GPU-Assisted Learning on an Autonomous Marine Robot for Vision-Based Navigation and Image Understanding

Travis Manderson[1] and Gregory Dudek[2]

***Abstract*— We present a GPU-based integrated robotic platform that enables collision avoidance, navigation, and image understanding on a single underwater vehicle. The platform enables observational tasks such as coral reef health assessment by enabling simultaneous operation of multiple image analysis taskswhile navigating in close proximity to obstacles. The integration of a GPU allows us to leverage deep neural networks for collision avoidance and automated object detection and classification while a general purpose CPU processes images to perform visual Simultaneous Localization and Mapping (SLAM). In this paper, we describe the system architecture and summarize experimental results for coral detection and collision-free navigation.**

I. INTRODUCTION

In this paper, we describe the software architecture and associated hardware for a lightweight high-performance robotic system that depends on real-time computer vision. Our vehicle is a GPU-based integrated platform that enables collision avoidance, navigation, and image understanding on a single vehicle. The platform enables observational tasks such as coral reef health assessment by enabling simultaneous operation of multiple automated detection and classification tasks along visual SLAM. Many state-of-the-art marine applications can leverage deep networks for sensor processing. In our work, we focus on image processing using Deep Convolutional Neural Network (DCNN) for environmental assessment in shallow water where vision-based navigation, data analysis, and measurement solutions can have major advantages over other sensing modalities in terms of both performance and overall system efficiency where classical methods fall short [1]. On the other hand, the disadvantage of using computer vision as the primary sensing modality is the challenging nature of carrying out computer vision underwater and the computational cost associated with it.

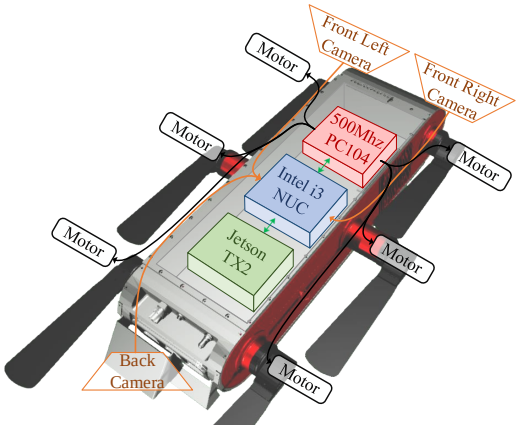

Fig. 1: Overview of the Aqua robot and functional block diagram showing its main computers, cameras, and motors.

Mobile Robotics Laboratory, School of Computer Science, McGill University, Montreal QC, Canada
{[1]travism,[2]dudek}@cim.mcgill.ca

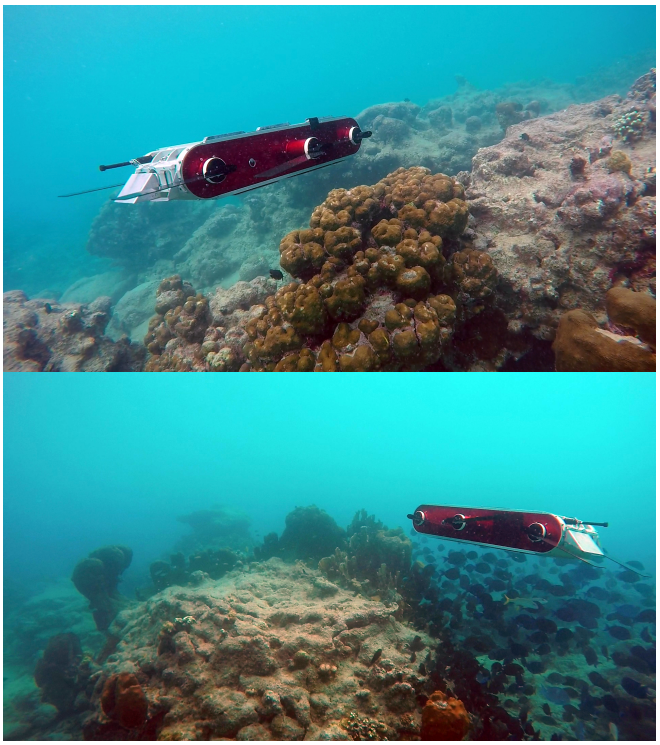

Fig. 2: Aqua robot swimming in close proximity to coral reef while avoiding collisions.

