Navigation in the Service of Enhanced Pose Estimation

Ran Cheng, Travis Manderson, David Meger and Gregory Dudek

1 Introduction

This paper addresses robust vision-based odometry for underwater robotics by autonomously adjusting the robot trajectory in real-time to optimize the quality of ongoing visual feedback. It is well-known that accurate Visual Odometry (VO) depends on both the presence of sufficient smooth surfaces with manageable reflectance functions and the availability of detectable rigid structures or appearance variations. More generally, some locations in the world are suited to good VO performance while others are not. Our system finds trajectories that pass over such usable visual content by evaluating a localization quality metric (quality score) on points from a forward-facing camera's image. By planning forward-looking paths that optimize an associated quality score, a downward-facing camera running VO is able to localize our robot more accurately. While our approach is based on a particular underwater vehicle and imaging arrangement, its essential elements can be generalized readily.

Fig. 1 Example coral environment where sandy regions will lead to poor visual odometry estimates.

Fig. 2 Our vision-based robot swimming over a coral reef.

The robotic vehicle used in this work is a fully-autonomous marine system that uses vision for collision avoidance, navigation, and image understanding on a single vehicle. This vehicle is a variant of the "Aqua" class of six-flipper swimming robots [10, 15] configured with two forward-facing cameras and a downward-facing camera (see Figure 6) as well as an on-board GPU module (NVidia Jetson TX2) that is used to run a Deep Convolutional Neural Network (DCNN). The platform is targeted to observational tasks such as coral reef mapping and health assessment.

School of Computer Science, McGill University, Rm 318, 3480 Rue University, Montreal QC, H3A 0E9, Canada
{travism,rancheng,dmeger,dudek}@cim.mcgill.ca

Underwater robots operating in complex environments such as coral reefs often rely on accurate odometry for tasks like autonomous surveying and collision avoidance. VO is appealing for pose estimation due to its simplicity and energy efficiency, as well as the fact that cameras are exteroceptive sensors (at least in the subset of underwater regions where there is good visibility). We use vision as our primary sensing modality because it is passive, low energy, compatible with our very compact vehicle, and can be used in a shallow-water reef environment, which is acoustically complex. On the other hand, coral reefs are a challenging environment for vision-based algorithms due to: 1) poor visual conditions caused by low light, water turbidity, and floating artifacts, 2) featureless regions that appear plain or uniform (such as sand), 3) caustics caused by refractive effects at the water surface, and 4) non-rigid objects.

One way to evaluate coral reef health is by determining the amount of live coral, which we have previously measured using image data [27]. Aqua is small, lightweight, highly maneuverable, and capable of swimming autonomously in close proximity (tens of centimeters) to coral without collision [25]. Our long-term goal is to deploy Aqua to perform fully-autonomous geometric coverage of underwater environments for surveying tasks such as coral reef health. To this end, we require reliable and accurate vision-based odometry. Although many visual-odometry systems exist, they rely on visual texture to compare photometric differences between frames to estimate change in pose, and when uncoupled from trajectory planning, they are likely to encounter situations where poor texture results in incorrect pose estimates.

2 Related Work

Numerous authors have demonstrated the importance of active motion planning for navigation [2, 31, 22]. Information-theoretic reasoning is often utilized [12, 38, 6] to guide the robot along mapping paths that optimize environment coverage and localization accuracy, both with laser-based [5, 33] and visual maps [36, 7]. Gaze control can improve localization quality by capturing images with visually-salient content [13], with many features visible [20] and with reliable local features [9].

Recently, methods that learn to predict visual reliability from self-supervised robot performance data have been considered. Gurau *et al.* [16] utilize object detector performance along previously visited paths to train an introspective predictor of test-time recognition rates at nearby locations, yielding confidence estimates for self-driving behavior. Our own previous work trained a texture classifier capable of making a greedy decision to orient a single camera towards the most reliable visual content [28, 26]. This current paper extends that work significantly, training more flexible deep models through the use of a realistic simulator in conjunction with real open-ocean data, creating an association between two cameras and planning longer navigation paths. This use of simulation data for vision-based algorithm development is a direction we have also explored in the context of marine target tracking [23].

Robot localization in underwater environments is often based upon acoustic or computer vision sensing [32, 19, 35, 21]. Particularly notable is the Direct Sparse Odometry (DSO) [11] algorithm, which we employ in our own work. DSO is one of a family of methods, reaching back to the early work of Horn, that exploit a combination of feature point extraction and tracking to compute 3D structure from motion.

