Identifying in-water obstacles is fundamental for safe navigation of Autonomous Surface Vehicles (ASVs). This paper presents a model-free method for segmenting individual in-water objects (e.g., swimmers, buoys, boats) and shorelines from LiDAR sensor data. To reduce the computational requirement, our method first converts the 3D point cloud into a 2D spherical projection image. Then, an algorithm based on the integration of a breadth-first search and a variant of a hierarchical agglomerative clustering segments the points according to different objects. Our method addresses the sparsity and instability of the point cloud in the aquatic domain – a characteristic that makes the methods developed for self-driving cars not directly applicable for in-water obstacle segmentation, as demonstrated in our experiments. Our method is compared with other state-of-the-art approaches and is validated both in simulation and in real-world ASV deployments, with different objects and encountering scenarios. The proposed method is effective in segmenting in-water obstacles not known a priori, in real-time, outperforming other state-of-the art methods.**

I. Introduction

The goal of this paper is to detect and segment objects from real-time LiDAR data, mounted on Autonomous Surface Vehicles (ASVs) operating in aquatic environments, such as lakes and oceans – see Fig. 1. This task is fundamental for ASVs' situational awareness to safely navigate around other obstacles, such as buoys, boats, and swimmers.

Current work on in-water obstacle detection primarily uses cameras. Publicly available datasets in the marine domain are solely comprised of camera images [1]–[3]. However, cameras have limitations: (1) restricted range and Field of View (FOV) and (2) restricted visibility in dynamic conditions, e.g., fog, rain, and nighttime. Systems fusing cameras with other sensors typically require additional high computation, such as GPU for deep learning methods [4], [5].

LiDAR is more expensive and constrained in its applicability in some scenarios (e.g., extremely bad weather) compared to RADAR, but provides more accurate information and can operate better when conditions are not ideal for cameras. Self-driving cars have been using LiDAR-based methods, showing success in object segmentation – the process of identifying 3D points belonging to the same object – such as pedestrians or other cars in many scenarios [6], [7]. The aquatic domain introduces a number of challenges to such methods: compared to urban domain, LiDAR data are more likely to be affected by sparsity and instability [8], [9], making segmentation difficult. In the ASV's case, there are no measurements available about the water surface due to the attenuation of the signal, ego-motions of both ASV and in-water objects are affected by water dynamics, and detection of objects is typically much farther (in order of tens of meters), because of marine traffic safety and vehicle maneuverability, making less dense and stable point cloud data compared to the ground case where vehicles are within 5 to 10 m. Fig. 2 and the Appendix show a number of examples of point clouds from the Velodyne VLP-16 in the two different domains, highlighting this issue. A couple of methods recently appeared for ASVs, addressing static obstacles detection [10] or navigations in urban waterways [11]. Reliable approaches for in-water obstacles segmentation are needed to make ASVs operate safely in other type of environments that are not as structured, such as lakes and oceans and where obstacles are not known a priori.

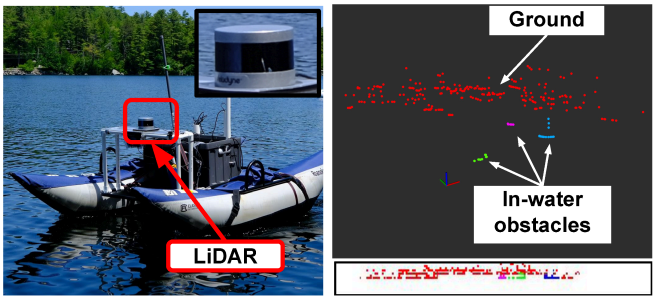


Fig. 1: Robotic boat *Catabot* with Velodyne VLP-16 scanner (left) performs LiDAR-based obstacle detection and segmentation based on the spherical projection image (right).

We present an efficient LiDAR-based obstacle detection and segmentation algorithm, tailored to the LiDAR data in the aquatic domain. The proposed method generates a **2D spherical projection image** from raw **3D point clouds**, finds connected components with Breadth-First Search (BFS), and clusters those components to segment in-water obstacles. Our method segments the objects based on compatibility of geometric distributions – orientation with respect to the sensor, orientation between neighboring clusters, and minimum separation distance between neighboring clusters. This method enables robustness to occlusion between objects, sparsity, and instability in the marine domain. The output is a point-wise segmentation, distinguishing **3D point clouds** into each object as shown in Fig. 1. Our algorithm was compared with other state-of-the art approaches to analyze the performance of segmenting in-water obstacles of which data collected by diverse real-world scenarios. For validation

The authors are with Department of Computer Science, Dartmouth College, Hanover, NH USA {mingi.jeong.gr, alberto.quattrini.li}@dartmouth.edu

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. DOI: 10.1109/IROS51168.2021.9636028

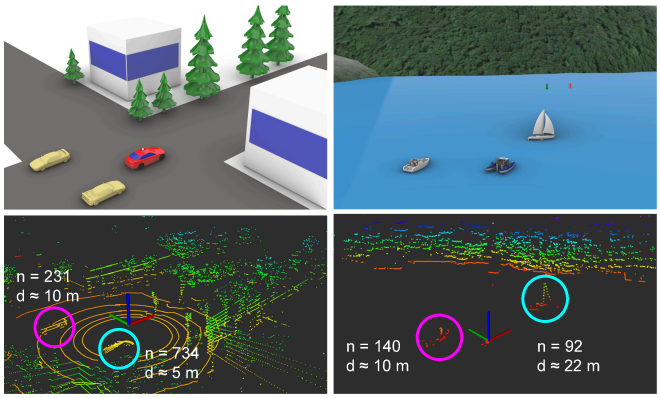


Fig. 2: Schematic diagram of a typical scenario in urban and aquatic environments and raw real LiDAR (Velodyne VLP-16) point cloud data comparison between a self-driving car[1] (left) and our ASV (right). In aquatic environments obstacles are typically farther than in urban ones, resulting in sparser point clouds. d is distance of an object from the scanner and n is the number of point clouds for the two obstacles (circles in cyan and magenta).

and applicability, we used a Velodyne VLP-16 LiDAR, one of the most common scanners with relatively low cost (∼4K USD) rather than higher resolution models which are extremely expensive – e.g., 32, 64 channels (∼30K, ∼80K USD). The proposed method showed to reliably segment objects, outperforming other state-of-the art methods, as well as showed efficient capability for real-time application to computationally limited mobile platforms.