The experimental platform we use is a novel variant of the Aqua class of robots [2], [3] configured with two forward-facing cameras and a downward-facing camera (see Figure 1) as well as an integrated GPU compute module (NVidia Jetson TX2) that performs inference on the DCNN. With this platform, we can simultaneously perform three tasks in real-time: 1) collision avoidance, 2) localization, and 3) coral detection. Figure 2 shows Aqua swimming while performing collision avoidance and target selection on the GPU.

The ability of our platform to perform this unique combination of tasks enables us to track coverage while avoiding obstacles and coral-depleted regions and focus on areas that are richer in coral. In our previous work, we demonstrated the ability to perform automatic coral detection by combining extracted texture information with a Support Vector Machine

to predict the presence of coral in an image region [4]. Although this technique worked well for the sole task of automatic coral detection, its computational requirements are ill-suited to run in real-time alongside other vision tasks on the same CPU. With recent advances in miniaturized GPU compute modules, automatic coral detection can be performed on more flexible models based on DCNNs.

The NVidia Jetson TX2 currently provides the best combination of low-power and performance in an embeddable subsystem, making it appropriate for integration into the Aqua platform where, like many small submersibles, physical space and power are both at a premium. Our approach is to first use the deep learning framework TensorFlow [5] deployed on the TX2 GPU for vision-based inference using a DCNN. We then pass the results off to an integrated Intel Core i3 general purpose processor running the Robot Operating System (ROS) for navigation and high-level planning, with low-level control, motor feedback, fail-safe operations, and other tasks delegated to an additional processor running the real-time Xenomai extension to the Linux operating system.

II. Related Works

Many underwater vehicles exist, such as the Seaglider [6], Spray [7] and Nereus [8]. As noted in [9], however, much of the focus of Autonomous Underwater Vehicless (AUVs) design has been in the area of reliability, endurance, and safety. The design of the Aqua robot puts particular emphasis on extreme maneuverability, portability (such that it can be deployed by a single operator) and vision-based computation. This work does not elaborate on the mechanics of the robot, but instead focuses on the computational and software characteristics as it pertains to the vision pipeline.

The ability to localize underwater is an important capability of any submersible vehicle. The size, weight, and power limitations of the Aqua vehicle make it particularly challenging (i.e. unappealing) to use bulky exteroceptive sensors, such as sonar or ultra-short baseline acoustic positioning. Visual Odometry (VO) has the promise of using only cameras to perform localization, which is ideal due to the simplicity of this approach in terms of hardware. It is, however, susceptible to the lighting, texture, and turbidity conditions of the surrounding environment. Several researchers have focused on improving underwater SLAM [10], [11], [12], and in our own work, we have examined and evaluated several existing and state-of-the-art VO systems including ORBSLAM [13] and Direct Sparse Odometry (DSO) [14] for underwater use. Most recently, we focus on the use of

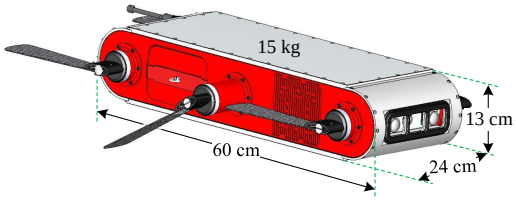

Fig. 3: CAD drawing of the Aqua robot.

