Semantically-Aware Strategies for Stereo-Visual Robotic Obstacle Avoidance

Jungseok Hong[1], Karin de Langis[2], Cole Wyeth[3], Christopher Walaszek[4], and Junaed Sattar[5]

***Abstract*— Mobile robots in unstructured, mapless environments must rely on an obstacle avoidance module to navigate safely. The standard avoidance techniques estimate the locations of obstacles with respect to the robot but are unaware of the obstacles' identities. Consequently, the robot cannot take advantage of semantic information about obstacles when making decisions about how to navigate. We propose an obstacle avoidance module that combines visual instance segmentation with a depth map to classify and localize objects in the scene. The system avoids obstacles differentially, based on the identity of the objects: for example, the system is more cautious in response to unpredictable objects such as humans. The system can also navigate closer to harmless obstacles and ignore obstacles that pose no collision danger, enabling it to navigate more efficiently. We validate our approach in two simulated environments: one terrestrial and one underwater. Results indicate that our approach is feasible and can enable more efficient navigation strategies.**

I. INTRODUCTION

An autonomous robot navigating an unstructured, mapless environment typically avoids obstacles by utilizing range information received from its sensors. When a danger of collision arises, the robot adjusts its trajectory to ensure safety. The state-of-the-art techniques in obstacle avoidance effectively prevent collisions, but they are unaware of the semantic properties of the obstacles they are avoiding [1]. Consequently, they are unable to exploit a more in-depth understanding of the objects in the environment to navigate more efficiently without compromising safety. In this paper, we augment an obstacle avoidance module's capabilities by incorporating semantic information and show experimentally that this results in more efficient navigation. We refer to this augmentation as Semantic Obstacle Avoidance for Robots (SOAR).

Semantic information is often used to aid robot navigation in structured environments [2], [3]. In this work, we propose that semantic information can also be useful for obstacle avoidance in unstructured environments. For example, a robot may want to consider that a living thing, like a person or a dog, has the potential to start moving, even if it is currently stationary. The robot may also want to consider that some obstacles may not actually pose a collision danger: for instance, a depth map may detect plastic balls in the robot's direction of motion, but the robot can safely collide with the balls. When the robot is able to recognize that different objects pose different collision dangers, it can choose a path that maximizes efficiency without jeopardizing safety. In fact, semantic obstacle avoidance is desirable in several robotic applications with unstructured environments, including:

- **Robotic wheelchairs** need to generally stay clear of obstacles, but they may want to give extra clearance to objects like doors that have the potential to suddenly swing forward. On the other hand, if the user wants to dock at a table, the wheelchair needs to allow itself to get very close to it.
- **Autonomous underwater vehicles** (AUVs) are employed by marine biologists to observe endangered species of mussels, which are usually embedded within rock formations, requiring AUVs to get much closer.
- **Diver-following AUVs** need to avoid obstacles while recognizing bubbles emanating from the diver's flippers do not actually pose a collision danger.

It is imperative to note that even if a map of the environment is available, the presence of dynamic objects (*e.g.*,

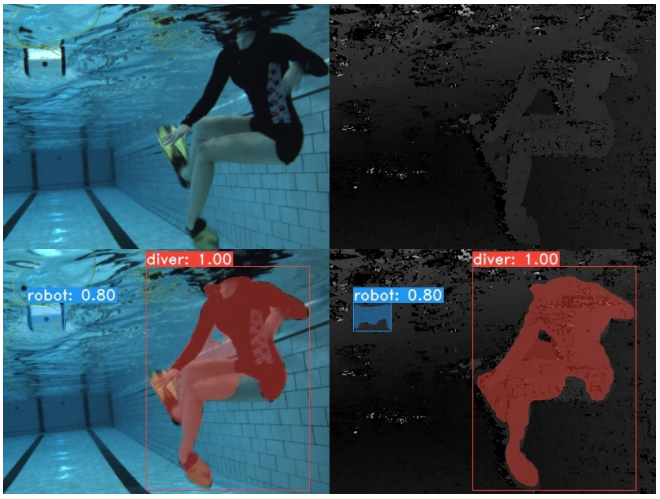


Fig. 1: Four levels of scene understanding in underwater human-robot collaborative missions, shown top-to-bottom, left-to-right: observation with no understanding (RGB image), distance-aware observation without scene understanding (depth estimation), instance-aware scene understanding (instance segmentation), and depth-instance-aware scene understanding (depth-instance segmentation). We propose a depth-instance-aware approach for obstacle avoidance.