This paper provides the following contributions:

- A model-free approach to segment point clouds in aquatic domains with connected components and clusters via fast and effective processing of 2D spherical images;
- A real-time online application of in-water obstacle detection and segmentation performed both in simulations and in field experiments with a real ASV;
- A feasibility and robustness validation through diverse scenarios in lakes and oceans, covering different coastal lines and several types of obstacles, e.g., boats, kayaks, swimmers, buoys, and docks.

Towards a full real-time obstacle avoidance system, we plan as future direction to extend the proposed algorithm by the registered data over the span of the past frames, for multi-object tracking.

II. Related Work

Point cloud segmentation – i.e., classification of 3D points to objects – has been pushed forward by advances in the self-driving car industry and the availability of datasets collected in urban environments – e.g., KITTI [12]–[14]. Two classes of methods can be identified: one purely based on LiDAR (e.g., [6], [15]–[17]), and one where multiple sensors are fused (e.g., [18]–[20]). These approaches can be again categorized into two data processing formats: 2D representation after applying a transformation to the point cloud [6], [16], [18], [20], and 3D representation, either directly using the raw 3D point clouds [17] or some approximation methods, e.g., voxel [15], [19]. Given the sparsity and instability of the 3D point cloud in the aquatic domain [8], [9], as highlighted in the previous section, we select a 2D representation for computational efficiency.

For ASVs, point cloud segmentation work appears to be at its early stage, with no datasets readily available. Current methods mainly focus on obstacle detection, identifying if an obstacle exists or not, with camera sensors [21]. Recently, a couple of algorithms fused LiDAR with other sensors (RADAR and camera) for maritime detection [4], [22]. The main focus of [4] was situational awareness and avoidance with comprehensive sensor system, making it highly expensive for computation and design. Haghbayan *et al.* [22] addressed the maritime object detection and recognition problem. Their method was tested on recorded dataset only and not on-board. As ASVs are mainly deployed in unknown local environments [23], with not complete information of existing obstacles, compared to commercial vessels deployed in charted oceans, approaches using prior information are not suitable for ASVs deployed in those areas. A recent LiDAR-based method [10] has shown to detect static obstacles using occupancy grid mapping. Another LiDAR-based study [11] applied an open-source package [24] to operate in urban waterways. Urban waterways provide structure that can be exploited to identify areas outside of the water, e.g., buildings, to then segment the remaining points for detecting the in-water obstacles. Muhovič *et al.* [25] proposed a method that uses point clouds from an ASV's stereo camera to fit the water plane, but does not segment obstacles.

This paper addresses the problem of "how can an ASV, operating in an unstructured environment where obstacles are not known *a priori*, detect and segment the surrounding scene, e.g., waterborne obstacles and shorelines, by only using low-cost 3D LiDAR?". The focus is on small ASVs, with limited computational resources available on-board.

III. Obstacle Segmentation Approach

Our obstacle segmentation approach is model-free and is based on transforming a 3D point cloud, by applying a spherical projection for efficiency of traversal, and integrating a BFS and a variant of hierarchical agglomerative clustering to find connected components by a graph search among sparse and instable marine point clouds.

While our method works with any 3D LiDAR sensor that provides point cloud data (x, y, z of the measured points), to ground the following presentation, we consider our custom-made ASV *Catabot* [26] – Fig. 1 (left). *Catabot* is equipped with a relatively low-cost Velodyne VLP-16 LiDAR with a 360° horizontal and 30° vertical FOVs and other proprioceptive and exteroceptive sensors, including GPS, IMU, and sonar. The LiDAR is set at a height of $0.3\,\mathrm{m}$, with a $1\,\mathrm{m}$ for x, z translation from the lowest part of the body frame, to

[1]Velodyne LiDAR sample data for Monterey highway `https://velodynelidar.com/`

minimize occlusions from its own ASV body and to optimize sensor reading stability. To eliminate 3D points from the robot itself, the first pre-processing step uses *crop box filter*.

A. Spherical Projection Transformation

The filtered 3D point cloud input is converted into a *2D spherical projection image*. The spherical projection is represented as a 2D matrix where each row and column corresponds to a given angle in a vertical and horizontal FOV, respectively, and the value is the distance [6], [16]. This representation allows the algorithm to exploit locality of nearby points for efficient data traversal, compared to 3D point clouds [7]. The image size, especially the column length, can be downsampled. Scale value can be decided upon available computational power and point cloud distribution, as discussed in Section IV. Given the sparsity of marine point clouds, which do not require high-resolution 2D spherical images, we use 16×512 images for the spherical projection. The column width setup is identical to downsampling with horizontal resolution of about $0.7°$ (compared to VLP-16's 1800 points and $0.2°$ horizontal resolution). Other values, discussed in Section IV, were experimentally tested.

B. Connected Component Searching

We use a BFS algorithm on the 2D spherical image (4-connected) to find a first coarse set of connected components, satisfying geometrical conditions – angle or distance threshold (additional consideration compared to [6]) between neighboring points to capture points of the same object. BFS has a time complexity of $\mathcal{O}(n)$ (n number of pixels).

One simplified aspect of the aquatic domain LiDAR data is that there is no need for ground plane labeling and segmentation, necessary for ground vehicles [6], [7], [13], [27]: we observe that no measurements are available about the water surface, due to attenuation of the laser signal.

