Autonomous Robots (2023) 47:1367–1385
https://doi.org/10.1007/s10514-023-10138-0

UVS: underwater visual SLAM—a robust monocular visual SLAM system for lifelong underwater operations

Marco Leonardi[1] · Annette Stahl[1] · Edmund Førland Brekke[1] · Martin Ludvigsen[2]

Received: 12 October 2020 / Accepted: 21 August 2023 / Published online: 22 September 2023

Abstract
In this paper, a visual simultaneous localization and mapping (VSLAM/visual SLAM) system called underwater visual SLAM (UVS) system is presented, specifically tailored for camera-only navigation in natural underwater environments. The UVS system is particularly optimized towards precision and robustness, as well as lifelong operations. We build upon Oriented features from accelerated segment test and Rotated Binary robust independent elementary features simultaneous localization and mapping (ORB-SLAM) and improve the accuracy by performing an exact search in the descriptor space during triangulation and the robustness by utilizing a unified initialization method and a motion model. In addition, we present a scale-agnostic station-keeping detection, which aims to optimize the map and poses during station-keeping, and a pruning strategy, which takes into account the point's age and distance to the active keyframe. An exhaustive evaluation is presented to the reader, using a total of 38 in-air and underwater sequences.

Keywords Underwater VSLAM · Visual SLAM · Loop closure · Monocular SLAM · Underwater navigation

1 Introduction

Navigation systems are critical components of most autonomous and non-autonomous robotic systems, and alternative localization systems are welcome, if not necessary, especially in environments where satellite navigation is most likely not available. In this work, we present a novel visual SLAM system for underwater camera-only navigation in natural environments.

This work was supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254—NTNU Centre for Autonomous Marine Operations and Systems (AMOS).

✉ Marco Leonardi
marco.leonardi@ntnu.no

Annette Stahl
annette.stahl@ntnu.no

Edmund Førland Brekke
edmund.brekke@ntnu.no

Martin Ludvigsen
martin.ludvigsen@ntnu.no

[1] Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems NTNU AMOS, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

[2] Department of Marine Technology, Applied Underwater Robotics Lab, NTNU NTNU AMOS, Trondheim, Norway

Underwater imaging is a complex matter, particularly in a natural environment, as many factors intervene in lowering the image quality, like back-scattering from suspended particles and distortion effects introduced by light passing through different materials (water, glass, air). Nevertheless, there are several positive aspects that can be exploited, like a smooth motion of the vehicle, a much lower vehicle speed compared to other kinds of robotic platforms, and the capability to employ more computational resources, due to much lower restrictions on weight and available power compared to popular robotic platforms like multi-rotors. Our focus is on reliability, continuity, and robustness in the pose estimation, while generating enough mapping information for navigation, including obstacle avoidance. Our system is built upon ORB-SLAM [36], which is currently one of the most successful state-of-the-art visual SLAM systems.
The main contributions of this paper are:

- Three view initialization procedure, which does not use a model selection procedure
- A fast, exact solution applied to nearest neighbor search for triangulation, increasing tracking robustness

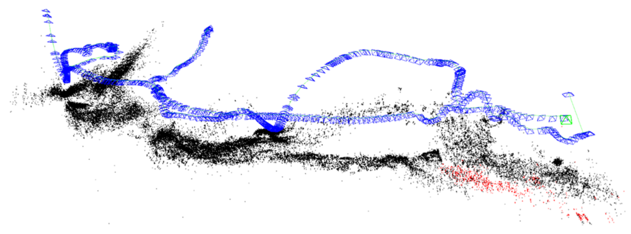

Fig. 1 VSLAM estimate generated on sequence 4 of the Aqualoc dataset. No other tested monocular VSLAM state-of-the-art system is able to produce a meaningful result for this sequence

- A model that predicts the camera pose in absence of visual information and map consistency
- Partial synchronization between front-end and back-end
- A procedure to detect station-keeping operations while using the available computing power for performing a global bundle adjustment
- The same descriptor-matching solution applied for triangulating new points is applied for loop closures detection
- A pruning procedure that enables lifelong operations

In summary, by strengthening the worst-performing elements of ORB-SLAM in underwater sequences and by adding new functionalities which specifically target the underwater scenario, we improved the overall performance of the system both in terms of median root mean square error (RMSE) by 34.6% and in terms of loop closure quantity by 33.3%. As a result, we can produce SLAM estimates on sequences where it was not possible before, see Fig. 1.

In the following section, a literature review is presented, together with an evaluation of systems for underwater visual navigation. Subsequentially, an overview in Sect. 2 of the proposed system and dedicated Sects. (3–9) for each contribution are presented. Finally, in Sect. 10 results are compared against ORB-SLAM and Direct Sparse Odometry with Loop Closure (LDSO) using in-air datasets and a newly published underwater dataset that targets visual odometry (VO) and visual SLAM.

1.1 Related work

1.1.1 Monocular visual odometry and SLAM

Recently [21, 31, 39], utilized edge-based features within the VO/VSLAM pipeline, but pointed out that these methods do perform better than point-only methods only in artificial environments. On the contrary underwater natural environments do rarely contain geometrically well-defined edges like in an office or street scene, so using edge features in this context can introduce false correspondences.

Work in visual odometry includes direct methods like direct sparse odometry (DSO) [8] and fast semi-direct monocular visual odometry (SVO)-2 [12]. These methods rely either partially or even completely on the change of pixel intensity values. Usually, these methods are not able to achieve the same performance, in terms of quality of the estimation of geometry-based VO/VSLAM methods like ORB-SLAM, as they require photometric calibration to remove vignetting effects, control, and modeling of exposure, gain, and luminosity (amongst other camera-specific parameters). Direct methods operating in underwater environments would need to take into consideration that the intensity of pixels is a function of the observation distance (given that attenuation of light in water is wavelength dependent). Challenges would also arise from a likely uneven active illumination of the scene and the presence of dynamic elements in the scene, like fish swimming in front of the camera.

Several artificial intelligence techniques that exploit convolutional neural networks (CNNs) have been proposed for performing VO/VSLAM [27, 34, 44]. Also, the direct fusion of inertia measurement unit (IMU) measurements has been demonstrated to be possible and effective [5]. These methods have the advantage of not requiring any knowledge about the camera and can provide a metric reconstruction with a monocular camera. Training CNNs in such systems do require a ground truth, but unfortunately, ground truth data for natural underwater environments are not available. Producing ground truth data for underwater positioning is very challenging and in the case of real image sequences in general of lower quality compared to what can be achieved with in-air motion capture systems or real-time positioning (RTK-GPS) solutions. However, synthetic underwater data sets [19, 47] exist.

1.1.2 Underwater specific

Due to various challenges present in underwater vision, many underwater VO/VSLAM and photogrammetry-specific solutions have been proposed that make use of a stereo system [3, 16, 20, 26, 40, 43, 46] whereas monocular system solutions are rarely available. However, reduced systems, like visual pose estimation approaches in underwater scenarios do exist [7, 41].

Recently, a monocular geometry-based approach [9] that makes use of the Lucas-Kanade optical flow to identify keypoint matches has been published; while this approach requires low computational resources, it has its limitations regarding the tracking system. Because of its local formulation, keypoint matches are restricted to similar locations in pixel coordinates. Even if a camera with a high frame rate is used and constant illumination is possible, in natural underwater environments it is very likely that a series of images is highly blurred or obstructed, further limiting the ability to correctly match keypoints. Without a motion model, the tracking would be lost and a new initialization from scratch would be needed. In addition, [9] does not take into consideration loop detection and correction.

Underwater feature-based visual SLAM has been already of interest in [1], where a scale-invariant feature transform (SIFT)/speeded-up robust features (SURF)-based extractor is proposed, preceded by a non-uniform light correction and a normalization. The authors do not describe a visual SLAM method for underwater applications, but their focus lies on finding regions of interest (RoI) which can be utilized as landmarks for loop detection and correction. The idea of using only salient color RoI does limit strongly the capability of [1] to close loops as shown in our work. Features present on sand can be used to safely close loops, thanks to co-visibility keyframe analysis. Deep learning techniques have been investigated in order to detect and close loops in underwater applications, specifically, convolutional autoencoders [25, 32].

The use of monocular ORB-SLAM in an underwater environment has been investigated in [18], in this work, the behavior of ORB-SLAM is analyzed with challenging sequences, captured with a monocular camera mounted on a remotely operated vehicle (ROV). The authors suggest enhancing ORB-SLAM with sub-mapping capabilities, which is also presented in this paper, with the addition of a motion model that updates the current pose (see Sect. 6). In ORB-SLAM keypoints detected in underwater scenes could belong to various elements, not *suitable* to be used for ego-motion tracking and mapping, for this reason, a deep learning-based approach has been developed to reject such keypoints [24].

ORB-SLAM has been found to be the best-performing visual SLAM by the authors of the Aqualoc dataset [10], justifying our goal of improving upon it for underwater monocular VSLAM applications.

2 System overview

In this section, a brief overview of the UVS system is given (a graphical overview of the system is shown in Fig. 2); details are explained in the following sections.

When a new frame is recorded, it is inserted into a first-in-first-out (FIFO) queue. In the initialization phase, a *three-view initialization* is attempted utilizing the frames in the queue. In the case initialization is successful, the system will continue processing this queue as long as the triangulation of new points is completed. This queue avoids frames to be dropped, trading the real-time constraint for tracking robustness. A discussion about this trade-off is presented in the *partial synchronization between front-end and back-end* Sect. 5. This queue runs on an independent thread and performs the ORB feature extraction and descriptor calculation.