In many domains, the tight coupling of visual cues and trajectories has been considered - for example, to allow a previously-taken path to be replayed [8] or to allow servo-control to be coupled to visual cues [3]. Other authors have considered learning per-viewpoint detection rates as a guide for motion during object search [29]. Active guidance is also a crucial factor for a team of cooperatively exploring robots with communication constraints [4].

3 Approach

Our approach to planning trajectories for accurate navigation is to use the wide field of view of forward-looking cameras to predict eventual VO quality for distant points when observing such points in the environment from the downward-facing camera of the robot. In short, the front facing cameras allow us to see and estimate what will be soon approaching in various parts of the visual field. Since the overall goal is to swim in close proximity over coral, the downward-facing camera provides better VO estimates than running on the forward-facing images.

3.1 Visual Odometry

Visual odometry refers to the processes of computing pose updates for the vehicle from information captured by the camera(s). While several powerful methods exist for this computation, they all depend on a visually suitable environment including good illumination, a suitable number of visual features at the right range of scale, constraints on the rigidity of the material being observed, and other attributes. The suitability of the environment for good VO depends on illumination, vehicle speed, geometry, material properties of the local environment, and other features.

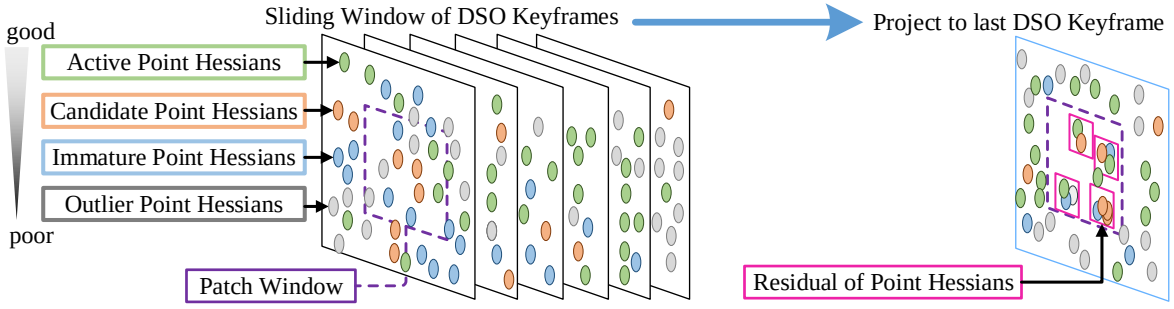

Fig. 3 Pipeline of the DSO quality score prediction.

We apply a variant of the DSO algorithm [11] as our Visual Odometry estimator, with enhancements (outlined below) to permit an estimate of tracking reliability at each frame. More precisely, we track each frame in a sliding window and compute a Hessian for each point being tracked (referred to as a Point Hessian (H)) as shown in Fig. 3. The Point Hessian is a data-structure in DSO that contains the Hessian of the temporally tracked inverse depth and intensity values. The residual (or difference between the estimated and measured value) of the Point Hessian dictates to which class it belongs: "Active", "Candidate", "Immature" or "Outlier" point, as described in the DSO literature.

We introduce a measure for tracking quality, b, which is derived from the classification of the Point Hessians in the current frame and the entropy [34] of those in a center region of the current frame. The entropy, E at frame k is then defined as the sum of normalized probabilities that a Point Hessian belongs to a particular class, c:

$$E(k) = \sum_{i=0}^{N^H} \sum_{c \in C} \frac{P_c ln(P_c)}{N^C}, \qquad C = \{\text{'Active', 'Candidate','Immature','Outlier'}\}$$

where P_c is probability of each class: $P(c \mid H(i))$, and $H(i)$ is one of N^H Point Hessians in the current frame and N^C is the total number of classes (5). We define a quality score, g, per frame, k, as a normalized sum of all

weighted Point Hessians in a centered patch in the frame:

$$g(k) = \frac{N^H}{\eta} \sum_{c \in C} \frac{1}{w_c n(c,k)}$$

where η is the maximum weight over the previous frames ($max\big(g(m)\big)$, $m \in (0 : k-1)$), w_c is a weight associated for each class (with poorer classes being penalized more as shown in Fig. 3), and $n(c,k)$ is the number of Point Hessians in class c within a window at the center of the image. Finally, the quality score is:

$$b(t) = \sum_{k=t-N^w}^{t} (g(k) - E_t(k))$$

where N^w is the number of frames in the sliding window (approximately 10 in practice).