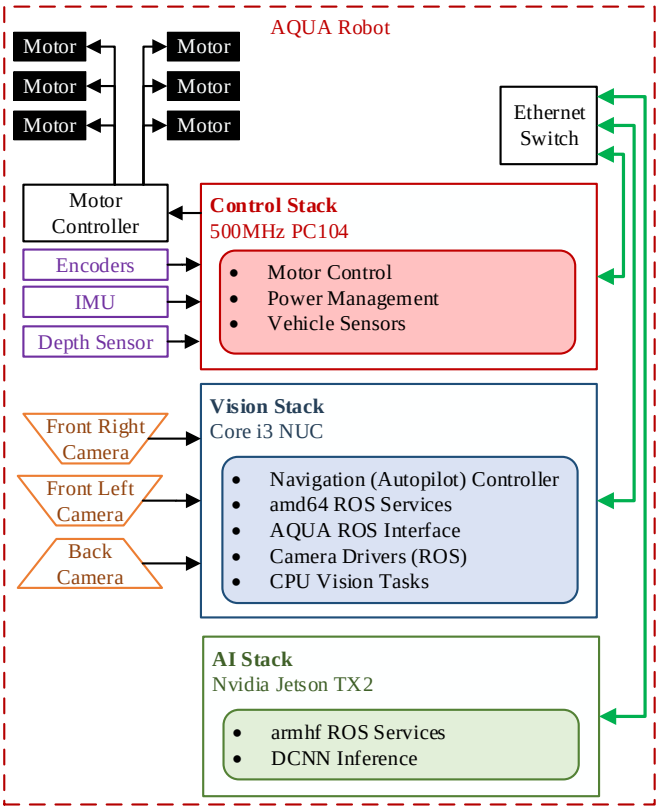

Fig. 4: Block diagram of the Aqua robot components showing the connection of the motors, internal sensors, and cameras to the robot computers, as well as an overview the computational components and their networking.

DSO using the downward-facing camera due to the planar motion when the robot is swimming near the sea floor.

Our overall goal is to robustly navigate and avoid obstacles using a purely vision-based approach that relies on human experts to provide examples of good navigation. Early work in this area was done by Pomerleau [15], who used examples generated in simulation to predict steering angles for autonomous driving of a van. Since then, several researcher have used similar techniques on real-world data to train autonomous controllers for self-driving cars [16] and aerial vehicles [17]. Our collision avoidance and navigation approach bears resemblance to these techniques, but differs in that we first collect a set of individual images, and then hand label them with the expected action.

III. Hardware and Software Architecture

The Aqua and Aqua2 robot families are comprised of several similar models of hexapod robots that are intrinsically able to either swim or walk on land when suitably configured. The Aqua models used in this paper are research vehicles developed at McGill University, while the Aqua2 models are commercial variants built and sold by Independent Robotics. In this work, we employ a model (Aqua-5) of the Aqua robot (as shown in Figure 3) - a 15 kg autonomous underwater hexapod that is 64 cm long, 24 cm wide, and 13 cm high, and uses six flippers to swim and perform 6DoF maneuvers.

To capture images, it contains two USB 3.0 IDS Imaging UI-3251LE global-shutter cameras in the front (configured as a stereo pair) and one in the back configured to look downward using an external mirror. Each camera has a 4 mm lens resulting in an underwater field of view of 107° horizontal and 91° vertical. We typically down-sample each camera image from its native 1600 pixels wide and 1200 pixels high resolution to 800 pixels wide and 600 pixels high resolution.

Figure 4 shows a block diagram of the three internal computers inside Aqua along with their connections to the motors, internal sensors, and cameras. The robot has gigabit Ethernet that networks all three computers together through a switch.

A. Control Stack

The Control Stack is a 500 MHz PC104 computer that runs Linux and is responsible for real-time control and gait selection. We use a variant of Linux enhanced with the Xenomai framework to enable hard real-time scheduling and control. Xenomai enhances the native POSIX kernel interface with a supplementary API for real-time scheduling, which we use for precise motor control and sensor selection.