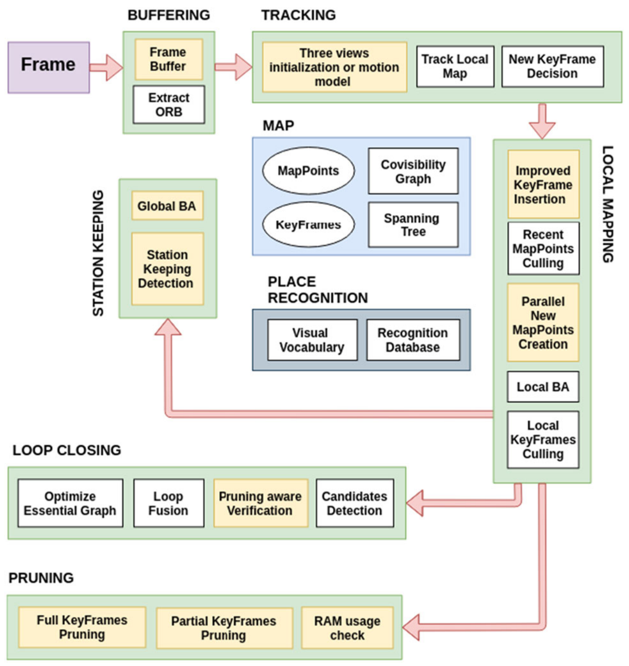

Fig. 2 Block diagram of the UVS system: in purple the system input, in green the different threads, in blue the map representation, in yellow the innovations to ORB-SLAM (Color figure online)

The ORB features are extracted homogeneously in the frame utilizing a grid in the same way as in ORB-SLAM.

If the tracking is lost while frames are collected, a constant velocity *motion model* is applied to predict the current camera pose. When initialization is available the estimation process will restart from the last camera pose. Following the same criteria as present in ORB-SLAM ([35] section V, subsection B, D), several attempts to track the local map are performed, and evaluations are made to see if it is convenient to insert a new keyframe.

In case a new keyframe is inserted, new map points are created considering neighbor keyframes in the covisibility graph. The new map points are created by considering all the possible keypoint matches between keyframe pairs. Points that do pass chirality, epipolar, parallax, scale consistency, and reprojection tests are considered as high-quality points. These points are inserted into the map and are subject to the local bundle adjustment (BA) procedure.

At the end of the tracking, the *station-keeping detection* procedure runs on an asynchronous thread. The goal is to identify a situation where the robot is held stationary by a dynamic positioning system so the camera is looking constantly at the same scene. In such a situation the creation of new map points and keyframes is not necessary, and the computational resources can be allocated to perform map and pose optimization.

Two more procedures run concurrently: the *loop closing* thread and *the pruning thread*. The loop closing thread is in charge of detecting loop closure candidates and eventually performing loop fusion and essential graph optimization. The pruning thread is managing the memory resources, by pruning first map points which are further away, then eventually entire keyframes and all their observed map points.

3 Three-view initialization

Correct map initialization is a crucial step in VO/VSLAM feature-based algorithms, as the tracking starts from the initial map. In ORB-SLAM, initialization is performed by analyzing two views. The first relative pose is estimated with a homography or a fundamental matrix. A choice is made between these two models considering a score that is based on symmetric transfer errors [17]. The necessity of eventually using a homography to estimate the relative camera pose is needed by the fact that, for collinear points, the estimation of the fundamental matrix builds a set of possible solutions [17]. We see three problems with this solution:

1. This procedure selects the homography model on not perfectly planar surfaces, which leads to an approximate solution of unknown magnitude, considering the lack of scale information.
2. Visual SLAM, in general, is a three-view problem: estimating the relative scale requires scene overlap in three frames.
3. In a typical scenario, the underwater VSLAM system considers a camera pointing downwards, towards the seabed. This means that the main motion is a side-way motion, for which the estimation of a fundamental matrix with a 6-, 7-, and the 8-point algorithm is not recommended [38].

For this reason, we propose a three-view initialization procedure with a unique model for both planar and non-planar scenes, based on the estimation of the essential matrix exploiting the 5-point algorithm from [38] (it would be however preferable to use the iterative version of the 5-point algorithm [30], due to potential ill-conditioning of the high degree polynomials involved in the original formulation of the 5-point algorithm). By calculating the essential matrix, using > 2 views, and knowing the intrinsic parameters, a unique solution can be calculated in every possible case (cf. [38], Table 1 shows the degrees of ambiguity in the face of planar degeneracy for pose estimation).

For each frame, the keypoints and the descriptor are extracted, and only the frames where at least 100 keypoints can be detected are retained, as presented in ORB-SLAM [35].

A combination of random sample consensus (RANSAC) and chirality tests is used to find the unique essential matrices and the correctly triangulated corresponding keypoints. The *optimal triangulation method* is employed ([17] Algorithm 12.1) to estimate the initial map. A further test [28] on the sum of a pair of two view parallax is performed to ensure the quality of the solution, as a high parallax limits the uncertainty on the initial map and poses. To limit the computational complexity, the first two views which would satisfy all the original initialization criteria as a fundamental matrix-based initialization in ORB-SLAM [35] are found. A map is created and a third view is searched in the pool of available frames to guarantee uniqueness, improve the 3D point estimate, and improve the chances that no initialization has to be performed soon again. The third view has to be subsequent to both the other two views. Immediately after initialization, a full BA is performed, refining poses and 3D points.

4 Complete keypoint set matching

When a new keyframe is inserted into the system by the mapping thread, new 3D points are triangulated using key-frames present in the covisibility graph. A simple and effective way to match keypoints is a *brute force* approach, where the descriptor associated with each keypoint is compared with other descriptors, utilizing the Hamming distance. A brute force matching guarantees that every pair of descriptors is compared, providing an exact solution to the matching problem, at the cost of having to perform roughly $\mathcal{O}(n^2)$ comparison operations (assuming n keypoints in both frames subject to this operation). In ORB-SLAM brute force matching also happens, but is restricted to those features that belong to the same vocabulary tree, speeding up the search. While this operation does provide a sensible speed-up compared to a brute force, it limits the number of new keypoint candidates leading to a sparsity of the map and therefore reduces the tracking robustness. Vocabularies are sensitive to training, eventually contributing to an overall uneven SLAM performance through different datasets and sequences.

The goal is to avoid using the vocabulary, but still, be able to perform fast descriptor matching. A crucial fact is that the actual speed of the binary descriptor matching is very sensitive to how it is performed: naïve methods can be a hundred times slower than deeply optimized methods, which use proper data structures or exploit specific instruction sets. Specifically, utilizing 64 bits integers, unrolled loops, and $X86\_64$ instruction $\_popcnt64()$ it is possible to obtain a 3X speedup compared to the brute force binary descriptor matcher available in OpenCV [4].

This descriptor matcher represents already a 60X speedup compared to a naïve implementation. All together this implementation is 180X faster than a naïve comparison. Considering the matching time of the distributed bag-of-word 2 (DBoW2) [13] based ORB-SLAM match, it is just 30% slower on average (7.6 milliseconds against 10 milliseconds). A public implementation of this matcher is available [2]. As the matching procedure involves only a pair of keyframes at the time, this process can be easily parallelized by launching a thread for each pair of keyframes.

5 Partial front/back-end synchronization

In ORB-SLAM back-end and front-end are running asynchronously. This allows the system to be as responsive as possible to new frames. Unfortunately, the back-end and the front-end are dependent on each other when it comes to estimating the next camera pose: if the back-end did not create enough map points in time, the tracking will be lost, regardless of the quality of visual information in the frames. ORB-SLAM is strictly dependent on the hardware configuration. In ORB-SLAM, the ability to estimate new camera poses depends on the ability of the back-end to estimate new map points, if the latter does not happen fast enough, the tracking could be lost. In UVS the front-end and back-end are partially synchronized. UVS does accept new frames only when a new keyframe insertion, and so the creation of new map points, has been completed. If frames arrive before the triangulation is completed, they are placed in the FIFO queue. Using this strategy the dependency from the hardware configuration is not eliminated, as a fast framerate can fill the FIFO queue, and so the SLAM estimates could be outdated, considering the current pose of the robot. The dependency of the hardware configuration is shifted from real-time performance towards tracking robustness and loop closure quantity and quality.

6 Motion model

ORB-SLAM uses a constant velocity motion model, see Eq. (1). This motion model is used only for tracking the local map given the current frame. The motion model is implemented as a differential frame-to-frame camera pose, by the mean of a homogeneous transformation matrix:

$$\begin{aligned} V_c^w &= X_c^w P_w^c \\ X_c^w &= V_c^w P_c^w \end{aligned} \tag{1}$$

Here, $P \in SE(3)$ represents the previous pose, $X \in SE(3)$ the current pose, and $V \in SE(3)$ the differential pose between the two, where SE stands for the special Euclidean group. While this functionality is preserved in our system, this motion model is used also for a different purpose. When the tracking is lost, such as in the case of a complete temporary occlusion (and a few keyframes additional to the initial ones are created), ORB-SLAM enters into a re-localization mode. In this mode, no re-initialization is possible, with the consequence, that even in presence of frames with which initialization, mapping, and tracking would be possible, no actions are performed and all the potential map and camera poses are lost (see Fig. 3). Our solution to this problem is to continuously update the current pose of the camera by using the previously described motion model, while a new initialization is attempted. If a new re-initialization is successful, then the previous map and graph information is retained, and the system restarts tracking and mapping. This process does not block the attempts of relocalization, instead, relocalization attempts continue, and if relocalization is successful, the motion model predictions are discarded. If the camera motion is in contradiction with the motion model, it is obvious that the new map estimate starts with a substantially wrong pose. It is worth mentioning that in case of loop closure, a global map can be created and a valid SLAM estimate can still be achieved, thanks to the loop alignment procedure and the essential graph optimization. In concrete situations of camera-only underwater navigation of autonomous underwater vehicles, this approach is likely to be very effective, as the motion is mostly unidirectional, with mild accelerations, making a constant velocity motion model a valid model.

7 Loop closure

Loop closure in ORB-SLAM is composed of three main parts: loop detection, loop validation/loop correction, and optimization. The loop detection procedure ensures that a loop is found in several consecutive keyframes. This procedure produces loop candidates which are on average seventy times more numerous than the number of corrected loops, both in in-air and underwater datasets. The loop validation proceeds to compute a similarity transform with the keyframes involved in the eventual loop closure. Using the found transformation, a geometric validation of the loop is performed, and if the validation is successful, the loop is accepted, and a loop correction can be initiated. In ORB-SLAM, loop detection and loop validation both rely on the bag-of-words (BoW) model, but in two different ways: loop detection uses BoW vectors to compute a similarity score between keyframes, while loop validation does use it to speed up feature matching. Speeding up the matching process by using DBoW2 decreases significantly the number of valid matches that can be obtained, and this can cause a loss of a perfectly valid loop closure. As the loop validation is run in the loop closure thread, the small extra computational effort required to perform a complete keypoint set matching is not impacting directly the tracking and mapping thread, considering the method described in Sect. 4.

