Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Deep learning based keypoint rejection system for underwater visual ego-motion estimation ⋆

Marco Leonardi * **Luca Fiori** ** **Annette Stahl** *

* *Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, NTNU AMOS, Trondheim, Norway. e-mails: marco.leonardi@ntnu.no, annette.stahl@ntnu.no*
** *Department of Information Engineering and Mathematics, University of Siena, Italy, e-mail: luca.fiori@student.unisi.it*

Abstract: Most visual odometry (VO) and visual simultaneous localization and mapping (VSLAM) systems rely heavily on robust keypoint detection and matching. With regards to images taken in the underwater environment, phenomena like shallow water caustics and/or dynamic objects like fishes can lead to the detection and matching of unreliable (unsuitable) keypoints within the visual motion estimation pipeline. We propose a plug-and-play keypoint rejection system that rejects keypoints unsuitable for tracking in order to obtain a robust visual ego-motion estimation. A convolutional neural network is trained in a supervised manner, with image patches having a detected keypoint in its center as input and the probability of such a keypoint suitable for tracking and mapping as output. We provide experimental evidence that the system prevents to track unsuitable keypoints in a state-of-the-art VSLAM system. In addition we evaluated several strategies aimed at increasing the inference speed of the network for real-time operations.

Keywords: keypoints, monocular SLAM, VSLAM, Visual Odometry, VO, underwater navigation, underwater SLAM, noise filtering

1. INTRODUCTION

Visual odometry and VSLAM have been successfully applied in the underwater domain, as for example for ship-hull inspection Kim and Eustice (2013), Kim and Eustice (2009) or autonomous underwater navigation Bonin-Font et al. (2015). Other previous work relies mainly on inertial navigation system (INS) measurements Krys and Najjaran (2007), while a very modern work proposes feature based visual odometry Ferrera et al. (2019). The interest in this topics is continuously growing and lately a dataset dedicated to real-time visual underwater localization has been published by Ferrera et al. (2018). In addition, many SLAM systems for dynamic non-underwater environments have been presented by Bescos et al. (2018), Cui and Wen (2019), and Tan et al. (2013). Unfortunately most of these systems are very sensitive to moving objects in the scene. Therefore, our system aims to prevent keypoints detected at moving objects or dynamic textures, which in a state-of-the-art approach might have been considered for ego-motion estimation and mapping. Keypoints are prevented by using the prior learned by a neural network, so that, depending on the VO/VSLAM system, the tracking is not affected by the presence of moving objects or dynamic textures in the scene.

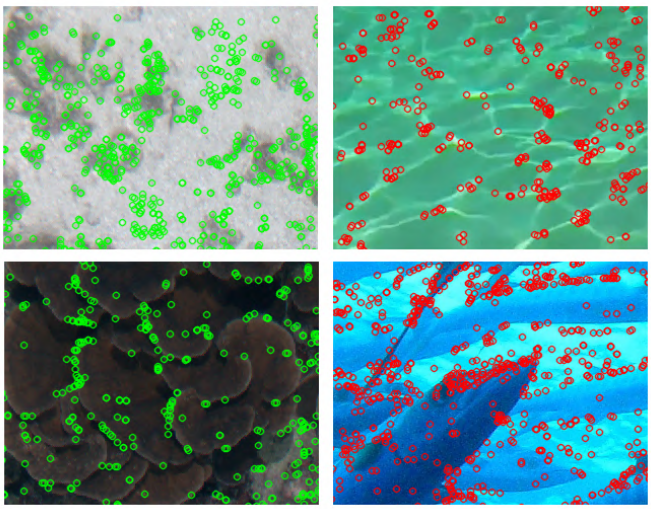

Fig. 1. Keypoints labeled with our system: Keypoints classified as *suitable* are circled in green, keypoints classified as *unsuitable* are circled in red. Top left: sandy seabed, top right: caustics, bottom left: seabed with vegetation, bottom right: a *school* of fishes.

We provide experimental evidence of this statement in the result section (section 4).

⋆ This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU.

