Vision-Based Autonomous Underwater Swimming in Dense Coral for Combined Collision Avoidance and Target Selection

Travis Manderson, Juan Camilo Gamboa Higuera, Ran Cheng, Gregory Dudek

***Abstract*— We address the problem of learning vision-based, collision-avoiding, and target-selecting controllers in 3D, specifically in underwater environments densely populated with coral reefs. Using a highly maneuverable, dynamic, six-legged (or flippered) vehicle to swim underwater, we exploit real time visual feedback to make close-range navigation decisions that would be hard to achieve with other sensors. Our approach uses computer vision as the sole mechanism for both collision avoidance and visual target selection. In particular, we seek to swim close to the reef to make observations while avoiding both collisions and barren, coral-deprived regions. To carry out path selection while avoiding collisions, we use monocular image data processed in real time. The proposed system uses a convolutional neural network that takes an image from a forward-facing camera as input and predicts unscaled and relative path changes. The network is trained to encode our desired obstacle-avoidance and reef-exploration objectives via supervised learning from human-labeled data. The predictions from the network are transformed into absolute path changes via a combination of a temporally-smoothed proportional controller for heading targets and a low-level motor controller. This system enables safe and autonomous coral reef navigation in underwater environments.**

We validate our approach using an untethered and fully autonomous robot swimming through coral reef in the open ocean. Our robot successfully traverses 1000 m of the ocean floor collision-free while collecting close-up footage of coral reefs.

I. INTRODUCTION

In this paper, we present and validate an integrated system for vision-based navigation combined with collision avoidance and visual target detection and selection. Specifically, our system relies solely on computer vision to select a navigational path that maximizes the value of its visual view while simultaneously avoiding collisions in a highly cluttered environment that is also visually challenging. Such visual challenges include optical artifacts such as floating particulate matter, reflective effects, and (optical) caustics. Furthermore, maintaining system control is complicated by exogenous forces acting on the vehicle, such as surge currents. We develop and test our navigational system in an underwater environment where the objective is to perform close-range coral reef monitoring, classification, and observation (see Figure 1). Our technique, however, is well suited for a wider range of applications.

Underwater environments have large variations in appearance, visibility and floating particulate matter which makes our objective of navigating safely while performing close observations of structures labeled as interesting to a human domain expert, difficult. Thus, we use a learning-based approach to vision-based navigation. We first collect open-loop image data from swim-throughs by a human domain expert to generate 'good' and 'bad' navigation scenarios. The images from these scenarios are then manually annotated off-line and used to train the appearance-based controller we employ. We also employ state-of-the-art vision-based SLAM methods to post-process our image data and reconstruct the trajectories.

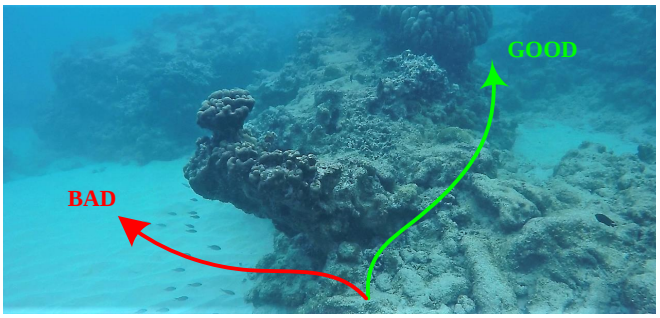

Fig. 1. Example of the underwater environment with two possible navigation paths. The desirable path is shown in green (right). The undesirable path is shown in red (left). A demonstration video can be found at `https://youtu.be/jjkP0SYiF7M`

In our experiments, we use a hexapod underwater robot that autonomously swims over coral reefs in the open ocean. This robot is highly maneuverable with a turning radius of roughly 7 cm, which allows us to maintain a distance of roughly 10-50 cm between the robot and the reef structures. As a result, this underwater navigation task more closely resembles a typical drone-flying task in terms of control demands and operational parameters than the standard navigation task for underwater vehicles. To our knowledge, this close-approach navigation in six degrees of freedom (6DoF) is the first instance of vision-based underwater navigation with such meticulous performance requirements.