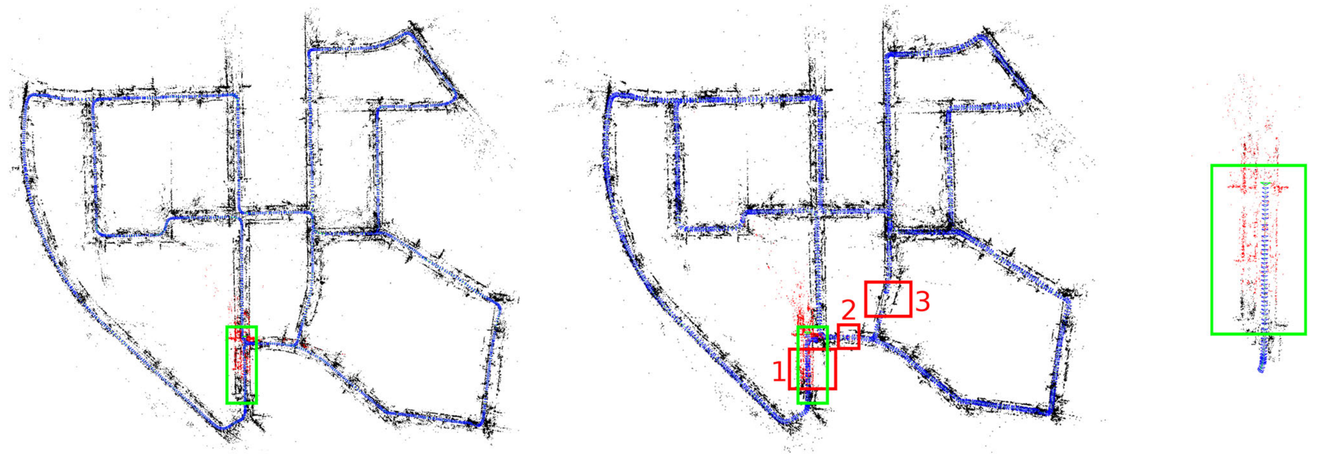

Fig. 3 Three different SLAM estimates generated on sequence 00 of the KITTI dataset are presented. On the left, the map was generated by ORB-SLAM. In the center, the map generated by UVS, where a series of frames have been skipped to force the system to lose the tracking. The sections where this skipping has been performed, are the ones surrounded by red bounding boxes and enumerated. On the right, is the map generated by ORB-SLAM, where the same frame-skipping operation is performed. Due to the relative scale difference between the left/central and the right figure, corresponding sections are surrounded by a green bounding box (Color figure online)

One could expect that re-training the BoW vocabulary would lead to higher quality and quantity of loop corrections, but our experiments have only confirmed this hypothesis to a minor degree. A 1 million-word vocabulary trained with underwater images (Scott Reef 25 and Tasmania O'Hara 7 [11]) shows no improvements in loop detection and correction on the Aqualoc sequences compared to the vocabulary supplied with ORB-SLAM. A vocabulary trained with the same settings on the Aqualoc dataset does show improvements compared to the one supplied with ORB-SLAM. It is able to produce only 60% of the correct matches produced by the UVS system.

8 Station-keeping detection

A typical operation in marine robotics is station-keeping. During this operation, the robot tries to keep its position stationary relative to an object or to the environment. It is popular to perform this operation using vision sensors [6, 23, 33]. As the scene is almost static, many visual SLAM operations, like keyframe insertion and new map point triangulation, are not required. Ideally, it would be beneficial for SLAM systems to receive a message from the control system about the initialization and termination of the station-keeping operation, keeping in mind that there could be a delay when the station-keeping is initiated and when the visual scene is actually static. Furthermore, it is not always practical to interface the control system with the SLAM system. Therefore, we propose to detect an ongoing station-keeping operation (a situation where tracking is successful and no relative motion can be estimated from images for a certain time period), and use this information to allocate CPU time to perform global BA, improving the poses and the map.

We propose an approach to detect station-keeping in absence of absolute scale information, looking at three indicators:

– the ratio r of commonly observed map points Obs between an initial frame f_1 and the following frames, see Eq. (2)
– the Mahalanobis distance $d_{\vec{\theta}}$ over the attitude derivatives $\frac{\mathrm{d}\vec{\theta}}{\mathrm{d}t}$, see Eq. (3)
– the average angle α_{avg} between velocity vectors $\vec{v}$, see Eq. (4)

The n frames $f_i, i = 1, ..., n$ taken into consideration define the *station-keeping detection window*. The station-keeping detection window size n is user-defined and depends on the desired sensibility goal. As a general indication, it is advisable to set n to 15 times the fps, so that false detections are avoided. Station-keeping is detected when these indicators verify a series of conditions, see Eq. (5). Here, follows a detailed description of these indicators: the ratio r of commonly observed map points is given by

$$r = \frac{(Obs_{f_1} \cap Obs_{f_2} \cap ... \cap Obs_{f_n})}{\mathrm{card}(Obs_{f_1})}, \qquad r \leq 1, \tag{2}$$

where Obs_{f_i} represents the set of map points observed by frame $f_i, i = 1, ..., n$ as a set of map point IDs and $\mathrm{card}(Obs_{f_1})$ represents the number of observed map points by the frame f_1. As a consequence, the nominator of Eq. (2) represents the intersection set, consisting of those map points with distinct IDs which can be identified in all the station-keeping window frames. In an ideal station-keeping situation, all the frames in consideration would observe the same map points, and the ratio would be equal to one. The Mahalanobis distance $d_{\vec{\theta}}$ of the estimated Gaussian distribution over the attitude derivatives, calculated as frame-to-frame attitude differential and expressed as Euler angles, here represented as

$$d_{\vec{\theta}} = \sqrt{(\vec{0} - \vec{\mu}_{MLE})^T \Sigma_{MLE}^{-1} (\vec{0} - \vec{\mu}_{MLE})}, \tag{3}$$

it should be equal to 0, as we expect a robot that is performing station-keeping to be stationary without performing any rotational motion. The Gaussian distribution $\mathcal{N}(\vec{\mu}, \Sigma)$ is calculated using a maximum likelihood estimator (MLE):

$$\vec{\mu}_{MLE} = \arg\max_{\vec{\mu}} \mathcal{N}\left(\left[\frac{\mathrm{d}\vec{\theta}}{\mathrm{d}t} w_a\right] \mid \vec{\mu}\ \Sigma\right),$$
$$\Sigma_{MLE} = \arg\max_{\Sigma} \mathcal{N}\left(\left[\frac{\mathrm{d}\vec{\theta}}{\mathrm{d}t} w_a\right] \mid \vec{\mu}\ \Sigma\right),$$

where $w_a = \frac{1}{3}$ represents a normalization factor, expressed in degrees. A smaller w_a implies that more rotation is tolerated when it comes to detecting station-keeping. The last indicator is the average angle between velocity vectors, inside the station-keeping detection window:

$$\alpha_{avg} = \frac{1}{n-1} \sum_{i=1}^{n-1} \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_{i+1}}{\|\vec{v}_1\| \, \|\vec{v}_{i+1}\|}\right) \tag{4}$$

where $\vec{v}_i$, $i = 1, ..., n-1$ is the velocity vector associated with the frame f_i.

During the robot's forward or backward movement, all of these velocity vectors would point roughly in the same direction and, as a consequence α_{avg} should be close to zero. When the robot is turning, α_{avg} should be higher than a few degrees, note the attitude is also changing. In an ideal station-keeping situation, α_{avg} would have a high value, as a stationary robot would in theory produce uniformly distributed velocity vectors, and attitude changes would be small. Taking into consideration α_{avg} together with r, the ratio of commonly observed map points and $d_{\vec{\theta}}$, an indicator of the attitude derivatives, is important, since the robot when moving through a large landscape with low velocity, could result in small attitude changes and several frames could observe the same map points. In such a case, the evaluation of α_{avg} allows distinguishing a moving robot from a stationary one.

Finally, the three conditions which have to be satisfied for the station-keeping to be detected are:

$$\begin{cases} r > 0.9 \\ d_{\vec{\theta}} < 1 \\ \alpha_{avg} > 45^\circ \end{cases} \tag{5}$$

As station-keeping is detected, no new keyframes are created and global BA starts. When the number of tracked points in the last frame goes under 50, then the global BA is aborted, and tracking and mapping continue as normal. Furthermore, as the station-keeping ends, n frames have to pass (where n represents the station-keeping window) before station-keeping can be detected again.

9 Pruning

As an underwater robot is exploring an open environment, the data generated by a visual SLAM system is growing unbounded. All this data is naturally residing in the system memory, which is limited and crucial for the operative system and other programs. It follows that a pruning procedure is obligatory in a system that claims to be able to perform life-long operations. Pruning not only bounds the system memory requirements but also bounds the computational complexity of global operations, like loop correction and global bundle adjustment. The pruning strategy presented in this paper introduces a new data structure: the partially pruned keyframes. A partially pruned keyframe is a keyframe that observes only a fraction of the map points that it was observed before the partial pruning operation. A central operation in visual SLAM is loop closure. The pruning strategy that is going to be described tries to keep re-localization and loop correction possible on partially pruned keyframes. Partial pruned keyframes allow to lower the memory footprint of the UVS system, and at the same time, allow to keep vital information for loop closure and path planning. The map points are pruned in such a way that the remaining points are uniformly distributed, in this way we keep valid information for the same region, where these map points are present for future endeavors.

In addition, surface points uniformly sampled do maximize the probability of a correct terrain reconstruction, within the limits of the Nyquist-Shannon sampling theorem.

The partial pruning procedure retains 33% of the points originally observed by a keyframe. When a keyframe gets partially pruned, a Boolean flag is set and the covisibility graph weights are updated accordingly. As soon the first pruning is initiated, the loop detection thresholds are lowered by 66%, and the loop verification procedure utilizes the per-keyframe Boolean's flag to take into consideration the lower amount of observed map points.