The authors are with the Department of Computer Science and Engineering, Minnesota Robotics Institute, University of Minnesota–Twin Cities, 100 Union St SE, Minneapolis, MN, 55455, USA. {[1]jungseok,[2]dento019,[3]wyeth008, [4]walas013,[5]junaed}@umn.edu.

*This work was supported by the US National Science Foundation awards IIS-#1637875 & IIS-#1845364, the UMII-MnDRIVE Fellowship, the MnRI Seed Grant, and Nvidia GPU Grant.

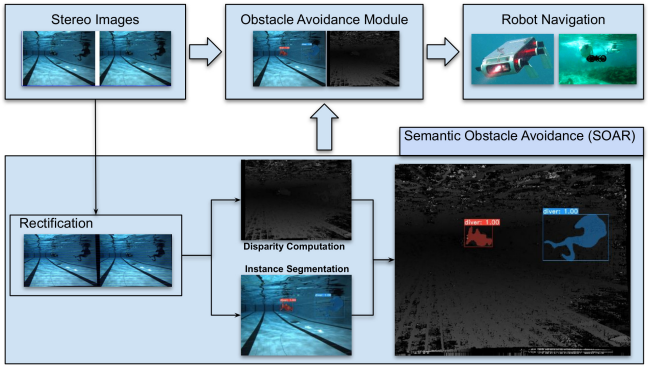


Fig. 2: Illustration of our obstacle avoiding approach which fuses depth and semantic information for selective avoidance. The input is a pair of stereo images, which is used to both compute disparity and, through YOLACT, generate semantic labels for obstacles in the scene. Fusing these, a robot has both depth estimates and semantic information about potential obstacles, enabling it to select navigation strategies depending on the nature of the obstacle.

in the above scenarios, people, wheelchairs, and fish) will require information beyond the "free-and-occluded" labels that are usually provided by maps. Semantic maps [2] and semantic scene understanding [4], [5], [6], [7] may provide additional information, but they would not account for dynamic objects in the environment. For field robots, particularly in sensory-deprived environments (*e.g.*, underwater), such maps are often nonexistent, and real-time scene understanding is still an open problem.

SOAR uses instance segmentation, which identifies specific object instances for each pixel in the visual scene, and fuses it with depth (*i.e.*, distance) information to provide semantically-aware obstacle information to obstacle avoidance modules. We adopt YOLACT [8] as the instance segmentation module for our pipeline, as shown in Fig. 2, because it is the first state-of-the-art model to run in real-time with reasonable accuracy. While useful, object detection is not appropriate for this task as bounding boxes generated by these algorithms will contain spurious information from scene background and other objects. While semantic segmentation approaches are useful, they do not discriminate between object instances, and this information is needed for the proposed approach of semantically-aware object avoidance.

In this paper, we make the following contributions:

- Propose a pipeline for combining depth estimation and instance segmentation for semantic obstacle avoidance,
- Develop a semantically-aware obstacle avoidance algorithm to keep flexible distances from objects,
- Create an instance segmentation dataset of underwater obstacles to train an instance segmentation model, and
- Demonstrate the efficiency of the proposed pipeline in both underwater and terrestrial simulated environments.

II. RELATED WORK

A. Instance Segmentation

Research in object detection has studied models to improve accuracy while keeping real-time inference speed since the appearance of YOLO [9], one of the first real-time object detection models. However, instance segmentation poses more complex challenges, and achieving good accuracy in real-time has been difficult. FCIS [10] is the first end-to-end CNN-based model for instance segmentation. It is built on R-FCN [11] and utilizes position-sensitive inside/outside score maps to generate instance segmentation proposals. Mask R-CNN [12], which is an extension of Faster R-CNN [13], performs segmentation in a two-stage process by generating Region of Interest (RoI) proposals first and then creating a mask based on the RoI from the first stage. PANet [14] improves the accuracy of segmentation from Mask R-CNN by enriching information propagation. MS R-CNN [15] outperforms Mask R-CNN by adding a *MaskIoU head* to align the scores of the masks. Although the aforementioned models show accurate results, their two-stage-based structures make real-time instance segmentation infeasible. In order to overcome the structural problem, YOLACT [8] conducts two predictions in parallel: mask prototypes and per-instance mask coefficients. Then, the predictions are combined linearly to yield masks. This allows a single-stage structure and inference in real-time with reasonable accuracy.