Let's consider the example in Fig. 3 (top) which shows a magnified part of a real point cloud: there are two objects, a boat with a fisherman and a sailboat. After BFS, each of them have separate components despite being part of the same object – c1 and c2 are the boat hull and the fisherman, respectively; c3 and c4 are the sailboat and mast. In general, BFS traversal might result in more segmented components than the number of objects. Indeed, the whole point cloud depicted in Fig. 3 (top) results in 8 components returned, where only two objects were in the magnified part of the entire frame. This is due to the fact that more sparsity in point clouds leads to pixels in a spherical image be disjoint with blank gaps. Also, BFS might over-segment an object by splitting it into multiple components; this observation was noticed by Bogoslavskyi and Stachniss [6] when laser reflections were mostly parallel to a line connecting sensor and object. Waterborne obstacles typically have curved external shapes [8], [25]. Such a shape results in the angle threshold condition for neighborhoods failing to capture the group as a single component. This is uncommon in the urban case, but it occurs frequently in the marine domain.

Algorithm 1 Connected Component Clustering

Input:
- hash table for connected component F containing forest component f_i after BFS step in Section III-B
- spherical image matrix R with x,y,z data of point clouds
- threshold for dot product (ϵ_1)
- threshold for angle (ϵ_2)
- threshold for distance (ϵ_3)

Output: hash table for labeled clusters L

$L \leftarrow$ empty
for every f_i in F **do**
  if L is not empty **then**
    normalized sensor to centroid vector norm(c_i) by R and f_i
    Matrix $Dot_i \leftarrow$ dot_product(norm($c[1:i-1]$), norm(c_i))
    $d_{long} \leftarrow \max(\text{dist}(c[1:i-1]), \text{dist}(c_i))$
    $d_{short} \leftarrow \min(\text{dist}(c[1:i-1]), \text{dist}(c_i))$
    Matrix $Theta_i \leftarrow \arctan\left(\frac{d_{short} * \sin(\arccos(Dot_i))}{d_{long} - d_{short} * Dot_i}\right)$
    Matrix $Dist_i \leftarrow \text{dist}(c[1:i-1] - c_i)$
    Set $Mask_i \leftarrow$ find element for $(Dot_i \geq \epsilon_1)$ **AND** $\{(Theta_i \geq \epsilon_2)$ **OR** $(Dist_i \leq \epsilon_3)\}$
    merged cluster set $M_i \leftarrow$ empty
    for every l_k in L.items() **do**
      if $\exists(Mask_i \cap l_k.\text{value}())$ **then**
        M_i.add(l_k.key())
    $L \leftarrow$ update(L, M_i, f_i)
  else
    $L \leftarrow$ update(L, f_i)
return L

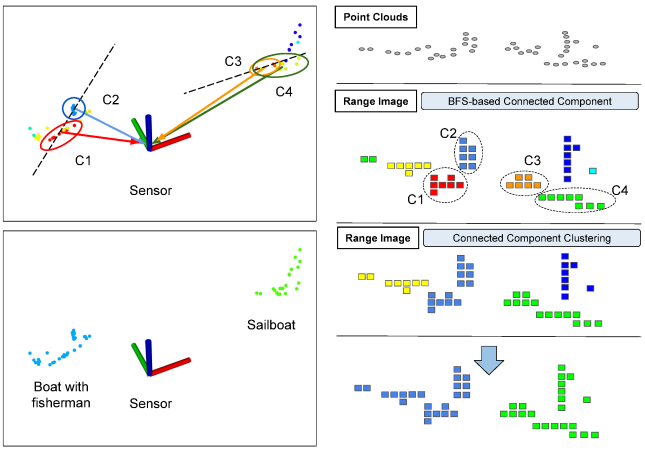


Fig. 3: Clustering and labeling example of point clouds from actual data in marine domain (Lake Sunapee, NH). *top:* BFS only method [6] returns many discrete components despite being one single object, due to marine specific point cloud characteristics. We marked a few parts of entire components for visibility. *bottom:* the connected components are merged by our proposed method to return accurate segmentation.

Despite over-segmenting, BFS is useful to reduce the overall number of components to be considered by the next step – clustering – thus improving the overall efficiency. Its impact will be evaluated in the experimental section.

C. Component Clustering and Labeling

The connected components found by BFS become a large-scale node in the graph space. We propose Algorithm 1 to reduce the number of components, based on a variant of hierarchical agglomerative clustering. This step on top of BFS makes our proposed algorithm better segment in-water object by considering the entire scene, e.g., even ground as

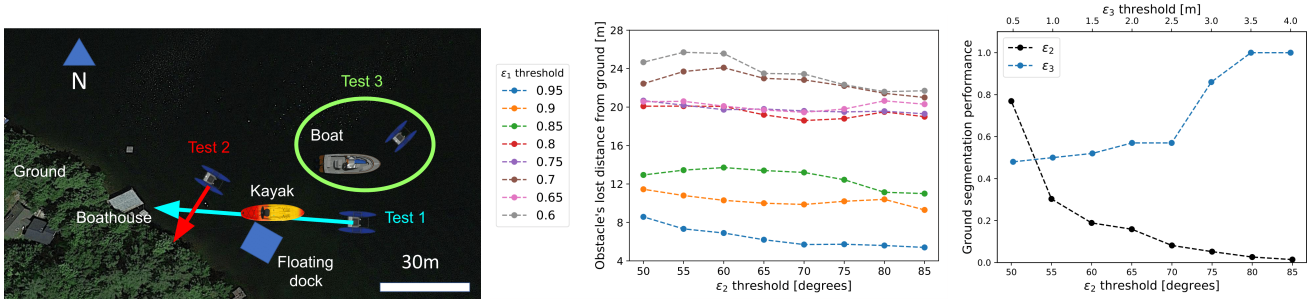


Fig. 4: Controlled test (Lake Sunapee, NH) and performance plot. *Left:* Test 1 to 3 environment setup. *Mid:* Test 1 result for ϵ_1, ϵ_2, *Right:* Test 2, 3 result for ϵ_2, ϵ_3 with fixed ϵ_1. ASV and obstacles are scaled up for visibility.

a large object cluster, than the others aimed at urban scenes like [6].