Vision-based odometry (VO) in 6DoF has been considered by many authors [1], [2], [3], [4], [5], [6]. In this work, our emphasis is not on VO per se, but on safe close-range navigation combined with surveillance of desirable content. However, many of the constraints from VO still apply, including the need to account for observing useful content. On the other hand, some objectives of our navigation task go beyond the capabilities of traditional VO methods, including control feedback for navigation and assurance of collision avoidance. In addition, we need to compute a 6DoF navigation solution in the absence of GPS or range

Mobile Robotics Laboratory, School of Computer Science, McGill University, Montreal, Canada
{travism,gamboa,rancheng,dudek}@cim.mcgill.ca

data and in the presence of non-rigid structures, floating particular matter, and lighting variations due to caustics (refraction through the water surface) or exogenous forces such as currents. These requirements invalidate some of the assumptions of classical methods based on a combination of visual features and inertial input, making them difficult to apply. Note that in underwater environments, GPS is essentially impossible while other sensing mechanisms such as sonar tend to be relatively bulky and slow relative to the close-range scale of the operation in which we are interested.

II. RELATED WORK

Behavioural cloning is one of the earliest techniques used for training a vision-based controller from human-provided examples. Pomerleau [7] trained a feedforward neural network controller to predict the angle of the steering wheel for driving a van. The network was trained using supervised learning based on data collected in simulation. A similar setup has been replicated for training deep convolutional neural network controllers for self-driving cars using non-simulated data collected while driving a real vehicle [8].

Recent related methods for aerial vehicles have a similar setup but differ in the way data is collected. Loquercio et. al [9] used a dataset collected by bicycles and cars for autonomous flying in urban environments. Giusti et. al. [10] used a dataset collected with head-mounted cameras to train a controller that predicted steering angles for an aerial vehicle. A similar data collection system was used by Smolyanskiy et. al [11] to also train a controller for the lateral offset of an aerial vehicle. Our work is closely related to [11] in that we use a similar controller architecture, but differs in the data collection setup, how the system is trained, and the deployment environment.

A related method that does not rely on training a deep neural network controller is the visual 'teach-and-repeat' framework by Furgale et. al [12]. In this framework, an autonomous vehicle is driven through a desired path while collecting images along the way. These images are used to build a locally consistent map of the environment and a reference path to be tracked. Using this map, an autonomous rover can determine how far it is from the reference path and use this information as an error signal for a path-tracking controller.

The experiments we describe in this paper also bear close resemblance to the DAGGER algorithm [13]. DAGGER proceeds by training a controller from an initial set of expert demonstrations. This controller is applied on the target system, possibly leading the system to states that were not observed in the initial demonstrations. To continue, an expert must label the new states that were visited, thus providing the actions that he or she would have taken in these novel situations. The new labeled data is then added to the initial dataset to train a new controller. This procedure is repeated iteratively until the controller achieves satisfactory performance. In this study, we trained our controller in a similar fashion, repeatedly deploying the controller and labeling new data as it was generated. However, our expert demonstrations

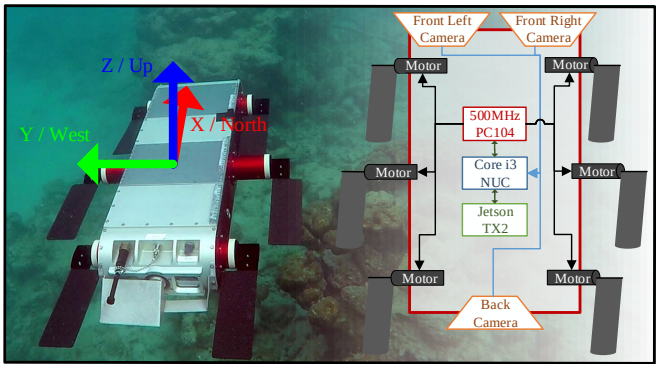

Fig. 2. Aqua robot (left) shown with a system block diagram of the motors, cameras, and three computer components (right).

came from multiple experts with slightly different ideas of how the robot should act in novel situations. Furthermore, the labels we obtained did not correspond directly to the desired actual actions taken by the robot; rather, they represented the desired relative change in the path of the robot.

Recent work by Shkurti et. al. [14] used a neural network controller to control the same underwater vehicle used in this work. Their controller consisted of a single class object detector trained to detect divers or other underwater robots and a linear controller for keeping the detected bounding boxes centered around the optical axis of the camera. In such an application, the task is defined by the controller architecture and does not provide obstacle avoidance. Such a method may be combined with ours for safe multi-robot/multi-diver convoying.