The utilized system memory is checked with a system call when a new keyframe is inserted. No pruning is performed until the utilized memory reaches 90% of the set threshold. All the pruning operations start from the keyframe which is further away from the active keyframe and proceed following the creation order through the *essential graph*. When the pruning has partially pruned all the keyframes that do not belong to the local map, then keyframes are pruned entirely and the covisibility graph is updated.

10 Results

In this section we present a comprehensive analysis comparing our modifications of the monocular ORB-SLAM(2) to the original work, using in-air and underwater datasets. Analysis of underwater datasets includes also the state-of-the-art direct VSLAM LDSO [14], which adds loop closure capabilities to the original DSO [8]. In addition, we will present a comprehensive analysis of each stated contribution, to highlight the strengths and weaknesses of the UVS system. Regarding benchmarking using public datasets, we will analyze several sequences from the KITTI dataset [15], the TUM-RGB-D dataset [42], the Aqualoc dataset [10] and the RTMVO dataset [9]. Two additional private datasets were used to further verify and highlight the pros and cons of the contributions proposed in this paper. Every single experiment was performed 5 times, and the median value was reported. Reporting median values over several experiments is necessary, as ORB-SLAM, as well as UVS, contains several stochastic mechanisms so that results cannot be exactly repeated. Examples of these stochastic mechanisms are the BRIEF descriptor and the non-convex optimization.

Here is a list and a brief description of the restricted datasets:

1. Herkules Relict Front-Camera: A 30 fps, and 1080p video from the front camera used to manually guide an ROV is supplied. The sequence starts by looking at the empty sea, then it meets the relict of a fishing ship. As the ROV is positioned so that the front camera is facing a side of the ship [26], the ROV activates dynamic positioning. This sequence is used to demonstrate the ability of the UVS system to detect the station-keeping, using only visual information.
2. Down-looking camera on the NTNU light autonomous underwater vehicle (LAUV) "Fridtjof" from OceanScan-MST registered at Kjerringholmen North, Norway: The sequence is captured approximately at 4 fps at a resolution of 1376x1032 pixels. In this sequence, a LAUV slowly dives with a small pitch angle towards the seabed and then keeps a constant distance to it with a constant velocity, for a certain amount of time. It shows at the start of the sequence that the seabed is mostly composed of rocks, but becomes slowly sandy.
   This sequenced is used to demonstrate the ability of the UVS system to perform VSLAM in the occurrence of low visibility and low frame-to-frame scene overlap (on average, every 10 frames the system is presented with a completely new scene). Natural and artificial illumination of the scene is very poor, as the seabed is located at a depth between 40 and 60 ms (Norwegian waters) for which the illumination system of the LAUV is not powerful enough. A per-frame "ground-truth" is provided, thanks to the LAUV navigation system.

10.1 Overall performance analysis

In this subsection, we analyze several key performance indicators. The results are presented separately for in-air datasets and underwater datasets, see Table1 and Table 2, respectively. All tests have been performed on an i7 5960X @ 3.0Ghz platform with 32Gb of DDR4 and 2133 Mhz. Regarding the trajectories, the following operations are performed before calculating the median keyframe RMSE:

1. KITTI: 7 degree of freedom (DoF) alignment with the method of [45].
2. TUM RGB-D: Unfortunately, the ground-truth data given is not as friendly as the KITTI, as the ground-truth data provided does not match exactly the frames in terms of timestamps. To compare the estimated trajectories, each keyframe is compared to an element in the ground truth, considering the closer timestamp. The trajectories are then aligned with the 7 DoF method of [45].
3. Aqualoc: Ground truth is provided for every 5th frame. The solution here proposed is to linearly interpolate these poses to obtain per-frame ground truth information. After that, the trajectories are aligned with the 7 DoF method of [45].
4. RTMVO: Same setting as the Aqualoc dataset.

Together with the raw data, we summarize the data in terms of gain/loss percentage moving from ORB-SLAM to UVS. This summarized data is composed of sequences where both ORB-SLAM and UVS did initialize within the first 5% of images of a sequence. In all the tables, positive changes are highlighted in bold italics, neutral changes are in bold, and negative changes are highlighted in italics.

For each table column, the argumentation is as follows:

1. The decrease in the median keyframe RMSE is positive, as the UVS system did generate a trajectory closer to the ground truth.
2. The increase in the number of map points is positive, as it can potentially produce a higher amount of loop closures and positively impact robustness.
3. The decrease/increase in the number of detected loop closures is neutral, as it is arguable that a higher number of detected loop closures could lead to a higher number of corrected loop closures. In Table 1 it is shown that the number of detected loop closures is still over two orders of magnitude higher than the number of corrected loop closures and UVS always corrects an equal or higher number of loops. Thus, the higher number of detected loops in ORB-SLAM are loops that do not pass the geometric verification.
4. The decrease in the number of keyframes is positive since the UVS system is in general able to initialize earlier than ORB-SLAM and creates on average a higher amount of key-frames (the reader should compare the map point/key-frames metric, which shows how a brute force match increases the number of 3D points and so tracking robustness). For this reason, this metric is marked as neutral.
5. The increase in the map point/keyframe metric is positive, as it indicates that the keyframe insertion procedure is more effective at creating map points, lowering the chances of losing track in the incoming frame.

Note, all numbers listed depend directly on when the initialization occurs. In this case, one algorithm initializes at the start of a sequence and another initializes in the middle of a sequence, the resulting numbers cannot be used for a direct comparison. For this reason, all the results presented in this paper are marked red between parentheses in the mean RMSE column (initialization occurs more than 5% of the total number of frames later). An example is the *fr1_xyz* sequence, where ORB-SLAM initializes roughly in the middle of the sequence.

The in-air results are presented in Table 1. UVS performs better with respect to ORB-SLAM in terms of the median RMSE and does exceptionally well regarding the KITTI 09 sequence, where it is able to close the loop consistently. In the KITTI dataset, the overall number of map points created by UVS is slightly inferior, since a much lower amount of keyframes is created by UVS (-18%).

An explanation of these results is that the number of keypoints matched using a brute force approach is higher and of higher quality, requiring fewer keyframes to be created.

Regarding the TUM-RGBD dataset, we first have to inform the reader that the number of keypoints extracted is 1000 for both systems, instead of the ordinary 2000, following the original ORB-SLAM configuration file for this dataset. In this indoor dataset, the DBoW2-based matching in ORB-SLAM seems to be very challenging, especially compared to its performance for the KITTI dataset. This can be observed in the number of map points/keyframes, as it is 12% higher utilizing UVS, which translates to a 3X improvement with respect to the results in the KITTI dataset. The sensible lower amount of keypoints translates into a more difficult initialization, which occurs in four sequences (cf. Table 1). A high correlation can also be observed between the number of map points and the median RMSE.
The underwater results in Table 2 show that UVS is able to outperform ORB-SLAM in almost every sequence, and all the summarized key indicators are positive. On average, UVS performs more than 2X better in terms of the median RMSE compared to ORB-SLAM. Especially good performances can be observed in the RTMVO dataset, where UVS is 3.5X closer to the ground truth compared to ORB-SLAM. The number of map points is also always higher, even when the overall number of keyframes is negative, as in the Aqualoc dataset. In several Aqualoc sequences, UVS is able to close more loops than ORB-SLAM. In the second sequence of the RTMVO dataset, ORB-SLAM loses tracking and re-localizes, caused most likely by the high density of small fish swimming in front of the camera. LDSO has been run without enforcing any real-time constraints and without using the “fast” option, with automatic “crop” enable, as suggested by the author. LDSO is not able to run on most of the Aqualoc sequences and performs poorly regarding the other sequences listed. LDSO is able to run on all the sequences of the RTMVO dataset like UVS but produces by far the poorest results in terms of the median RMSE. LDSO is able to produce a much denser representation compared to ORB-SLAM, but it isn’t able to run in real time. A direct method like the one described in DSO/LDSO shows the potential to be utilized as a *densifier*, beneficial for performing underwater robotic navigation and visual obstacle avoidance.

10.2 Three-view initialization

Similarly to ORB-SLAM, we analyze the initialization behavior for the sequence *fr3_nostructure_texture_far*.

Thanks to the three-view initialization, a unique solution is provided even for planar scenes. In this sequence, the UVS system initializes always correctly, around frame 19.

10.3 Motion model

We propose the following experiments regarding the analysis of the efficiency and the limitation of the motion model:

- Case 1: The motion model is predicting correctly the motion of the camera.
- Case 2: The motion model is predicting wrongly the scale of the movement.
- Case 3: The motion model prediction is not correct, in terms of scale, translation, and rotation.

All the experiments were conducted on the KITTI 00 sequence, as it provides a simple and familiar way to visualize the VSLAM estimate. The results are present in Table 3.

Regarding Case 1, we replace the original frames of the sequence (see listing below) with fully black frames, we call this operation *obfuscation*. This is performed for three different parts of the sequence:

- 20 frames starting from frame 8
- 20 frames starting from frame 130
- 20 frames starting from frame 252

These three different sub-sequences have been selected as the motion of the camera in these sub-sequences can be approximated by a linear uniform motion, and so the motion model predicts correctly.

Regarding Case 2, we perform a similar operation to the one presented in Case 1, with the only difference that instead of obfuscating the original frames, we skip these frames, we call this operation *disruption*. By doing so, we simulate a situation where the tracking is lost and the motion model underestimates the scale of the movement several times during the initial part of the sequence and therefore putting at risk the entire SLAM estimate. Nevertheless, the median RMSE is only marginally higher (cf. Table 3). While the UVS system is able to recover and produce maps and poses of nearly identical quality from the one generated by ORB-SLAM on the original sequence, ORB-SLAM produces less than 5% of the map and the poses compared to UVS. A visual representation of the estimates is shown in Fig. 3.

Regarding Case 3, we perform the disruption of 120 frames starting from frame 8, this removes from the sequence the first right turn. The right turn involves also a change in the speed of the car, which results in a severely mispredicted position, orientation, and scale. This disruption is too severe for the system to recover: as soon as a loop closure occurs that involves the missing turn, the essential graph optimization induces that parts of the map grow in scale, destructing the entire estimate.

