Synthetically Trained 3D Visual Tracker of Underwater Vehicles

Karim Koreitem, Jimmy Li, Ian Karp, Travis Manderson, Florian Shkurti, Gregory Dudek
School of Computer Science, Centre for Intelligent Machines
McGill University
Montréal, Canada
{karimkor, jimmyli, ianfkarp, travism, florian, dudek}@cim.mcgill.ca

***Abstract*—We present a method for visually detecting and tracking the 3D pose of autonomous underwater vehicles, which aims to enable robust multi-robot convoying. We follow the approach of tracking-by-detection, which combines the robust, drift-free nature of object detection with the temporal consistency of tracking algorithms. Central to our method is a multi-output convolutional network that jointly predicts whether the target robot is present in the image (classification), the 2D bounding box around the target in the image plane, and the 3D orientation of the target. This, combined with camera intrinsic parameters and prior knowledge of the robot's absolute scale, allows us to recover the full 6-degree-of-freedom pose (translation and orientation) of the target robot. To train our network, we use only synthetic images rendered using the Unreal game engine, which is a cost-effective way to produce a large training set without the need for laborious manual annotations. Our evaluation analyzes the impact of orientation offset on 3D detection accuracy, and demonstrates successful generalization of the learned model to real underwater photographs of the target robot.**

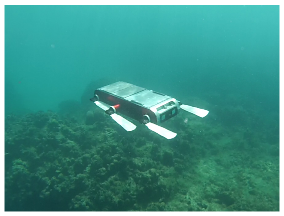
***Index Terms*—3D object detection, visual tracking, robot convoying, underwater robotics, data synthesis, Unreal game engine**

I. Introduction

We present a vision-based approach for tracking the 3D pose (position and orientation) of an autonomous robot in underwater environments. Our tracker enables robust multi-robot convoying, which has been studied extensively in the robotics research community under a variety of settings ranging from self-driving cars [1] to aerial drones [2], but relatively little in the marine context. In the context of automated surveillance of marine ecosystems, convoying enables the deployment of highly configurable heterogeneous robot teams in which each robot collects data using different sensing devices. In contrast to deploying a single highly-capable robot, this distributed approach is more scalable and less prone to a single point of failure [3].

While prior work has used fiducial markers on the target to simplify the visual detection task [1], our method relies solely on the natural appearance of the target. In comparison, our approach overcomes difficulties due to the marker not always being visible as the target robot is seen from a variety of poses. Moreover, in contrast to previous methods that require expensive hardware such as mobile beacons [4] or acoustic sampling [5] to achieve localization underwater, we capitalize on commodity RGB cameras.

This work was funded by the NSERC Canadian Field Robotics Network (NCFRN). The authors are affiliated with the Mobile Robotics Lab and the Center for Intelligent Machines (CIM) at McGill University.


(a) An Aqua robot pictured underwater in Barbados

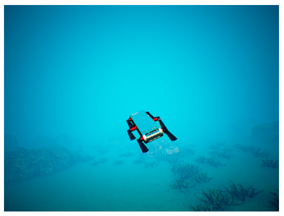

(b) An Aqua robot simulated in Unreal Engine

Fig. 1: The Aqua robot: a highly maneuverable amphibious hexapod.

Our solution employs *tracking-by-detection*: the relative position and orientation of a robot is detected at every frame, and integrated temporally via filtering. The detector we employ is a convolutional neural network (CNN) trained on synthetic images rendered using a CAD model of the robot. The trained network jointly regresses the robot's orientation as a quaternion, as well as a tightly fitting bounding box around the robot in the image plane. Given the camera's intrinsic parameters and the absolute scale of the robot, we can use the detected orientation and bounding box to directly estimate the 3D translation of the target robot. In this way we acquire the full 6-degree-of-freedom (6-DOF) pose of the target.

