Journal of
Marine Science and Engineering

MDPI

Article

Towards Multi-Robot Visual Graph-SLAM for Autonomous Marine Vehicles

Francisco Bonin-Font † **and Antoni Burguera** *,†

Systems, Robotics and Vision Group, Department of Mathematic and Informatics, University of the Balearic Islands, 07122 Palma, Spain; francisco.bonin@uib.es

* Correspondence: antoni.burguera@uib.es

† These authors contributed equally to this work.

Received: 16 May 2020; Accepted: 10 June 2020; Published: 14 June 2020

Abstract: State of the art approaches to Multi-robot localization and mapping still present multiple issues to be improved, offering a wide range of possibilities for researchers and technology. This paper presents a new algorithm for visual Multi-robot simultaneous localization and mapping, used to join, in a common reference system, several trajectories of different robots that participate simultaneously in a common mission. One of the main problems in centralized configurations, where the leader can receive multiple data from the rest of robots, is the limited communications bandwidth that delays the data transmission and can be overloaded quickly, restricting the reactive actions. This paper presents a new approach to Multi-robot visual graph *Simultaneous Localization and Mapping* (SLAM) that aims to perform a joined topological map, which evolves in different directions according to the different trajectories of the different robots. The main contributions of this new strategy are centered on: (a) reducing to hashes of small dimensions the visual data to be exchanged among all agents, diminishing, in consequence, the data delivery time, (b) running two different phases of SLAM, intra- and inter-session, with their respective loop-closing tasks, with a trajectory joining action in between, with high flexibility in their combination, (c) simplifying the complete SLAM process, in concept and implementation, and addressing it to correct the trajectory of several robots, initially and continuously estimated by means of a visual odometer, and (d) executing the process online, in order to assure a successful accomplishment of the mission, with the planned trajectories and at the planned points. Primary results included in this paper show a promising performance of the algorithm in visual datasets obtained in different points on the coast of the Balearic Islands, either by divers or by an *Autonomous Underwater Vehicle* (AUV) equipped with cameras.

Keywords: multi robot; Simultaneous Localization and Mapping; visual loop closure; image global signatures

1. Introduction and Related Work

Simultaneous Localization and Mapping (SLAM) [1] is an essential task for *Autonomous Underwater Vehicles* (AUV) to achieve successfully and precisely their programmed missions. SLAM consists of building a map of the environment and, at the same time, estimating its own pose within this map. SLAM is presently a de facto localization standard for any kind of autonomous vehicle. Laser range finders or sonar were the sensor modality of choice at first [2–4]. However, research turned to computer vision as soon as price and capabilities of cameras made it possible [5], since cameras provide higher temporal and spatial data resolutions and richer representations of the world.

However, large-scale or long-term operations with a single robot equipped with cameras generate huge amounts of visual data that can collapse the vehicle computer, if they are not treated properly. A common strategy to overcome this problem is to explore the areas of interests in different,

separated missions, so-called *sessions*, run with a single robot in different time periods (a Multi-session configuration [6]) or with several robots running simultaneously (Multi-robot configurations [7]). Therefore, any low capability of a robot to operate robustly during long periods of time can be alleviated by running different transits with different agents, at the same time through common areas, and joining all individual estimated trajectories in a single coordinate frame. Multi-robot systems also increase robustness in case of failure of any of the robots; however, they need complex coordination and multiple localization systems. Typical applications using teams of robots include aerial surveillance [8], underwater exploration [9], maintenance of industrial infrastructures or intervention in archaeological sites [10], among others.

The first approaches to Multi-robot SLAM were based on particle filters [11], and introduced the concept of *encounters* as the relative pose between two robots that can mutually recognize each other and determine their relative poses. These *encounters* are introduced as additional pose constrains in the particle filter. Some Multi-robot approaches are based on the *Anchor-nodes* [12,13] proposal, which defined two concepts unconsidered for multiple trajectories until that moment: (a) the *Anchor*, defined as the offset of a complete trajectory with respect to a global system of coordinates, and (b), an *encounter*, re-defined as a transformation between two different poses of two different robots that observe the same part of the environment, but without being necessarily that both robots recognize themselves. In visual-based systems this can be achieved, for instance, detecting overlapping scenes. In Multi-robot systems, *encounters* represent additional constraints between different graphs corresponding to different sessions.

Schuster et al. conceived a very precise SLAM approach to localize a team of planetary rovers equipped with an *Inertial Measurement Unit* (IMU) and a stereo camera [14]. IMU data, visual odometry and wheel odometry are integrated in a local localization *Extended Kalman Filter* (EKF) and the 3D point clouds of all robots computed from their respective stereo views are, firstly stored in each agent, and then matched to be joined in global 3D maps. The use of stereo vision and advanced techniques for 3D feature matching and alignment complicate considerably the whole system and generate huge amounts of data to work with and to be exchanged. This solution turns out to be very difficult for underwater missions, given the limited options for fast communication in this media.

Saeedi et al. offered an extensive survey of Multi-robot systems and strategies, pointing also towards the upcoming trends and challenges [15], such as extending the systems to dynamic and/or large-scale environments or increasing the number of agents in the working teams.

Another issue to consider in SLAM is the detection of loop closings and their use to correct the robot trajectory estimated by means of dead-reckoning sensors, such as, inertial units, acoustic beacons, laser-based or visual odometers. Loop closing is the problem of recalling revisited scenes, and approaches to visual loop closure detection try to recognize the same scene in different images, taken at different and relatively distant time instants, regardless evident differences on scale or view point [16]. In single session SLAM, since the robot pose is continuously estimated, the search for images candidate to close a loop with a *query* (from now on called intra-session loop closings) is constrained to a region around the robot pose associated with that query [17]. In contrast, multi-robot loop detection, i.e., the detection of loop closings among different sessions of different robots (from now on called inter-session loop closings), cannot rely on the AUV poses to constrain the search since, at first, the relative poses between sessions is unknown. Consequently, it seems that every *query* of one session would need to be compared with all the images obtained until that moment in the other sessions, increasing considerably the time dedicated for this task, and the amount of visual data to be exchanged.

Exchanging image hashes instead of entire images or sets of image salient points is a way to reduce data transfer requirements in Multi-robot configurations. Hash functions are usually used to authenticate messages sent between a source and a receiver, so that the later can verify the authenticity of the source. Conventional hashing algorithms are extremely sensitive to the hashed messages. A change in 1 bit of the input message causes dramatic changes on the output. Applications of

hashes, understood as exposed before, include image retrieval in large databases, authentication and watermarking, among many others [18]. However in applications of scene recognition, localization or visual loop closing detection, it is accepted that similar or overlapping images are expected to produce similar or close hashes while distinct images produce clearly distinctive hashes, being this concept known are perceptual image hashing [19–23]. In particular, McDonald et al. [6] proposed to detect loop closings using a solution based on *Bag of Words* BoW [24] combined with iSAM [12] for batch map optimization, and Negre et al. [22] showed how their new global image descriptor HALOC outperformed other techniques, such as BoW and VLAD [20], in the task of loop closing detection with image hashes. From now on, this text uses equally hash or global image descriptor to refer the same concept.

All these aforementioned references apply hashes to detect loops in SLAM applications for single robots. However, now, our interest is focused on the Multi-robot systems, and the application of global image signatures to find loops between images captured by different robots that operate in a same mission on a common area of interest. A few authors have already explored this idea. For instance, *Decentralized Visual Simultaneous Localization and Mapping* (DSLAM) [25] is a powerful tool for pose-graph Multi-robot decentralized applications in environments where absolute positioning is not available. DSLAM reduces every image to its NetVLAD (a Neural Network Architecture for Place Recognition) global descriptor [26]. To find loop closings, DSLAM seeks, for every *query* of one robot, the image of another robot whose NetVLAD descriptor presents the shortest distance to the descriptor of the *query* and this distance is below a certain threshold. This process is done for every frame of every robot that is inside a predefined cluster. DSLAM uses ORB-SLAM [27] for continuous localization, which includes a global image charectrization based on BoW for initial odometric estimates, and ORB [28] feature matching and RANSAC to calculate the 3D transform between confirmed visual loop closings.