Fig. 2. Example of *unsuitable* (left) and *suitable* (right) keypoint image patches. *Unsuitable* image patches can contain fishes, a crab, seaweed, caustics, clean water with high gradient zones, and the effect called *marine snow* (first row, third image to the right). *Suitable* image patches can contain for example man-made objects and different sea bed types.

We extract patches centered around each detected keypoint as input for a convolutional neural network (CNN) and train it to distinguish if the keypoint is *suitable* for tracking or not. We define keypoints as *suitable* which are detected at (static) surfaces, that are not supposed to move (e.g. rocks), and exclude those keypoints - defined as unsuitable - which are detected at moving objects, textures or dynamic physical phenomena (eg. fish, caustics, etc.; see Fig. 1). Thereby, keypoint neighborhood image scene information serves as context for the CNN, which is trained to distinguish if the detected keypoint belongs to a moving surface or not. This prior knowledge based selection makes it possible to forward only keypoints to a motion estimation pipeline that are exploitable for a reliable ego-motion estimation.

The two main contributions presented in this paper are:

- A fast supervised way to generate a dataset for keypoint classification
- A novel method for reliable keypoint selection for underwater visual ego-motion estimation

1.1 Related Work

In Wangsiripitak and Murray (2009) a parallel implementation of monoSLAM was presented, that incorporates a 3D object tracker into the SLAM system. Moving features are thereby not included into the map and features that are known to be occluded by objects are deleted, but as this work does not detect moving objects, non-stationary features apart from the ones laying on the tracked moving object will still corrupt the SLAM estimation. In a similar way, Riazuelo et al. (2017) implemented an object tracker dedicated to people tracking. These methods would detect those *a priori* dynamic objects, but fail to react on movements of *a priori* static objects.
In Tan et al. (2013) the authors compare features between keyframes and the current frame for 3D structure validation. This method fails when *a priori* dynamic objects remain static (like e. g. lifeless marine fauna).
Recently, an ORB-SLAM Mur-Artal et al. (2015) based approach was presented that operates reasonable well in dynamic environments Cui and Wen (2019). While ORB-SLAM is already implicitly robust to small changes in the scene, the authors improve the robustness of the algorithm by comparing image patches around the re-projected 3D point between the current frame and the reference frame. However, while the system is reported to increase the tracking performance in dynamic environments, it's not able to cope with scenes where many dynamic objects with a coherent motion pattern are present (e.g. a school of fishes). The authors suggest the use of object recognition techniques to improve the robustness, since dynamic scenes can generate a valid apparent ego-motion.
Similar to the work presented in this paper DynaSLAM Bescos et al. (2018), and ORB-SLAM2 Mur-Artal and Tardós (2017), a monocular approach uses Mask R-CNN He et al. (2017) which is state-of-the-art for object instance segmentation. The deep network architecture is designed to classify and to reject keypoints associated with regards to a list of dynamic object labels (i.e. person, bicycle, car, etc.). Unfortunately, DynaSLAM doesn't work in real-time, as Mask R-CNN alone runs at around 195 ms per image on a Nvidia Tesla M40 GPU He et al. (2017). Only a few underwater labeled datasets are available, while to the best of our knowledge no datasets exist with respect to semantic segmentation OByrne et al. (2018).
Very recently, an underwater VO and feature-based SLAM approach Ferrera et al. (2019) was published not taking into account the dynamics of underwater environments, exposing the system to be highly sensible to fishes, shallow water caustics and other dynamic objects present, including seaweed transported by current.
Our method falls into the category where keypoints laying on *a priori* dynamic objects will always be removed, even if the objects are most likely going to remain static for the time they are going to be observed (e.g. lifeless marine fauna). We argue that the *a priori* dynamic objects are extremely unlikely to remain static in the underwater environment, and that these objects should never be used

for Visual SLAM as they could be only momentarily static (e.g. a fish resting on the seabed).