A 1 kHz control loop is responsible for reading the depth sensor, Inertial Measurement Unit (IMU) (Microstrain 3DM-GX1), and motor encoders as well as running the low-level PID motor controller. Each of the six flippers can be independently controlled and normally involves using periodic motions of each flipper and setting its motion parameters with respect to an offset angle, oscillation amplitude, and frequency, which results in a thrust vector at each hip (ie, the attachment point between the motor and flipper). This approach allows for a variety of gaits and provides the robot with the ability to perform many behaviors, such as hovering in place or executing a complex trajectory.

B. Vision Stack

The Vision Stack is a general purpose Intel i3 based computer (Intel NUC6i3SYH) that is primarily responsible for communicating to the Control Stack, running high-level navigation tasks, and capturing images from the camera. The use of an i3 CPU provides an attractive tradeoff between computational performance and power dissipation. This computer runs the Ubuntu operating system with ROS services. Custom ROS nodes are used to expose the robot state from the Control Stack using UDP communication as well as send gate commands to the motor controller. Additionally, images are captured from the three cameras and published using ROS image transport, which are available to subscriber nodes on either the Vision Stack itself or the AI Stack. An autopilot controller node is used to receive target orientation, depth, and velocity messages that control the motion of the robot by sending commands to the Control Stack [18]. Additional computation on the Vision Stack can be used for CPU-orientated vision tasks or high-level planning algorithms.

In our work, we use the Vision Stack to run a modified version of the DSO algorithm to perform VO [19]. DSO runs on two threads and is better suited to run on the CPU than the GPU compute module. Our cameras normally run at 10 or 15 FPS, which allows all our image processing tasks to run in real-time while Aqua swims at speeds between 0.5 and 1 m/s.

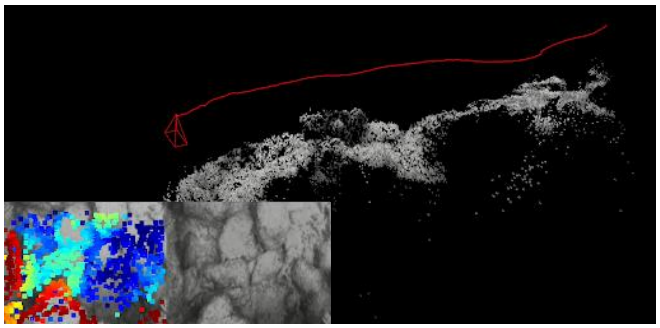

Fig. 5: Sample of a trajectory produced using DSO on the downward-facing camera.

C. AI Stack

The AI Stack is an NVidia Jetson TX2 (hereafter referred to as the TX2) mounted on an Auvidea J120 carrier board and connected using only power and a generic network interface that minimizes the hardware integration. This configuration gives us maximum flexibility by not integrating hardware (eg, the USB 3.0 cameras) directly to the unit. This configuration also eases the process by which the rapid advances in GPU architecture can be accommodated via upgrades. The TX2 also runs Ubuntu with the ROS services and receives robot images by subscribing to the images sent from the Vision Stack. Robot control commands from the AI Stack are sent back to the autopilot controller on the Vision Stack. The strength of the TX2 is its ability to run low-latency inference on machine learning algorithms. Our particular focus is on the use of artificial DCNNs. In particular, we have evaluated a number of image understanding tasks using network architectures based on variants of the Resnet-18 [20] architecture and using the Tensorflow software library.

IV. Experimental Validation

Our experimental validation is focused on three image-processing tasks: 1) running VO on the Vision Stack, 2) collision avoidance and navigation running on the AI Stack, and 3) live-coral detection that is also able to run on the AI Stack.

A. Visual Odometry

To demonstrate the effectiveness of VO on Aqua, we ran a modified version of DSO on the Vision Stack using 40 minutes of recorded video taken from the downward-facing camera on Aqua. This data was collected in the open ocean in close proximity to coral on the cost of Barbados. A sample of the trajectory produced using DSO is shown in Figure 5. The video frame rate is 10 FPS and the trajectory is produced in real-time. On several occasions, DSO got lost due to poor texture regions of the sea-floor (most notably sandy

Fig. 6: Cumulative trajectories produced using DSO of swimming paths in Barbados.

regions). We have added an automatic reset feature to DSO where it tries to re-initialize upon becoming lost. However, in the ideal case, we want continuous tracking for robust localization and coverage data. As ongoing research, we are actively working on methods to avoid texture-poor regions to maintain good visual tracking. The paths of all trajectories produced in Barbados are shown in Figure 6 overlayed on an aerial image of the coral reef where the experiments were performed.