Training a convolutional network typically requires a large training dataset, but manually labelling 3D ground truth on real images is labor-intensive, and requires the labeller to learn to use sophisticated software tools. Thus, we opt to generate our entire training data synthetically. We follow the domain randomization approach of [6], and generate a large variation of non-photo-realistic training images featuring the robot rendered with a variety of textures and on different background patterns. We then test our trained model on a test set of both synthetic and real images containing hand-labeled 3D poses. Only having to annotate test set images substantially reduces our data management cost. Overall, our detection method and training regime are widely applicable to many multi-robot convoying systems, since a CAD model of the robot is usually readily available and the only onboard sensor needed is a calibrated RGB camera.

We validate our method using amphibious hexapod robots shown in Fig. 1. These belong to the Aqua family of robots [7], which achieve a high degree of maneuverability underwater by synchronously actuating six flippers. The on-board inertial measurement unit (IMU) and pressure sensor provide input to the autopilot [8], which is capable of closed-loop control over the desired depth, attitude, and thrust. The autopilot and the rest of the Aqua sensor suite are operated on top of the Robot Operating System (ROS) [9] framework. Images captured by the front-facing RGB camera serve as input to our tracking pipeline, which can run at frame rate on the robot's laptop-grade dual-core Intel i3 processor and NVidia GPU (Jetson TX2) [10].

We conduct experiments both in simulation using renderings of the robot in synthetic aquatic environments, and on real footage of the robot collected underwater. In our simulated runs we analyze the robustness of our tracker by varying the complexity of the trajectory executed by the target robot. Experiments on real images show that our learned model successfully generalizes and can be utilized to reliably implement multi-robot convoying underwater.

II. Related Work

A. Visual Tracking and Convoying

There is an extensive literature on visual tracking. Many model-free tracking algorithms are designed to cope with unforeseen object instances. For example, in [11], the algorithm is initialized with a bounding box around an arbitrary object to be tracked, and the algorithm is expected to adapt to the target's appearance changes throughout tracking. Our approach is closer to model-based tracking, in which a model of the target is built ahead of time. Like [12] we use a CAD model of the target to train a discriminator but instead of using hand-designed edge and vertex features, we train a convolutional network to recognize the target.

Our method falls in the category of tracking-by-detection. The target is first detected independently at every frame, and the detections are then integrated via tracking. This approach has been shown to effectively combine the temporal consistency of tracking with the robust, drift-free nature of object detection [13].

In the domain of visual convoying, fiducial markers have been added to the target to make detection and tracking easier [1]. However, this approach is often susceptible to the marker going out of view, so we opt to directly model the natural appearance of the target as in [14]. Explicit signalling behaviors executed by the target has also been exploited to improve the efficiency and robustness of multi-robot convoying [15]. We find this to be a promising direction for future work, especially since it has seen little adoption in the underwater environment.

In our own prior work, we have demonstrated an underwater convoying system based on tracking-by-detection that operates on the 2D position of the robot in the image plane [16]. In this work we track the 3D pose of the target, which allows us to potentially leverage a more geometrically detailed motion model to better predict the motion of the target, ultimately resulting in more robust tracking. This is becoming especially relevant as we have successfully demonstrated complex exploration behaviors on the Aqua robot in which the robot stays close to corals while avoiding collision and barren uninteresting regions [17]. Convoying under this setting requires visually tracking the robot while it carries out complex maneuvers.

B. 3D Object Detection

Recently there has been substantial interest to move beyond 2D bounding box detection and to infer 3D information about objects in the image. While earlier work relies primarily on detecting hand-crafted features (i.e. SIFT, HOG) in the image and matching them to features on known 3D object models [18] [19] [20], newer methods often leverage convolutional networks, either to directly learn a mapping from pixels to pose information [21] [22], or to use features produced by the network to facilitate subsequent pose optimization [23]. A variety of object representations have been tried, including 3D object centroid [6], 3D bounding box [21], 3D skeleton [22], and CAD model instances retrieved from a database [23].