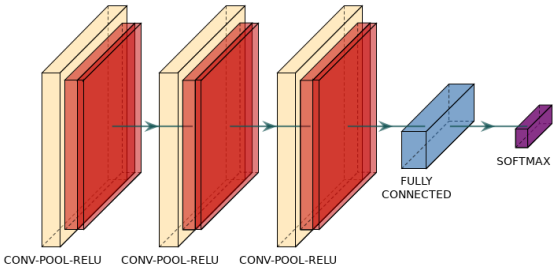

Fig. 3. Representation of the network architecture: the first three layers are a stack of CONV-POOL-RELU layers, extracting features for the fully connected layer. The output layer is a softmax layer.

Compared to methods that use deep neural networks for object-instance segmentation, our approach does not need to create masks for the training procedure, and the overall execution time is also positively affected, thanks also to the highly-parallelizable nature of the problem. Patch creation and analysis is not directly tied to image resolution, making a network trained in this way more future-proof.

2. DATABASE CREATION

We composed our dataset of underwater images from the following datasets: the *Tasmania Coral Point Count*, the *Scott Reef 25* and the *Tasmania O'Hara 7* from the Australian Centre for Field Robotics for Field Robotics (2009), the *An Underwater Observation Dataset of Fish* Eickholt (2018) and images from the *Herkules wreck site* coming from several field surveys the authors conducted Leonardi et al. (2017). For our tests 110 images out of this dataset were selected and in total 13158 patches were extracted (see Fig. 1), where 60% were used for the training set, 20% for the validation and 20% for the test set. Possible underwater dynamic image scenes are presented, including various fish species, crabs, algae, floating particles, as well as illumination changes and effects and man-made floating objects.

The dataset is created utilizing a manual labelling procedure. For each image, up to 1000 keypoints are extracted using the Oriented FAST and Rotated BRIEF (ORB) Rublee et al. (2011) feature detector by setting the FAST threshold to 20, which is a commonly used value.

Images are proportionally scaled, keeping the aspect-ratio, considering the first image as reference; this process aims to consistently scale context information in the patches centered in the keypoints.

We implemented a software interface to ease the process of differentiation between *suitable* keypoints (laying on *a priori* static objects) and *unsuitable* keypoints (laying on *a priori* dynamic objects), see Fig. 4. Our software visualizes the image and shows the ORB extracted keypoints as dots. In the interface the user can draw multiple independent polygons as successions of segments. Keypoints inside the polygons are marked *unsuitable*, while keypoints outside the polygons are marked as *suitable*. Image patches of 257×257 pixels (cf. Fig. 2) are extracted around each keypoint. The size of the patches has been selected in a process of trial and error, with the goal of jointly maximizing the contextual information and the inference time performance of the resulting network. All patches corresponding to *suitable* keypoints are stored separately from those corresponding to *unsuitable* keypoints.

3. NETWORK ARCHITECTURE

In this section we describe the details of the network architecture as illustrated in Fig. 3. In our implementation the keypoint patches have been resized to 65×65 pixels. This choice was based on empirical tests, since the patches where originally 257×257 pixels to preserve context information. For a higher resolution (129×129 and 97×97 pixels) of the patch the network did not gain in precision, instead decreasing its performance in both memory and speed. We recommend, that the size of the extracted rectangular patches should be around 2/10 of the image width, considering a maximum aspect ratio of 16:9. The idea is that it should be possible for a human operator to visually analyze and discriminate successfully the patches. The proposed network is a convolutional neural network, counting three *convolutional* (CONV) layers, one *fully-connected* (FC) layer and a *soft-max* layer.

The first layer is a CONV layer, consisting of 16 filters with a kernel size of 3×3, stride of 1 in each direction, zero filling padding strategy, followed by a 2×2 *max-pooling* (POOL) and a *rectified linear unit* (RELU) layer for adding non-linearity. The second and third layer are identical to the first one. It follows a *flatten* layer which is needed to format the input for the following FC RELU layer. The FC layer (128x128) is followed by a soft-max RELU layer with two outputs, representing the *suitable*/*unsuitable* posterior class probabilities, given the input and the network configuration. The neural network architecture is inspired by the original LeNet-5 LeCun et al. (1998) network, with a different amount of layers and a different layer dimensioning due to the more complex discrimination problem. Different network topologies and configurations have been tried, the one we just presented is the network that provided us with the best ratio between the amount of network weights and classification performance.