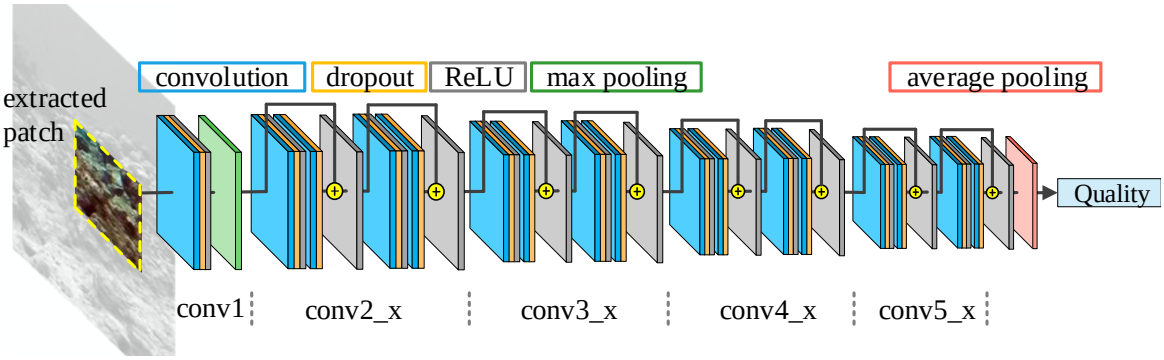

Fig. 4 Deep Convolutional Neural Network architecture which is based on the Resnet-18 model. The input into the network is an image patch extracted from the whole image and the output is a quality score.

3.2 Predicting Localization Quality

Our quality score has a diagnostic form in relation to DSO. It requires visual odometry to be running real-time and must actually experience potential loss of visual quality for the quality score to drop. We aim to prevent the robot from ever experiencing such circumstances at deployment time, and therefore we utilize a learned classifier trained to predict localization quality in advance by analyzing images from a forward-looking camera. This approach allows our system to be pro-active and select paths such that localization performance is never reduced at deployment time.

Our target trajectory predictor is based on a DCNN that predicts the quality score for a set of points observed from a forward-looking image (using a small patch centered at these points), and chooses a heading based on the highest quality points. The DCNN is based on the Resnet-18 architecture [17, 18] as shown in Fig. 4, which performs well on an auxiliary GPU (NVidia Jetson TX2) installed in our vehicle. We introduce dropout layers [37][14] to prevent over-fitting and use the mean squared error cost function for training.

Finally, the robot is controlled using a tuned PID [25] controller that takes temporally averaged image scores and their location to control the yaw and pitch of the robot. A low-level gait controller [30] translates these angular commands to motions of the six flippers.

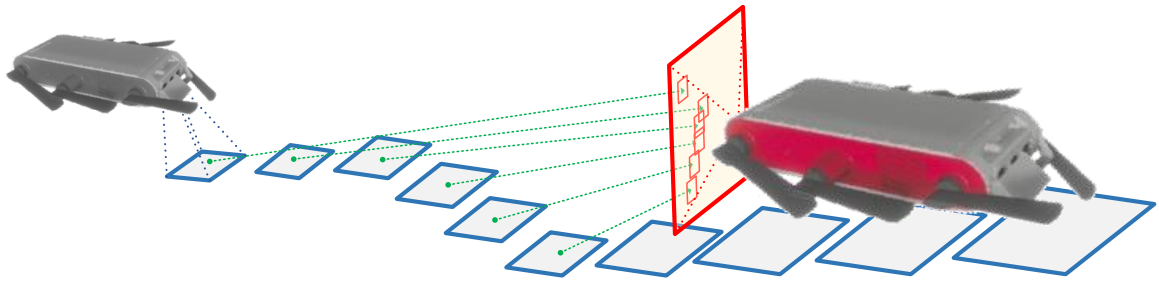

Fig. 5 Image patches are generated by back projecting downward-facing image locations to the front camera image frames.

3.3 Training

Our predictive quality network is trained in a self-supervised fashion by moving the robot over a wide range of terrain and correlating the localization quality with patches in the forward-looking camera via the robot's self-calibration. For each image in the training data set, training labels for the DCNN are generated by back-projecting the relative 3D location of the downward-facing images into the history of front images, as shown in Fig. 5. For each point projected into the front image, a patch is extracted, along with the quality score described above.

The back-projection of a downward facing image at time t into a forward facing image at time t' is given by:

$$u_{t',t}, v_{t',t} = \pi(T_w^c(t')\mathbf{P_b}(\mathbf{t})) \tag{1}$$

where π is a function that projects a 3D point relative to the front camera into the image plane, T_w^c is the transformation from a point in the world to the front camera, P_b is the position of the downward-facing image in world coordinates, and $u_{t',t}$ and $v_{t',t}$ are the pixel locations at time t' of the downward-facing patch at time t.