A similar test is set up on sequence 5 of the Aqualoc dataset. In this sequence, we take into consideration the frames from frame 900 to frame 940. Similarly to the frames in the KITTI dataset, in these frames, the camera movements reassemble a rectilinear uniform motion, and so the motion model predictions are close to the ground truth.

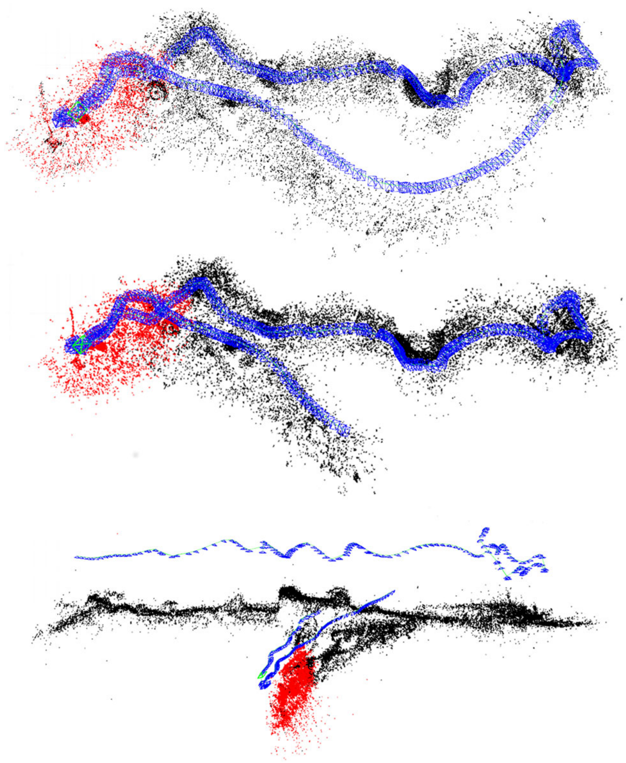
Fig. 4 Several UVS estimates of the Aqualoc 05 dataset: On the top is the XY view of the normal estimate of UVS, in the middle is the XY view (Case 4 and 5), on the bottom is the XZ view (Case 6 and 7)

Similarly to the cases previously described, several cases are analyzed:

- Case 4: The frames are obfuscated.
- Case 5: The frames are disrupted.
- Case 6: In addition to the first set of frames, another set of frames is considered for obfuscation, the frames between 2900 and 2940.
- Case 7: In addition to the first set of frames, another set of frames is considered for disruption, the frames between 2900 and 2940.

In Case 4 and Case 5, the loop closure enables a valid map and trajectory, without any tangible effects on the accuracy. Cases 6 and 7 show a limitation of this system: the tracking is lost twice, resulting in two disconnected maps. The consequence is that, if a loop closure occurs, only one of the two maps benefits from it. A visual representation of these cases can be observed in Fig. 4.

10.4 Keypoint matching

In the overall performance, we observed how a complete set of keypoint matchings allows the production of denser maps and therefore lowers the chances of tracking loss. A positive

Table 1 Key performance indicators of UVS and ORB-SLAM on in-air sequences from the *KITTI* (outdoor) and the *TUM-RGBD* (indoor) dataset

System	Dataset	Median RMSE	Map p	D. loops	C. loops	KeyFrames	Map p. / KF	fps
ORB-SLAM	KITTI 00	5.84 m	96610	297	4	1578	61.2	10 fps
UVS	KITTI 00	6.53 m	104433	245	4	1435	**72.7**	10 fps
ORB-SLAM	KITTI 01	✗	–	–	–	–	–	–
UVS	KITTI 01	✗	–	–	–	–	–	–
ORB-SLAM	KITTI 02	13.49 m	116438	182	2	2086	55.8	10 fps
UVS	KITTI 02	10.23 m	107258	129	2	1727	**62.1**	10 fps
ORB-SLAM	KITTI 03	0.70 m	18119	114	0	287	63.1	10 fps
UVS	KITTI 03	0.83 m	18833	106	0	255	**73.8**	10 fps
ORB-SLAM	KITTI 04	0.58 m	6714	65	0	109	61.6	10 fps
UVS	KITTI 04	1.21 m	6644	43	0	99	**67.1**	10 fps
ORB-SLAM	KITTI 05	6.58 m	56960	135	3	952	59.8	10 fps
UVS	KITTI 05	4.21 m	52693	141	3	781	**67.4**	10 fps
ORB-SLAM	KITTI 06	16.66 m	22348	102	1	407	54.9	10 fps
UVS	KITTI 06	12.84 m	22270	215	1	345	**64.5**	10 fps
ORB-SLAM	KITTI 07	1.91 m	27073	218	1	401	67.5	10 fps
UVS	KITTI 07	1.93 m	25337	105	1	326	**77.7**	10 fps
ORB-SLAM	KITTI 08	29.81 m	100916	225	0	1791	56.3	10 fps
UVS	KITTI 08	25.98 m	94448	196	0	1491	**63.3**	10 fps
ORB-SLAM	KITTI 09	39.24 m	42727	181	0	757	56.4	10 fps
UVS	KITTI 09	6.27 m	39949	134	**1**	623	**64.1**	10 fps
ORB-SLAM	KITTI 10	5.36 m	27621	112	0	489	56.4	10 fps
UVS	KITTI 10	5.67 m	25289	88	0	398	**63.5**	10 fps
ORB-SLAM to UVS	KITTI all	***−16%***	*−3.6%*	*−23.3%*	***+9%***	**-18%**	***+4.4%***	-
ORB-SLAM	fr1_xyz	0.69 cm (✗)	1544	0	0	28	55.14	30 fps
UVS	fr1_xyz	0.74 cm	3221	24	0	66	48.80	30 fps
ORB-SLAM	fr1_desk	10.9 cm	2818	24	0	58	48.53	30 fps
UVS	fr1_desk	11.8 cm	3994	35	0	78	**51.2**	30 fps
ORB-SLAM	fr1_floor	✗		–	–	–	–	–
UVS	fr1_floor	3.16 cm	8352	100	2	165	50.61	30 fps
ORB-SLAM	fr2_360_kidnap	28.3 cm	2120	30	1	40	53	30 fps
UVS	fr2_360_kidnap	26.1cm	4597	74	1	81	**56.75**	30 fps
ORB-SLAM	fr2_desk	5.9 cm	6963	225	0	162	42.98	30 fps
UVS	fr2_desk	7.5 cm	8844	205	0	204	**43.35**	30 fps
ORB-SLAM	fr2_desk_person	22.3 cm (✗)	2426	42	0	68	35.67	30 fps
UVS	fr2_desk_person	0.6 cm	5167	119	0	125	41.33	30 fps
ORB-SLAM	fr2_xyz	0.22 cm	1548	0	0	35	44.22	30 fps
UVS	fr2_xyz	0.19 cm	1901	0	0	42	**45.26**	30 fps
ORB-SLAM	fr3_long_office	3.59 cm	9242	155	1	187	49.42	30 fps
UVS	fr3_long_office	0.72 cm	10340	134	1	180	**57.44**	30 fps
ORB-SLAM	fr3_nstr_tex_far	4.6 cm (●)	(1431)	0	0	(33)	(43.36)	30 fps
UVS	fr3_nstr_tex_far	3.6 cm	2286	11	0	36	**63.5**	30 fps
ORB-SLAM	fr3_sit_halfsph	22.5 cm	1839	0	0	44	41.79	30 fps
UVS	fr3_sit_halfsph	8.4 cm	2474	0	0	44	**56.22**	30 fps
ORB-SLAM	fr3_sit_xyz	0.88 cm	1107	0	0	25	44.28	30 fps

Table 1 continued

System	Dataset	Median RMSE	Map p	D. loops	C. loops	KeyFrames	Map p. / KF	fps
UVS	fr3_sit_xyz	0.68 cm	1379	0	0	29	**47.55**	30 fps
ORB-SLAM	fr3_str_tex_near	1.13 cm	3481	56	0	52	66.9	30 fps
UVS	fr3_str_tex_near	1.09 cm	4026	54	0	59	**68.23**	30 fps
ORB-SLAM	fr3_walk_halfsph	2.0cm	1321	0	0	44	30.02	30 fps
UVS	fr3_walk_halfsph	1.29 cm	1979	6	0	59	**33.54**	30 fps
ORB-SLAM	fr3_walk_xyz	✗		–	–	–	–	–
UVS	fr3_walk_xyz	1.13 cm	1235	0	0	33	37.42	30 fps
ORB-SLAM to UVS	TUM-RGBD all	***-27%***	***+31.2%***	**+11.5%**	***0%***	**+19.4%**	***+12.6%***	–
ORB-SLAM to UVS	In-Air all	***−21.5%***	***+13.8%***	**−5.9%**	***+4.5%***	**+0.7%**	***+8.5%***	–

The mark ✗ represents a sequence where the system was unable to initialize, while the same mark between parentheses means that the system was able to initialize only after 5% of the frames in the sequence have passed. The mark ● represents a case where ORB-SLAM should have not initialized as indicated in [35], but unexpectedly it did initialize around half the sequence

correlation between the number of map points and the overall accuracy is found. In Table 4 we observe that the matching using a brute force approach is only 0.2X slower in terms of CPU time than the matching based on DBoW2 as in ORB-SLAM.

10.5 Partial synchronization

Partial synchronization was not required for the in-air and underwater datasets. Its effectiveness comes into play when frames arrive in the system at a higher rate than the ones at which tracking and keyframe insertion can occur.

Simplifying, this situation happens when the frame rate is too high. An analysis of how the increase in fps affects the system is present in Table 5. Even if the UVS system is utilizing a 30% slower on-average keypoint matching procedure, it is able to clearly outperform ORB-SLAM when the framerate is increased, both in terms of median RMSE and in terms of the ability to complete a sequence without losing the camera localization. When using a thread for each set of features to match, the framerate can be increased even more, without a significant decrease in accuracy.