Two neighboring connected components can be merged by the following ***compatibility*** conditions meeting (a) **AND** {(b) **OR** (c)}: **(a)** assuming objects distributed in a 3D world, the components must lie sufficiently parallel and close to each other: more formally $|\phi_i \cdot \phi_j| \geq \epsilon_1$ where ϕ_i, ϕ_j is a normalized vector of the line joining the sensor to a component's centroid c_i, c_j, and ϵ_1 is a dot product threshold. ϵ_1 can be represented as a cosine threshold, given $\phi_i \cdot \phi_j = \|\phi_i\| * \|\phi_j\| * \cos\theta_{ij} = \cos\theta_{ij}$ where θ_{ij} is the acute angle between ϕ_i, ϕ_j. In Fig. 3, both (c_1, c_2) and (c_3, c_4) pairs satisfy this condition. **(b)** The line joining the centroids should have a sufficiently large angle with respect to the line from the sensor to the centroid with longer distance: $\arctan\left(\frac{d_{short} * \sin(\theta_{ij})}{d_{long} - d_{short} * \cos(\theta_{ij})}\right) \geq \epsilon_2$ where d_{short}, d_{long} is a vector length from the sensor to each component's centroid, and ϵ_2 is an angle threshold. As an intuitive example, components of point clouds, such as fore hull and aft hull belonging to a boat, form an almost perpendicular line with respect to the sensor. In Fig. 3, (c_1, c_2) meets this compatibility, merging the pair, whereas (c_3, c_4) fails, being almost in line. **(c)** the distance between the centroids should be close enough: $\|c_i - c_j\| \leq \epsilon_3$, where ϵ_3 is a distance threshold. Note the OR between conditions (b) and (c), necessary to be robust when (b) is not met by geometric shapes, e.g., boat side hull. ϵ_3 could be set considering the *Catabot*'s size, so that, although there might be two separate objects, they are merged together to prevent the robot considering it as a passage. In Fig. 3, both (c_1, c_2) and (c_3, c_4) meet this compatibility. Any component of the fisherman boat that does not meet the compatibility with a component of the sailboat (or vice versa) is a failure of (a) and (c) compatibility.

The algorithm uses matrix data structures for efficient calculations, avoiding pairwise comparisons. The algorithm's time complexity is in the worst case $\mathcal{O}(n^2)$ and best case $\Omega(n)$. Both the worst case – each point is a separate object – and the best case – entire points are in one object – are rare, as some points are likely to be part of a certain object.

Similar to other literature on segmentation, there are heuristic thresholds $\epsilon_1, \epsilon_2, \epsilon_3$. We experimentally optimized those parameters with a sensitivity analysis during controlled experiments and tested in other scenarios.

Each connected component is associated to a label. The segmented clusters can be transformed into a 3D point cloud format for visualization as [13], [16] shown in Fig. 1. Bounding boxes are typically used for segmentation. However, given the curvature and sparsity of the point clouds in the aquatic domain, points belonging to different objects could end in the same box. Thus, we select point-wise labeling.

IV. Results and Evaluation

No public dataset including LiDAR point clouds in aquatic domain is available. Hence, first, we deployed *Catabot* to collect point cloud data in diverse aquatic domains and situations. From the controlled dataset, we optimized the parameters. Validation of in-water obstacle segmentation is done in both simulation and real-world experiments, including ASV on-board real-time processing. Finally, we demonstrated the performance of the proposed algorithm compared to state-of-the-art methods.

A. Dataset Characteristic

Under two geographic categories – saltwater (Jan. 2020, in Carribean Sea) and freshwater (weekly, Apr. 2020 - Oct. 2020, in Lake Sunapee, NH and {Lake China, Sabattus, Auburn}, ME) – we have collected data of different encounters with diverse objects. The dataset is about 493 GB of LiDAR, RGB images, IMU, GPS, and sonar. Objects include swimmers, buoys (ball, pillar), power boats, water skiers, kayaks, floating docks, and sail boats. For encounters, we have head-on, crossing, overtaking, and overtaken situations, as defined by the navigation rules [28]. In addition to different ego-motions of the ASV (stop, move, turn, heavy roll, and pitch), the dataset includes diverse scenarios under non-controlled environments (e.g., a power boat approaches the ASV from a dock) and controlled environments (e.g., an obstacle intentionally maneuvers in a specified direction and speed). The dataset will be opensource[2].

B. Calibration Test

We evaluated the sensitivity of our method by changing parameters in Section III under controlled scenarios: (1) losing detection when an obstacle (3 m kayak) approaches the shore (Obstacle-move, ASV-not move) – when the ASV can start to detect an in-water object departing from the

[2]https://github.com/dartmouthrobotics/asv_detection_dataset.git

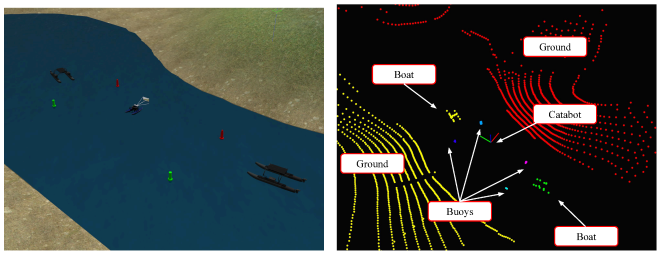


Fig. 5: Simulated environment in Gazebo for *Catabot* with buoys, shorelines, other boats (left) and segmentation result by our proposed method (right).

TABLE I: Evaluation of the proposed method's accuracy with state-of-the art methods. Note that all methods are used with default parameters for consistency. The run time is on average for tested sequences in the dataset. *denotes the width of spherical image projection. The approaches are – Euclidean Clustering (EC), Fast 3D method (F3D), and Depth Clustering (DC).

Approach	EC [24]	F3D [7]	DC [6]	Ours 512*	Ours 1024*	Ours 1800*
mIoU [%]	26.25	23.13	18.93	33.56	55.18	**84.47**
Time [ms]	140	21	**16**	30	57	86

shore; (2) segmenting a ground into one object when the ASV is close to the shore, i.e., approximately $20\,\mathrm{m}$ – how well the ASV can segment a large scale ground; and (3) segmenting an obstacle ($6\,\mathrm{m}$ boat) into one object when the point clouds show sparsity due to the hull shape – how well the ASV can cluster in-water object(s) without over-segmentation.