In the target tracking domain, we are able to assume that the physical dimensions of the target is known. We leverage this in our detection method, wherein we train a network to output only the target's orientation and 2D bounding box; we then combine this information with the known scale to directly optimize the target's translation. As in [24], we regress the orientation as a quaternion.

C. Training on Synthetic Data

Modern object detectors based on convolutional networks are heavily dependent on abundant training data. Compared to traditional detection tasks that typically output only 2D bounding boxes in the image plane, 3D pose detection introduces an expanded output space, which leads to even greater data requirements and much more labor-intensive ground truth annotation procedures. Highly sophisticated software tools are typically required to characterize the 3D pose of objects that appear in images, which adds to the challenge of creating sufficient training data [25].

To sidestep the need for data labelling, there has been considerable interest in using synthetic images rendered from CAD models as training data [26] [27] [28]. Peng et al. find that a pre-trained network that is fine-tuned on synthetic data can better adapt to new tasks than directly training a network for the new task using few labeled real images [29]. We take a similar approach by bootstrapping our network with weights from the VGG network trained on ImageNet [30], and retraining on synthetic data.

We are also inspired by the work on domain randomization by Tobin et al. [6], which demonstrates that when a model is trained on a large set of unrealistic images that exhibit sufficient variability, the real world can simply be considered as another variation. This approach is especially relevant for the underwater domain since factors such as light absorption and reduced visibility caused by suspended sediment are difficult to simulate. Our synthetic training data renders the target robot using a variety of textures and an assortment of background patterns in order to help our learned model generalize to real images.

III. Method

A. Pose Estimator

We define a multi-output convolutional network that consists of a robot classifier, an orientation regressor (in quaternion form) and a bounding box detector. We train the network on monocular images generated synthetically using the robot's CAD model and various backgrounds constructed in the Unreal [31] game engine. The classifier simply outputs a probability p of the image containing the robot. The orientation regressor head of the network outputs the robot's orientation as a normalized quaternion vector $q = (w, x, y, z)$. The bounding box detector outputs a vector b outlining the diagonal coordinates of the bounding box: $(x_{min}, x_{max}, y_{min}, y_{max})$. These coordinates are normalized with respect to the width and height of the image to lie in $[0, 1]$. Note that directly regressing 3D relative translation will prevent the trained network from operating on cameras with varying focal lengths, so we compute it instead from the estimated orientation and bounding box, as we will discuss later.

We initialize our pose estimation network with the VGG16 architecture as it has been demonstrated to perform well on visual classification tasks [30]. We use VGG weights from pre-training on ImageNet. Our images are resized to $(224, 224, 3)$ to match ImageNet's scaling. We discard the VGG fully connected layers (FC) and augment the network for orientation regression with four FC-ReLU layers, the bounding box regression with two FC-ReLU layers and classification with two FC-ReLU layers.

The loss function we minimize combines the binary cross-entropy for the classification, the L2-norm of the four coordinates of the bounding box and the L2-norm of the orientation quaternion with lambda scale factors to balance the losses. Formally, it is defined as:

$$L = \lambda_b * L_b + \lambda_q * L_q + \lambda_c * L_c \tag{1}$$

where the L2-norms are defined as

$$L_q = \frac{1}{2n} \sum_x ||q_{gt} - q||^2 \tag{2}$$

and

$$L_b = \frac{1}{2n} \sum_x ||b_{gt} - b||^2 \tag{3}$$

and the binary cross-entropy is defined as

$$L_c = -\sum_x (p_{gt} \log(p) + (1 - p_{gt}) \log(1 - p)) \tag{4}$$

To obtain the relative translation t^* of the robot, we solve the following minimization problem:

$$t^* = \underset{t}{\operatorname{argmin}} \|b - \pi(q, t)\|_2 \tag{5}$$

where $b = (x_{min}, x_{max}, y_{min}, y_{max})$ is the detected bounding box, and $\pi(q, t) = (x^{\pi}_{min}, x^{\pi}_{max}, y^{\pi}_{min}, y^{\pi}_{max})$ is the projected bounding box of the robot at translation t with orientation q. Note that computing $\pi(q, t)$ requires that the camera's intrinsic parameters and the robot's absolute scale are known.