III. SYSTEM OVERVIEW

AQUA robots are a small, human-portable, six degrees-of-freedom hexapods with six flippers, as shown in Figure 2 [15], [16]. These robots are capable of complex trajectories in confined underwater spaces, including dense coral reef. Sensors on the robot include an Inertial Measurement Unit (IMU), depth sensor, and three IDS Imaging UI-3251LE global-shutter cameras with 4 mm lenses (the effective field of view underwater is 107° horizontal and 91° vertical), two forward-looking cameras at the front and one at the back (configured to look down using a mirror). Onboard computation includes: 1) a 500 MHz PC104 computer running a real-time operating system dedicated to low-level motor control and autopilot systems [16]; 2) an Intel NUC with an i3 processor used to capture camera images, do image processing, and run high-level control algorithms; and 3) a Jetson TX2 Module used to run neural networks. All three

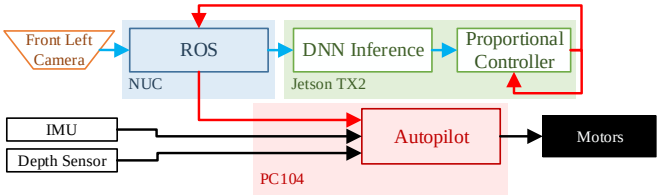

Fig. 3. Flowchart showing the main sensors, computers and motors inside AQUA, along with the connections of the main components of our system.

computers communicate over TCP using gigabit Ethernet.

A flowchart of our autonomous system is shown in Figure 3, which begins with the forward-looking camera where images are captured and processed by the NUC computer using custom Robot Operating System (ROS) nodes. The images are sent to the Jetson TX2 using ROS messages, which also has the ROS services installed, where they are processed by our neural network model as described in section V. The model predicts relative steering angles in the form of changes to the yaw and pitch. These predictions are used by a proportional controller that weighs the predictions and performs temporal filtering (as described in section VI) to output actions in the form of target yaw and pitch angles expressed in world-frame. The actions are relayed to the autopilot, which orientates the robot to the target yaw and pitch angles.

IV. DATA COLLECTION

Our dataset consisted of 12,870 labeled images collected on the west coast of Barbados. We collected video footage by treating the robot as a hand-held camera and recording several desirable sequences of coral. We also recorded undesirable sequences such as approaches to coral closer than we wanted the robot to swim or footage of the sandy sea-floor or the water surface. From these video sequences, we saved individual frames that were then hand-labeled to indicate the desired change in the path of the robot.

Our labeling tool worked by displaying a single image frame to the labeler (a human) as shown in Figure 4 who used the arrow keys to label each image with a desired change in the yaw and pitch angles that one would expect the robot to swim. Although this labeling task may be viewed as subjective, we generally applied the following guidelines to promote consistency between the labelers:

1) Avoid obstacles and ensure the maximum change in yaw and/or pitch when extremely close.
2) Explore interesting coral.
3) Avoid sandy and plain regions.
4) Steer away from the water surface.
5) Steer away from the sea floor.

To augment the size of the training set and add variation in our dataset, we applied image mirroring (and changed the corresponding labels), randomly added noise, and changed the luminosity to account for change in lighting conditions.

V. MODEL OVERVIEW

The vision-based controller used in our experiments is based on a convolutional neural network (CNN) trained to predict steering actions for the robot for a given input image. These actions represent the *desired* degree of change in the orientation of the robot (yaw and pitch) while swimming at a constant forward speed in the water. We based our architecture (as illustrated in Figure 5) on the Resnet-18 architecture [17], which has been used successfully for similar visual navigation tasks [11].

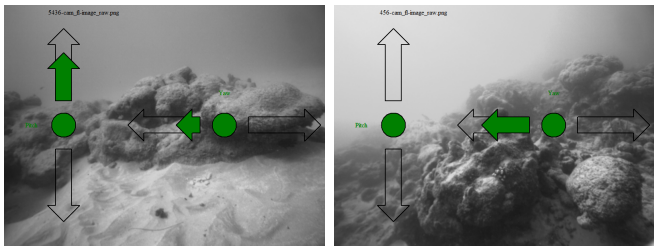

Fig. 4. Two examples of our Graphical User Interface labeling tool.

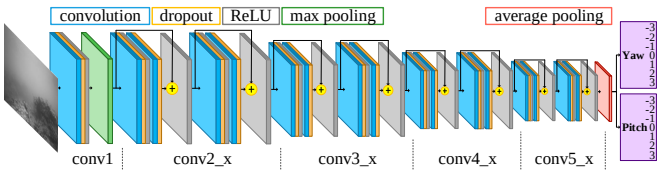

Fig. 5. An overview of the neural network architecture used for the steering action classifier. It takes as input, an image from the robot's camera and predicts relative yaw and pitch steering angles.