For Test (1), the kayak followed a straight line between the ASV and the shore. We used a portable laser device for measuring distance during the experiment and used the recorded data to calculate the distance when the kayak was lost as a segmented obstacle by the proposed algorithm. Test (1) is vital for finding the principal parameter ϵ_1, noted in Section III-C. As shown in Fig. 4, this test demonstrates that as ϵ_1 decreases, the method tends to better identify the obstacles closer to the ground. For example, from $8.58\,\mathrm{m}$ the kayak was "lost" at $\epsilon_1 = 0.95$, and from $24.67\,\mathrm{m}$ it was lost at $\epsilon_1 = 0.6$, both under fixed ϵ_2 $50°$. In addition, with fixed ϵ_1, as ϵ_2 increases, the method tends to perform better.

For Test (2), we monitored how the proposed algorithm segments a horizontally expanded ground into one object. We extracted point cloud frames when the ASV is about $20\,\mathrm{m}$ from the shore line. Here, the performance was quantified by how many separate clusters were identified given that the ideal number of clusters is 1. With good $\epsilon_1 = 0.95$ identified by Test (1), large ϵ_2 drastically decreased the segmentation performance. Even if ϵ_2 ensures the capability to distinguish one object – see Fig. 4 (mid) from the background scene (ground) – it should be tuned to prevent over-segmenting the ground, as shown in Fig. 4 (right).

Finally, Test (3) found a reasonable distance threshold to segment in-water obstacles with diverse shapes. This test is aligned with the observation and approach motivated in Section III. As shown in Fig. 4 (right), with fixed $\epsilon_1 = 0.95$, our method was able to segment the detected obstacle – $6\,\mathrm{m}$ motor boat – into one object over ϵ_3 as $3\,\mathrm{m}$.

Based on these tests, we selected ϵ_1 as 0.95, ϵ_2 as $50°$, and ϵ_3 as $3.5\,\mathrm{m}$.

C. Simulation

For repeatable tests, we used a realistic 3D simulator, Gazebo [29]. The simulated environment has relative ideal condition, i.e., less water dynamics, less sensor noise, and less point clouds constrained by the sensor in the simulator, compared to the real-world. We used realistic in-water obstacle models (Fig. 5): *Catabot* model (length: $2.4\,\mathrm{m}$, beam: $1.4\,\mathrm{m}$) with Velodyne VLP-16 scanner, boat (length: $4.9\,\mathrm{m}$, beam: $2.4\,\mathrm{m}$) and buoy[3] (radius: $0.25\,\mathrm{m}$, height: $1\,\mathrm{m}$), and terrestrial shape[4]. The total number of object clusters were 8. Due to some limitations of the Gazebo Velodyne library, measurements per vertical channel were set as 1024 points, about $4°$ horizontal resolution. Despite this discrepancy with the actual sensor model, to evaluate our method's robustness, we conducted the tests with the parameters identified from the calibration test. Fig. 5 (right) confirms that the proposed algorithm can robustly segment simulated waterborne obstacles into individual labels. Average running time was around $80\,\mathrm{ms}$ and *Catabot* was able to segment two obstacles at approximately $4\,\mathrm{m}$ apart based on the parameters in Section IV-B. As shown in the next section, our method is able to perform on-board with significantly lower computation time, indicating an effect of the simulator on the computation time.

D. Field Experiment

1) Segmentation Evaluation: Segmentation performance is summarized in Table I. Given the lack of LiDAR-based segmentation in water, we evaluated with other state-of-the-art algorithms originally aimed for ground vehicles:

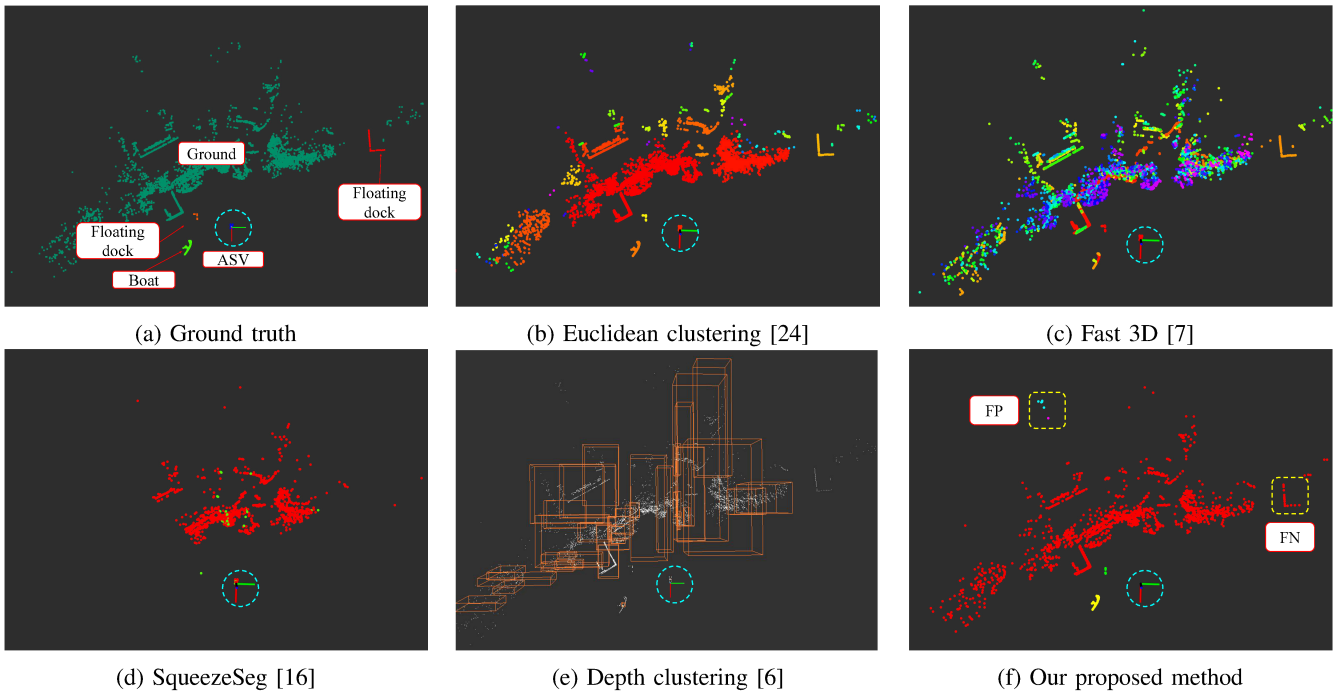
- algorithms that primarily use *spherical projection* [6], [16]. [6] is completely model-free denoted as DC in Table I. [16] uses deep learning and was only qualitatively analyzed due to no annotated dataset;
- algorithms that don't need any prior information and directly use 3D point clouds [7], [24], represented as EC, F3D in Table I.