3.1 Training procedure

The training of the network was performed over the dataset described in section 2, with the patches resized to 65×65 pixels. The *batch size* was set to 64 and the network is trained for 1000 epochs. The set of weights which is retained is the one that performs best in terms of validation loss. The optimizer used is adaptive moment estimation (ADAM) Kingma and Ba (2014), with a learning rate of 0.001 and $\epsilon = 1e-10$. The loss function used in both training and validation is the mean square error (MSE). The target follows the *one hot encoding*. In this way the inference output of the network approximates posterior probabilities Richard and Lippmann (1991). We obtained an accuracy of 96.7% on the test set. In Table 1 the final confusion matrix is presented.

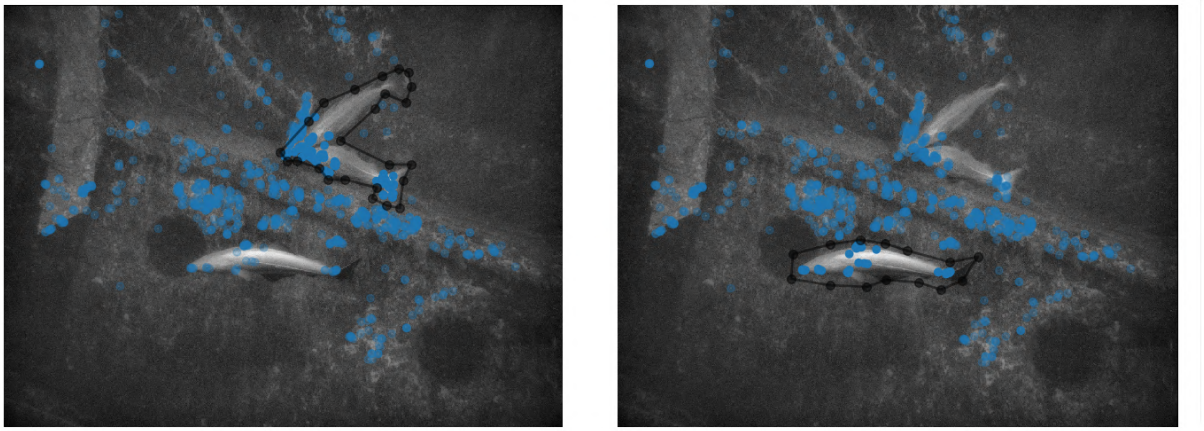

Fig. 4. Software interface procedure for the selection of *unsuitable* keypoints. During this procedure, not all of the *unsuitable* keypoints are selected by drawing only one polygon. As soon as the first polygon is selected, the procedure asks to eventually draw another polygon. In the picture on the left, first a polygon is drawn to enclose the keypoints on the fishes in the top-right quadrant, then in the picture on the right a second polygon is drawn to select the keypoints laying on the remaining fish.