During convoying, we run a Kalman-Filter based tracker on the follower robot to integrate the detected translation and orientation of the leading robot and the bounding box over time. We use a PID controller to interface with the autopilot on the follower so that it maintains a fixed pose relative to the leader while both robots swim.

IV. Evaluation

We first evaluate the pose estimation network in isolation on a real underwater test set by measuring mean angle errors across the dataset as well as their distributions over angles. We then evaluate the tracking performance by analyzing the mean orientation errors across entire trajectories of varying complexity using the Kalman-Filtered pose and bounding box estimates. We do this on both synthetic and real trajectories.

A. Pose Estimation Dataset

Our pose estimation dataset consists of a mixture of synthetic images generated with Unreal Engine [31] and real manually annotated underwater images collected during field trials at McGill University's Bellairs Research Institute in Barbados. We restrict the training set to only use synthetic images. This setup highlights our goal of avoiding dependence on the tedious manual annotation process by relying on easily generated synthetic data for training. Our experiments display the network's ability to generalize to real images.

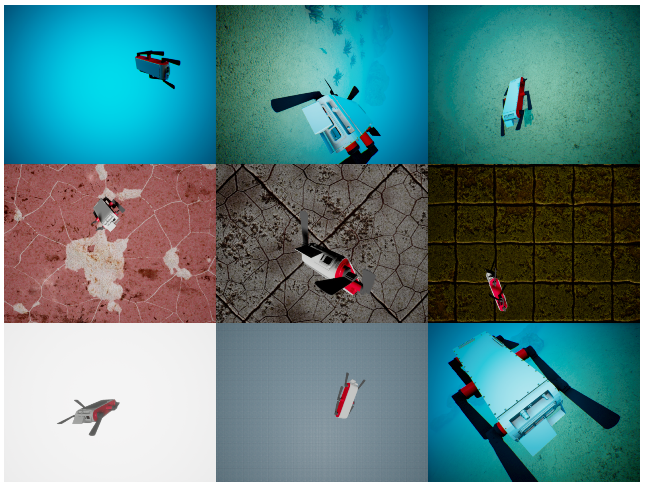


Fig. 2: Image samples from synthetic training data generated using Unreal.

1) Unreal synthetic dataset: The training data consists of 40000 synthetic images generated in Unreal [31] with 45 lighting variations (including varied angle and intensity), 2 robot chassis materials (matte and shiny), 2 custom parts variations on the robot on 4 sets of backgrounds, including a simulated custom-designed underwater world, a simulated pool, and random textures.

In order to increase the realism of the simulated environments, professionally made photo-scanned assets and textures are used from Quixel Megascans [32], including rocks, coral, sand, fish, and plant life. The simulated underwater environment used for both dataset generation and evaluation are inspired by previous footage and data from field trials in Barbados and is approximately $0.5km$ by $0.5km$. The simulated pool environment is created to match the dimensions of the McGill University pool. Visual effects within Unreal Engine such as Exponential Height Fogs and Post Process Materials are added and tuned to mimic the visibility and hues of the real environments.

The Aqua CAD model is randomly placed in the field of view using a randomly generated pose with $[-85°, 85°]$ bounds on the three rotation axes (roll, pitch, yaw) and $[0.5m, 2.0m]$ bounds on the robot's distance from the camera. Image samples of the training data are shown in Fig. 2.

2) Real underwater dataset: Our test set consists of 1000 real images collected during underwater field trials off the west coast of Barbados. The images are captured from diver-held GoPro cameras and an Aqua robot's on-board camera. We annotate the 6-DoF pose of the robot in each of these images using a custom-built annotator, which allows the user to mark keypoints on the robot assigned from the CAD model. The annotator then iteratively fits a wireframe to the robot using its known dimensions.