We treated the task of learning the controller from data as a classification task. The goal was to predict yaw and pitch classes θ and ϕ; where both were elements of the set, C=$\{-3, -2, -1, 0, 1, 2, 3\}$. These classes represented an *unscaled* desired degree of change in the angle, with positive/negative labels denoting a change in the anti-clockwise/clockwise direction. For example, $\theta = 0$ corresponded to no change in yaw, $\theta = -1$ corresponded to turning slightly to the right, and $\theta = 3$ corresponded to a large turn to the left. The meaning of these predictions in absolute terms was a function of the mental state of the individuals labeling the data and the scaling factor introduced in the controller that used the network predictions. We chose seven as the number of yaw and pitch classes to provide consistent labels, which has been commonly shown in psychology literature [18].

Since our datasets were relatively small for the number of parameters ($\sim$13 000 labeled samples for 10 million parameters), we opted to regularize our model via approximate Bayesian variational inference. We introduced dropout layers [19] after every convolutional layer in the network and changed the objective function to correspond with approximate variational inference, as described in [20]. In particular, we used the Concrete Dropout approximation [21], as it allowed for fitting the dropout rates for each convolutional layer in a data dependent manner. We also added a penalty term for predictions that were overconfident and performed *label smoothing*, as it is suggested in [11].

More specifically, we started with a neural network f with parameters $\mathbf{w}$, which produced two predictive categorical probability distributions, $\mathbf{f}^{(\theta)}$ and $\mathbf{f}^{(\phi)}$, for the yaw and pitch steering actions. We were given a dataset D consisting of tuples of input images and one-hot encoded labels[1] $\langle(\mathbf{x}_1, \theta_1, \phi_1), ..., (\mathbf{x}_N, \theta_N, \phi_N)\rangle$. In this scenario, the cost

[1]A vector of size equal to the number of classes, where the position indexed by the corresponding class label is set to one while the other entries are set to zero.

function for training was the following:

$$\mathcal{L}(D, \mathbf{w}) = \mathcal{L}_{\text{pred}}(D, \mathbf{w}) + \lambda_1 \mathcal{L}_{\text{reg}}(D, \mathbf{w})$$
$$\mathcal{L}_{\text{pred}} = \sum_{(\mathbf{x}_i, \theta_i, \phi_i) \in D} l(\mathbf{f}_i^{(\theta)}, \theta_i, \mathbf{w}) + l(\mathbf{f}_i^{(\phi)}, \phi_i, \mathbf{w})$$

where the regularization term corresponds to the KL-divergence term in [21]. The prediction loss for yaw steering actions was given by

$$l(\mathbf{f}_i^{(\theta)}, \theta_i, \mathbf{w}) = -\sum_j \theta_{ij} \log f_{ij}^{(\theta)} - \lambda_2 \sum_j f_{ij}^{(\theta)} \log f_{ij}^{(\theta)}$$

where first term corresponds to the cross-entropy between the network predictions and the smoothed labels, and the second term is the penalty for overconfident predictions, which aims to maximize the entropy of the predictive distributions. The loss for the pitch steering actions was similar. The hyper-parameters λ_1 and λ_2 determined the weights of the KL-divergence regularization and the entropy penalty and were selected manually.

We kept dropout enabled at test time and applied temporal smoothing to the output predictions. By keeping dropout enabled, we aimed to avoid making decisions on predictions that were too uncertain. When dropout was disabled, we obtained the expected prediction averaged over the posterior distribution of the parameters $\mathbf{w}$. With dropout enabled, we obtained predictions from samples of the posterior distribution. When the model received images that were significantly different from the ones observed during training, we expected the distribution over predictions to have high variance (corresponding to high modeling uncertainty). By using samples from this distribution over predictions, we let modeling uncertainty affect the behaviour of the robot. We did temporal smoothing to avoid sudden changes on the motor commands applied to the robot.

VI. CONTROLLER

Our controller was responsible for converting the model's predictive distributions to target yaw and pitch angles, y^{θ} and y^{ϕ}. This conversion was done after temporal smoothing by selecting the class with the maximum probability after adding a bias weight to the class associated with the previous action taken. Our predicted action class for yaw was then given by:

$$\hat{y}^{(\theta)} = \underset{k \in 0 \ldots 6}{\text{argmax}}(\mathbf{f}^{(\theta)} + w_b \hat{Y}_{t-1}^{(\theta)})[k]$$

where k is the class index, w_b the bias weight, and $\hat{Y}_{t-1}^{(\theta)}$ is the one-hot encoded class of the previous action.