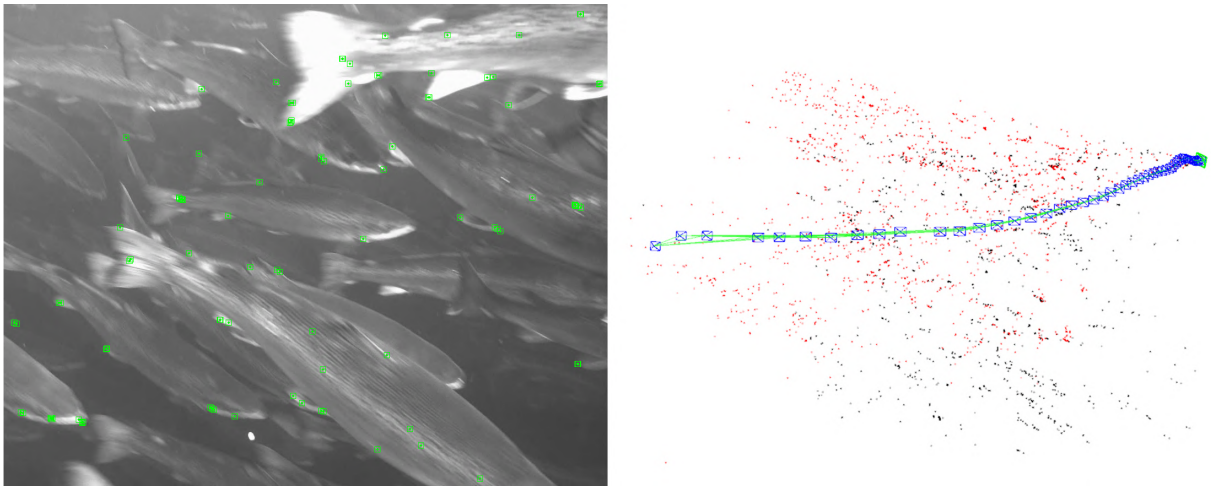

Fig. 5. Example of ego-motion estimation through a fish swarm present in the LAKSIT sequence 1, generated by monocular ORB-SLAM with a tuned initialization procedure as described in this paper. On the left the current image with the tracked keypoints and on the right the 3D map with the camera poses (in blue) with the covisibility links (in green). Initialization, mapping and tracking operations that follow rely entirely on keypoint correspondences that are laying on moving objects. To be noted that keypoints highlighted in green on the left part of the image are the keypoints currently tracked by ORB-SLAM, here they are not indicating *suitable* keypoints found by our neural network.

Patches: 2631	Pred. Suitable	Pred. Unsuitable
Suitable	1532 - TP	19 - FN
Unsuitable	68 - FP	1012 - TN

Table 1. Confusion matrix calculated over the test set (cf. section 2): True positive (TP), True negative (TN), False positive (FP), False negative (FN). Percentages: FP 2.6%, FN 0.7%

4. RESULTS

In this section we present results in terms of performance in drift reduction and in terms of inference speed, with different execution configurations and different network floating point precision. During all the tests no pre-processing, like histogram manipulations or brightness augmentation, is performed. For analyzing the capabilities of drift reduction, we analyzed several footages recorded by a static camera inside a fish cage, used at a fish farming site (LAKSIT project Føre, M., Frank, K., Svendsen, E., Schellewald, C., Sunde, L.M., Alfredsen, J.A. and Stahl, A. (2017), to this date the dataset is not publicly available, but it may be released in the future). From the LAKSIT dataset, we selected 3 image sequences composed of 3120 images each with a resolution of 1280×1024 pixels. We compare our system also with DynaSLAM Bescos et al. (2018), which is based on ORB-SLAM. The monocular ORB-SLAM is not always able to initialize on such sequences, because the apparent relative motion is not always consistent or the image is too blurred. Therefore, in

order to obtain a higher initialization rate and in order to demonstrate that 3D map-points are generated from keypoints which belong to fishes, we decreased the amount of keypoints required for the initialization step to 25. Since ORB-SLAM enters the *relocalization mode* only after the tracking is lost (and a minimum of 5 keyframes have been generated), several different sub-sequences have been chosen from the sequences, in order to find the longest drift accumulated in each sequence (see Fig. 5). The results are presented in Table 2.

With this keypoint rejection system, the ORB-SLAM system does not have to (re-)initialize, and drift does not accumulate over time, as most of the features detected at the moving objects would not be considered for mapping and tracking.
Approaches aiming to perform keypoint rejection for visual ego-motion estimation (like the one presented in this paper) using only *a priori* knowledge, may also reject *suitable* keypoints. Examples of this behaviour are shown in Fig. 6. We assume that in the underwater natural environment it is unlikely that *a priori* dynamic objects will remain static, and for this reason our approach is negligible impaired by this effect.

DynaSLAM provides two different rejection systems for the keypoints: a geometric test and a deep learning based method. While the geometric test is ineffective when elements in the scene are moving in a coherent way, the deep learning based method, which masks regions that should belong to moving objects, is effective in reducing the drift, even if the network is not trained for underwater scenarios.
Our system was also tested using the RT MVO dataset Ferrera et al. (2019) without benchmarking (unavailability of open source support). DynaSLAM as well as ORB-SlAM were tested with our proposed keypoint rejection system on the RT MVO data set and a similar performance for both systems with respect to the estimated trajectories were obtained. This is because the RT MVO sequences does show a very clear sea bed and the moving objects present in the sequences are only very small fishes, not showing for example any *schooling behaviour*.