B. Trajectory generation

1) Unreal synthetic trajectories: In order to evaluate our tracking performance in simulation, we generate a dataset of synthetic videos showcasing the robot performing a variety of trajectories in the Unreal underwater environment. This allows us to quickly compare our tracker's projected trajectory to the Aqua's actual swimming path without the tedious process of annotating real underwater footage frame by frame.

In Unreal Engine, we use the custom-designed underwater world described in Sec. IV-A1 as the backdrop with a gamepad-controlled simulated Aqua model. Exactly 2 meters behind the Aqua model, we place a simulated camera that serves as the tracking camera. Default settings in Unreal Engine cause the camera to translate and rotate perfectly with the object it is following, so we introduce artificial lag that allows the target Aqua to partially escape the camera frame, but to never fully leave the camera view. We then record four distinct videos of the Aqua maneuvering the simulated underwater environment from the point of view of the tracking camera, restricting different rotation axes for each. This provides us with trajectories showcasing the robot rotating only along a) yaw, b) yaw and pitch, c) yaw and roll, and finally d) roll, pitch, and yaw. Along with each saved frame from the tracking camera, ground truth information including the 6-DoF pose and bounding box of the target Aqua and the pose of the camera are stored as a ROS bag file and can be replayed on the real robot if necessary.

2) Real underwater trajectories: We generate a dataset of underwater trajectories from footage captured by an Aqua of a secondary target Aqua in a multitude of underwater environments in Barbados. Unlike the synthetic trajectories, these real trajectories do not display a high amount of variability across the roll and pitch axes and do not include as much clutter. In each video, we annotate every 10 frames and evaluate the trajectory errors over the annotated frames. These trajectories are used in the next section to evaluate tracking performance.

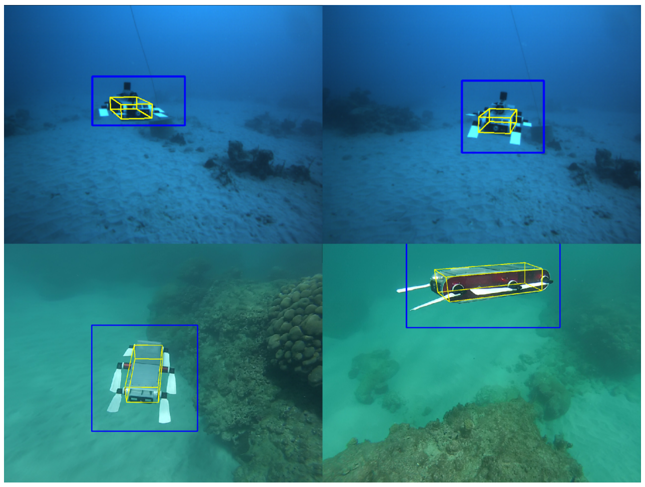


Fig. 3: Sample detections and pose estimates on test images.

V. Results

We summarize results from evaluating both the 3D detection and pose estimation in isolation on our real images test set and the tracking performance on synthetic and real trajectories.

A. Pose Estimation

Qualitative detection results showcasing the wireframe of the robot overlaid on test images are presented in Fig. 3. The images used for this evaluation are real underwater images collected in Barbados from the test set described in IV-A2. We summarize the performance of our model on the test set in Table I. The table shows that pitch dominates the rotational error.

TABLE I: Base metrics evaluated over the real underwater test set collected in Barbados IV-A2.

Mean Rotation Error	**Mean Roll Error**	**Mean Pitch Error**	**Mean Yaw Error**
23.51°	7.29°	12.05°	5.87°

We also measure recall of the orientation estimate from the test set over a number of θ thresholds, 30° usually being the default. Our results are presented in Table II.

TABLE II: Orientation estimate recall for different θ thresholds.