B. Collision Avoidance and Navigation

Unsupervised learning is an active area of research. However, the most effective solutions depend on supervised learning methods where good ground truth solutions are available for each example image at training time. In the case of collision avoidance and target selection, we used a DCNN based on the Resnet-18 architecture and trained it using a large set of images labeled with the steering action one would expect the robot to take when encountering such a scene. The possible actions were categorized into seven yaw classes and seven pitch classes, each of the set -3, -2, -1, 0, 1, 2, 3 representing *unscaled* degrees of steering angles.

The training data consisted of three image types: 1) images that were very close to an object and the label indicated

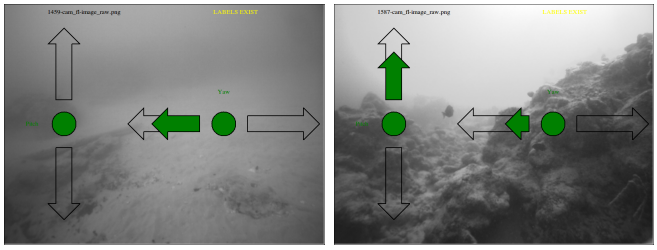

Fig. 7: Examples of two images and their associated label used to train the DCNN for collision avoidance and target selection. In the left image, the label indicates that the robot should swim left to avoid the plain sandy region. In the right image, the label indicates to swim left and upward to avoid hitting the coral (obstacle).

an action for the robot to move away from the obstacle, 2) images of coral or other scenes of interest where the label indicated an action for the robot to move closer for observation, and 3) images of poor texture regions such as the sandy sea-floor and ocean surface with the label indicating the robot should avoid these areas. Two example images and their associated action labels are shown in Figure 7. In general, the objective was to select steering directions that minimized the chances of collision, but did not veer away unnecessarily from visual features.

We trained the Resnet-18 based DCNN to predict a preference distribution over the yaw and pitch steering angles based on the current image and a small window over recent history. The network was trained on approximately 13 000 images using cross-entropy between labeled data and was regularized using a KL-divergence term. The network was pre-loaded with the weights provide by [21] and training was done on an NVidia Titan Xp GPU and took five hours.

The trained model was loaded into the TX2 where inference was performed on the DCNN. The predicted action was sent to the Vision Stack autopilot to autonomously control the robot. Our experiments resulted in 40 min of autonomous swimming (with some very minimal intervention), covering 1080 m in the open-ocean in close proximity to coral while avoiding obstacles. The average distance to objects throughout the experiments was 43 centimeters [19].

C. Coral-Reef Classification

To validate live-coral classification on the TX2, we used labeled data from our previous work [4] and trained a DCNN similar to IV-B. The dataset consisted of images labeled as either containing live-coral or not, as exemplified in Figure 8. The dataset contained 1 117 training images and 2 036 test images. Using the DCNN, the live-coral prediction accuracy was 80.46%. Although this result is lower than in our previous work, we suspect that the lower accuracy is due to the relatively small number of training examples ($\sim$1 000) compared to the number of parameters for the network ($\sim$10 million). Figure 9 shows the accuracy and loss on the training data as the DCNN was trained over 10 000 iterations.