The idea of *Cloud Computing* is applied in some cases to alleviate the computational charge needed for a set of robots to localize themselves and map the environment running a software architecture based on a multi-layer cloud platform [29]. In this later reference, robots use ORB-SLAM for self-localization and the multi agent SLAM is tested with the KITTI [30] public dataset and using a quadrotor drone in an outdoor environment. A few solutions integrate inertial with image data to perform Multi-robot graph SLAM. In [31], ORB visual features are tracked along consecutive frames and integrated together with the motion given by an IMU in a graph optimization context. BoW is also used to detect candidates to close inter-session loops. The BoW-based global image descriptor of a query image is compared to the global descriptor of all other images of the other agents, selecting a set of candidates to close inter-session loops with the query. Afterwards, a brute-force feature matching with RANSAC is applied to confirm the candidates or to reject them. Experiments in [31] are performed with aerial robots in industrial environments.

Previous references have been tested only in terrestrial indoor and outdoor environments. The literature is extremely scarce in Multi-robot SLAM addressed, implemented and tested in underwater scenarios with AUVs [9,32]. Underwater computer vision is affected by several challenging problems, such as flickering, reduced range, lack of illumination, haze, light absorption, refraction, and reflection. These limitations increase the need for more robust visual SLAM approaches which start with accurate camera calibrations. Accuracy in the processes of camera calibration is critical to reduce drift in the visual odometry and increase precision in the pose transform obtained from images that close loops [33,34]. Furthermore, none of the papers cited previously consider the potential impact of limited communications among robots, because, either they are applied in ground or aerial robots or they simply assume full, high-bandwidth connectivity. This supposal is clearly unrealistic in underwater environments, where blue-light laser communications need to be highly directive and acoustic USBL modems work, on average, at 13 Kbps for long range devices and up to 65 Kbps in mid-range devices. Additionally, the later speeds are not suitable to transmit medium-high resolution visual data between two robots without a previous compression. For instance, Pfingsthorn et al. [35]

proposed a visual pose-graph SLAM approach in which compressed JPEG-format images are send via acoustic links only between robots that can mutually recognize their positions and are viewing overlapping areas. Paull et al. [36] refuse the use of images for SLAM and trust all the localization process to an acoustic modem for data transmission and instruments that give relatively small amounts of data, if compared with images: compass, *Doppler Velocity Log* (DVL) and a Side Scan Sonar. Besides, they also apply a new strategy to marginalize unnecessary local information to reduce the dimensions of the transferable packages.

In the context of the ongoing national project TWINBOT (*TWIN roBOTs for Cooperative Underwater Intervention Missions*) [37], diverse missions of exploration and cooperative intervention have to be run using one or several AUVs in underwater areas with multiple appearances and different benthic habitats. In this project, accurate, fast and reliable robot localization, loop closing and navigation algorithms are crucial for the success of their missions. This paper presents a new approach to Multi-robot visual graph-SLAM, especially designed for 2'5D configurations, where vehicles move at a constant altitude with a camera pointing downwards, with the lens axis constantly perpendicular to the ground or to the vehicle longitudinal axis. This condition simplifies the visual system to 3 *Degrees of Freedom* (DoF): two for an in-plane translation (x, y) and another for rotation in yaw (θ). This simplification fits with aerial and underwater vehicle configurations, if the navigation altitude is large enough compared with the heigh of the terrain relief [38,39]. However, now tests have been made only with underwater datasets since the research developed by our team, in general [40], and the TWINBOT project in particular, is applied entirely and solely underwater, and this approach emerged as a solution to be applied on the robots that participate in our project missions.

The approach presented now includes several contributions that represent clear advantages with respect to the existing solutions, namely:

(a) As in [25], images are reduced to global descriptors decreasing drastically the amount of visual data to be exchanged among robots; however, the global descriptor used now is HALOC [22] instead of NetVLAD. The construction of HALOC is simpler and faster than NetVLAD, consisting in projecting all image features on a base of orthogonal vectors, without any need for tedious and long training tasks. This is a clear advantage over the previous work, since HALOC already showed to outdo VLAD [20] and BoW, in speed and performance for loop closing detection, in both terrestrial public benchmaks and underwater environments. HALOC also showed a performance better than ORB-SLAM, only underwater. Additionally, extensive experiments with HALOC performed in marine areas partially colonized with seagrass [41,42] also revealed an excellent efficiency, capacity and utility for loop closing detection in this type of environments.

(b) A second important contribution is the simplification of the whole system with respect previous approaches. Ours does not require neither the computation of relative poses among robots, nor a specific strategy to limit their communication and interaction. At every SLAM iteration, the quantity of bytes to be exchanged between robots is so small that this will not necessarily limit the communication between all agents that participate in the mission, if needed.

(c) The global procedure includes local and global SLAM tasks, with a map joining process in between. The advantage of this point lies on the flexibility to choose the moment at which the map joining is performed, giving priority to local routes as accurate as possible, or delaying the major corrections once all maps have been joined.

(d) The present approach goes one step beyond its predecessors, since the joined graph incorporates and reflects, online, the successive poses of all robots that move simultaneously.

(e) The localization and motion problem is simplified to 2D. Furthermore, it avoids complex multi-layer software architectures or Cloud computing strategies present in previous solutions.

(f) One of the principal objectives has been the reduction of the computational requirements of the algorithm, since, in general, they are limited in lightweight underwater vehicles. Running the algorithm online onboard the vehicles is a must, since it is especially addressed to multi-robot configurations, and these configurations imply controlling, mapping and guiding several robots

moving simultaneously, where usually, one centralizes the processing of the localization data of the whole group.

Furthermore, although they are not directly novel contributions, it is worth mentioning two additional advantages in the implementation: (i) similarly to [32] or [35], once maps of different robots are joined, standard graph-based topology representations are used, where images form nodes and transforms between two images (being from consecutive frames or between two images that close a loop) form edges or links, and (ii) the graph is optimized by means of standard *g*2*o* [43]; this standardization facilitates the exchange of the different modules on a variety of software platforms and their reuse among different implementations.

Although this is out of the scope of this paper, this vision-based algorithm can complement the navigation facilities of underwater vehicles equipped with multiple types of sensors. In fact, this algorithm can integrate additional sensorial data in the first estimation of the vehicle motion, combining visual odometry with other means of laser or sonar-based dead reckoning [44,45].

The source code of the whole approach has been made publicly available for the scientific community in several GitHub repositories, together with a simple underwater dataset to test the whole procedure. Links to sources are provided in Section 3.

Section 2 contextualizes and details all algorithms proposed to: (a) estimate the visual odometry, (b) detect intra- and inter-session loop closings, (c) perform the local trajectory-based SLAM and (d) join maps and optimize the global graph. Section 3 presents some qualitative and quantitative preliminary results. Finally, Section 4 concludes the paper and gives some indications of ongoing and upcoming tasks to continue and improve this work.

2. Materials and Methods

2.1. Overview

The proposed localization module is based only in vision, with no intervention of either dead-reckoning navigation instruments, such as IMU or DVL, or global positioning systems, such as GPS for surface vehicles or Ultra-short Baselines (USBL) for underwater vehicles.

The structure of the proposed system is as follows:

(1) Let us simplify the problem assuming that there are, for instance, two vehicles moving simultaneously over the same area of interests in such a way that there is no possibility of collision, and that part of the area will be explored by both robots.

(2) The approach starts by estimating the trajectory of each robot motion, separately, applying the trajectory-based visual-SLAM strategy included in the multi-session scheme of Burguera and Bonin-Font [46]. Let us refer to this step as the *intra-session SLAM*. The indicated trajectory-based scheme implies that the trajectory of each robot is estimated by means of compounding [47] successive displacements calculated from one point to the next. These successive displacements form the state of an Extended Kalman filter (EKF) which is updated using the transforms given by the confirmed loop closings. In our particular case, this displacement corresponds to the visual odometry calculated between consecutive images, and the images candidate to close a loop with the current image are found comparing the corresponding image hashes and confirmed by a RANSAC-based algorithm applied on a brute-force visual feature-matching process. The main difference with respect to [46] is that while in a multi-session localization procedure the trajectory of the currently running robot is joined to another trajectory already completed and available in its totality, now, in a Multi-robot scheme, both robots are moving at the same time to complete a mission in which both participate simultaneously. When joined, both trajectories are incomplete, and continue running until all robot missions are finished.