	60°	45°	30°	22.5°	15°	7.5°
Recall	0.89	0.79	0.57	0.40	0.21	0.03

A plot of a randomly sampled subset of angle errors relative to their respective angle value shows a concentration of errors below the 20° mean orientation error. While pitch dominates the rotational error as per Table I, Fig. 4 shows that pitch errors tend to jump after the 50° pitch angle. This might be explained by the loss of view of some important key features of the robot's back. As the robot's top plate is plain aluminum with very little variation, our network might find it more difficult to extract enough features to distinguish pitch angles beyond this mark. The behavior is slightly less present with the yaw axis and almost non-existent in the roll axis which in particular maintains a good view of the back plates of the robot.

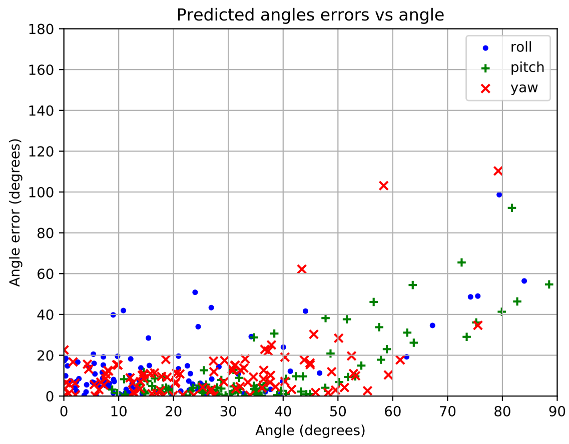


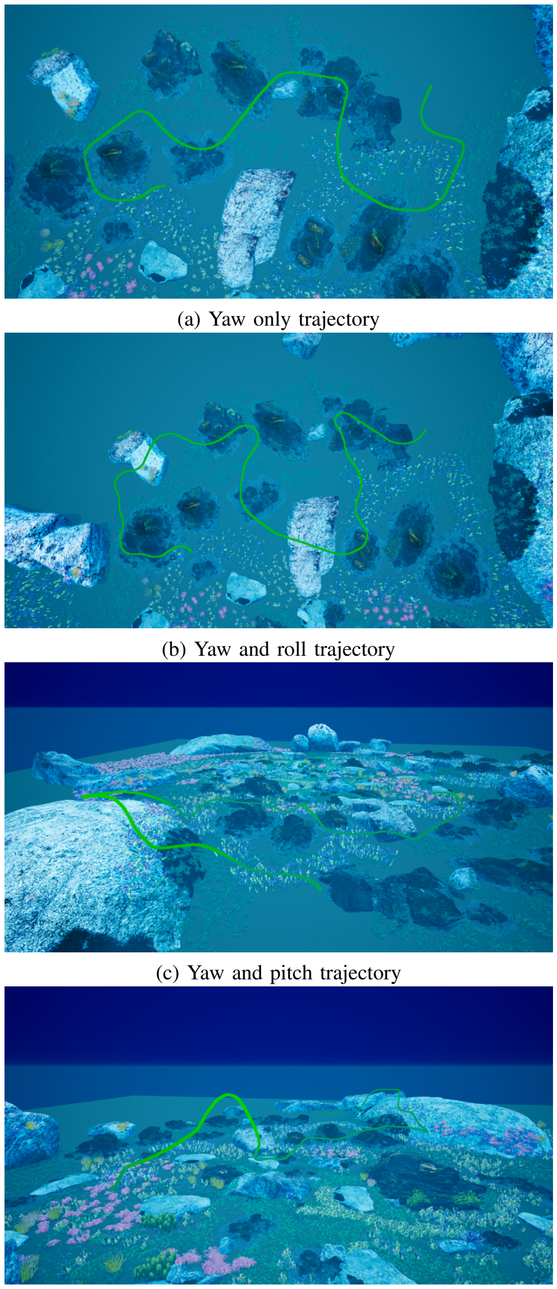
Fig. 4: Angle errors vs angle values for yaw/pitch/roll from a subset of the test set.