Our method performs a point-wise segmentation and does not use a bounding box. Thus, we evaluated the performance with a metric variant at point level based on mean intersection over-union (mIoU) metric for accuracy [30]:

$$mIoU = \frac{1}{k} * \sum_{i=1}^{k} \frac{p_{ii}}{\sum_{j=1}^{k} p_{ij} + \sum_{j=1}^{k} p_{ji} - p_{ii}} \qquad (1)$$

where k is the number of objects, p_{ii} is correct segmentation for object i, and p_{ij}, p_{ji} is incorrect segmentation as object i or j is predicting each other. When a method finds several

[3] https://robotx.org/
[4] https://github.com/Intelligent-Quads/iq_sim



(a) Ground truth (b) Euclidean clustering [24] (c) Fast 3D [7]

(d) SqueezeSeg [16] (e) Depth clustering [6] (f) Our proposed method

Fig. 6: Qualitative comparison of each method for segmentation in the aquatic environment by bird's eye view – Lake Sunapee, NH. The ASV location is marked with a circle in Cyan. The color represents each labeled obstacle except Depth clustering method. The depth clustering marks each obstacle within a boundary of orange box as the default visualization output. Ground truth has in total 4 obstacle clusters: a boat, two docks, and ground. Our method only missed the floating dock on the right side (False Negative) and segmented points far away from ground as discrete ones (False Positive). Note that False Negative case from our algorithm is considered to be a component belonging to ground as a conservative way for the navigational safety by ϵ_3 described in Section III-C. See the supplementary video for details.

segmentation for one object, we assume the largest *IoU* one as the predicted one. Note that our proposed method is just segmenting in-water objects, not differentiating each label, i.e., semantics. The main goal of segmentation is to avoid anything in the water. This is similar to other related work [6], [7], [24], which do not classify the type of objects.

Therefore, we monitor overlapping points on a segmented object with ground truth segmentation. Among the entire dataset, we extracted 10 representative sequences for evaluation. The sequences include different types of obstacles – buoy, boat, swimmer, kayak, etc. – and diverse coast lines from the ocean or lake. We manually annotated the point clouds at every frame for ground truth. The low performance values by the state-of-the-art methods can be explained by (1) the ground plane removal algorithm for a ground vehicle is not suitable for the marine domain, and (2) the clustering algorithm is not enough to encompass the sparse point clouds in water as a group. See Fig. 6 for qualitative method comparison. The total number of object clusters in the example scene are 4. The maximum dimension of the objects were on a large scale (ground: $\approx 177\,\mathrm{m}$, boat: $\approx 10\,\mathrm{m}$, and floating dock: $\approx 5\,\mathrm{m}$). They are detected by the maximum coverage of the sensor.

One of the methods closest in spirit to ours [6] showed over-segmentation on the objects after the ground plane removal – see in Fig. 6(e). This effect did not improve despite tuning the main parameters (angle threshold). Only a few partially segmented parts changed, while the overall segmentation performance did not change. Furthermore, it had difficulty in segmenting the curved boat, as described in Section III. The simplest algorithm [24] based on the Euclidean distance in 3D space also poorly segmented the sparse marine point clouds (Fig. 6(b)) due to its range parameter not covering the sparsity, and had an extremely slow performance due to its radius search procedure (Table I). The incorrect ground removal or detection cases were more noticeable in Fig. 6(c),(d). In Fig. 6(c), the ground fitting algorithm resulted in too many remaining points segmented as false separate objects by its scan-line run algorithm [7]. Note that we only qualitatively compared the learning-based method [16] with the original trained models (Fig. 6(d)). We plan to make own annotated dataset for the aquatic domain.

Column Pixel	512	1024	1800
TPR [%]	90.22	89.84	91.73
Ground Points [ea]	1563	2726	4204
Actual Ground Points [ea]	5580		
Nearest Distance Mean [m]	-0.079	-0.023	-0.025
Standard Deviation [m]	0.757	0.693	0.595

TABLE II: True Positive Ratio (TPR) and number of point clouds belonging to the ground at one time stamp as per different width for spherical projection images in our algorithm.

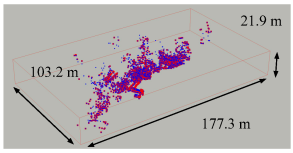


Fig. 7: Comparison of point clouds distribution between actual ground (Blue) and downsampled by using 512 pixels (red) in one time stamp.

TABLE III: Running time analysis for the proposed method on offline and online platform as per 360° full scanning.

Platform	Location	Average Running Time [ms]	Standard Deviation [ms]
Laptop	Barbados	35.26	7.98
	Lake Sunapee	35.28	14.88
Mobile	Lake Sunapee	36.84	16.60

In our experiments, we also checked the impact of downsampling the spherical projection image. The extracted and converted points from spherical images are known to have a small discrepancy with the original data [31]. The low *mIoU* performance in our method when considering a downsampled spherical image results from the "gaps", compared to the total ground truth points. Therefore, we considered another metric, the *True Positive Ratio* (*TPR*), and compared it with another downsampling factor and original size. Table II and Fig. 7 show the results. Our algorithm showed that *TPR* maintains about $90\,\%$ for all the tested pixel values. To check applicability of the converted point clouds against the raw ground truth point clouds, we measured the nearest neighbor distance between the two point cloud groups. The result shows that for all cases, the nearest distances are within the reasonable range from the ground truth point clouds, which gives a different insight on performance metrics (*mIoU*, *TPR*) for our algorithm. In other words, by using a reasonable pixel downsampling to cover the detected objects, the performance of clustering is not significantly affected.