Since $\hat{y}^{(\theta)}$ is an *unscaled* representation of the desired change in yaw angle, we converted $\hat{y}^{(\theta)}$ to the scaled yaw angle action (given in the robot body-frame) by multiplying it by one-sixth of the camera's horizontal field of view (FOV^H) and proportional gain, $P^{(\theta)}$:

$$y^{(\theta)} = P^{(\theta)} \hat{Y}^{(\theta)} \frac{\text{FOV}^H}{6}$$

The same process was done for the pitch $y^{(\phi)}$ using the vertical field of view. We sent the target pose to the autopilot in world-frame coordinates, so y^{θ} and y^{ϕ} were converted to world-frame by the transformation ${}^{W}T_{R}{}^{R}T_{C}$, where ${}^{W}T_{R}$ was the transformation from the robot body-frame (initialized to the IMU state when an experiment started) and ${}^{R}T_{C}$ was the transformation from the camera coordinate system to the robot body-frame, which we estimated using the camera-IMU calibration tool, Kalibr [22].

Fig. 6. An example image of the environment where the experiments took place. The poor-quality image exhibits low light, absence of color variation, and high turbidity.

VII. EXPERIMENTAL VALIDATION

The field experiments described in this paper were performed on the west coast of Barbados (approximately 13°11'30.00" N, 59°38'30.00" W). Our data collection was done on three separate days in different regions of a coral reef. The validation trials were performed on two additional days. Figure 6 shows an example image where the experiments took place. As described in IV, we collected training data of similar environments where the validation experiments took place, but not in the exact same locations to avoid overfitting. Since the training data was collected over a number of days, we were able to capture multiple underwater conditions in terms of lighting and turbidity, which vary with the weather.

Our CNN was initialized using the weights provided by [11], and training our CNN took five hours for 10 000 iterations on an NVidia Titan Xp. The hyperparameters of our network are shown in Table I. Our network is implemented using TensorFlow, and on the Jetson TX2, we processed the images in real-time at 10 Hz. Although the accuracy reported during training was only 41%, this result is for seven classes, unlike [9] which reported the accuracy on a binomial class, and [11] which was for three classes. Furthermore, when attempting to avoid obstacles, it is possible that ambiguity may exist in which direction to go. However, this ambiguity is handled by our temporal smoothing.

A. Parameter Tuning

Prior to validating the performance of our model, we manually tuned the control parameters, w_b, $P^{(\theta)}$ in two stages. The first stage used informal heuristics on dry land,

Learning Rate	**Batch Size**	λ_1	λ_2	**Label Smoothing**
0.0001	128	0.00001	0.1	0.1

TABLE I
CONVOLUTIONAL NEURAL NETWORK HYPER PARAMETERS.

Trial	Time	Distance Travelled	Percentage of Path Containing Coral		Average Distance to Obstacles
	(m:s)		>10%	25%	
1	6:30	150 m	95.53%	75.14%	0.55 m
2	6:30	160 m	97.16%	81.20%	0.48 m
3	7:52	220 m	100.00%	88.45%	0.34 m
4	11:10	270 m	92.61%	67.07%	0.46 m
5	7:53	280 m	99.90%	84.32%	0.37 m

TABLE II

TIME, DISTANCE, AND STATISTICS OF CORAL COVERAGE AND OBSTACLES FOR OUR EXPERIMENTAL TRIALS.

followed by refinement passes in the ocean. We tuned $P^{(\phi)}$ by performing three tuning trials in the open sea where the robot was allowed to operate until an overly-close approach to an obstacle was observed by a supervising human diver (at which point we physically intervened to stop the robot). At such a point, the gain was modified. We repeated this process for approximately 20 minutes for each trial. Initially (and for the first tuning trial), w_b was set to zero and the initial parameters for $P^{(\theta)}$ and $P^{(\phi)}$ were based on our previous work on robot tracking [14] and chosen to be 0.5. Without any bias term w_b, the robot oscillated, particularly in the yaw axis, and in a couple instances would have collided with coral. This result can be attributed to ambiguity when avoiding obstacles. In some instances, such as an obstacle directly in front of the robot, either a left or right turn is equally acceptable, which can lead to unstable behavior if only instantaneous decisions are used. In such situations, temporal stabilization is needed and the w_b introduces hysteresis, thus biasing the decision to continue the maneuver that was chosen during the previous time window.