Seq.	ORB-SLAM	DynaSLAM	DynaSLAM-M	Ours
1	29.13m	45.57m	7.75m	0m
2	11.24m	0m	0m	0m
3	14.68m	22.32m	3.09m	0m

Table 2. Absolute translation error (ATE) accumulated by ORB-SLAM with respect to three sequences out of the LAKSIT dataset, without and with the keypoint rejection system described in this paper (norm of the translation vector between the two initialization frames set to 1). ATE of 0m means that no initialization did occur.

The next experiments show the result of the different execution configurations with respect to floating point precision, parallel instances of the network and batch inference (we refer to Table 3). Note, due to the incompatibility to the Tensorflow *frozen model* with a multi-thread session creation procedure, no test has been performed with a parallel implementation running half precision floating point models.

Table 3 shows the results of our network with respect to execution speed using a desktop PC featuring 64Gb of RAM, a Nvidia GeForce RTX 2080 Ti and an Intel Core i9-9900K 8 Core @ 3.6Ghz, using a Python 3 implementation. The results are evaluated over a set of 5426 patches extracted from 10 test images. The various configurations evaluated differ by the type of processing unit utilized (CPU or GPU), the floating-point precision of the model, the number of threads launched (each executing a different instance of the graph; each instance elaborates an equal amount of patches), and whether we pre-loaded all the test patches in a single feeding dictionary. Regarding the floating-point precision, we tested the standard model created by Tensorflow, which uses 32 bit floating-point values both for constants and variables in the graph against an optimized version using 16 bit floating-points generated using the Nvidia TensorRT library. The values of mean and standard deviation in the table are computed over the entire images, averaging 543 keypoints (thus patches) each.

In Table 4 we compare the inference speed of DynaSLAM (Mask R-CNN) and the network used in this paper. The mean inference time has been obtained averaging over the time that took DynaSLAM to generate 100 masks from underwater images. Our approach is eight times faster on CPU and almost four times faster on GPU.

Our experiments showed that, given our network architecture and hardware, it is possible to achieve the highest inference speed with FP32 operations and 32 parallel network inferences on the GPU, performing at around 16 FPS.

5. CONCLUSION AND FURTHER WORK

In this paper a keypoint classification system is used to improve the robustness of visual ego-motion estimation in underwater environments. The procedure involved uses a deep convolutional neural network in order to classify each keypoint exploiting the surrounding image patch information. Only the keypoints which are classified as *suitable* for tracking will be retained for motion estimation.

The approach has been verified by comparing the drift accumulated by ORB-SLAM with a static camera recording a dynamic image scene of fishes showing a *schooling behaviour* with and without the keypoint rejection system described in this paper. The approach has been verified and benchmarked also against DynaSLAM, a state-of-the-art monocular visual SLAM system which accounts for dynamic environments.

We also discussed the inference performance for real-time operation by using state-of-the-art optimization frameworks for deep neural networks and parallel inference on CPU and GPU. The results show that the best performance can be achieved with a GPU acceleration, a floating point 32 network with 32 parallel sessions, even if the performance flattens out already under 16 parallel sessions. The method proposed in this paper enables feature-based underwater visual ego-motion estimation systems to oper-