(3) Simultaneously to both *intra-session SLAM* tasks, the system also searches for inter-session loop closings using HALOC. The global signature of each new image captured by each robot is compared with the global signatures of all images of the other robot. Once a certain number of inter-session

loop closings are confirmed, both routes are joined in a single graph. Let us refer to this step as the *Map Joining*.

(4) Once joined, the global map (graph) must be completed with the successive poses of both robots until the end of both sessions. Furthermore, the trajectory-based localization approach applied to both agents separately is not longer valid. Each new displacement of both robots is included in the form of new nodes on the graph. Each new node of the graph will follow the direction of motion of each vehicle, which means that the graph will grow in two different directions, according to the two different trajectories, but forming a single entity. Let us denote this step as the *Multi-robot SLAM*.

(5) The global graph completion and optimization is done using a pose-based scheme, i.e., all new nodes corresponding to each robot will contain their successive global poses with respect to the origin of a unique world coordinate system, while the links between nodes will contain the displacement between them. After the map joining, only the inter-session loop closings are used to optimize the global map. If one loop closing between different sessions is confirmed, its resulting transform is used as an additional constraint between two nodes to optimize the whole graph. In our case, the optimization solver used is the *g*2*o* implementation [43] of the *Levenberg Marquardt* algorithm [48,49].

2.2. Intra-Session SLAM and Map Joining

2.2.1. Visual Odometry

Figure 1 shows the global idea behind this first step of the approach. The visual odometry gives the estimated 2D motion between consecutive images by means of a SIFT feature detection and matching procedure.

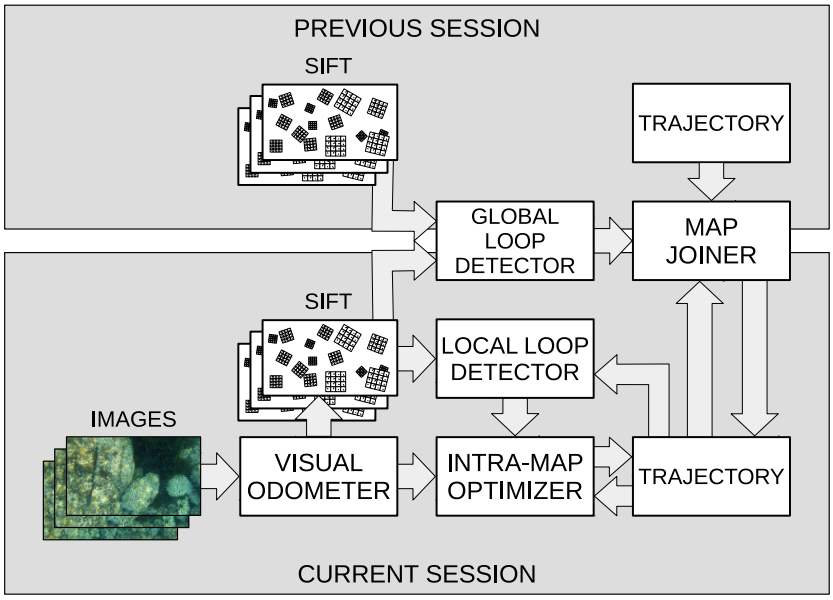


Figure 1. Sigle Session SLAM overview.

Algorithm 1 shows the RANSAC-based method used to register two images, i.e., get the transform (if it exists) between them in translation and rotation. $apply\_altitude()$ is a function that converts image feature coordinates from pixels to meters by considering the altitude at which the AUV navigates, as well as the camera parameters.

Algorithm 1: RANSAC approach to estimate the motion $\hat{X}_B^A$ from image I_A to image I_B.

Input:
f_A, f_B: SIFT features in images I_A and I_B
a_A, a_B: Altitudes corresponding to I_A and I_B
C: Set of correspondences
K: Number of iterations to perform
N_{corr}: Number of correspondences to be randomly selected
N_{min}: Minimum number of correspondences to consider a roto-translation as candidate
ε_{corr}: Maximum allowable error per correspondence

Output:
$fail$: Boolean stating if failed to find $\hat{X}_B^A$
$\hat{X}_B^A$: The estimated roto-translation

begin
  $f'_A \leftarrow apply\_altitude(f_A, a_A)$;
  $f'_B \leftarrow apply\_altitude(f_t, a_B)$;
  $\varepsilon_B^A \leftarrow \infty$; $fail \leftarrow true$;
  for $i \leftarrow 0$ **to** $K-1$ **do**
    $R \leftarrow$ random selection of N_{corr} items from C;
    $X \leftarrow \arg\min_T \sum_{(i,j)\in R} ||T \oplus f'_{A,i} - f'_{B,j}||$;
    $\varepsilon \leftarrow \sum_{(i,j)\in R} ||T \oplus f'_{A,i} - f'_{B,j}||$;
    foreach $(i,j) \in (C-R)$ **do**
      if $\|X \oplus f'_{A,i} - f'_{B,j}\| < \varepsilon_{corr}$ **then**
        $R \leftarrow R \cup \{(i,j)\}$;
      end
    end
    if $|R| > N_{min}$ **then**
      $X \leftarrow \arg\min_T \sum_{(i,j)\in R} ||T \oplus f'_{A,i} - f'_{B,j}||$;
      $\varepsilon \leftarrow \sum_{(i,j)\in R} ||T \oplus f'_{A,i} - f'_{B,j}||$;
      if $\varepsilon < \varepsilon_B^A$ **then**
        $\varepsilon_B^A \leftarrow \varepsilon$; $\hat{X}_B^A \leftarrow X$; $fail \leftarrow false$;
      end
    end
  end
end

The idea behind this algorithm is that correct correspondences lead to the same roto-translation while wrong feature matchings lead to different and wrong roto-translations. The algorithm selects a random subset R of correspondences from the total number of correspondences C between two images, and then computes the roto-translation X and the subsequent error ϵ using only this subset. Afterwards, if the error introduced by the non-selected matchings of C is below a threshold ϵ_{corr}, then, these matchings are included in R. If at any moment, the number of elements in R surpasses a threshold N_{min}, the roto-translation and the error are computed again using this expanded R. If the error is below the smallest error obtained until this moment, the roto-translation is kept as a good model. This process is iterated a certain number of times. If partial roto-translations are inconsistent and R never reaches the minimum number of items required, the algorithm will not return any transform, but a boolean called $fail$ set to $true$. The obtained transform can be assumed to be the odometric displacement between consecutive images and the trajectory of the robot between steps i and j (X_j^i) (assuming step j being subsequent to i) can be estimated using the compounding $\oplus$ operator of the successive odometric displacements ($X_{i+1}^i, X_{i+2}^{i+1}, \ldots, X_j^{j-1}$), as described in [50]:

$$X_j^i = X_{i+1}^i \oplus X_{i+2}^{i+1} \oplus \cdots \oplus X_j^{j-1} \quad j > i \ . \tag{1}$$

2.2.2. Local Loop Detection and Trajectory Optimization

Local loops are those found within a single SLAM session. The Loop Candidates set (LC_t) is the set of images I_i that may close a loop with the last gathered image I_t obtained in each running trajectory. This set is built by searching in a region within a predefined radius δ [38] around the current robot pose as estimated by the odometry:

$$LC_t = \{i : ||X_t^i||_2 \leq \delta, i < t-1\} \tag{2}$$

where X_t^i is computed by Equation (1).

Every image contained in the set of loop closing candidates ($I_i \in LC_t$) is registered with I_t using Algorithm 1, in order to build the set of local loops LL_t, being $LL_t = \{Z_t^i : i \in LC_t \cap \neg fail(i,t)\}$, where $\neg fail(i,t)$ indicates that Algorithm 1 did not fail to get a roto-translation Z_t^i between I_i and I_t.

The trajectory estimation obtained by means of compounding the successive odometric displacements between two points A and B will most likely not coincide with the direct transform between images obtained in A and B provided by the image registration process of Algorithm 1, if A and B close a loop:

$$X_{A+1}^A \oplus X_{A+2}^{A+1} \oplus X_{A+3}^{A+2} \oplus \cdots \oplus X_{B-1}^{B-2} \oplus X_B^{B-1} \neq Z_B^A \tag{3}$$

due to the drift introduced by the visual odometry and the error inherent to the transform directly obtained from the image registration procedure. Figure 2 illustrates these concepts.