B. Obstacle Avoidance

Obstacle avoidance, unsurprisingly, has seen significant development (*e.g.*, [16], [17], [18]) given its importance in safe robot navigation. Here, we focus specifically on sensor-based approaches where no information about the environment is available beyond what is received from sensors (see [1] for a complete discussion). Sensor-based approaches typically plan a short-horizon trajectory at every time step [19], [20]. A classic obstacle avoidance technique is the Artificial Potential Field, first proposed by [21]. This technique assigns artificial repulsive fields to obstacles and attractive fields to goals, thereby guiding the robot toward a goal while simultaneously avoiding obstacles. Other approaches include vector field histograms (VFH) [22], receding horizon control [23], and Voronoi diagrams [24].

Most obstacle avoidance that incorporates semantic information is focused on developing socially-aware responses to human obstacles, *e.g.*, [25], [26], [27]. Similar to our approach, [26] instructs the robot to avoid humans more than inanimate objects. However, their work is focused on path planning in mapped environments and uses model-based methods to estimate the human's location.

Another approach for obstacle avoidance in marine robotics, based on conditional imitation learning, is presented in [18]. This approach uses data collected from expert users to learn what navigational action to take given an input image, but does not explicitly model different behaviors for different types of obstacles.

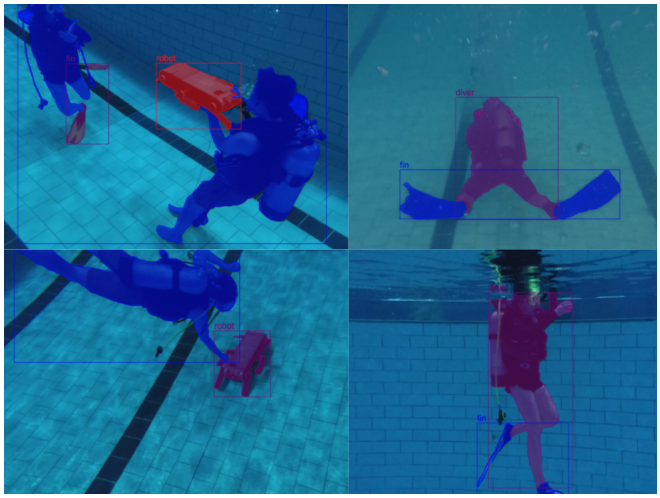


Fig. 3: Examples of labeled training images from our underwater dataset showing three classes: *diver, robot*, and *fin* (the colors for each class are randomly selected per image).

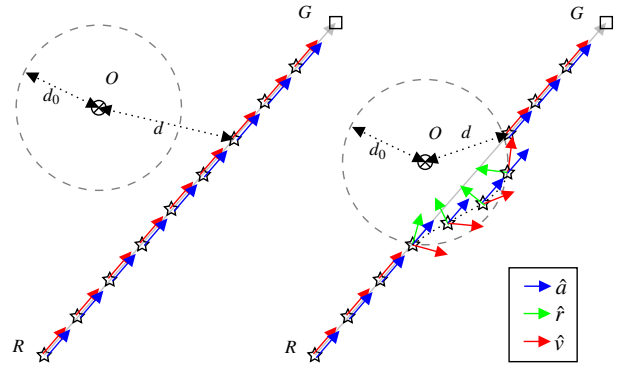


Fig. 4: Example trajectories of a robot around an obstacle using the proposed approach. The repulsive potential $\hat{r}$ affects the robot only if $d \leq d_0$. The position of object O is represented by a cross, the goal position G by a square, and the position of robot R over time with stars. (left): when the obstacle is not along the direct path from the robot to the goal, it does not affect robot navigation. (right): the robot's direct path to the goal brings it closer than d_0 distance to the obstacle; our algorithm forces the robot to circumnavigate around it.

III. METHODOLOGY

The proposed approach incorporates both instance segmentation and depth information to intelligently avoid obstacles in unstructured and dynamic environments, with the goal to optimize robot paths (*e.g.*, in terms of distance traveled, energy spent, and time taken) without compromising obstacle avoidance capabilities. The system obtains object labels and pixel locations from instance segmentation and fuses the information with depth information to assign clearance distances to obstacles.