2) Running Time Analysis: We evaluated how fast our proposed algorithm can support real-time object detection, considering that the laser scanner used returns measurements at $10\,\mathrm{Hz}$. First, we performed an ablation study. In the example sequence of Fig. 6, we ran the algorithm with or without BFS as a prior step. Among 5744 point clouds in total, the running time by using BFS as a prior step was $72\,\mathrm{ms}$ with 460 components fed to the hierarchical clustering process, whereas the running time by not using BFS was $297\,\mathrm{ms}$ with 1604 components fed to the hierarchical clustering.

Then, we analyzed the experimental run time on the collected datasets and the ASV on-board computer during navigation. The running time of the proposed algorithm is summarized in Table III. For recorded data processing, the testing computer had an Intel Core i7-7700HQ $2.80\,\mathrm{GHz}$ and $16\,\mathrm{GB}$ memory. We measured the running time to process each $360°$ scan. For ASV on-board processing, the computer is an Intel NUC with an Intel Core i7-8559U Processor $4.50\,\mathrm{GHz}$ and $16\,\mathrm{GB}$ memory. Note, a state-of-the-art method [16] in Table I only applied the monitoring field of view as $90°$ forward direction, whereas our proposed method uses the entire $360°$ scan. We also tested by applying the same monitoring field of view as [16], i.e., $90°$ thus reducing the running time into $21.19\,\mathrm{ms}$ on average with standard deviation as $13.39\,\mathrm{ms}$. Also, the method by [6] using only BFS step in Table I is approximately two times faster than ours. That method does not correctly segment in-water obstacles, as discussed in the previous subsection.

V. Discussion and Future Steps

Our proposed model-free method for in-water obstacle detection and segmentation qualitatively and quantitatively outperforms other state-of-the-art methods in the aquatic domain in segmenting obstacles. While the other methods targeted for AGV first removes a planar ground and focuses on objects only above it, our proposed algorithm, tailored to marine specific applications, considers all objects scanned by the sensor. This procedure could give the algorithm relatively more input data for segmentation than AGV's algorithms, but we optimized the algorithm with some approximation and heuristics while achieving a good accuracy performance.

We will extend the proposed LiDAR based method by fusing other on-board available sensors: GPS, IMU, RGB camera, etc. to supplement information for an ASV. We plan to increase its capability, in false positive cases of a few reflected point clouds on the water surface noted in [25] and Fig. 6, by comparing with registered data. In relation to the proposed algorithm, we will collect more operation data under adversarial conditions – wave, rain, and other inclement weather. The dataset collected by our ASV equipped with multiple sensors under diverse conditions will contribute to the robust autonomy of robotic boat's operation.

In the end, we will develop multi-object tracking, connecting it with our previous work [32] on collision avoidance planning, towards autonomous surface vehicles.

Appendix
Point clouds of in-water obstacles

To see the sparsity and instability of point clouds in marine domain, we extracted examples of in-water obstacles from our dataset and compared in two ways in Fig. 8: (1) ground domain vs. marine domain; and (2) variation of point clouds within $1\,\mathrm{s}$ time lapse under the same distance from the sensor.

Acknowledgement

We thank Monika Roznere for her help with the experiments and the Eliassen family for our access to the experimental sites. This work is supported in part by the Burke Research Initiation Award and NSF CNS-1919647, OIA1923004.

References

[1] B. Bovcon, J. Muhovič, J. Perš, and M. Kristan, "The mastr1325 dataset for training deep usv obstacle detection models," in *Proc. IROS*, 2019.

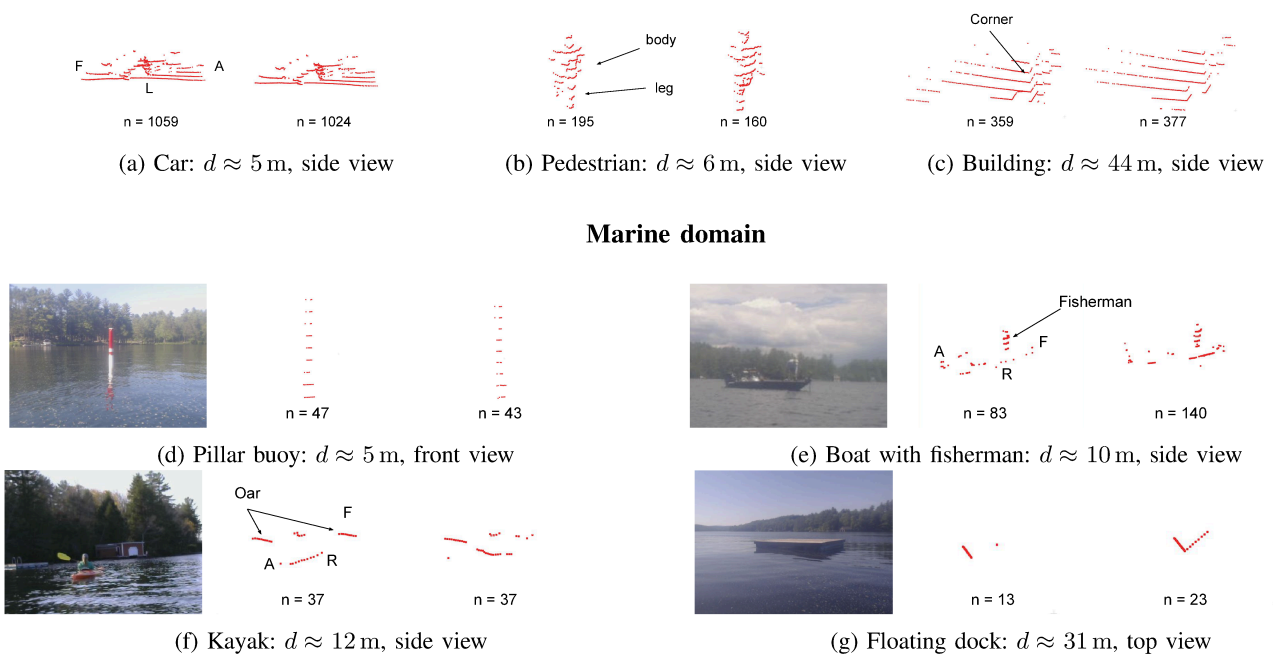
[2] E. Gundogdu, B. Solmaz, V. Yücesoy, and A. Koç, "MARVEL: A Large-Scale Image Dataset for Maritime Vessels," in *Computer Vision – ACCV*, 2016.