The robot is controlled using a PID [25] controller that takes temporally averaged image scores and their location to control the yaw and pitch of the robot. A low-level gait controller [30] translates these angular commands to motions of the six flippers.

4 Experiments

Overall, our method processes images in real time to maintain a navigation estimate and close planning and control loops at a number of levels. We perform accurate VO based on the downward looking "Back Camera" images while estimating localization quality in the region ahead of the robot by filtering the "Front Camera" images using a DCNN, whose output feeds longer horizon path planning for the purpose of guiding the robot to regions that support accurate localization.

Our experimental approach is composed of two components: 1) a preliminary proof-of-concept study in the open ocean and 2) a series of rigorous simulation studies allowing repeatable quantitative assessment in a state-of-the-art simulation environment. While our primary focus is achieving reliable performance in the open ocean, the simulation is critical as it allows access to large volumes of reliable ground truth that is challenging to collect and imprecise in field conditions.

Open ocean experiment This experiment served to: 1) generate quality scores in real conditions to validate the methodology and generate initial training data for the DCNN used proceeding experiments, and 2) test the integrated hardware (NVidia Jetson TX2) running DCNN inference alongside the VO algorithm.

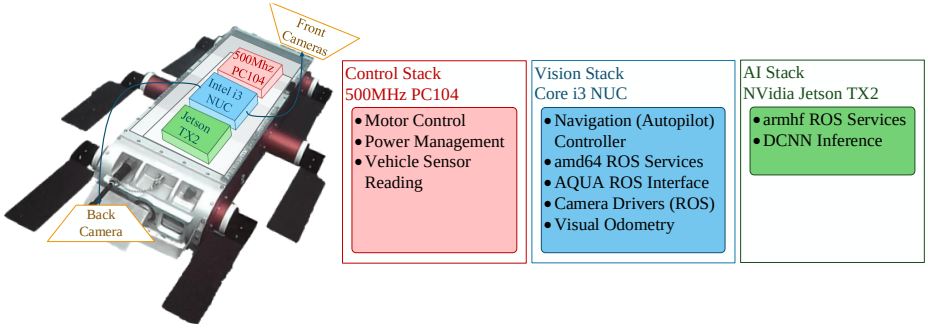

Fig. 6 Block diagram of the Aqua robot and the three computers. The three computers are connected by high-speed Ethernet and the cameras are connected by USB 3.0 to the Intel NUC computer.

Our experimental platform is an Aqua robot (as shown in Fig. 6) where the cameras are operated by three efficient, yet powerful computers that are networked using Gigabit Ethernet. The PC104 computer (Control Stack) uses the real-time Xenomai extension to the Linux operating system and performs low-level motor control, power management and vehicle orientation and depth estimation from an internal Inertial Measurement Unit (IMU) and depth sensor. An Intel i3 NUC computer (Vision Stack) running Robot Operating System (ROS) captures the images (at 15 Hz) from the global-shutter and hardware-synchronized cameras, runs the VO algorithm and performs the high-level navigation tasks. An NVidia Jetson TX2 GPU compute module (AI Stack) which is also running ROS, employs TensorFlow [1] for vision-based inference using the DCNN. The inference results are transmitted to the Vision Stack where they are used in the navigation planner.

Fig. 7 An example comparison of the real and simulated underwater ocean environments. The left image is the real underwater environment taken during our open-ocean experiment and the right image is an example taken from our simulator, showing its close resemblance to the real environment.

Simulation experiments We developed a simulator with two objectives: 1) perform end-to-end validation of our system beginning with image data characteristic of an ocean environment to control outputs to move the robot, and 2) quantitatively evaluate the effectiveness of our system using the ground truth available. Our simulator is based on Unreal Engine 4 and involves a kinetic model for control of the Aqua robot and a visual-inertial sensor model to obtain vision and inertial data. Ground truth, images, and control are communicated using custom ROS2 and ROS1 messages with the same definition used on the Aqua robot. This approach allows us to use the exact inference and control code in both simulation and real-world. We have also built a customizable ocean environment, which also allows us to vary the environmental conditions such as the water turbidity, water current, and brightness. We have used this simulator to tune the inference model as well as control parameters. In separate work, we have demonstrated that this simulator is faithful enough to train vision-based methods that then perform well in genuine marine imagery [24, 23]. As seen in Fig. 7, the simulated environment bears close resemblance to the real underwater, ocean environment.