A. Information Fusion using a Stereo Camera

We use a stereo camera as our only sensor to acquire (1) pixel-level masks and labels of each object using instance segmentation, and (2) depth information. Once both are acquired, we fuse them to provide semantic information to an obstacle avoidance module (see Fig. 2).

1) Instance Segmentation with Transfer Learning: We choose YOLACT [8] as the base instance segmentation module due to its real-time inference and competitive accuracy. We use ResNet50-FPN as a backbone network for achieving maximum inference speed since robots are likely to use low-power computation units (*e.g.*, an Nvidia Jetson TX2) to perform semantic inference. We collect a total of 2,263 images, of which 1,682 are labeled with *diver*, *robot*, and *fin* (*i.e.*, diver's flippers) classes; see Fig. 3 for training images labeled using the Supervisely [28] tool. In addition, we use 581 images from the SUIM dataset [29] which has *diver*, *fish*, and *robot* classes. We refine a pre-trained YOLACT model, initially trained with the MS COCO dataset [30], with this additional data.

2) Depth Estimation: Our depth estimation process is as follows:

1) We perform stereo rectification to obtain a transform matrix **R**, projection matrix **P**, and disparity-to-depth mapping matrix **Q** for each camera using a camera matrix **K** and distortion parameters **D** from each camera, a rotation matrix **R** between the first camera coordinate and the second camera coordinate, and a translation vector **T** between two cameras.
2) Next, we remove distortion from each image using the **K**, **D**, **R**, and **P** matrices.
3) After rectifying each pair of images, we run stereo matching to generate a disparity map.
4) Lastly, we estimate the depth information from the disparity map using **Q**.

B. Obstacle Avoidance

Our obstacle avoidance algorithm is inspired by the Artificial Potential Field (APF) method [21]. APF uses an attractive potential f_a to guide the robot toward the goal and a repulsive potential f_r to push the robot away from obstacles. The attractive potential is calculated as:

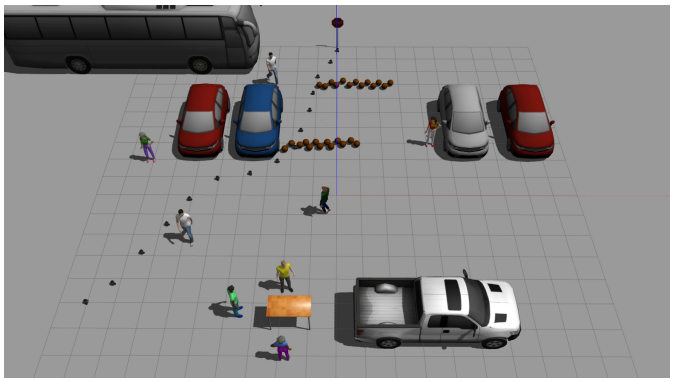
$$f_a(x) = c(|x - x_g|)^2 \tag{1}$$

Here c is a scaling constant, x is the current robot position, and x_g is the goal position. The repulsive potential is calculated as

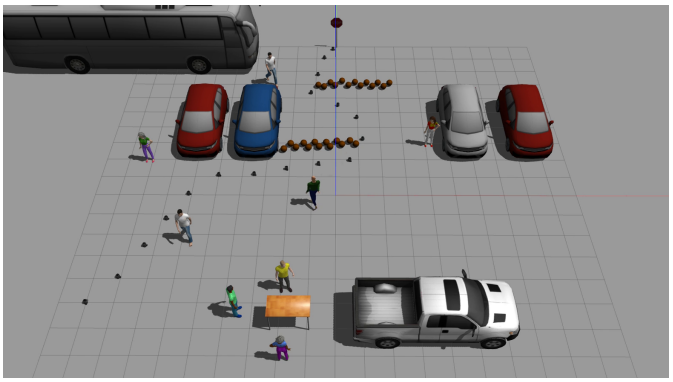
$$f_r(x) = \begin{cases} \eta\left(\frac{1}{p(x)} - \frac{1}{d_0}\right)^2 & \text{if } p(x) \leq d_0 \\ 0 & \text{if } p(x) > d_0 \end{cases} \tag{2}$$

where η is a constant, $p(x)$ is the closest obstacle to the position x, and d_0 is the largest distance from the obstacle at which the robot can sense the obstacle's repulsive force.