Other tuning actions were needed to assure that the robot would be able to pitch up sufficiently rapidly to avoid a forward obstacle by moving vertically. Likewise, ocean surge currents (back and forth currents) vary rapidly over the reef and can be particularly strong between pairs of obstructions (which is the same phenomenon that leads to "rip tides" that are a risk to human swimmers). In order to account for observed difficulties due to surge (especially in rough sea conditions), some parameters were also used to increase maximum swimming speed and responsiveness. The upshot is that local variations in sea conditions imply tradeoffs between responsiveness, safety, battery life, and data quality.

B. Model Evaluation

In a series of evaluation trials in the open ocean, the robot was able to swim continuously, avoid collisions, and successfully collect the desired image data. However, in two instances, the combination of rough seas, very close approach to the coral, and sudden changes in current provoked an intervention by the supervising divers. Whether these interventions were actually required is impossible to determine. To completely avoid the potential of collision, the operating distance would need to be increased or sea conditions selected to reduce surge currents. We estimate that over the course of the five trials, the robot swam a total distance of more than one km navigating through dense coral in the open sea. The trial times and distances are tabulated in Table II.

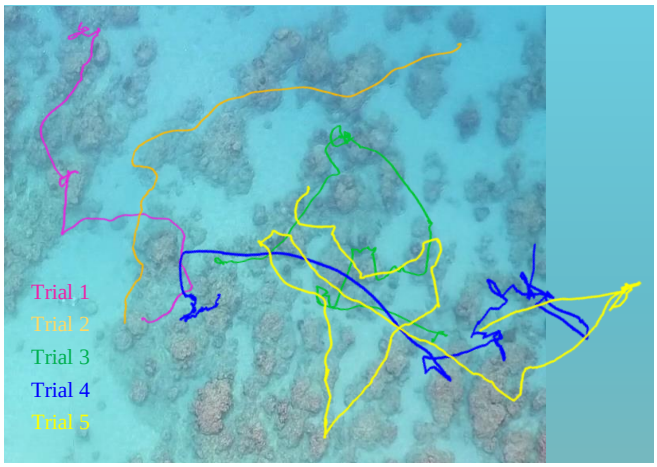

Fig. 7. Planar projections of estimated paths executed during five trials of our experiments (each in a different color). An aerial view of the reef is used as a background to show underlying coral structures and was not available in acceptable quality for the right edge of the region being displayed.

The robustness of our approach is validated by the time and distance that the robot swam without collision. However, to further evaluate the model's ability to navigate the robot through dense coral and compare it to future work, we post-processed the data using three different techniques to gain several performance metrics. First, we ran a modified version of the Direct Sparse Odometry (DSO) algorithm [6] to estimate the trajectory the robot executed. Second, we performed stereo matching between the left and right front cameras based on Oriented FAST and Rotated BRIEF (ORB) feature matching (our implementation is similar to that described in [5]) to evaluate the distance to the nearest potential obstacle in front of the robot. Third, we measured the fraction of each image that was coral using automated image analyses described in our previous work [23].

1) Trajectory Estimation: The robot's trajectory for each trial was measured using recorded video from the back downward-looking camera. Due to the low-texture in sandy regions of the sea floor, DSO is unable to track the point Hessians[2] and comopfute the re-projection error. To overcome this inability, we re-sampled and normalized the distribution of point Hessians. To account for the scale invariant nature of monocular visual odometry, we manually adjusted the scale to correspond with the observed distance to the sea floor.

The computed trajectories for each trial are shown in Figure 7 overlayed on an image taken from the sky. The alignment of the paths is done by our best estimate. The top image in Figure 8 shows the path and reconstruction of the environment for the second trial and bottom image shows an example of the robot's path avoiding an obstacle.

2) Depth Estimation: For each stereo pair of images that we recorded from the front and left cameras, we performed depth estimation to nearby objects by robustly matching

[2]See page six of [6] for a definition of the Hessian.

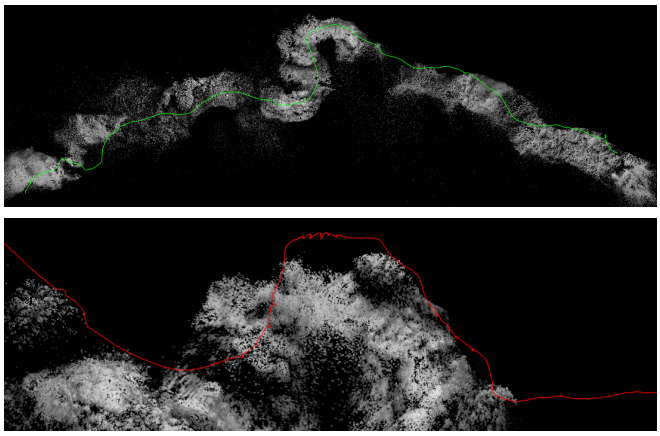

Fig. 8. Sample trajectory of trial two produced using DSO. The bottom image is a closeup view as the robot avoids an obstacle.