[3] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, 2017.

[4] J. Han, Y. Cho, J. Kim, J. Kim, N.-s. Son, and S. Y. Kim, "Autonomous collision detection and avoidance for ARAGON USV: Development and field tests," *J. Field Robot.*, 2020.

[5] B. Bovcon, J. Muhovič, J. Perš, and M. Kristan, "Stereo obstacle detection for unmanned surface vehicles by IMU-assisted semantic segmentation," *Robot. Auton. Syst.*, vol. 104, 2018.

[6] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *Proc. IROS*, 2016.



(a) Car: $d \approx 5\,\mathrm{m}$, side view (b) Pedestrian: $d \approx 6\,\mathrm{m}$, side view (c) Building: $d \approx 44\,\mathrm{m}$, side view

(d) Pillar buoy: $d \approx 5\,\mathrm{m}$, front view (e) Boat with fisherman: $d \approx 10\,\mathrm{m}$, side view

(f) Kayak: $d \approx 12\,\mathrm{m}$, side view (g) Floating dock: $d \approx 31\,\mathrm{m}$, top view

Fig. 8: Comparison of raw 3D point cloud examples by the same VLP-16 scanner between ground (a)-(c) and marine domain (d)-(g). For each object, two measurements of point clouds were also compared within $1\,\mathrm{s}$ under the same distance. Each view angle of point clouds was adjusted for best comparison regardless of the RGB image of the obstacle. d: distance of an object from the scanner, n: number of point clouds, F: fore, L: left, R: right, A: aft. Note that we used the following datasets (a): Velodyne Monterey highway[1], (b)-(c): Stevens-VLP-16 [14], (d)-(g): our own dataset.

[7] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in *Proc. ICRA*, 2017.

[8] R. Halterman and M. Bruch, "Velodyne HDL-64E lidar for unmanned surface vehicle obstacle detection," in *Unmanned Systems Technology XII*, vol. 7692, SPIE, 2010.

[9] J. R. Kidd, "Performance evaluation of the Velodyne VLP-16 system for surface feature surveying," M.S. thesis, University of New Hampshire Durham, 2017.

[10] D. Thompson, E. Coyle, and J. Brown, "Efficient lidar-based object segmentation and mapping for maritime environments," *IEEE J. Ocean. Eng.*, 2019.

[11] W. Wang, B. Gheneti, L. A. Mateos, F. Duarte, C. Ratti, and D. Rus, "Roboat: An autonomous surface vehicle for urban waterways," in *Proc. IROS*, 2019.

[12] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. CVPR*, 2012.

[13] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. ICCV*, 2019.

[14] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IROS*, 2018.

[15] 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," *Robot. Auton. Syst.*, vol. 83, 2016.

[16] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. ICRA*, 2018.

[17] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3d laser data," in *Proc. ICRA*, 2008.

[18] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," in *CVPR Workshop on Autonomous Driving*, 2019.

[19] H. Cho, Y. Seo, B. V. K. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *Proc. ICRA*, 2014.

[20] L. Xiao, R. Wang, B. Dai, Y. Fang, D. Liu, and T. Wu, "Hybrid conditional random field based camera-LIDAR fusion for road detection," *Information Sciences*, vol. 432, 2018.

[21] L. Steccanella, D. D. Bloisi, A. Castellini, and A. Farinelli, "Waterline and obstacle detection in images from low-cost autonomous boats for environmental monitoring," *Robot. Auton. Syst.*, vol. 124, 2020.

[22] M. Haghbayan, F. Farahnakian, J. Poikonen, M. Laurinen, P. Nevalainen, J. Plosila, and J. Heikkonen, "An efficient multi-sensor fusion approach for object detection in maritime environments," in *Proc. ITSC*, 2018.

[23] R. Polvara, S. Sharma, J. Wan, A. Mann going, and R. Sutton, "Obstacle avoidance approaches for autonomous navigation of unmanned surface vehicles," *Journal of Navigation*, vol. 71, no. 1, 2018.

[24] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. ICRA*, May 2011.

[25] J. Muhovič, R. Mandeljc, B. Bovcon, M. Kristan, and J. Perš, "Obstacle tracking for unmanned surface vessels using 3-d point cloud," *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, 2020.

[26] M. Jeong, M. Roznere, S. Lensgraf, A. Sniffen, D. Balkcom, and A. Quattrini Li, "Catabot: Autonomous surface vehicle with an optimized design for environmental monitoring," in *Proc. OCEANS*, 2020.

[27] K. Liu, W. Wang, R. Tharmarasa, J. Wang, and Y. Zuo, "Ground surface filtering of 3d point clouds based on hybrid regression technique," *IEEE Access*, vol. 7, 2019.

[28] International Maritime Organization, *Convention on the international regulations for preventing collisions at sea (COLREGs)*, 1972.

[29] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IROS*, vol. 3, 2004.

[30] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vision*, vol. 111, no. 1, Jan. 2015.

[31] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, *Deep learning for 3d point clouds: A survey*, 2020.

[32] M. Jeong and A. Quattrini Li, "Risk vector-based near miss and real-time obstacle avoidance for autonomous surface vehicles," in *Proc. IROS*, 2020.