D. Simulation Data for Training

One classic shortcoming of supervised learning using a DCNN is that it can require large amounts of training data with ground truth annotation that can be sometimes

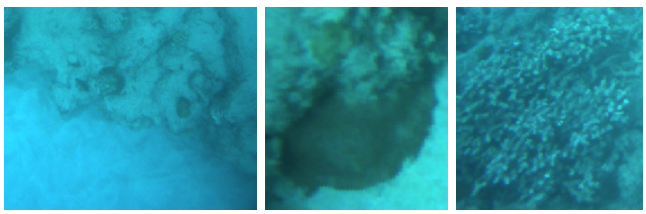

Fig. 8: Examples of data used to train the coral classifier. The left image has no live-coral, while the center and right image contains live coral.

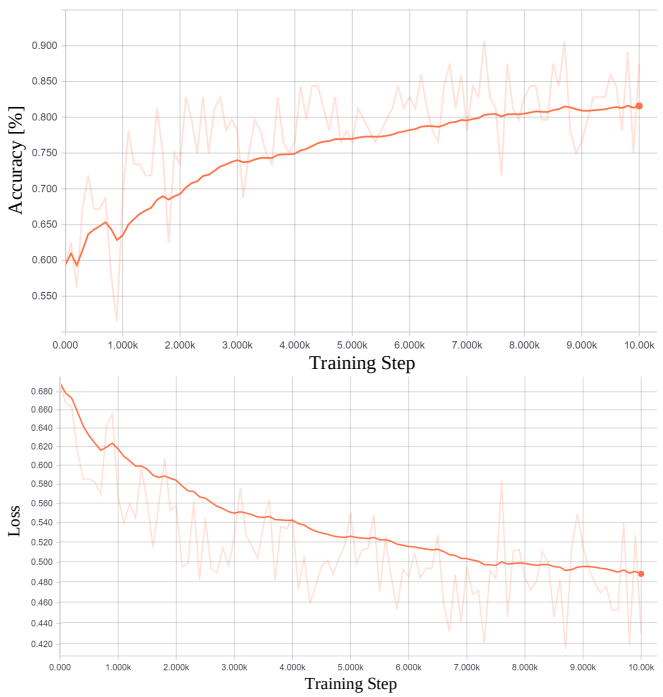

Fig. 9: Training accuracy and loss for the DCNN used to predict live-coral.

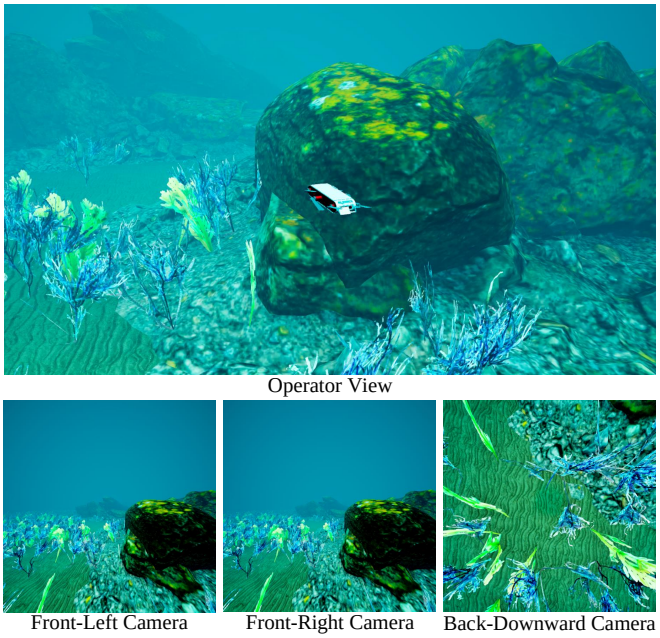

Fig. 10: Example of simulation data showing the operator view, and the view from the three cameras.

difficult to collect in the open ocean. To account for this deficiency, we have developed an underwater simulator (see Figure 10 for an example image of the operator view and three camera views) complete with the Aqua kinematic and sensor models to create large amounts of ground truth-labeled data to train the deep models. This simulator allows for the synthesis of perfect ground truth in diverse scenarios, including potentially critical situations that are rarely seen in real life. Such simulated data, however, must be augmented with some real data to avoid generating fragile solutions that may leverage imperfections in the simulation environment. Our simulator further relieves us from the burden of hand-labeling images for recognition tasks such as object detection and pose-estimation, classification, and scene geometry. This synthetic data has been successfully used in [22] to track underwater vehicles.