5 Results

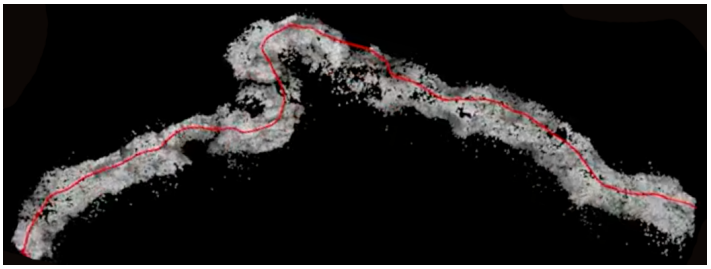

Fig. 8 Illustrative reconstruction from a deployment in the Caribbean.

We ran DSO and our quality score scoring system on our preliminary open-ocean data as well as simulated data as described in Sec. 4. We collected ∼74 minutes (in segments of two to 10 minutes) of data from autonomous swimming in the open ocean near coral over the course of five days (on the coast of Barbados) at varying locations under varied weather conditions (and visibility) and swimming motions. We successfully ran DSO on this dataset and validated the quality score by examining the distribution of scores, visually inspecting obvious good and poor image patches for VO, and verifying that VO failures were preceded with extremely low quality scores. An illustrative path and reconstruction on our preliminary open-ocean data is shown in Fig. 8.

Table 1 DCNN Classification Accuracy on Simulated Validation Dataset

Patch Size	**40px**	**60px**	**80px**	**100px**	**120px**
Accuracy	90.59%	91.46%	93.13%	92.56%	92.37%

Our simulated dataset consisted of 51,892 training images, 21,567 validation images, and 10,104 test images. Our Resnet based DCNN predicted the quality score as one of five classes. The model was initialized with the pre-trained weights after training on ImageNet [17] and the parameters of the model were: batch size: 25, l2 regularization weight: 1e-5, momentum: 1e-4, initial learning rate: 1e-5 and learning rate decay: 1e-3.

Table 1 shows the classification accuracy of our DCNN on a validation dataset collected using our simulator over a number of patch sizes. All patch sizes resulted in accuracy over 90%. The smaller patch sizes allowed our DCNN to perform faster while being more granular in our estimate. A patch size of 80 x 80 pixels resulted in the maximum accuracy of 93.13%. We suspect that the larger patch sizes performed slightly worse because they were capturing a larger region of the scene and may have contained ambiguous textures such as sand and coral in the same image patch.

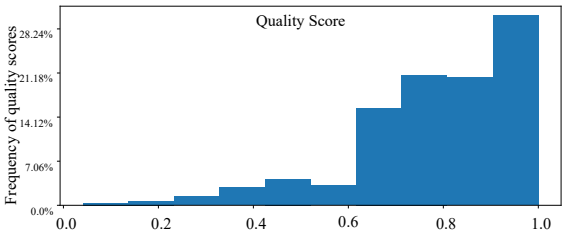

Fig. 9 Sample score Histogram swimming with trajectory prediction (this is a good result since mass is concentrated on the right).

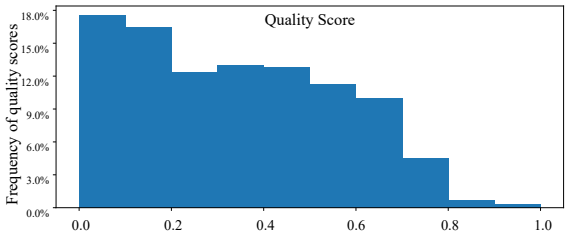

Fig. 10 Sample score Histogram swimming without trajectory prediction (this is a less useful result since mass is concentrated on the left).

The effectiveness of our trajectory predictor can be seen in Fig. 9 and 10, which show the distribution of quality scores collected from autonomous swimming in our simulator with and without our trajectory prediction. The concentration of high quality scores sampled when running our trajectory prediction (as compared to running without it) demonstrates the effectiveness of our quality score estimator and trajectory planner. The quality scores on our preliminary open-ocean experiment is similar to Fig. 10.

Fig. 11 Three simplified examples of the image patch classification from the DCNN inference (note that the class overlay visualization removes much of the image texture). The numbers represent the unscaled score prediction, where the highest number (4) represents the highest quality score. In each example, it can be seen that the highest quality prediction is over coral regions and a much lower quality is predicted for seagrass, sand and water.