10.6 Station-keeping detection

Table 6 shows the in-air performance of the station-keeping detection. The proposed set of conditions (see Eq. 5) is able to detect when the car stops at crossroads. A nice plot of the evolution of the three station-keeping indicators in the KITTI 00 sequence is present in Fig. 5.
Unfortunately, neither the Aqualoc nor the RTMVO dataset contains any sequence which does reassemble a station-keeping situation, and so we resort to our previously described “Herkules Relict Front-Camera” sequence. In this sequence, station-keeping is detected, as the ROV is kept stationary by an operator and the camera is looking at the ship. In addition, the station-keeping is interrupted multiple times. The proposed station-keeping detection is able to detect the absence of motion and a persistent stationary state. At the same time, this allows the system to continue its operation as soon as the scene starts to evolve, without loss of tracking.

10.7 Pruning

To evaluate the effectiveness of the pruning we performed two different experiments. The first experiment aims to evaluate the impact of partially pruned keyframes on loop closing. To achieve this goal we avoid performing full key-frame pruning, but partial pruning is always active (memory target of 0 Megabytes). The second experiment aims to simulate a realistic situation, where pruning starts in the middle of the sequence.

To analyze the effects of the pruning, we present to the reader a table with the main indicators, Table 7, using one in-air and one underwater sequence. In addition, an exceptional case is presented, where instead of the previously described 66% map point pruning, the percentage is raised to 90%, see Fig. 6. While the number of map points decreases accordingly, no significant disturbances can be observed on the median keyframes RMSE, especially in the underwater dataset, where, thanks to the high scene overlap of the frames, no decrease in accuracy is observed. Due to the adjustments on the loop validation of partially pruned keyframes, no loops are missed.

10.8 Kjerringholmen North dataset

The *Kjerringholmen North* is a different dataset with regards to the previously presented ones. It is recorded by an autonomous underwater vehicle instead of a manually controlled ROV, like the Aqualoc or the RTMVO sequences. As previously mentioned, most of the trajectories of the LAUV

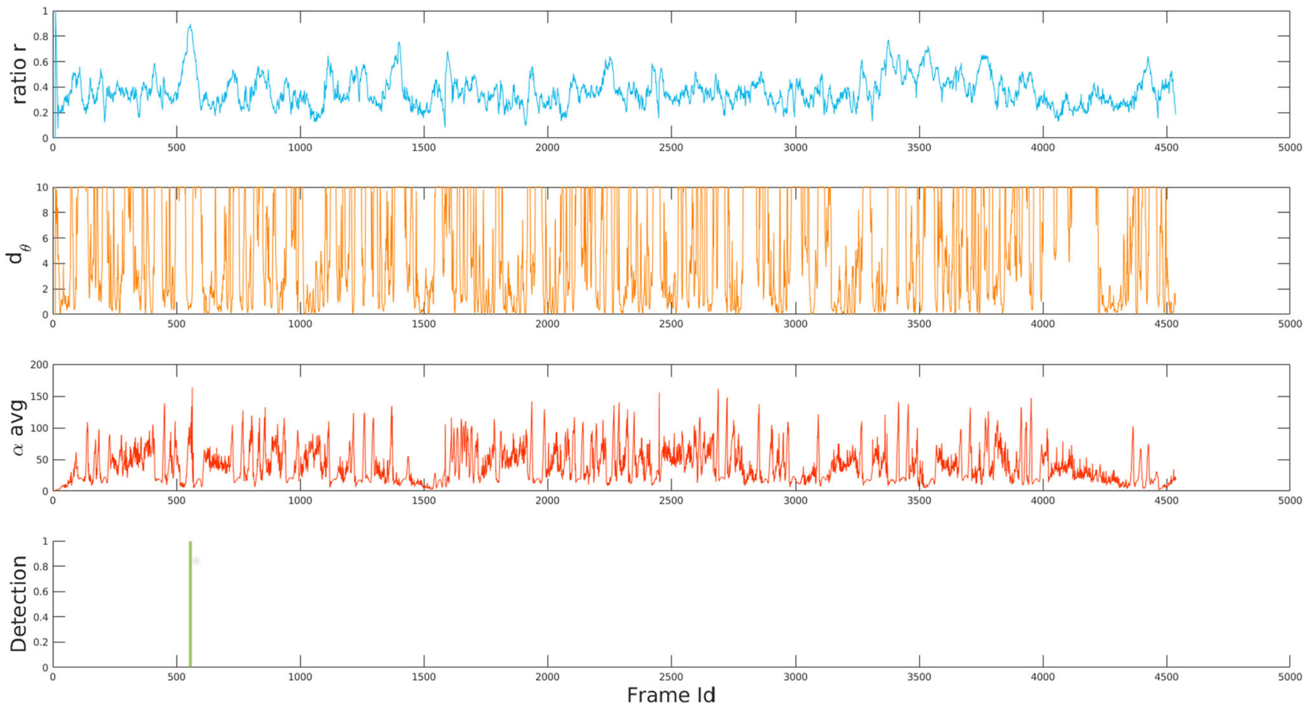

Fig. 5 Station-keeping indicators on the sequence 00 of the KITTI dataset, obtained with a station-keeping detection with $n = 10$. Station-keeping is detected when the car briefly stops at a crossroad. $d_\theta = 10$ to improve the visualization

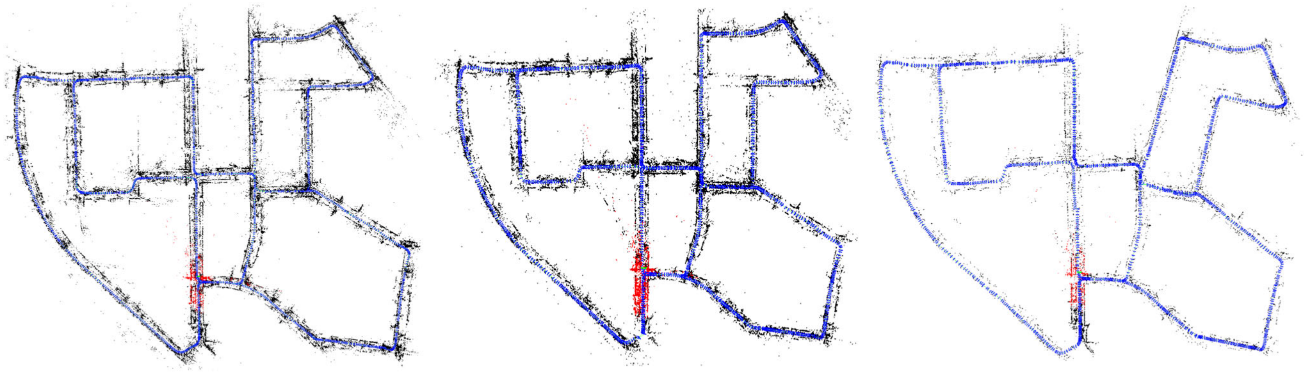

Fig. 6 Left: Original SLAM estimate produced by the ORB-SLAM system. Center: UVS estimate, where a pruning of 66% of the map points is performed after half of the sequence has been processed by the system. No significant difference can be observed. Right: SLAM estimate produced by the UVS system, when partial keyframe pruning is set to 90%, and the memory target is set to 0 megabytes. The loop closures occur in the same quantity and at the same position compared to ORB-SLAM, even with 66% fewer map points. However, the top-right part of the map is incorrectly rotated: Sparsifying an already sparse representation can lower the quality of the final estimates. Note, that this is a particular case, as many runs with continuous 90% pruning did not contain any trajectory alternation

Fig. 7 Visual representation of the sequence *Kjerringholmen North*. A sub-sequence is selected, representing the typical, mostly rectilinear path of LAUV

Table 2 Key performance indicators of UVS and ORB-SLAM on underwater sequences from the *Aqualoc* and the *RTMVO* dataset

System	Dataset	Median RMSE	Map p	D. loops	C. loops	KeyFrames	Map p. / KF	fps
LDSO	Aqualoc 01	✗	–	–	–	–	–	–
ORB-SLAM	Aqualoc 01	✗	–	–	–	–	–	–
UVS	Aqualoc 01	**0.44 m**	17564	361	1	210	**83.6**	20 fps
LDSO	Aqualoc 02	✗	–	–	–	–	–	–
ORB-SLAM	Aqualoc 02	0.44 m	32016	284	1	402	79.6	20 fps
UVS	Aqualoc 02	**0.19 m**	39212	178	**2**	216	**181.5**	20 fps
LDSO	Aqualoc 03	✗	–	–	–	–	–	–
ORB-SLAM	Aqualoc 03	0.031 m	20401	209	1	209	97.6	20 fps
UVS	Aqualoc 03	**0.025 m**	23804	186	1	221	**107.7**	20 fps
All others	Aqualoc 04	✗	–	–	–	–	–	–
UVS	Aqualoc 04	**0.29 m**	40610	166	1	400	**101.5**	20 fps
LDSO	Aqualoc 05	0.70 m	2268303	43	0	1595	**1422**	16 fps
ORB-SLAM	Aqualoc 05	0.14 m	44100	171	1	324	136.1	20 fps
UVS	Aqualoc 05	**0.11 m**	43379	179	1	331	131.0	20 fps
LDSO	Aqualoc 06	0.78 m	660066	0	0	443	**1490**	16 fps
ORB-SLAM	Aqualoc 06	0.11 m	20700	141	0	187	110.69	20 fps
UVS	Aqualoc 06	**0.09 m**	27427	141	**1**	208	131.8	20 fps
All systems	Aqualoc 07	✗	–	–	–	–	–	–
ORB-SLAM to UVS	Aqualoc all avg	***−28.95%***	***+14.2%***	**−17.7%**	***+66.6%***	**−14.9%**	***+25.2%***	-
LDSO	RTMVO 01	8.29 m	251001	0	0	484	518.59	16 fps
ORB-SLAM	RTMVO 01	8.97 m	20007	378	0	212	94.37	16 fps
UVS	RTMVO 01	**1.52 m**	24726	336	0	242	106.30	16 fps
LDSO	RTMVO 02	5.50 m	138463	0	0	268	516.65	16 fps
ORB-SLAM	RTMVO 02	4.47 m	6281	108	0	82	76.59	16 fps
UVS	RTMVO 02	**1.75 m**	8695	124	0	96	90.57	16 fps
LDSO	RTMVO 03	7.53 m	19982	0	0	64	312.21	16 fps
ORB-SLAM	RTMVO 03	**0.36 m**	761	12	0	23	33.08	16 fps
UVS	RTMVO 03	0.56 m	1074	14	0	26	41.30	16 fps
LDSO	RTMVO 04	4.48 m	269114	2	0	575	468.02	16 fps
ORB-SLAM	RTMVO 04	4.01 m	16099	343	0	201	80.09	16 fps
UVS	RTMVO 04	**1.10 m**	20223	318	0	239	84.61	16 fps
LDSO	RTMVO 05	4.37 m	151504	0	0	348	435.35	16 fps
ORB-SLAM	RTMVO 05	✗	–	–	–	–	–	–
UVS	RTMVO 05	**1.44 m**	13933	285	0	178	78.27	16 fps
ORB-SLAM to UVS	RTMVO all	***−40.2%***	***+26.8%***	**−6.1%**	**0%**	**+16.4%**	***+13.6%***	–
ORB-SLAM to UVS	Underw. all avg	***−34.57%***	***+20.5%***	**−11.9%**	***+33.3%***	**+0.75%**	**+19.4%**	–

The mark ✗ represents a sequence where the system was unable to initialize

are rectilinear, see Fig. 7. While ORB-SLAM does not initialize at all, LDSO initializes, but fails in the state estimate, see Table 8. In underwater imaging applications, it is common to use contrast limited adaptive histogram equalization (CLAHE) to improve the quality of underwater images, especially in low-light conditions.