Afterwards, a process of global optimization is run to get a trajectory that best combines the pose constraints imposed by the set of local loops (LL_t) and the odometry. As mentioned before, the trajectory of the robot is the state vector of a *Iterative Extended Kalman Filter* (IEKF). Each new odometric displacement (X_t^{t-1}) computed between the last image and the previous one is used to augment the state vector at time t (X_t^-): $X_t^- = \begin{pmatrix} (X_{t-1})^T & (X_t^{t-1})^T \end{pmatrix}^T$. In the prediction stage of the IEKF the state vector does not change at all.

If LL_t is not empty, the trajectory is optimized performing the Update stage of the IEKF using the set of loop closings as measurements. The observation function h_t^i associated with each measurement Z_t^i (the transform of each real loop closing) can be defined as $h_t^i(X_t^-) = X_{i+1}^i \oplus X_{i+2}^{i+1} \oplus \cdots \oplus X_{t-1}^{t-2} \oplus X_t^{t-1}$, being the innovation of the IEKF for each measurement: $Z_t^i - h_t^i(X_t^-)$. With all this elements, one can iterate the classical equations of an EKF to get the optimized trajectory, until $||X_{t,j}^- - X_{t,j-1}^-|| < \gamma$, where j represents the last iteration and γ is a predefined threshold. The classical format of the IEKF involves iterating until the changes between consecutively estimated states are below a certain threshold. However, in the experiments we verified that after a certain and almost constant number of iterations, the filter already converged with a difference between consecutive results below the threshold ($||X_{t,j}^- - X_{t,j-1}^-|| < \gamma$). Because of that, it was decided to repeat all the experiments with a defined number of iterations, in order to limit the number of executions to be done and save computational resources. In any case, both options can be used in other circumstances, depending on the needs and environmental conditions of each different system and field case.

2.2.3. Inter-Session Loop Closings

The main problem for joining two trajectories of two robots operating simultaneously is the lack of geometric relation between their corresponding sessions. Every robot geo-localizes itself with respect to the origin of its own trajectory, but it has no knowledge about the origin of the other trajectories. By means of finding inter-session loop closings, i.e., images that show, partially or totally, the same area, but taken by two different robots in two different sessions running simultaneously, the maps of the two robots can be joined in a single one [46]. Due to this lack of geometrical relation between the two trajectories, the search of the loop closing candidates of one robot to close a loop with the last image captured by the other robot cannot be restricted to a certain area. In this case, one should

compare the last image gathered by one of the robots with all images taken by the other robot, from the start of the mission until the current moment. Applying a brute force feature matching algorithm between all these involved images is unfeasible for online applications, due to the great amount of computing resources and time needed. One way to alleviate this problem is reducing all images to global descriptors. As in [46], all images of both sessions are reduced to their HALOC global descriptor. The size of HALOC is fixed in 384 floats since the size of the used projective orthogonal vectorial space is 3 [22]. This length is independent of the number of visual features found in each image. The global descriptor of every new image of one of the sessions (called the query image) is compared with the global descriptor of all and each of the images taken during the other session. According to [22], those 5 images that give the lowest L1-norm of the difference between their hash and the query hash, and this norm is lower than a certain threshold δ' are taken as the inter-session loop closing candidates (GC_t):

$$GC_t = \{i : ||H_i - H_t||_1 \leq \delta', \forall I_i \in V_p\} \tag{4}$$

being V_p the set of images taken by one of the robots from the start of its session until the current moment, H_i the hash of each of these images and H_t the hash of the query image. The value of δ' will be selected experimentally.

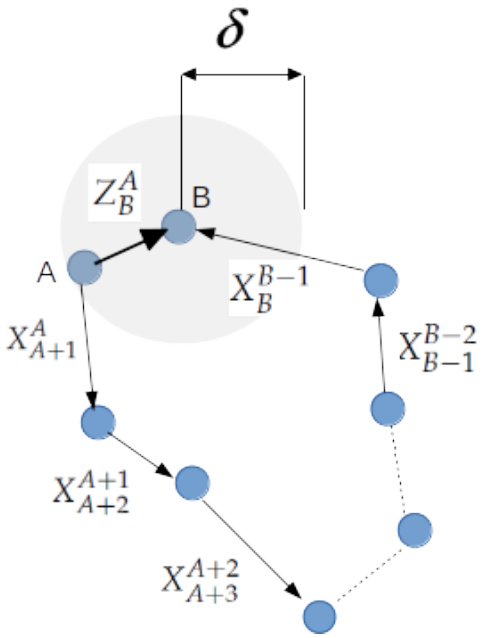


Figure 2. In theory, the transform between A and B, if both close a loop should be very close to the transform obtained compounding the odometric displacements X_{i+1}^i.

Once the set of candidates is established, the true positives are confirmed by means of the RANSAC-based Algorithm 1, forming the definitive set of images (GL_t) of the first session that, in principle, close a loop with the last image of the second session, as: $GL_t = \{Z_t^i : i \in GC_t \cap \neg fail(i,t)\}$, being Z_t^i the transformation found by Algorithm 1. Inter-session loops are accumulated at every iteration of both single SLAM sessions. These transforms Z_t^i, are, in fact, a set of geometrical relations between the two different sessions. Assessing the performance of HALOC in loop closing detection, in terms of accuracy, recall and fall-out, is out of the scope of this paper, since it has already been presented in [22,42] with considerable good results underwater.

2.2.4. Map Joining

As mentioned in the previous section, the loop closings between different sessions can be used to infer the geometrical relation between the two trajectories of the two robots that perform both missions

simultaneously. The objective now is to align, at a certain moment, both surveys in a single global graph, and, maintain this single graph from the moment of the joining to the end of both missions.

Let X_1 denote the trajectory of one of the sessions. Let the first and last images of this trajectory be denoted as I_{1s} and I_{1e}, respectively. Let X_2 denote the trajectory of the second session and let us denote its first and last images as I_{2s} and I_{2e}, respectively. Let us denote the number of accumulated inter-session loop closings at a certain moment t as K. Let us also denote this set of loop closings as Z_G, each one relating one image of the first session with another image of the second session.

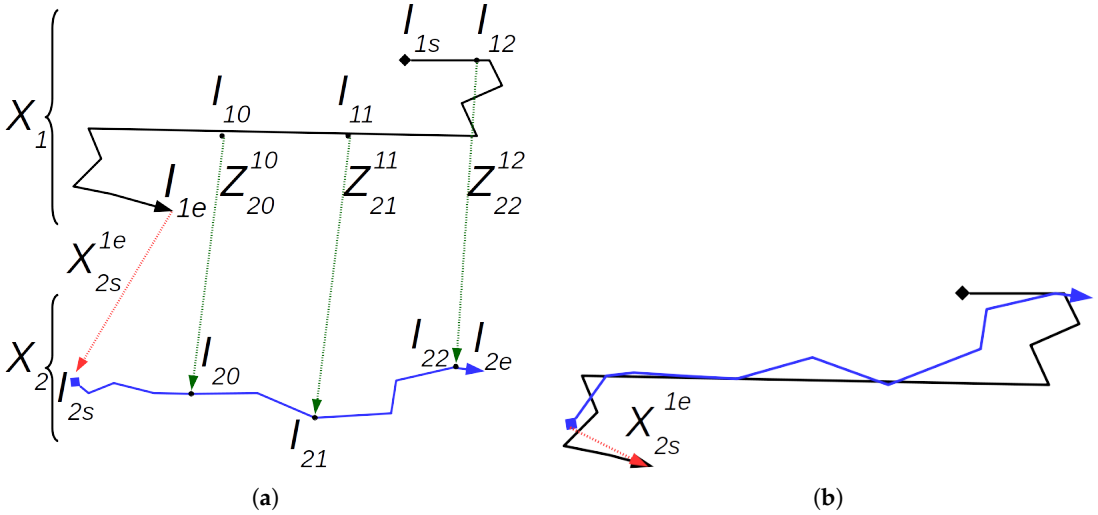
$$Z_G = \begin{pmatrix} (Z_{20}^{10})^T & (Z_{21}^{11})^T & \dots & (Z_{2K-1}^{1K-1})^T \end{pmatrix}^T \tag{5}$$

where each Z_{2i}^{1i} represent a transform from image I_{1i} of the session 1 to image I_{2i} of session 2, or what is the same, the transforms of the loop closings. Each Z_{2i}^{1i} belongs to a certain GL_t.