ORB features between images and triangulating their position. From these features, we measured the approximate distance to potential obstacles in the robot's path. Note that this approach only measured objects directly in front of the robot due to the narrow FOV (107° x 91°). There are many times when the robot passed less than 10 cm from objects that were either beside or below the robot. Figure 9 shows the distance to the closest objects for each trial. We observed that on average, the robot maintained a distance of 43 cm from coral. There were also very few instances where the distance to coral was further than 200 cm. It is expected that in some instances, due to the nature of the environment, there may be no alternative but to head in the direction of distant coral.

We also calculated the mean position of nearby obstacles by averaging the position of each of the ORB features weighted by their inverse distance. Figure 10 shows a sample of the first trial with the obstacle's average (horizontal) position along with the yaw action (where the yaw has been

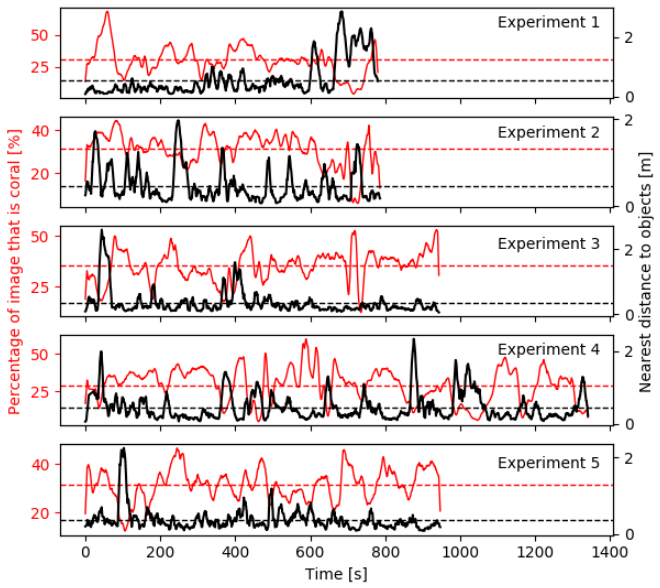

Fig. 9. The coral coverage per frame and distance to nearest objects per experiment. On average, the robot maintained 30% of the image as coral and swam 43 cm from the nearest coral.

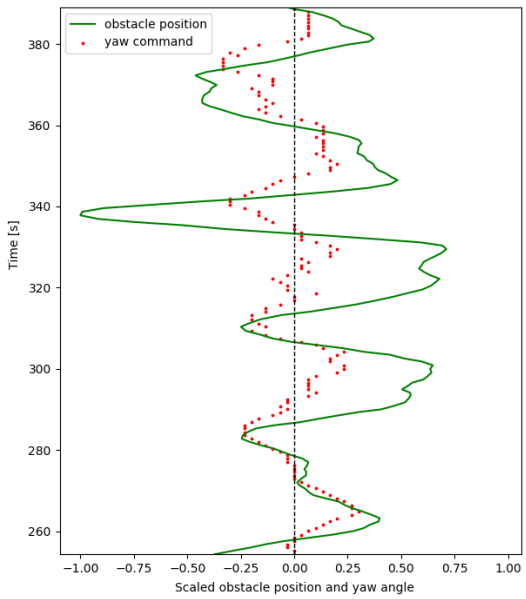

Fig. 10. A sample showing the correlation between the coral and obstacle position versus the yaw action. Note that the yaw has been inverted for clarity.

inverted and both the yaw and average position have been normalized to increase readability). It can be seen that there is a correlation: when an obstacle is on one side of the image, the robot steers in the other. We observed this motion in very confined spaces - however, nearby objects are not the only factors in the yaw control.