The *Kjerringholmen North* dataset contains images recorded under poor light conditions. Therefore, this sequence represents a good example to analyze the different settings of the CLAHE.

The CLAHE used in Table 8 is provided by OpenCV, where the first element represents the clip limit and the second one the tile size. The table presented provides evidence of how a correct CLAHE setting could further improve performance.

Table 3 Results from different kind of *obfuscations* and *disruptions* of the KITTI sequence 00 showing the median RMSE

System	Sequences	Case	Median RMSE
ORB-SLAM	KITTI 00	–	5.84 m
UVS	KITTI 00	–	6.53 m
ORB-SLAM	KITTI 00	All cases	✗
UVS	KITTI 00	1	*+47.8%*
UVS	KITTI 00	2	*+28.2%*
UVS	KITTI 00	3	*+875.5%*
ORB-SLAM	Aqualoc 05	-	0.14 m
UVS	Aqualoc 05	–	0.11 m
ORB-SLAM	Aqualoc 05	All cases	✗
UVS	Aqualoc 05	1	***−0.8%***
UVS	Aqualoc 05	2	*+2.2%*
UVS	Aqualoc 05	3	✗
UVS	Aqualoc 05	4	✗

Table 4 A comparison between UVS (Full) and ORB-SLAM (DBoW2) on the procedure to match keypoint descriptors, in terms of running time and amount of valid matches

Matching	Sequence	Time	Triangulations
DBoW2	KITTI 00	0.77 ms	128
Full	KITTI 00	0.95 ms	168
DBoW2	fr1_desk	0.58 ms	141
Full	fr1_desk	0.80 ms	208
DBoW2 to Full	In-Air	*+29.6%*	***+39.7%***
DBoW2	Aqualoc 05	0.46 ms	144
Full	Aqualoc 05	0.55 ms	195
DBoW2	RTMVO 01	0.49 ms	99
Full	RTMVO 01	0.53 ms	134
DBoW2 to Full	Underwater	*+13.6%*	***+35.3%***
DBoW2 to Full	All avg	*+21.6%*	***+37.5%***

11 Conclusion and further work

11.1 Conclusion

We presented a series of modifications to the monocular ORB-SLAM system, in order to achieve a better performance regarding underwater environments in terms of robustness, loss of tracking, and lifelong operations. As this series of modifications and extensions is quite extensive, we defined a new term for the system: Underwater Visual SLAM (UVS). Several tests have been performed to show how these modifications impact the performance of ORB-SLAM, utilizing publicly available datasets, both in-air and underwater. Our system outperforms both ORB-SLAM and LDSO in underwater environments, both in terms of the median RMSE and in terms of the ability to close loops. Compared to ORB-SLAM and LDSO, UVS is able to operate in poorer light conditions, with lower scene overlap between frames and also it tolerates loss of tracking, all without the need for any image pre-processing. Several generic issues with ORB-SLAM have been addressed, leading to a system that performs better even in-air.

11.2 Further work

Systems that are dependent on hardware resources, like every VSLAM system available today, could benefit from a benchmark dataset that aims to determine the maximum framerate for a particular hardware configuration.

In addition, integer overflow must also be addressed. New points and new keyframes do use an integer-based unique ID. During the execution of VSLAM, this integer will eventually overflow and so a proper strategy must be implemented to avoid this problem.

While the integration of ORB-SLAM with global navigation satellite system (GNSS) and IMU has been already investigated [22, 37], it would be interesting to investigate sensor fusion for ORB-SLAM regarding underwater-specific sensors, like a Doppler velocity log (DVL).

Table 5 This table shows how the partial synchronization in UVS allows it to run at a higher framerate than ORB-SLAM on the same hardware platform, with a relatively small cost in terms of accuracy

System	Sequence	fps	median RMSE
ORB-SLAM	KITTI 00	10	5.84 m
UVS	KITTI 00	10	6.53 m
ORB-SLAM	KITTI 00	14	*+2.1%*
UVS	KITTI 00	14	*+1.6%*
ORB-SLAM	KITTI 00	> 14	✗
UVS S	KITTI 00	23	*+22.5%*
UVS S	KITTI 00	> 23	✗
UVS M	KITTI 00	26	*+1.7%*
UVS M	KITTI 00	> 26	✗
ORB-SLAM	Aqualoc 05	16	0.14m
UVS S	Aqualoc 05	16	0.11 m
ORB-SLAM	Aqualoc 05	17	✗
UVS S	Aqualoc 05	17	≈ **0%**
ORB-SLAM	Aqualoc 05	> 17	✗
UVS S	Aqualoc 05	20	*+78.2%*
UVS S	Aqualoc 05	> 20	✗
UVS M	Aqualoc 05	32	*+1.0%*
UVS M	Aqualoc 05	> 32	✗

The mark ✗ is shown when the tracking is lost at least once

Table 6 Scale-agnostic detection of a pseudo-stationary situation in the KITTI dataset

Sequence	fps	n	Frame range
KITTI 00	10	10	557–570
KITTI 05	10	10	2348–2356
KITTI 08	10	10	4014–4045
KITTI 00	10	20	560–568
KITTI 05	10	20	2351–2357
KITTI 08	10	20	✗
KITTI 00	10	30	✗
KITTI 05	10	30	2366–2357
KITTI 08	10	30	✗
KITTI 05	10	≥70	✗

The frame range column shows for which frame the station-keeping is detected and for which frame it is terminated

Table 7 Effects of map point pruning with different settings on the median RMSE, the global amount of map points and corrected loop closures

Sequence	begin	P. %	m. RMSE	Map p	C. l
KITTI 00	–	–	5.84 m	104433	4
KITTI 00	2276 4541	66%	*−8.7%*	**−66.4%**	4
KITTI 00	0	66%	*+17.8%*	**−72.3%**	4
KITTI 00	0	90%	*+19.2%*	**−89.4%**	4
Aqualoc 05	–	–	0.11 m	43379	1
Aqualoc 05	1833 3457	66%	≈ ***0*** *%*	**−42.5%**	1
Aqualoc 05	0	66%	≈ ***0*** *%*	**−45.8%**	1
Aqualoc 05	0	90%	≈ ***0*** *%*	**-84,7%**	1

Table 8 Indicators of SLAM estimates for the *Kjerringholmen North* dataset

System	CLAHE	Frames	m. RMSE	Map p
UVS	–	692–1448	9.64 m	28641
UVS	2 – 8x8	**673–1840**	9.43 m	36420
ORB-SLAM	any setting	✗	no init	–
LDSO	any setting	✗	wrong est	–

Pre-processed images utilizing CLAHE to enhance the tracking capability of UVS

Most of the underwater cameras have a flat port enclosure integrated which represents a substantial contribution to the underwater 3D structure estimate and the relative camera motion. A pinhole camera model correction map generator for underwater imaging, for cameras with flat port enclosures, has been recently presented [29]. Preliminary experiments regarding the correction model show that the water refraction index and the distance of the focal point of the camera from the flat port are very difficult to estimate. The model can be effectively calibrated by using a checkerboard placed underwater and performing a grid search on these parameters by minimizing the re-projection error.

Acknowledgements We would like to thank Dr. Øystein Sture and Dr. Stein M. Nornes, for their help in collecting the datasets *Herkules* and *Kjerringholmen North*. Furthermore, we would like to thank the *Australian Centre for Field Robotics*, especially the marine robotics group for providing the *the Scott Reef 25* dataset, which was used to re-train the DBoW2 vocabulary. We would like to thank the *Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier* (LIRMM), et al. for releasing datasets that target specifically underwater VO/VSLAM, like the *Aqualoc* and the *RTMVO* datasets. Finally, we would like to thank the authors of *ORB-SLAM* for making the VSLAM system freely available to the community.
This work was supported by the Research Council of Norway (RCN) funded projects AILARON (#262741), NTNU AMOS (#223254) and AROS (#304667).