V. Conclusion

In conclusion, we have presented a GPU-assisted robot that is capable autonomous collision-free navigation and image understanding by running inference on DCNNs using the integrated TX2. We have shown that the Aqua robot is capable of simultaneously swimming autonomously, avoiding obstacles, navigating towards coral-dense regions, performing classification of live-coral, and running visual odometry to maintain an estimate of its location. We hope that these capabilities will enable robot-assisted environmental monitoring such as autonomous evaluation for coral reef health.

References

[1] J. Horgan and D. Toal, "Computer vision applications in the navigation of unmanned underwater vehicles," in *Underwater vehicles*. InTech, 2009.

[2] G. Dudek, M. Jenkin, C. Prahacs, A. Hogue, J. Sattar, P. Giguere, A. German, H. Liu, S. Saunderson, A. Ripsman, S. Simhon, L. A. Torres-Mendez, E. Milios, P. Zhang, and I. Rekleitis, "A visually guided swimming robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton AB, Canada, Aug. 2005, pp. 1749–1754.

[3] P. Giguere, Y. Girdhar, and G. Dudek, "Wide-Speed Autopilot System for a Swimming Hexapod Robot," in *Canadian Conference on Computer and Robot Vision (CRV)*, 2013. [Online]. Available: http://www.cim.mcgill.ca/ yogesh/publications/crv2013.pdf

[4] T. Manderson, J. Li, D. Cortés Poza, N. Dudek, D. Meger, and G. Dudek, *Towards Autonomous Robotic Coral Reef Health Assessment*. Cham: Springer International Publishing, 2016, pp. 95–108.

[5] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[6] C. C. Eriksen, T. J. Osse, R. D. Light, T. Wen, T. W. Lehman, P. L. Sabin, J. W. Ballard, and A. M. Chiodi, "Seaglider: a long-range autonomous underwater vehicle for oceanographic research," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 424–436, Oct 2001.

[7] J. Sherman, R. E. Davis, W. B. Owens, and J. Valdes, "The autonomous underwater glider "spray"," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 437–446, Oct 2001.

[8] L. L. Whitcomb, M. V. Jakuba, J. C. Kinsey, S. C. Martin, S. E. Webster, J. C. Howland, C. L. Taylor, D. Gomez-Ibanez, and D. R. Yoerger, "Navigation and control of the nereus hybrid underwater vehicle for global ocean science to 10,903 m depth: Preliminary results," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 594–600.

[9] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics Automation Magazine*, vol. 19, no. 1, pp. 24–39, March 2012.

[10] P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu, "Experiments with underwater robot localization and tracking," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4556–4561.

[11] A. Kim and R. M. Eustice, "Perception-driven navigation: Active

visual slam for robotic area coverage," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3196–3203.
[12] S. Pi, B. He, S. Zhang, R. Nian, Y. Shen, and T. Yan, "Stereo visual slam system in underwater environment," in *OCEANS 2014 - TAIPEI*, April 2014, pp. 1–5.
[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.
[14] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," Mar. 2018.
[15] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.
[16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
[17] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, April 2018.
[18] D. Meger, F. Shkurti, D. C. Poza, P. Giguère, and G. Dudek, "3d trajectory synthesis and control for a legged swimming robot," in *Proceedings of the IEEE International Conference on Robotics and Intelligent Systems (IROS)*, 2014.
[19] T. Manderson, J. Gamboa Higuera, R. Cheng, and G. Dudek, "Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[21] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 4241–4247.
[22] K. Koreitem, J. Li, I. Karp, T. Manderson, F. Shkurti, and G. Dudek, "Synthetically trained 3d visual tracker of underwater vehicles," in *MTS/IEEE OCEANS*, Charleston, SC, USA, Oct. 2018.