B. Trajectories

We measure the root mean squared error (RMSE) of the translation and the orientation of the target's Kalman-filtered pose against the ground truth trajectories that were generated in Unreal or annotated manually. For synthetic trajectories, we restrict various axes of rotations in order to understand the impact of certain axes on the orientation RMSE. The results of our evaluation are summarized in Table III. We note that our calculated translation results in an average error of $< 1m$ while orientation RMSE is on average $< 30°$, consistent with our test set evaluation.

As mentioned in Sec. IV-B2, the real trajectories display a much lighter amount of clutter and much more predictable paths with less variations across the roll/pitch axes. This explains the overall inferior performance on the synthetic trajectories, which include fish animations, heavy rocks and corals and a variety of textures. However, the table demonstrates our system's ability to robustly track the robot in real underwater trajectories despite never training on real images.

To better visualize the trajectories, we include plots of the ground truth trajectories overlaid on the underwater environment for the synthetic trajectories in Fig. 5.



(a) Yaw only trajectory

(b) Yaw and roll trajectory

(c) Yaw and pitch trajectory

(d) Yaw, pitch and roll trajectory

Fig. 5: Plotted ground truth (green) trajectories.

VI. Conclusion

We presented a tracking-by-detection method for tracking the 3D pose of an autonomous underwater vehicle with an aim to improve and enable multi-robot convoying. Our method relies on a pose estimation multi-output convolutional neural

TABLE III: Kalman-Filtered pose errors over synthetic and real trajectories. The tracking camera is located within a range of $[0.5m, 2.0m]$ from the target.

Sequence	**Sequence Length (s)**	**Translation RMSE (m)**	**Orientation RMSE**	**Roll RMSE**	**Pitch RMSE**	**Yaw RMSE**
Synthetic Yaw Only	68	0.60	21.76°	10.92°	8.44°	19.20°
Synthetic Yaw + Pitch	75	0.61	32.86°	29.11°	25.5°	17.37°
Synthetic Yaw + Roll	70	0.79	21.65°	10.63°	14.52°	14.60°
Synthetic Yaw + Pitch + Roll	83	1.29	31.97°	26.57°	16.69°	16.93°
Real Barbados Underwater 1	70	0.72	17.59°	11.87°	4.59°	12.11°
Real Barbados Underwater 1	40	1.17	14.88°	11.74°	5.87°	7.12°
Real Barbados Underwater 1	30	0.40	20.96°	14.23°	7.19°	14.40°

network that jointly predicts the target robot's presence in the image, its 3D orientation and the bounding box that encapsulates the target. Combining this information with the robot's known scale and the camera intrinsics, we compute an estimate of the 3D translation of the robot in order to obtain the full 6-degree-of-freedom pose. We trained exclusively on synthetic training data generated in Unreal engine in order to bypass the tedious task of 3D pose manual annotations. Our system demonstrates the ability to transfer its performance from the learned model to a real underwater dataset and achieves a 23.51° mean rotational error over the entire dataset. Using our pose estimation network, we then apply a Kalman-filter on the pose and bounding box and evaluate our system on a variety of synthetic and real trajectories. In particular, we restrict various axes of rotation on the synthetic trajectories in order to isolate the errors across the axes. Our system achieves a mean translation RMSE of 0.79m and mean orientation RMSE of 23.09° over all the trajectories.

VII. Future Work

The rich geometric information recovered from 3D target tracking can be leveraged for gesture-based communication. Humans use gestures in a variety of settings when other forms of communication are difficult - some examples include aircraft marshalling in aircraft ground handling, and hand gesturing by scuba divers. In robot convoying, [15] leverages gestures performed by the target to communicate its intended heading to the follower, which makes the overall convoy more robust. A method for generic motion-based communication between robots with monocular vision is proposed by [33], in which the observer actively changes its position in order to unambiguously infer the target's trajectory and decode the intended message. We are inspired by this line of work, and we intend to design a visual communication system capable of transmitting generic messages based on 3D gesturing.