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital)

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aulinas, J., Carreras, M., Llado, X., Salvi, J., Garcia, R., Prados, R., & Petillot, Y.R. (2011). Feature extraction for underwater visual slam. In: OCEANS 2011 IEEE-Spain, pp. 1–7. IEEE.
2. Avetisyan, A. (2020). Fast binary descriptor matcher. https://github.com/skanti/HammingBruteForce. Accessed: 2020-01-01
3. Bonin-Font, F., Cosic, A., Negre, P.L., Solbach, M., & Oliver, G. (2015). Stereo slam for robust dense 3d reconstruction of underwater environments. In *OCEANS 2015-Genova*, pp. 1–6. IEEE
4. Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools, 25*, 120.
5. Clark, R., Wang, S., Wen, H., Markham, A., & Trigoni, N. (2017). Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*.
6. Cufí, X., Garcia, R., & Ridao, P. (2002). An approach to vision-based station keeping for an unmanned underwater vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. vol. 1, pp. 799–804. IEEE.
7. Duda, A., Schwendner, J., Stahl, A., & Rundtop, P. (2015). Visual pose estimation for autonomous inspection of fish pens. In *OCEANS 2015 - Genova*, pp. 1–6.
8. Engel, J., Koltun, V., & Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*(3), 611–625.
9. Ferrera, M., Moras, J., Trouvé-Peloux, P., & Creuze, V. (2019). Real-time monocular visual odometry for turbid and dynamic underwater environments. *Sensors, 19*(3), 687.
10. Ferrera, M., Moras, J., Trouvé-Peloux, P., Creuze, V., & Dégez, D. (2018). The aqualoc dataset: Towards real-time underwater localization from a visual-inertial-pressure acquisition system. arXiv preprint arXiv:1809.07076
11. for Field Robotics, A.C. (2009). Marine robotics datasets. http://marine.acfr.usyd.edu.au/datasets/. Accessed: 2019-06-01.
12. Forster, C., Zhang, Z., Gassner, M., Werlberger, M., & Scaramuzza, D. (2017). Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics, 33*(2), 249–265.
13. Gálvez-López, D., & Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics, 28*(5), 1188–1197.
14. Gao, X., Wang, R., Demmel, N., & Cremers, D. (2018). Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2198–2204. IEEE.
15. Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
16. Gracias, N., Garcia, R., Campos, R., Hurtos, N., Prados, R., Shihavuddin, A., Nicosevici, T., Elibol, A., Neumann, L., & Escartin, J. (2017). Application challenges of underwater vision. Computer Vision in Vehicle Technology: Land, Sea & Air pp. 133–160

17. Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge University Press.
18. Hidalgo, F., Kahlefendt, C., Bräunl, T. (2018). Monocular orb-slam application in underwater scenarios. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pp. 1–4. IEEE.
19. Hodne, L.M., Leikvoll, E., Yip, M., Teigen, A.L., Stahl, A., & Mester, R. (2022). Detecting and suppressing marine snow for underwater visual slam. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* pp. 5097–5105
20. Johnson-Roberson, M., Bryson, M., Friedman, A., Pizarro, O., Troni, G., Ozog, P., & Henderson, J. C. (2017). High-resolution underwater robotic vision-based mapping and three-dimensional reconstruction for archaeology. *Journal of Field Robotics, 34*(4), 625–643.
21. Jose Tarrio, J., & Pedre, S. (2015). Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision* pp. 702–710.
22. Kiss-Illés, D., Barrado, C., & Salamí, E. (2019). Gps-slam: An augmentation of the orb-slam algorithm. *Sensors, 19*(22), 4973.
23. Kuo, C.L., Chang, L.Y., Kuo, Y.C., Lin, C.H., & Lin, K.M. (2017). Visual servo control for the underwater robot station-keeping. In *2017 International Conference on Applied Electronics (AE)*, pp. 1–4. IEEE.
24. Leonardi, M., Fiori, L., & Stahl, A. (2020). Deep learning based keypoint rejection system for underwater visual ego-motion estimation. *IFAC-PapersOnLine, 53*(2), 9471–9477.
25. Leonardi, M., & Stahl, A. (2018). Convolutional autoencoder aided loop closure detection for monocular slam. *IFAC-PapersOnLine, 51*(29), 159–164.
26. Leonardi, M., Stahl, A., Gazzea, M., Ludvigsen, M., Rist-Christensen, I., & Nornes, S.M. (2017). Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles. In: OCEANS 2017-Aberdeen, pp. 1–10. IEEE.
27. Li, R., Wang, S., Long, Z., & Gu, D. (2018). Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7286–7291. IEEE
28. Longuet-Higgins, H. C. (1986). The reconstruction of a plane surface from two perspective projections. *Proceedings of the Royal Society of London. Series B. Biological Sciences, 227*(1249), 399–410.
29. Łuczyński, T., Pfingsthorn, M., & Birk, A. (2017). The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings. *Ocean Engineering, 133*, 9–22.
30. Lui, V., Drummond, T. (2013). An iterative 5-pt algorithm for fast and robust essential matrix estimation. In *BMVC*
31. Maity, S., Saha, A., & Bhowmick, B. (2017). Edge slam: Edge points based monocular visual slam. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2408–2417
32. Maldonado-Ramírez, A., Torres-Mendez, L.A. (2019). Learning ad-hoc compact representations from salient landmarks for visual place recognition in underwater environments. In *2019 International Conference on Robotics and Automation (ICRA)* pp. 5739–5745. IEEE.
33. Marks, R.L., Wang, H.H., Lee, M.J., Rock, S.M. (1994). Automatic visual station keeping of an underwater robot. In *Proceedings of OCEANS'94*, vol. 2, pp. II–137. IEEE.
34. Mohanty, V., Agrawal, S., Datta, S., Ghosh, A., Sharma, V.D., & Chakravarty, D. (2016). Deepvo: A deep learning approach for monocular visual odometry. arXiv preprint arXiv:1611.06069
35. Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics, 31*(5), 1147–1163.
36. Mur-Artal, R., & 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.
37. Mur-Artal, R., & Tardós, J. D. (2017). Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters, 2*(2), 796–803.
38. Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(6), 0756–0777.
39. Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., & Moreno-Noguer, F. (2017). Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4503–4508. IEEE.
40. Salvi, J., Petillo, Y., Thomas, S., & Aulinas, J. (2008). Visual slam for underwater vehicles using video velocity log and natural landmarks. In *OCEANS* 2008, pp. 1–6. IEEE.
41. Schellewald, C., Stahl, A., & Kelasidi, E. (2021). Vision-based pose estimation for autonomous operations in aquacultural fish farms. *IFAC-PapersOnLine, 54*(16), 438–443.
42. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D. (2012). A benchmark for the evaluation of RGB-D slam systems. In *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*.
43. Swirski, Y., Schechner, Y.Y., Herzberg, B., & Negahdaripour, S. (2010). Underwater stereo using natural flickering illumination. In *OCEANS 2010 MTS/IEEE SEATTLE*, pp. 1–7. IEEE.
44. Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pp. 6243–6252.
45. Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence, 13*(4), 376–380.
46. Wirth, S., Carrasco, P.L.N., & Codina, G.O. (2013). Visual odometry for autonomous underwater vehicles. In *2013 MTS/IEEE OCEANS-Bergen*, pp. 1–6. IEEE.
47. Zwilgmeyer, P.G.O., Yip, M., Teigen, A.L., Mester, R., Stahl, A. (2021). The varos synthetic underwater data set: Towards realistic multi-sensor underwater data with ground truth. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. pp. 3715–3723

Marco Leonardi has obtained a Master degree in Computer and Automation Engineering with 110/110 cum laude at the University of Siena, and his PhD at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology - NTNU, Norway and has been affiliated to the Center of Excellence for Autonomous Marine Operations and Systems - NTNU AMOS. During this time, he pursued research in underwater robotic vision, more specifically in the fields of Visual SLAM and visual obstacle avoidance. Currently he is a Performance Analysis Engineer working on mobile GPU performance analysis at Arm Holdings plc and co-founder of Exmarte AS, a startup that has developed a remote racing system and a visual timing system.

Annette Stahl is Head of the Robotic Vision Group at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology - NTNU, Norway. She is also an Affiliated Scientist of the Center of Excellence for Autonomous Marine Operations and Systems - NTNU AMOS. She received her PhD degree from the Heidelberg University, Germany in applied mathematics with the main focus on computer vision applications in relation to variational methods for motion estimation using physical prior knowledge. She spent two years as a postdoc at the School of Computing, Dublin City University - DCU, Ireland and three years at the Department of Mathematical Sciences, NTNU, Norway, where she worked on isogeometric analysis based methods for graphics and visualization. After this period she worked as a researcher at the High Performance Computing Group at NTNU and at SINTEF Ocean, Norway, where she was concerned with computer vision based aquaculture applications. In 2016, her was awarded an Onsager Fellowship from NTNU's Research Excellence. She is currently working within the field of robotic vision targeting underwater, on sea surface, on land, in air and space as well as indoor and industrial related robotic applications.

Martin Ludvigsen is Professor and founder of the Applied Underwater Laboratory (AUR-Lab) at NTNU, Trondheim, Norway and has since been managing the lab. Today, the AUR-Lab is considered an essential asset for multidisciplinary marine research at NTNU, facilitating a large body of research. He has also been working with underwater vehicles and operations in the industry inshore and offshore. Since 2014 he has held a Professorship in Underwater Technology at the Department of Marine Technology, NTNU while also being a founder of Blueye Robotics, a novel portable ROV start-up. He has extensive at-sea experience in arctic waters as well as in benthic environments associated with the Norwegian mid-ocean ridge. He has been involved in a variety of research project both on deep sea research in the upper water column and in the arctic deploying robotic underwater vehicles. Since 2015 has an Adjunct Assoc. Professorship in Marine Technology at the Svalbard University Centre (UNIS) in Longyearbyen. He is currently affiliated scientist of the Centre for Autonomous Marine Operations and Systems (NTNU AMOS).

Edmund Førland Brekke received the M.Sc. degree in industrial mathematics in 2005, and the Ph.D. degree in engineering cybernetics in 2010, both from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. From 2010 to 2014, he worked with the Acoustic Research Laboratory (ARL), NUS in Singapore as a postdoctoral Research Fellow. In 2014 he rejoined NTNU and the Department of Engineering Cybernetics as an Associate Professor. In 2015-2019 he has been project manager for the Autosea project, developing methods for sensor fusion and collision avoidance for autonomous surface vehicles. He is an affiliated scientist at the Centre for Autonomous Marine Operations and Systems (AMOS), a key scientist for the Autoferry project, which is focused on autonomous pedestrian ferries, a work package manager in the Centre for Research-based Innovation SFI Autoship, and project panager of the Autosit project, which is focused on automated situational awareness for autonomous surface vehicles. He is an associate editor of the IEEE Journal of Oceanic Engineering. Brekke's research interests are in the area of sensor fusion and situational awareness, with a particular focus on multi-target tracking.