To quantify the robustness of our approach, we performed several simulated experiments using our DCNN trajectory prediction controller and a naive way-point controller. Our DCNN trajectory prediction controller navigated the robot towards regions containing the highest quality scores while moving forward at a constant velocity. A simplified example of the DCNN inference is shown in Fig. 11 for three regions of the map (note that the class overlay visualization removes much of the image texture). Each experiment using the predictive controller resulted in a path that we compared against the ground-truth to quantitatively measure the relative

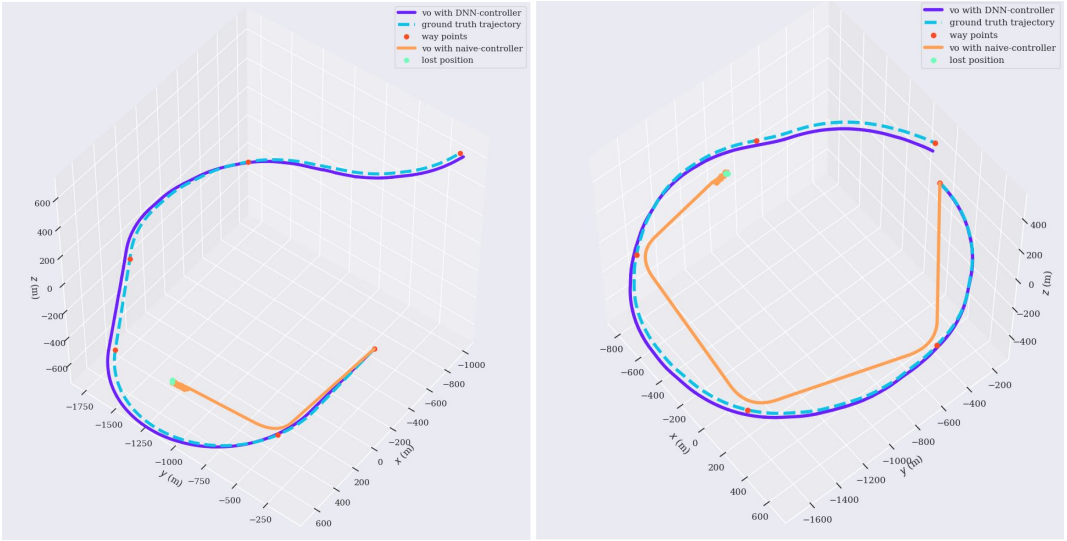

Fig. 12 Two example paths the robot swam with and without trajectory planning. In both cases, it can be seen that the VO path estimate (purple line) closely matches the ground-truth path (blue line). In the left example, the naive waypoint controller, reached the first waypoint, but DSO become lost before reaching the second waypoint (path represented by the orange line). Similarly, in the right example, using the naive waypoint controller, DSO gets lost before reaching the fourth waypoint.

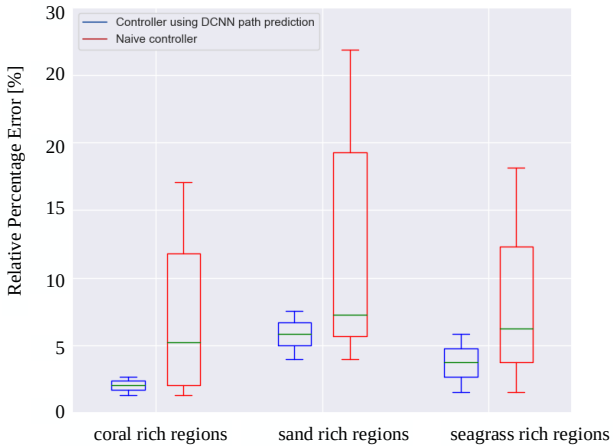

Fig. 13 Comparison of relative percentage error in coral, sand and seasgrass environments. Twenty experiments were performed in each of the three simulated environments which resulted in a total distance travelled of 162 km. The mean relative percentage error (RPE) in the coral rich environments was **2.35%** and 8.50% using our trajectory prediction controller and the native waypoint controller respectively. Likewise, **6.75%** and 17.50% for the sand rich regions, and **4.15%** and 11.25% for the seagrass rich regions. In all cases, our trajectory prediction controller outperformed the naive waypoint controller in both the mean error and the total variance.

error. For each path that the robot took using the predictive controller, we sampled a number of points to create a set of waypoints along the path. We then navigated the set of waypoints using a naive waypoint controller.

The naive waypoint controller moved at a constant forward velocity and used a proportional controller to iteratively point the robot towards the current waypoint until it was reached. To prevent VO error due to rapid rotations, we limited the angular rate of rotation. Once the current waypoint was reached within one meter, the robot navigated toward the next waypoint, repeating until all were reached, or DSO was lost (where the relative pose estimator was unable to converge due to a lack of image texture). An example of two of these experiments is shown in Fig. 12. In each example, the light blue line is the ground-truth path, the purple line is the VO path estimate of the robot while running the trajectory prediction controller, and the orange line is the path of the robot using the naive waypoint controller.