Let us define X_{2s}^{1e} as the transform, or the relative motion, from I_{1e} to I_{2s}. For every loop closing, ideally, $Z_{2i}^{1i} = X_{1e}^{1i} \oplus X_{2s}^{1e} \oplus X_{2i}^{2s}$, where X_{1e}^{1i} is the displacement from I_{1i} to I_{1e}, and X_{2i}^{2s} is the displacement from I_{2s} to I_{2i}, being:

$$X_{1e}^{1i} = (x_{1e}^{1i}, y_{1e}^{1i}, \theta_{1e}^{1i})^T \tag{6}$$
$$X_{2s}^{1e} = (x_{2s}^{1e}, y_{2s}^{1e}, \theta_{2s}^{1e})^T \tag{7}$$
$$X_{2i}^{2s} = (x_{2i}^{2s}, y_{2i}^{2s}, \theta_{2i}^{2s})^T \tag{8}$$

The proposal consists of an IEKF that will give the value of X_{2s}^{1e} that better matches all the loop closures found until the moment t. The state vector of the IEKF is just the transform X_{2s}^{1e}, the observation function for each loop closing will be $g_G^i = X_{1e}^{1i} \oplus X_{2s}^{1e} \oplus X_{2i}^{2s}$, and Z_{2i}^{1i} is the corresponding measurement. With this, one can form the innovation, and apply the classical EKF equations iteratively, as explained in Section 2.2.2. X_{2s}^{1e} is the transformation that can be used to join the two sessions, in such a way that the state vectors of both trajectories, formed by displacements, are joined by this recently computed transformation as: $X_J = \left((X_1)^T (X_{2s}^{1e})^T (X_2)^T\right)^T$, where X_J represents the joined trajectory, and $X_1 // X_2$ the state vector of the first and second trajectories, respectively, from their starting points until the instant t. The idea is illustrated in Figure 3.



(**a**) (**b**)

Figure 3. (**a**) Separated trajectories with intersession loop closings. (**b**) The joined trajectory.

Once both sessions have been joined, the trajectory-based schema is no longer valid, and the resulting map is transformed into a pose-based graph. All the robot displacements included in X_J are transformed into global poses that constitute each node of the global graph as:

$$X^i = \begin{cases} X_1^0 \oplus X_2^1 \oplus \cdots \oplus X_{i-1}^{i-2} \oplus X_i^{i-1} & i <= 1e \\ X_1^0 \oplus X_2^1 \oplus \cdots X_{1e}^{1e-1} \oplus X_{2s}^{1e} \oplus X_{2s+1}^{2s} \oplus \cdots \oplus X_i^{i-1} & i > 1e \end{cases} \quad (9)$$

where X^i is the pose associated with node i. The case for $i <= 1e$ refers to the global pose of a node corresponding to an element of the first trajectory, and the case for $i > 1e$ refers the global pose of a node of the second trajectory. All the displacements included in X_J become the links between successive nodes and each node of the graph is associated with its corresponding image.

2.3. Multi-Robot Graph SLAM

Let us assume that, (a) after both trajectories have been joined in a single graph, both robots are still running their own missions, and, (b) the successive poses of both robots must be included in the global graph as new nodes, each one following the corresponding trajectory.

The Multi-robot Graph-SLAM procedure detailed below follows the indications of [51], in terms of structure, node generation, inclusion of loop closings as additional pose constraints, and graph optimization, but in our case particularized for a Multi-robot configuration. The algorithm includes the next points:

(1) The local SLAM algorithm explained in Section 2.2 is continuously executed for both trajectories until them are joined when a certain number of inter-session loop closings have been accumulated. It is better to optimize the local trajectories every N frames, although there is only a couple of loop closings, in order to, when both sessions are joined the drift has already been reduced locally, as much as possible. Otherwise, trajectories could be joined before local optimizations have been applied, transferring local drifts to the global map.

Let us assume that the Multi-robot localization is centralized in the first robot, which will receive, from the second robot: (1) The set of visual features and the global descriptor of the last gathered image, (b) only if the map joining has to be done, the state vector and the last odometric displacement. The state vector is needed to be attached to the one of the first robot, if it is due. The last frame global descriptor is needed to find possible loop closures with frames of trajectory 1, the set of features is needed to confirm or reject the possible candidates, and the last odometric displacement of trajectory 2 will be used as a reference after the map joining.

(2) Once both trajectories have been joined in a single global graph, it is time to feed the map with the successive displacements of both robots. It is important to note that the last node of the graph corresponds to the last displacement of the second trajectory, since the first set of elements correspond to the trajectory of robot 1, then it comes the link between trajectories, and finally the elements of trajectory 2. Let us denote the identifier ID of the last node of the global graph as N_{n2}, where $n2$ represents the number of nodes in the graph, and is equal to the length of the joined state vector X_J ($|X_J|$). Accordingly, the ID of the graph node corresponding to the end of the trajectory 1 will be N_{n1}, where $n1 = |X_J| - 1 - |X_2|$.

Storing N_{n1} and N_{n2} is necessary, since they will be the points of the global graph from which the successive nodes corresponding to trajectories 1 and 2, respectively, will be placed according to the ongoing motion of both vehicles.

The set of iterated actions performed for the Multi-session SLAM are:

1. Let us denote the last (or next) computed odometric displacements of trajectories 1 and 2 as X_{n1} and X_{n2}, respectively. These displacements together with the last images of both session are stored in the system.
2. If trajectory 1 has not finished, add a new node (N_{n1+1}) to the graph, linked to N_{n1} with the transform X_{n1}. The global pose contained in this node will be: $X^{n1+1} = X^{n1} \oplus X_{n1}$. N_{n1+1} will be the last node of trajectory 1.
3. If the trajectory 2 has not finished, add a new node (N_{n2+1}) to the graph, linked to N_{n2} with the transform X_{n2}. The global pose of this node will be: $X^{n2+1} = X^{n2} \oplus X_{n2}$. N_{n2+1} will be the last

node of trajectory 2. The link between nodes X^{n1+1} and X^{n1}, and the link between nodes X^{n2+1} and X^{n2} will contain the values of X_{n1} and X_{n2}, respectively. Each new node added on the graph is associated in the code to its corresponding image, regardless the trajectory it belongs to. In this way, with the node ID one can find it associated image, and with an image identifier, one can find its associated node ID.

4. Search for inter-session loop closings between the last image of session 2 and all images of session 1 using the algorithm explained in Section 2.2.3. Those candidates of session 1 retrieved by HALOC that present a transform after the RANSAC discrimination process with several *inliers* lower than a pre-fixed parameter ($MinRansacInliers$), are discarded and considered false positives that can harm the result of the graph optimization. The rest are accumulated and considered true positives. Let us name the number of true positives that close a loop with the last image of trajectory 2 as N_{TP}. For each true positive, the system stores the next data: (a) The name of both images that close the loop, (b) the identifiers IDa and IDb of both nodes involved in the loop closing and (c) the transform between both images (Z_{IDbi}^{IDai}).
5. Let us denote the number of accumulated inter-session loop closings as N_{ALC}, initialized to 0 when both sessions are jointed. Then, $N_{ALC} = N_{ALC} + N_{TP}$. When $N_{ALC} = NIsLoopClosings$, where $NIsLoopClosings$ is preset at the beginning of the process, then the graph is optimized with all the new pose constraints, following the next steps:

   (a) Recover the node IDs of the images associated with each inter-session loop closing classified as true positive, and every corresponding transform.
   (b) Add one additional link in the graph between nodes $IDai$ and $IDbi$, which content is Z_{IDbi}^{IDai}, $\forall i, 1 \leqslant i \leqslant N_{ALC}$.

6. Run the graph bundle adjustment using the Levenberg Marquardt algorithm. Even if after a certain number of iterations no inter-session loop closings are found, the graph will be optimized as well, just to re-adjust the odometric trajectory estimates.
7. $N_{ALC} = 0$, $N_{TP} = 0$, $n1 = n1 + 1$ and $n2 = n2 + 1$.
8. Return to the first step, and iterate the process until both trajectories are finished. If one of the two trajectories finishes before the other one, the system keeps adding the corresponding nodes of the session that is still on course. Obviously, no additional inter-session loop closings will be found in this case, so every graph optimization will include only the pose estimates given by the visual odometry of the ongoing mission.