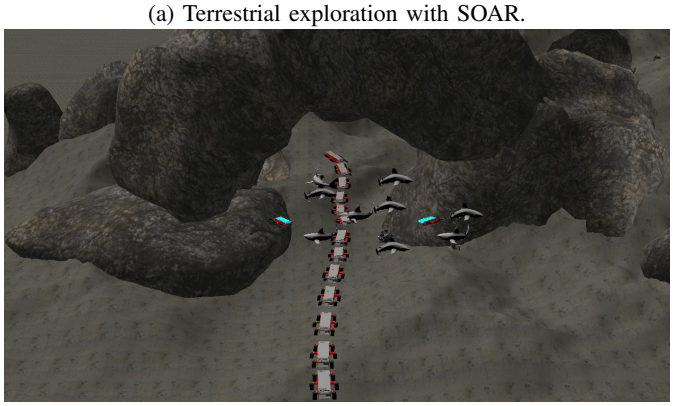
In our approach, we determine d_0 based on semantic information about the obstacles. For instance, we assign $d_0 = 0$ for objects we can ignore (*e.g.*, bubbles or sports balls) while assigning a larger value for the objects (*e.g.*, coral reef, robots, people) we intend to avoid. In our approach,



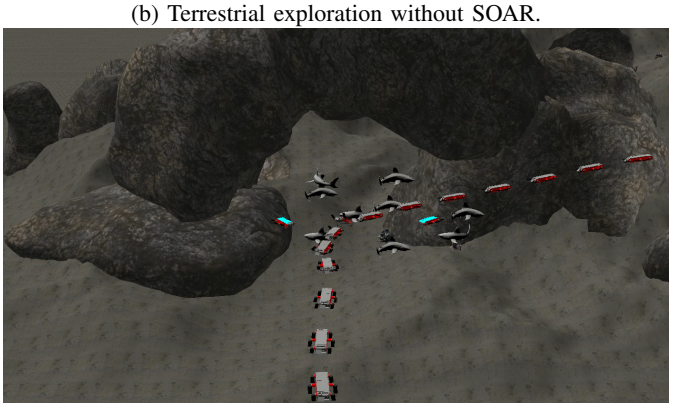
(a) Terrestrial exploration with SOAR.



(b) Terrestrial exploration without SOAR.



(c) Underwater exploration with SOAR.



(d) Underwater exploration without SOAR.

Fig. 5: Samples from the terrestrial and underwater simulation tests. In Fig. 5a and 5b, the Turtlebot moves from the bottom left corner to the upper middle area. In Fig. 5c and 5d, Aqua moves from the lower middle point to the middle of the arch. Fig. 5a and 5c show that SOAR finds more efficient paths to explore environments while avoiding significant obstacles.

unlike APF, the robot navigates around the boundary of an obstacle, keeping a constant distance of d_0 from the obstacle (Fig 4). The circumnavigation behavior is similar to the bug-2 navigation algorithm [31]. The robot, however, may face the challenge of maintaining fixed distances (*i.e.*, d_0) from obstacles when circumnavigating due to errors in state estimation and external forces (*e.g.*, surge for underwater robots operating in open waters and wind for aerial robots).

We introduce two unit vectors, $\hat{a}$ and $\hat{r}$, to implement the circumnavigation with the concept of attractive and repulsive potentials from APF. $\hat{a}$ points from the robot towards the goal and $\hat{r}$ points from the robot to the obstacle. With the two vectors, we update the robot's direction of movement $\hat{v}$ at any given point as follows:

$$\hat{v} = \begin{cases} \hat{a} & \text{if } |x_o - x| > d_0 \\ \hat{a} + c_1 c_2 \hat{r} & \text{if } |x_o - x| \leq d_0 \end{cases} \tag{3}$$

where c_1 is defined as a negative dot product between $\hat{a}$ and $\hat{r}$, and c_2 is an additional factor to keep the distance d_0 between the robot and the obstacle.

$$c_1 = -\hat{a} \cdot \hat{r} \tag{4}$$

We introduce the constant c_1 to make $\hat{v}$ perpendicular to $\hat{r}$ when the distance between the robot and the obstacle is d_0. However, due to the robot's momentum, the robot may still approach closer than d_0 to the obstacle. We use the additional factor c_2 to scale the repulsive component $c_1\hat{r}$ and to enforce distance d_0 from the obstacle:

$$c_2 = \frac{1-b}{d_0}|x_o - x| + b \tag{5}$$

Here x_o is the obstacle position, x is the robot position, and b is a constant greater than 1 that represents the maximum value $c_1\hat{r}$ can be scaled by. c_2 scales inversely with the robot's distance to the obstacle, and obtains a value between 1 and b as we approach the obstacle.