Experiment index	Configuration		Relative Percentage Error	
	Distance	Terrain	DCNN Trajectory Prediction	Naive Waypoint Controller
1	714 m	Coral Region	3.6%	12.8%
2	2156 m	Coral Region	7.6%	18.1%
3	1703 m	Sand Region	11.2%	30.4%
4	1442 m	Seagrass Region	6.9%	11.7%
5	1756 m	Sand Region	7.5%	17.3%
6	815 m	Seagrass Region	14.6%	22.1%
7	1677 m	Sand Region	9.1%	16.2%
8	561 m	Sand Region	5.3%	17.9%
9	2329 m	Seagrass Region	6.3%	12.4%
10	1503 m	Coral Region	3.7%	6.8%

Table 2 Ten example experiments from the set of 60 with the best results for the native waypoint controller showing the total distance travelled, terrain type, and the relative position error for both our trajectory prediction controller and the naive waypoint controller. In all cases, it can be seen that our trajectory prediction controller results in less error over the total path.

In total, we performed 60 experiments. These were divided evenly between three environments: 1) regions where the majority of the environment contained a mix of **coral**, 2) regions that contained a majority of **seagrass**, and 3) regions that contained a large amount of **sandy** bottom. Fig. 13 shows average position error (depicted as the relative percentage error) for the trajectory prediction controller (blue whisker) and the naive waypoint controller (orange whisker) for each environment. It can be seen that in all three cases, on average, the trajectory prediction controller outperformed the naive waypoint controller.

Finally, Table 2 tabularizes the 10 experiments with the lowest relative percentage error in the path using the naive waypoint controller. Note that in all cases, our trajectory prediction controller resulted in less position error, demonstrating the robustness of our method.

6 Conclusion

This paper has presented a real-time method for robust underwater navigation that uses long-to-medium term planning to optimize task completion in the context of using visual odometry. This addresses the important, but often under-appreciated issue, of picking routes in the world where estimation and observability are optimized. We have validated our method on real-world data using an underwater vehicle in the open ocean, and have performed extensive quantitative assessments by running the method in a photo-realistic simulator that simulates both the environment of the ocean and the robot physics. In that case, the use of our controller substantially improves localization and waypoint attainment. Our DCNN predicts a quality score for an image region that represents the performance of the VO at the corresponding position in the world with 93.13% accuracy. We simulated 162 km of autonomous swimming in several environments comparing our trajectory predictive controller against a naive waypoint controller. Our method outperformed all of the 10 best results using the naive controller by as much as 3.5 times.

In future work, we plan to conduct additional underwater sea trials in even more visually challenging and complex environments with the long-term goal of robustly building 3D models of underwater environments and performing autonomous surveys of coral reef health in close proximity. One of the primary challenges will be to identify a good source for ground-truth in these real-world conditions on our robot. We suspect that using simple sub-trajectories (such as constant forward translation), we can selectively simplify the correspondence problem between cameras to facilitate the acquisition of reliable training data.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI, vol. 16, pp. 265–283 (2016)
2. Aloimonos, J.: Purposive and qualitative active vision. In: Pattern Recognition, 1990. Proceedings., 10th International Conference on, vol. 1, pp. 346–360. IEEE (1990)
3. Azrad, S., Kendoul, F., Nonami, K.: Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. Journal of System Design and Dynamics **4**(2), 255–268 (2010)
4. Banfi, J., Li, A.Q., Basilico, N., Rekleitis, I., Amigoni, F., et al.: Multirobot online construction of communication maps
5. Bourgault, F., Makarenko, A., Williams, S., Grocholsky, B., Durrant-Whyte, H.: Information based adaptive robotic exploration. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2002)
6. Carrillo, H., Reid, I., Castellanos, J.A.: On the comparison of uncertainty criteria for active slam. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 2080–2087. IEEE (2012)
7. Chaves, S.M., Kim, A., Eustice, R.M.: Opportunistic sampling-based planning for active visual slam. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pp. 3073–3080. IEEE (2014)
8. Chen, Z., Birchfield, S.T.: Qualitative vision-based path following. IEEE Transactions on Robotics **25**(3), 749–754 (2009)
9. Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. IEEE transactions on pattern analysis and machine intelligence **24**(7), 865–880 (2002)