The idea is illustrated by Algorithm 2.

Algorithm 2: Multi-robot Visual Graph SLAM.

Inputs
X_{n1}, X_{n2}: Last odometric displacements of Trajectories 1 and 2 before the map joining.
N_{n1}, N_{n2}: Identifiers (ID) of the last graph nodes corresponding to trajectories 1 and 2, after the map joining.
X^{n1},X^{n2}: Global poses corresponding to nodes N_{n1} and N_{n2}, after the map joining. $X^{n1} = X^{n1-1} \oplus X_{n1}$
I_{n1}, I_{n2}: last images taken by robots 1 and 2 at instants $n1$ and $n2$, just before the map joining.
Parameters
MinRansacInliers: Minimum number of RANSAC inliers to consider a transform between two images as a true positive
NIsLoopClosings: Maximum number of accumulated inter-session loop closings.
N_{min}: Minimum number of correspondences to consider a roto-translation as candidate.
N_c: Number of image candidates to be searched in Trajectory 1 to close a loop with I_{n2}
Variables
I_{1j}: Jth image of the first session, candidate to close a loop with a query image, found using HALOC.
N_{TP}: Number of images of trajectory 1 considered as true positives that close a loop with I_{n2}
N_{ALC}: Number of accumulated inter-session loop closings.
IDa_n, IDb_n, Ia_n, Ib_n: Graph nodes involved in the nth inter-session loop closing and images corresponding to each node.
$Z_{IDb_n}^{IDa_n}$: Transform (x, y, θ) associated with the nth inter-session loop closing.
ListOfCandidates: Structure that contains the list of image candidates to close a loop with a given query. Every element of the structure stores the image Id, its HALOC hash, and the number of features.
$H_{trajectory1}$: List of Hashes (global descriptor) type HALOC of all images of trajectory 1.
Functions
$[X] = RansacEstimateMotion(I_1, I_2)$: returns the odometric displacement (X) between images I_1 and I_2 using Algorithm 1,
$H = hash(I)$: is the function of the HALOC library that returns the HALOC global descriptor H of image I
$[ListOfCandidates] = LibHALOC(H_{trajectory1}, N_c, H_{I2})$: is the function of the HALOC library that gets N_c candidates of the trajectory 1 to close a loop with the query I_2.
$AddRelativePose(Z_2^1, I_1, I_2)$: adds a new pose constraint (link with transform $Z_{I_2}^{I_1}$) between two graph nodes, I_1 and I_2
$OptimizeGraph()$: Does the global bundle adjustment of the whole graph using the Levenberg-Marquard algorithm.

begin
  $N_{ALC} = N_{TP} = n = 0$;
  Robot 1 takes the next image $\rightarrow In1 + 1$;
  Robot 2 takes the next image $\rightarrow In2 + 1$;
  H1=hash($In1 + 1$); H2=hash($In2 + 1$) ;
  $H_{trajectory1} \leftarrow$ H1 ;
  $[X_{n1+1}]$ = RansacEstimateMotion($In1, In1 + 1$).;
  $[X_{n2+1}]$ = RansacEstimateMotion($In2, In2 + 1$).;
  $X^{n1+1} = X^{n1} \oplus X_{n1+1}$. $N_{n1+1} \rightarrow$ node graph ID of X^{n1+1} ;
  $X^{n2+1} = X^{n2} \oplus X_{n2+1}$. $N_{n2+1} \rightarrow$ node graph ID of X^{n2+1} ;
  Store the correspondences $N_{n1+1} \rightarrow In1 + 1$ and $N_{n2+1} \rightarrow In2 + 1$; ;
  $[ListOfCandidates] = LibHALOC(H_{trajectory1}, N_c, H2)$;
  for $j \leftarrow 0$ **to** N_c **do**
    store $Ia_n = I_{1j}$, $Ib_n = I_{n2+1}$;
    $[Z_{IDb_n}^{IDa_n}] = RansacEstimateMotion(Ia_n, Ib_n)$;
    if *Number of Inliers between* Ia_n *and* $Ib_n \geqslant MinRansacInliers$ **then**
      $N_{TP} = N_{TP} + 1$;
      store $Z_{IDb_n}^{IDa_n}$ and IDa_n, IDb_n ;
      $n = n + 1$;
    end
  end
  N_{ALC}=N_{ALC}+N_{TP} ;
  $N_{TP} = 0$;
  if N_{ALC}=*NIsLoopClosings* **then**
    for $i \leftarrow (n - N_{ALC})$ **to** $n + N_{ALC}$ **do**
      $AddRelativePose\ (Z_{IDb_i}^{IDa_i}, IDa_i, IDb_i)$;
    end
    $N_{ALC} = 0$,
  end
  $OptimizeGraph()$;
  $n1 = n1 + 1$, $n2 = n2 + 1$;
end

3. Experimental Results

3.1. Experimental Setup

A set of preliminary experiments were performed simulating the Multi-robot configuration with real underwater data. Three different datasets were used. Every dataset consists of two different video sequences, partially overlapping. Sequences were recorded either by a diver or by an AUV, both carrying a bottom looking camera with its lens axis perpendicular to the longitudinal axis of the vehicle or of the diver. Divers or autonomous underwater vehicles moved at an approximate constant altitude with respect to the sea bottom. The lack of any other sensorial data which could be supplied by an AUV, makes the localization system a pure vision-based approach. The three datasets have been recorded in several coastal sites of the north and south of Mallorca, at depths between 5 and 13 m. All the environments where the datasets were grabbed present a great variety of bottom textures, including seagrass, stones, sand, algae, moss and pebbles. The first dataset was recorded in the north coast of the island, by a diver with an attached Gopro camera, pointing downwards, moving on the surface at an approximate constant altitude of 4 m. Let us refer to both video sequences of this first dataset as $S11$ and $S12$. The camera altitude was obtained at the beginning of the video sequence by means of a visual marker of known size, placed at the sea bottom, in the starting point of each trajectory.

A second dataset formed by two partially overlapping trajectories named $S21$ and $S22$ were recorded also in the north coast of Mallorca, also by a diver supplied with a Gopro, looking to the bottom, far from $S11$ and $S12$, swimming on the water surface, at an approximate constant altitude of 4 m. In this case, the initial altitude was computed thanks to the know dimensions of a structure formed by markers and PVC tubes placed at the sea floor in the origin of both trajectories. The video resolution was 1920×1080 pixels, grabbed at 30 frames per second (fps), and prior to their use, all images were scaled down to 320×180 pixels.

A third dataset, with two video sequences named $S31$ and $S32$, was recorded by a SPARUS II AUV [52] property of the University of the Balearic Islands, at 7.5 fps, moving at a constant altitude of 3 m, in an area of the south of the island with an almost constant depth of 16 m. The navigation altitude is obtained from the vehicle navigation filter which integrates a Doppler Velocity Log (DVL), an Inertial Measurement Unit (IMU), a pressure sensor, an Ultra Short Baseline (*USBL*) acoustic modem [53], and a stereo 3D odometer. This dataset permitted to test the approach in larger environments with complex imagery due to the presence of sea grass on the sea bottom. In particular, $S31$ was recorded during a trajectory of 93 m long, and $S32$ during a a trajectory of 114 m, covering both an approximate area of 300 M^2 each one.

Figure 4 shows some samples of images included in the three datasets.

All these images show how all regions are colonized with *Posidonia oceanica*, a seagrass that forms dense and large meadows. Images of dataset 3 show a lack of illumination which increases at larger depths. With these conditions, the feature matching process decreases its performance and affects directly the accuracy of the visual odometry and the loop closing detection using HALOC. In this type of marine environments and with our robot and its equipment, moving at approximately 1 knot at altitudes between 3 m and 5 m, in areas with a depth between 16 m and 20 m, gave a good tradeoff between image overlap and illumination conditions.