IV. EXPERIMENTS AND RESULTS

The ongoing COVID-19 pandemic prohibited field trial validations of the proposed method. However, we use realistic simulation using ROS Gazebo [32] worlds for both terrestrial and underwater cases to validate our algorithm. Because our goal is local navigation of unstructured, mapless environments, we choose goal points for each case with one condition: the goal point should be something visible to the robot when the robot is at its starting position if there is no obstacle between the robot and the goal points. Additionally, we use a mobile GPU (Nvidia Jetson TX2) with a stereo camera (Intel RealSense) to evaluate the performance of our model to mimic realistic robotic hardware.

TABLE I: Instance segmentation results (mAP) trained on our underwater dataset

	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
box	71.14	91.96	90.29	89.48	87.89	84.09	79.54	75.20	62.56	39.07	11.29
mask	69.38	93.72	93.51	92.14	90.47	85.64	81.71	69.39	55.94	29.55	1.71

A. *Simulated Terrestrial Trials*

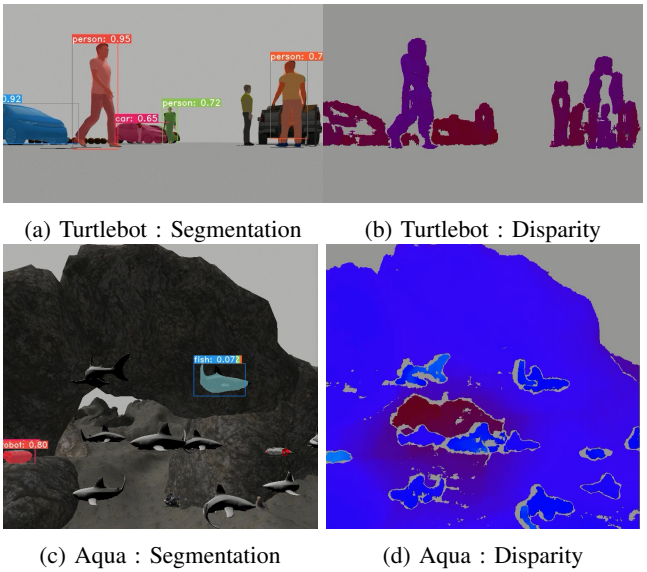
We have created a terrestrial world in Gazebo, simulating a parking lot environment. The scene was chosen to mimic a robot attempting delivery or curbside pickup from a departmental store, a relatively common occurrence in many parts of the world under the Coronavirus pandemic. The world has various types of objects, including *sports balls, cars, buses, people walking* or *standing*, and *tables*. We evaluate our model on a workstation equipped with an Intel i5-8600K CPU and an Nvidia GTX 1080 GPU. We add a stereo camera to a simulated Turtlebot robot to estimate depth and infer instance segmentation. To validate our algorithm, we intentionally block the shortest path from a robot to a goal point with *sports balls*, as shown in Fig. 5a and 5b, since we select the *sports ball* category as an object the robot can safely run into (*i.e.*, a *not-an-obstacle* object). The Turtlebot starts from the bottom left corner of the world and aims to reach the stop sign at the top. We use pre-trained COCO weights with YOLACT for instance segmentation. We use the modified APF-based obstacle avoidance algorithm as described in Section III under two conditions: receiving semantic information about obstacles (SOAR) and without receiving any semantic information about obstacles (non-SOAR). To measure the effectiveness of the semantic obstacle avoidance approach, we measure the travel time from a starting point to a goal point to evaluate each model's performance by running 10 tests for each case. We also note the path chosen by the robot in each of the SOAR and non-SOAR cases.