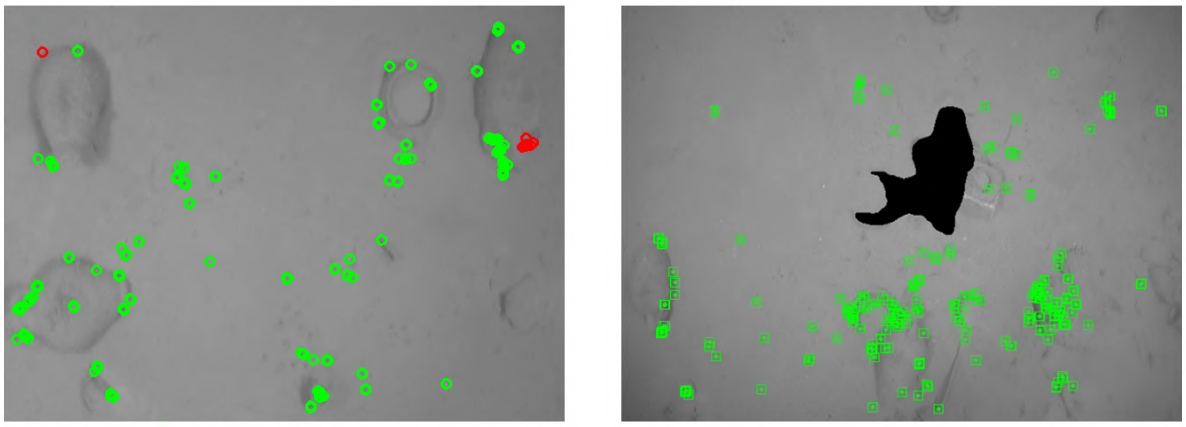

Fig. 6. This image shows the intrinsic risk of using the keypoint/area labeling algorithm: *suitable* keypoints/areas can be labeled as *unsuitable*, lowering the overall robustness of the visual motion estimation. Two different frames, coming from Sequence 1 of the RT MVO dataset are shown. On the left, circled in red, keypoints wrongly labeled as *unsuitable* from the network described in this paper are shown. On the right a masked area which did contain *suitable* seabed keypoints from DynaSLAM (Mask R-CNN) are shown.

Processing Unit	Mean	Std. Dev.	FP Precision	Threads	Batch
Graphic	0.134s	0.065s	32	1	✗
Graphic	0.080s	0.041s	32	4	✗
Graphic	0.067s	0.034s	32	16	✗
Graphic	**0.062s**	**0.031s**	32	32	✗
Graphic	0.066s	0.032s	32	64	✗
Graphic	1.139s	0.568s	32	1	✓
Graphic	1.139s	0.314s	*16*	1	✓
Central	0.377s	0.199s	32	1	✗
Central	0.413s	0.213s	*16*	1	✗
Central	0.213s	0.110s	32	8	✗
Central	0.201s	0.097s	32	16	✗
Central	3.401s	1.941s	32	1	✓
Central	3.359s	1.667s	*16*	1	✓

Table 3. Results of different network configurations for an inference speed analysis. *Batch* denotes that the inference is performed on all the patches in a single session.

Algorithm	Processing unit	Mean	Std. Dev.
Mask R-CNN	Central	1.665s	0.147s
This paper	Central	**0.201s**	0.167s
Mask R-CNN	Graphic	0.230s	0.284s
This paper	Graphic	**0.062s**	0.188s

Table 4. Inference speed of Mask R-CNN (the network present in DynaSLAM, which performs state-of-the-art object instance segmentation) versus the network present in this paper. The reported mean inference time corresponds to the best performing parallel implementation for both networks.

ate at close distance from regions of interest for underwater robotic applications, such as shipwreck or barrier reef monitoring, which are rich in marine life (causing dynamic image scenes). The proposed approach can be improved by using more training data representing unknown fish species, rock types, or other objects our CNN was not trained on. Data augmentation techniques, such as rotation, contrast manipulation and noise injection are likely to improve the generalization capabilities of the network. Therefore, it is advisable to train the network in such a manner that the false negative rate (i.e. the *suitable* keypoints that get classified as *unsuitable* keypoints) is as low as possible. Future work can also include optimizing the network for inference speed on more robotics-oriented computing platforms like the Nvidia Jetson AGX Xavier Nvidia (2019a) or even the Nvidia Jetson Nano Nvidia (2019b). Furthermore, the problem can be approximated by grouping neighbor keypoints and so reducing the amount of patches to be evaluated for each frame.