10. Dudek, G., Jenkin, M., Prahacs, C., Hogue, A., Sattar, J., Giguere, P., German, A., Liu, H., Saunderson, S., Ripsman, A., Simhon, S., Torres-Mendez, L.A., Milios, E., Zhang, P., Rekleitis, I.: A visually guided swimming robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1749–1754. Edmonton AB, Canada (2005)
11. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. In: arXiv:1607.02565 (2016)
12. Fox, D., Burgard, W., Thrun, S.: Active markov localization for mobile robots. Robotics and Autonomous Systems **25**(3-4), 195–207 (1998)
13. Frintrop, S., Jensfelt, P.: Attentional landmarks and active gaze control for visual slam. IEEE Transactions on Robotics **24**(5), 1054–1065 (2008)
14. Gal, Y., Hron, J., Kendall, A.: Concrete Dropout. In: Advances in Neural Information Processing Systems 30 (NIPS) (2017)
15. Giguere, P., Girdhar, Y., Dudek, G.: Wide-Speed Autopilot System for a Swimming Hexapod Robot. In: Canadian Conference on Computer and Robot Vision (CRV) (2013). URL http://www.cim.mcgill.ca/ yogesh/publications/crv2013.pdf
16. Gurau, C., Rao, D., Tong, C.H., Posner, I.: Learn from experience: probabilistic prediction of perception performance to avoid failure. The International Journal of Robotics Research (2017)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
18. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, pp. 630–645. Springer (2016)
19. Hidalgo, F., Bräunl, T.: Review of underwater slam techniques. In: Automation, Robotics and Applications (ICARA), 2015 6th International Conference on, pp. 306–311. IEEE (2015)
20. Jensfelt, P., Kristensen, S.: Active global localization for a mobile robot using multiple hypothesis tracking. IEEE Transactions on Robotics and Automation **17**(5), 748–760 (2001). DOI 10.1109/70.964673
21. Johnson-Roberson, M., Pizarro, O., Williams, S.B., Mahon, I.: Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. Journal of Field Robotics **27**(1), 21–51 (2010)
22. Juliá, M., Gil, A., Reinoso, O.: A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Autonomous Robots **33**(4), 427–444 (2012)
23. Koreitem, K., Li, J., Karp, I., Manderson, T., Shkurti, F., Dudek, G.: Synthetically trained 3d visual tracker of underwater vehicles. In: MTS/IEEE OCEANS. Charleston, SC, USA (2018)
24. Manderson, T., Dudek, G.: Gpu-assisted learning on an autonomous marine robot for vision based navigation and image understanding. In: MTS/IEEE OCEANS. Charleston, SC, USA (2018)
25. Manderson, T., Gamboa Higuera, J., Cheng, R., Dudek, G.: Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection (2018). Under Review
26. Manderson, T., Holliday, A., Dudek, G.: Gaze selection for enhanced visual odometry during navigation. In: Proceedings of the Conference on Computer and Robot Vision (CRV) (2018)
27. Manderson, T., Li, J., Cortés Poza, D., Dudek, N., Meger, D., Dudek, G.: Towards Autonomous Robotic Coral Reef Health Assessment, pp. 95–108. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-27702-8_7
28. Manderson, T., Shkurti, F., Dudek, G.: Texture-aware SLAM using stereo imagery and inertial information. In: Computer and Robot Vision. IEEE (2016)
29. Meger, D., Gupta, A., Little, J.J.: Viewpoint detection models for sequential embodied object category recognition. In: Proceedings of the International Conference on Robotics and Automation (ICRA) (2010)
30. Meger, D., Shkurti, F., Poza, D.C., Gigure, P., Dudek, G.: 3d trajectory synthesis and control for a legged swimming robot. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2014)
31. Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K., De Schutter, J.: A comparison of decision making criteria and optimization methods for active robotic sensing. In: International Conference on Numerical Methods and Applications, pp. 316–324. Springer (2002)
32. Ribas, D., Ridao, P., Tardós, J.D., Neira, J.: Underwater slam in man-made structured environments. Journal of Field Robotics **25**(11-12), 898–921 (2008)
33. Roy, N., Burgard, W., Fox, D., Thrun, S.: Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, vol. 1, pp. 35–40. IEEE (1999)
34. Roy, N., Gordon, G., Thrun, S.: Finding approximate pomdp solutions through belief compression. Journal of artificial intelligence research **23**, 1–40 (2005)
35. Sáez, J.M., Hogue, A., Escolano, F., Jenkin, M.: Underwater 3d slam through entropy minimization. In: Robotics and automation, 2006. ICRA 2006. Proceedings 2006 IEEE international conference on, pp. 3562–3567. IEEE (2006)
36. Sim, R., Dudek, G.: Effective exploration strategies for the construction of visual maps. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2003)
37. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
38. Stachniss, C., Grisetti, G., Burgard, W.: Information gain-based exploration using rao-blackwellized particle filters. In: Robotics: Science and Systems, vol. 2, pp. 65–72 (2005)