B. *Simulated Underwater Trials*

Our underwater world has been designed to mimic the ocean floor environment, including corals, rocks, and fauna. The world includes *fish, robots*, and an *underwater rock arch formation*. As shown in Fig. 5c and 5d, the *arch* is blocked by *fish* and *robots* to test the efficacy of the SOAR approach compared to non-SOAR. We select the *fish* category as a *not-an-obstacle* object class. We simulate the Aqua AUV [33], equipped with a stereo camera, to test our model with both SOAR and non-SOAR algorithms. The robot starts from the bottom of the world and aims to travel through the arch to the other side of the rock formation. We use the instance segmentation model trained on our own dataset (as mentioned in Section III-A.1) with the four classes (*i.e.*, *diver, robot, fish*, and *fin*). The hardware and trial configurations are unchanged from that of the terrestrial case.

C. *Results*

We use both quantitative and qualitative evaluation to test our semantic obstacle avoidance approach, as shown in Fig. 5 and Table II. Vision algorithms are implemented using the OpenCV [34] library.



(a) Turtlebot : Segmentation (b) Turtlebot : Disparity

(c) Aqua : Segmentation (d) Aqua : Disparity

Fig. 6: Instance segmentation and disparity estimation from the view of robots. Fig. 6a and 6b are captured from the Turtlebot in the terrestrial world, and Fig. 6c and 6d are from the Aqua robot in the underwater world.

1) Instance Segmentation: We train the instance segmentation model on a dataset of underwater imagery we collected, and also one that is openly available, as mentioned in Section III-A.1. We train for $800,000$ epochs which took five days on an Nvidia Titan XP GPU. Table I shows the average mAP (Mean Average Precision) score over IoU (Intersection over Union) thresholds from 0.50 to 0.95. With higher accurate localization (higher IoU thresholds), the mAP values decrease. Overall, the mAP values of bounding boxes are slightly higher than those of segmentation masks.

2) Simulated Terrestrial Trials: We achieve ≈ 20 *fps* while simultaneously running simulation and algorithms on the Nvidia Titan XP GPU. Table IIa shows the average travel time (in simulation time) from the starting point to the goal for both SOAR and non-SOAR algorithms. Although both algorithms can reach the goal, the non-SOAR algorithm takes 14% longer time than when using SOAR. Samples from each case are shown in Fig. 5a and 5b. Fig. 6a and 6b show how Turtlebot understands scenes during its exploration. The SOAR algorithm reduces travel time by utilizing non-obstacle information (*i.e.*, *sports ball*) obtained from the instance segmentation model. This demonstrates how instance segmentation information can assist in efficient exploration while safely avoiding obstacles. Information from bounding box detection and *semantic* segmentation is

TABLE II: Terrestrial and Underwater Simulation Trial Results

	SOAR		non-SOAR	
	Travel time(s)	Goal	Travel time(s)	Goal
1	89	✓	99	✓
2	93	✓	98	✓
3	83	✓	120	✓
4	90	✓	99	✓
5	87	✓	100	✓
6	85	✓	98	✓
7	90	✓	101	✓
8	88	✓	97	✓
9	87	✓	98	✓
10	89	✓	101	✓
Avg	88.1		101.1	

(a) Turtlebot

	SOAR		non-SOAR	
	Travel time(s)	Goal	Travel time(s)	Goal
1	74	✓	97	✗
2	69	✓	75	✗
3	73	✓	88	✗
4	70	✓	120	✗
5	69	✓	91	✗
6	68	✓	94	✗
7	70	✓	122	✗
8	72	✓	111	✗
9	70	✓	111	✗
10	74	✓	99	✗
Avg	70.9		100.8	

(b) Aqua

not sufficient to provide detailed information for a robot to explore environments, particularly with *intra-class occlusion.*

3) Simulated Underwater Trials: As our Gazebo world uses detailed hydrodynamic effects for the underwater simulation, we obtain ≈ 10 *fps* during our tests. Unlike the terrestrial case, the non-SOAR algorithm fails to reach the goal, as seen in Table IIb. This is because we terminate the non-SOAR algorithm in the following cases: 1) Aqua is heading in the wrong direction, 2) Aqua is stuck between rocks, and 3) it takes too long (≥ 2 minutes) to reach the goal. Sample cases from each scenario are captured in Fig. 5c and 5d. The cyan-colored robots shown are "obstacles", and immobile, to simulate a moving Aqua robot (silver/red colors) avoiding other robots in the field. With the SOAR algorithm, Aqua is able to reach the goal (*i.e.*, under the *arch*) by ignoring the group of fish. The system ignores them because of the semantic knowledge that fish do not present a collision danger. Fig. 6c and 6d show a snapshot of Aqua's view from the trials. The results show that for underwater domains, obstacle avoidance without understanding the scene could significantly extend the travel time at best and fail to reach the goal at worst.