Due to the particular texture of the Posidonia and the slight motion of its leafs caused by the currents, tracking stable visual features in consecutive overlapping frames is complicated and requires an accurate selection of the type of features and the feature detection/tracking parameters. Errors in this process will compromise the accuracy of the visual odometry and the image registration task for the loop closing confirmation. Previous pieces of work [22,41,42] already showed the high efficiency of SIFT features for underwater SLAM in areas colonized with Posidonia, in all the tasks involved in the process: visual odometry, image hashing with HALOC, loop closing detection, and pose refinement. Although SIFT feature detector is slower than other descriptors and delays the RANSAC-based

matching process of Algorithm 1, given the robustness and traceability of SIFT, and according to our experience, this additional processing time is preferable to obtain more reliable trajectories than using other simpler features that take less time than SIFT to be computed and tracked, but can cause larger inaccuracies in the camera trajectory estimation or in the image registration process.

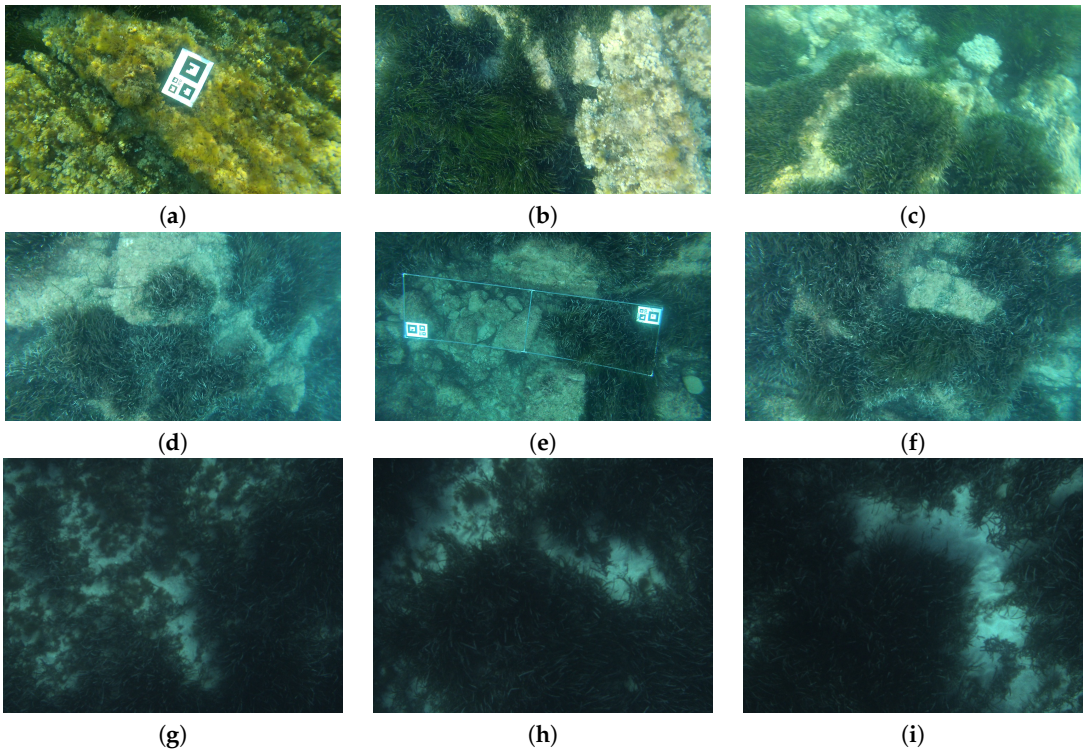


Figure 4. Examples of images of dataset 1, in (**a**–**c**), dataset 2, in (**d**–**f**), and dataset 3, in (**g**–**i**).

3.2. Experiments and Results

All experiments performed to get these first preliminary results were run offline, simulating the Multi-robot configuration with the visual data obtained in the sea. Key images of each video sequence that forms the different datasets mentioned in the previous section were extracted, indexed, stored separately in a hard disk, and processed consecutively according to the algorithms exposed in this paper.

Successive field experiments showed that, an overlap between consecutive images of 35% to 50% was necessary to obtain a robust visual odometry. On the other hand, reducing the number of images stored was also required in order to save as much memory space as possible. Consequently, a good trade off between both requirements was obtained selecting the keyframes of datasets 1 and 2 down-sampling the initial video frame rates at 1.1 fps, on average, and the dataset 3 at 3 fps. 226 key images were extracted from $S11$ and 199 from $S12$. 152 key frames were extracted from the video sequence $S21$ and 57 from $S22$. Finally, a total of 400 key images were extracted from dataset 3, 200 belonging to $S31$ and 200 to $S32$.

Local SLAMs are continuously executed for both trajectories in sequential steps of a predefined number of frames N; each local trajectory is accumulated and optimized from N to N frames, alternating both sessions every N frames. N is set differently for each dataset.

Figure 5 shows several samples of inter-session loop closings found by HALOC and confirmed by Algorithm 1.

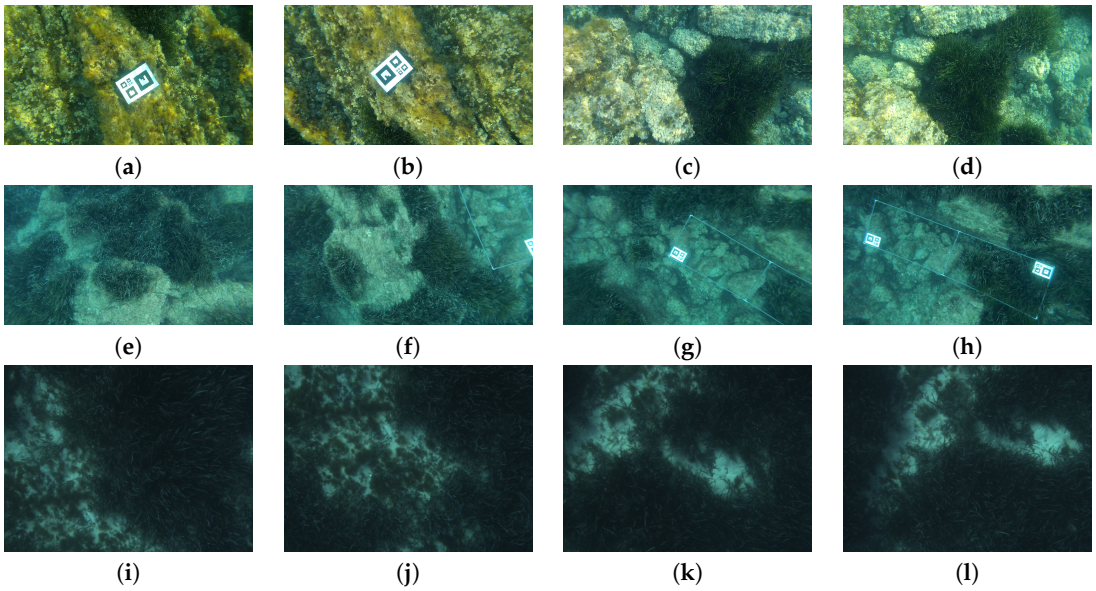


Figure 5. (**a**–**d**) Multi-session loops between sequences *S*11 and *S*12. (**e**–**h**) Multi-session loops between sequences *S*21 and *S*22. (**i**–**l**) Multi-session loops between sequences *S*31 and *S*32. (**a**) Closes a loop with (**b**,**c**) closes a loop with (**d**,**e**) close a loop with (**f**,**g**) close a loop with (**h**,**i**) closes a loop with (**j**,**k**) closes a loop with (**l**).