C. Coral Coverage

To measure the effectiveness of navigating coral regions as opposed to dull or sandy regions of the environment, we ran an automated coral classifier [23] on every image in our trials and recorded the percentage of the image classified as coral (see Figure 11). Since the robot spent most of its time navigating over coral while swimming flat, it is reasonable to expect at most half of the pixels in each image correspond to coral (the bottom of the image), unless the robot gets too close to an obstacle. Figure 9 shows the percentage of the image classified as coral at each frame, for each trial. On average, the robot swam while maintaining $\sim 30\%$ of the image as coral. Table II summarizes the path coverage of coral for two thresholds, 10% and 25%. With a 10% threshold, the robot kept corals in view $\sim 95\%$ of the time. With a 25% threshold, the coral coverage drops to as low as 67% for trial four, and as high as 88% for trial three.

To estimate the impact of the algorithm's effort to retain coral in the robot's view, we compared the amount of coral visible on the paths in Fig. 7 to trajectories that did not use visual feedback to enhance coral contact. To do this comparison, we generated several synthetic straight-line trajectories in the same regions of the reef and measured the fraction of frames in which coral was visible. These synthetic trajectories yielded an average coral visibility between 40% and 76% with a mean of 67%. These figures are substantially worse than those observed by the robot, which were between 95% and 100%.

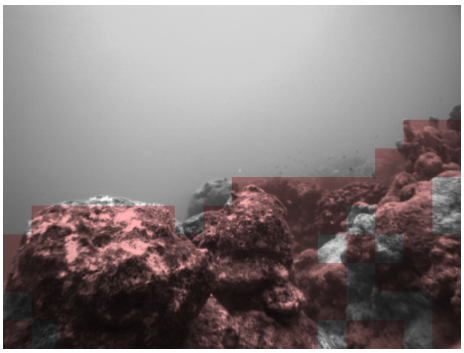

Fig. 11. Sample image with regions classified as coral shown with red overlay (note that the classification is imperfect). In this image, less than 50% of the image is coral and some rock has been misclassified as coral.

VIII. CONCLUSIONS

In this paper, we have described an approach to real-time vision-based robot navigation that combines collision avoidance, proximity-maintenance, and maximization of useful visual content. Unlike traditional (and impressive) methods for SLAM or visual odometry, our technique rapidly accounts for obstacles that are positioned too close to the vehicle's direction of motion; selects steering actions to avoid collisions; operates robustly in the presence of fish, soft moving coral, and non-rigid structures; and avoids oscillatory actions in the presence of ambiguous visual signals. Since formalization of all these constraints, as well as lighting variations and subjective preferences, is difficult to achieve, we employed a learning-based approach to path selection and demonstrated extensive collision-free navigation with good data acquisition properties. Notably, the quality and nature of the images we acquire make this a suitable controller for the collection of frames that are suitable for vision-based SLAM. In contrast, uncontrolled trajectories in the same regions often include numerous sequential frames with insufficient visual content for reliable long-term position estimation. We apply vision based SLAM as a post-processing stage to illustrate the trajectories that were executed.

To act consistently and decisively, we use temporal hysteresis in the controller. At present, we use a very short time window, but the manner in which this parameter could be modulated as a function of distance, speed, and uncertainty is a topic for future investigation.

In future work, we are interested in combining our approach with SLAM-based mapping and task-driven path planning to optimize data collection objectives. We hope that this work will lead to improvements in environmental modeling as well as pure robotics.

REFERENCES

[1] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, "Augmenting inertial navigation with image-based motion estimation," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 4326–4333.

[2] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1403–1410.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[4] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis, "On the consistency of vision-aided inertial navigation," in *Experimental Robotics*. Springer, 2013, pp. 303–317.

[5] 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.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

[7] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[8] 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.

[9] 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.

[10] A. Giusti, J. Guzzi, D. C. Cirean, F. L. He, J. P. Rodrguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, July 2016.

[11] 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.

[12] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, 2010.

[13] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[14] F. Shkurti, W. Chang, P. Henderson, M. Islam, J. Gamboa Higuera, J. Li, T. Manderson, A. Xu, G. Dudek, and J. Sattar, "Underwater multi-robot convoying using visual tracking by detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, Canada, September 2017, pp. 4189–4196.

[15] 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.

[16] 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

[17] 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.

[18] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychological Review*, pp. 81–97, 1956.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," in *4th International Conference on Learning Representations (ICLR) workshop track*, 2016.

[21] Y. Gal, J. Hron, and A. Kendall, "Concrete Dropout," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

[22] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[23] T. Manderson, J. Li, N. Dudek, D. Meger, and G. Dudek, "Robotic coral reef health assessment using automated image analysis," *Journal of Field Robotics*, vol. 34, no. 1, pp. 170–187, 2017. [Online]. Available: http://dx.doi.org/10.1002/rob.21698