4) Bench Test: We achieved ≈ 5 *fps* inference speed while running the SOAR algorithm on the Jetson TX2. We expect to achieve faster inference speed in more capable mobile platforms (*e.g.*, AGX Xavier).

5) Limitations: When the inference produced by instance segmentation incorrectly classifies an obstacle as a non-obstacle, a collision can occur. Additionally, the inference time needs to be sufficiently fast to capture the objects' motion. In other words, if an object moves far faster than the inference speed, it could cause the obstacle avoidance algorithm to fail.

V. CONCLUSIONS

In this paper, we propose an obstacle avoidance algorithm that incorporates both instance segmentation and depth information to perceive its surroundings from only a pair of stereo image as input. We are able to use the instance segmentation labels to inform a robot about which visible obstacles in its environment should be either avoided or ignored. Finally, we present the SOAR algorithm as a viable way to explore unstructured environments with the obtained visual information. We validate our algorithm on both terrestrial and underwater simulations; quantitative results show that our algorithm can lead to efficient and intelligent robotic navigation decisions in unstructured environments, which can result in extending the duration of robot operations.

We plan to extend this work in multiple directions. First, we intend to integrate a sonar sensor with the camera to make SOAR robust to poor visibility conditions. Sonar readings will be used to provide additional information about obstacle locations. Visual data will be exploited to fine-tune a robot's motion when the robot approaches obstacles in close proximity. Additionally, we will improve instance segmentation by reducing the model size and inference time; training the model with increasing the size of the dataset; and optimizing it. Lastly, we plan to implement of the proposed algorithm on a robotic platform for actual in-the-field validation.

REFERENCES

[1] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[2] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[3] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[7] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *Advances in Neural Information Processing Systems*, 2015, pp. 1990–1998.

[8] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: real-time instance segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9157–9166.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2359–2367.

[11] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[14] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[15] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring R-CNN," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6409–6418.

[16] S. Rahman, A. Q. Li, and I. Rekleitis, "SVIn2: An Underwater SLAM System using Sonar, Visual, Inertial, and Depth Sensor," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1861–1868.

[17] M. Xanthidis, N. Karapetyan, H. Damron, S. Rahman, J. Johnson, A. O'Connell, J. M. O'Kane, and I. Rekleitis, "Navigation in the Presence of Obstacles for an Agile Autonomous Underwater Vehicle," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 892–899.

[18] T. Manderson, J. C. G. Higuera, S. Wapnick, J.-F. Tremblay, F. Shkurti, D. Meger, and G. Dudek, "Vision-Based Goal-Conditioned Policies for Underwater Navigation in the Presence of Obstacles," in *Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[19] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[20] M. Hoy, A. S. Matveev, and A. V. Savkin, "Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1253–1266, 2012.

[21] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. Springer, 1986, pp. 396–404.

[22] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[23] P. Ögren and N. E. Leonard, "A provably convergent dynamic window approach to obstacle avoidance," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 115–120, 2002.

[24] E. Masehian and M. Amin-Naseri, "A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning," *Journal of Robotic Systems*, vol. 21, no. 6, pp. 275–300, 2004.

[25] A. F. Foka and P. E. Trahanias, "Probabilistic autonomous robot navigation in dynamic environments with human motion prediction," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 79–94, 2010.

[26] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[27] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3931–3936.

[28] Supervisely - Web platform for computer vision. Annotation, training and deploy. [Online]. Available: https://supervise.ly/

[29] M. I. Robotics and V. Laboratory, *Segmentation of Underwater Imagery Dataset*, http://irvlab.cs.umn.edu/resources. Accessed 10-31-2020.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[31] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403–430, 1987.

[32] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009. [Online]. Available: http://pub1.willowgarage.com/ konolige/cs225B/docs/quigley-icra2009-ros.pdf

[33] G. Dudek, P. Giguere, C. Prahacs, S. Saunderson, J. Sattar, L.-A. Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Ripsman, *et al.*, "Aqua: An amphibious autonomous robot," *Computer*, vol. 40, no. 1, pp. 46–53, 2007.

[34] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.