6. ACKNOWLEDGEMENTS

We would like to thank Christian Schellewald and Martin Føre for providing the three images sequences belonging to the LAKSIT project, the Australian Centre for Field Robotics, especially the marine robotics group for providing the *Scott Reef 25* and the *Tasmania O'Hara 7* dataset, the Institute for Marine and Antarctic Studies, University of Tasmania, for providing the *Tasmania Coral Point Count* and Jesse Eickholt, from the Central Michigan University, for providing the *An Underwater Observation Dataset of Fish*. This work was supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 - NTNU AMOS.

REFERENCES

Bescos, B., Fácil, J.M., Civera, J., and Neira, J. (2018). Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4), 4076–4083.

Bonin-Font, F., Oliver, G., Wirth, S., Massot, M., Negre, P.L., and Beltran, J.P. (2015). Visual sensing for autonomous underwater exploration and intervention tasks. *Ocean Engineering*, 93, 25–44.

Cui, L. and Wen, F. (2019). A monocular orb-slam in dynamic environments. In *Journal of Physics: Conference Series*, volume 1168, 052037. IOP Publishing.

Eickholt, J. (2018). An underwater observation dataset of fish. https://osf.io/3pyud/. Accessed: 2019-04-01.

Ferrera, M., Moras, J., Trouvé-Peloux, P., and Creuze, V. (2019). Real-time monocular visual odometry for turbid and dynamic underwater environments. *Sensors*, 19(3), 687.

Ferrera, M., Moras, J., Trouvé-Peloux, P., Creuze, V., and Dégez, D. (2018). The aqualoc dataset: Towards real-time underwater localization from a visual-inertial-pressure acquisition system. *arXiv preprint arXiv:1809.07076*.

for Field Robotics, A.C. (2009). Marine robotics datasets. http://marine.acfr.usyd.edu.au/datasets/. Accessed: 2019-06-01.

Føre, M., Frank, K., Svendsen, E., Schellewald, C., Sunde, L.M., Alfredsen, J.A. and Stahl, A. (2017). Final report on the project "Teknologi for nye datatyper og informasjon som beskriver situasjon og tilstand hos laksefisk i kommersielle merder (LAKSIT)" FHF Pnr. 901184. SINTEF report OC2017 A-104. Accessed: 2019-05-15.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

Kim, A. and Eustice, R. (2009). Pose-graph visual slam with geometric model selection for autonomous underwater ship hull inspection. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1559–1565. IEEE.

Kim, A. and Eustice, R.M. (2013). Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, 29(3), 719–733.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krys, D. and Najjaran, H. (2007). Development of visual simultaneous localization and mapping (vslam) for a pipe inspection robot. In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, 344–349. IEEE.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Leonardi, M., Stahl, A., Gazzea, M., Ludvigsen, M., Rist-Christensen, I., and Nornes, S.M. (2017). Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles. In *OCEANS 2017-Aberdeen*, 1–10. IEEE.

Mur-Artal, R., Montiel, J.M.M., and Tardos, J.D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5), 1147–1163.

Mur-Artal, R. and Tardós, J.D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5), 1255–1262.

Nvidia (2019a). Jetson agx xavier. https://developer.nvidia.com/embedded/buy/jetson-agx-xavier. Accessed: 2019-06-01.

Nvidia (2019b). Jetson nano. `https://developer.nvidia.com/embedded/buy/jetson-agx-xavier-devkit`. Accessed: 2019-06-01.

OByrne, M., Pakrashi, V., Schoefs, F., and Ghosh, B. (2018). Semantic segmentation of underwater imagery using deep networks trained on synthetic imagery. *Journal of Marine Science and Engineering*, 6(3), 93.

Riazuelo, L., Montano, L., and Montiel, J. (2017). Semantic visual slam in populated environments. In *2017 European Conference on Mobile Robots (ECMR)*, 1–7. IEEE.

Richard, M.D. and Lippmann, R.P. (1991). Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4), 461–483.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf.

Tan, W., Liu, H., Dong, Z., Zhang, G., and Bao, H. (2013). Robust monocular slam in dynamic environments. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 209–218. IEEE.

Wangsiripitak, S. and Murray, D.W. (2009). Avoiding moving outliers in visual slam by tracking moving objects. In *2009 IEEE international conference on robotics and automation*, 375–380. IEEE.