Acknowledgment

This work was funded by the NSERC Canadian Field Robotics Network (NCFRN).

References

[1] H. Schneiderman, M. Nashman, A. J. Wavering, and R. Lumia, "Vision-based robotic convoy driving," *Machine Vision and Applications*, vol. 8, no. 6, pp. 359–364, Nov 1995. [Online]. Available: https://doi.org/10.1007/BF01213497

[2] J. J. Lugo, A. Masselli, and A. Zell, "Following a quadrotor with another quadrotor using onboard vision," in *2013 European Conference on Mobile Robots*, Sept 2013, pp. 26–31.

[3] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Experiments in sensing and communication for robot convoy navigation," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 2, Aug 1995, pp. 268–273 vol.2.

[4] V. Chandrasekhar, W. K. Seah, Y. S. Choo, and H. V. Ee, "Localization in underwater sensor networks: survey and challenges," in *Proceedings of the 1st ACM international workshop on Underwater networks*. ACM, 2006, pp. 33–40.

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

[6] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 23–30.

[7] J. Sattar, G. Dudek, O. Chiu, I. Rekleitis, P. Giguère, A. Mills, N. Plamondon, C. Prahacs, Y. Girdhar, M. Nahon, and J.-P. Lobos, "Enabling autonomous capabilities in underwater robotics," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Nice, France, September 2008.

[8] 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.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[10] T. Manderson and G. Dudek, "Gpu-assisted learning on an autonomous marine robot for vision based navigation and image understanding," in *MTS/IEEE OCEANS*, Charleston, SC, USA, Oct. 2018.

[11] Q. Yu, T. B. Dinh, and G. Medioni, "Online tracking and reacquisition using co-trained generative and discriminative trackers," in *European conference on computer vision*. Springer, 2008, pp. 678–691.

[12] M. Manz, T. Luettel, F. von Hundelshausen, and H.-J. Wuensche, "Monocular model-based 3d vehicle tracking for autonomous vehicles in unstructured environment," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2465–2471.

[13] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[14] J. L. Giesbrecht, H. K. Goi, T. D. Barfoot, and B. A. Francis, "A vision-based robotic follower vehicle," in *Unmanned Systems Technology XI*, vol. 7332. International Society for Optics and Photonics, 2009, p. 73321O.

[15] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Experiments in sensing and communication for robot convoy navigation," in *iros*. IEEE, 1995, p. 2268.

[16] 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.

[17] 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.

[18] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 48–55.

[19] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing ikea objects: Fine pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2992–2999.

[20] S. Fidler, S. Dickinson, and R. Urtasun, "3d object detection and viewpoint estimation with a deformable 3d cuboid model," in *Advances in neural information processing systems*, 2012, pp. 611–619.

[21] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 808–816.

[22] C. Li, Q. Tran, M. Zia, G. Hager, and M. Chandraker, "Deep supervision with shape concepts for occlusion-aware 3d object parsing," in *CVPR*, 2017.

[23] H. Izadinia, Q. Shan, and S. M. Seitz, "Im2cad," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 2422–2431.

[24] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[25] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference Computer Vision (ECCV)*, 2016.

[26] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[27] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?" in *European Conference on Computer Vision*. Springer, 2016, pp. 202–217.

[28] B. Sun and K. Saenko, "From virtual to reality: Fast adaptation of virtual object detectors to real domains." in *BMVC*, vol. 1, no. 2, 2014, p. 3.

[29] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3d models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1278–1286.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[31] E. Games, "Unreal engine," https://www.unrealengine.com/en-US/what-is-unreal-engine-4, 1998-2018.

[32] Quixel, "Quixel megascans," https://megascans.se/, 2018.

[33] *Active Motion-Based Communication for Robots with Monocular Vision*, Brisbane, Australia, 2018 2018.