Once images and inter-session loops are available, the whole localization and mapping process starts. In other words, the sequence of actions is as follows:

1. For each dataset, extract the key images of both video sequences and store them in separated folders
2. For each dataset, compute the HALOC global descriptor of each image extracted from both video sequences.
3. For each dataset, compute and store in a file, the odometry, frame to frame, for both stored image sets, corresponding to both sessions.
4. At this point, for each dataset and for each of their sessions, the key frames and the odometry have been stored and related through successive identifiers. Thereafter, for each dataset run the local SLAM procedure, which:
    (a) Starts algorithm of Section 2.2, building the state vector of each session, by steps of N consecutive frames, using the displacements included in each odometry file.
    (b) For each newly gathered image (lets call it, the query image), searches for local loop closings on other images of the same dataset which positions are near the query. This search is done only among the images gathered before the query.
    (c) Optimize both local graphs according to Section 2.2.2.
    (d) For each image of trajectory 2 (called *the query*), the algorithm searches the best 5 HALOC loop closing potential candidates of trajectory 1. Each candidate, if any, is confirmed by means of Algorithm 1, and filtered out if the number of inliers is lower than the predefined threshold.
    (e) Accumulate the number of inter-session true loop closings.
    (f) When the number of accumulated inter-session loop closings is greater than a certain threshold, join both sessions in a single pose-based graph. That means transforming all members of the joined state vector in global poses and the corresponding graph nodes, associating to each node the corresponding image.
5. Run the Multi-robot SLAM procedure, according to Algorithm 2

(a) Obtain the rest of images from memory and add new nodes according to the successive odometry data of both sessions, as explained in Section 2.3.
(b) Search among all images of session 1 the best 5 candidates to close an inter-session loop with each new query of session 2, and filter out all those that do not present enough inliers after running Algorithm 1. HALOC is, obviously, the method used to find these candidates to close loops inter-sessions. Since each image will be associated with a node of the global graph, computing the transform between two candidates to close a loop and adding this transform between both nodes will be straightforward.
(c) Get the transform between pairs of images that constitute true positives (true loop closings).
(d) Add this transform to the graph as a new pose constrain, in the form of links between two nodes. The nodes will be those related with the images involved in the inter-session loop closing.
(e) Optimize the graph.
(f) Finish the process when all images from both sessions have been already used.

This simulation does not permit assessing anything related to execution times because, although the whole process works with real data, the Multi-robot configuration has been simulated, split into three different software packages of different natures. The purpose of these preliminary results is not giving an accurate set of quantitative and numerical results to assess the process in terms of execution time or trajectory accuracy. The aim of this section is presenting the implementation of a new approach and a set of preliminary results that provide: (a) a proof of viability and feasibility of the solution, (b) a proof of its utility and suitability to manage, in a single map, two sessions of two different robots that operate simultaneously, in a simple way, (c) a qualitative proof of concept and, (d) the source code and a dataset to be tested, open to further improvements.

Obtaining a ground truth trajectory underwater is a challenging task, unless one can install an infrastructure of acoustic beacons or Long Baseline (LBL) systems, which is costly and complicated to run and manage, and imposes spatial restrictions on the motion of the robots. In our experiments, there is no ground truth and no possibility to get it. The planned trajectories, in the case of those performed with the AUV, cannot be used as a ground truth either, since they differ substantially from the ones that the AUV ends up performing (which is usual in underwater robotics). In consequence, we cannot compare the trajectories estimated by the system with another one that serves as reference. In our case, robustness has been qualitatively validated by two means, (a) comparing the resulted global maps with the mosaics obtained with BIMOS [54] and, (b) comparing the direct transforms between images that close loops, obtained with Algorithm 1 with the transforms between the graph-nodes related to the same loop closing images. These two points have been already used and validated in previous pieces of work [46].

Figure 6a,b,d,e show, respectively, the trajectory of $S11$, $S12$, $S21$ and $S22$ estimated by the local SLAM procedure described in Section 2.2. Figure 6c,f show, respectively, the global graph obtained applying the Multi-robot SLAM procedure described in Section 2.3.

Figures 7 and 8 show four photo-mosaics corresponding to sequences $S11$, $S12$, $S21$ and $S22$. These photomosaics have been obtained using BIMOS [54], a mosaicing algorithm based on bags of binary words that already demonstrated a great performance in underwater environments [55,56].

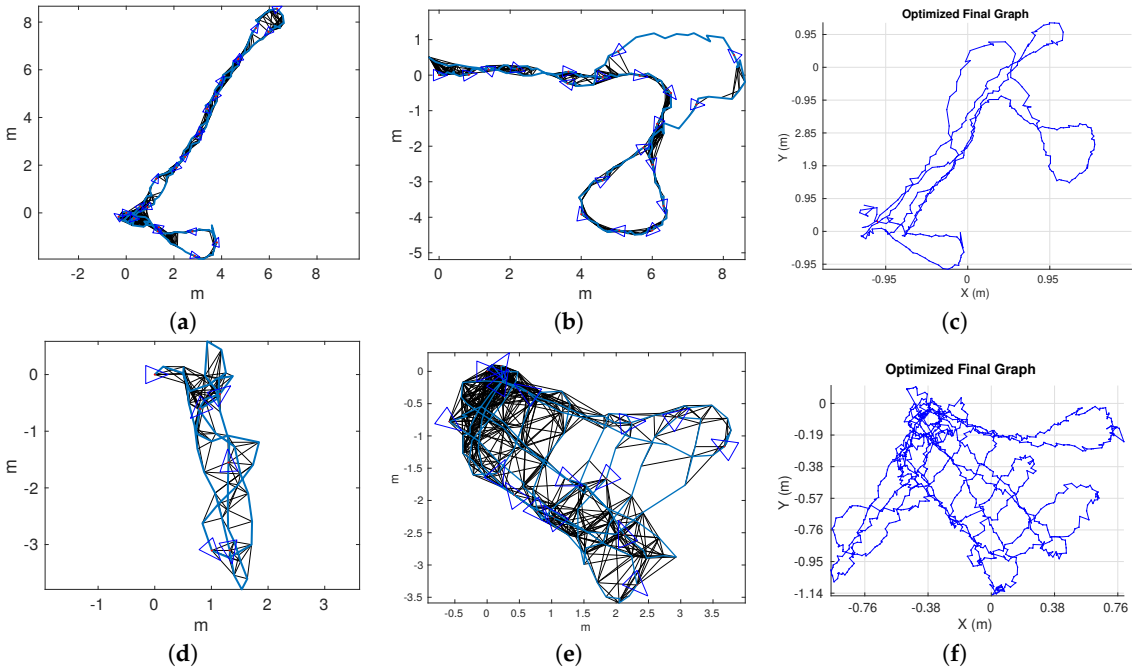
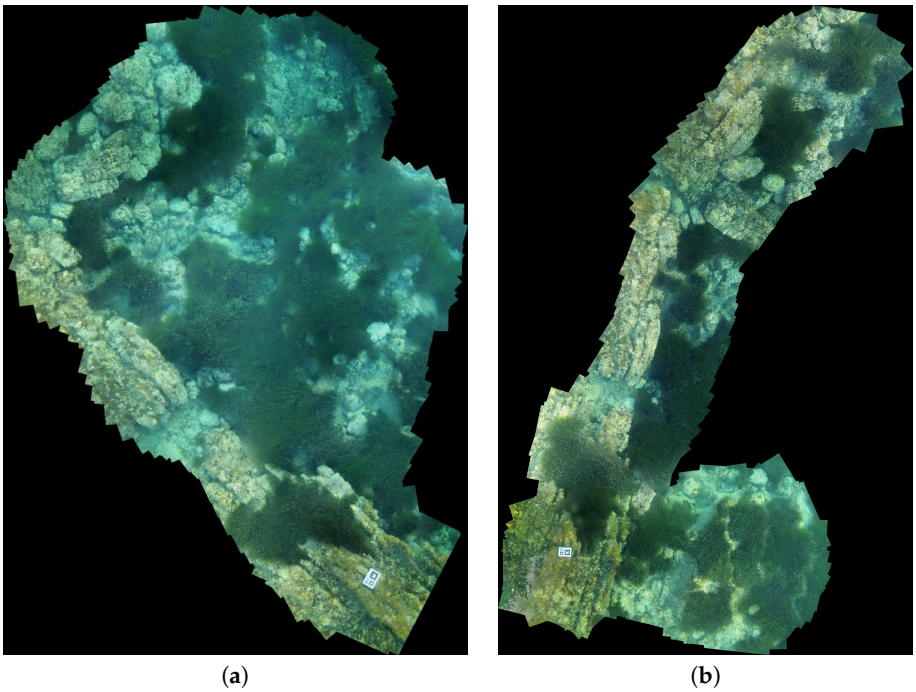


Figure 6. (**a**) Trajectory *S*11, (**b**) Trajectory *S*12, (**c**) Multi-robot final graph from *S*11 and *S*12. (**d**) Trajectory *S*21, (**e**) Trajectory *S*22, (**f**) Multi-robot final graph from *S*21 and *S*22.

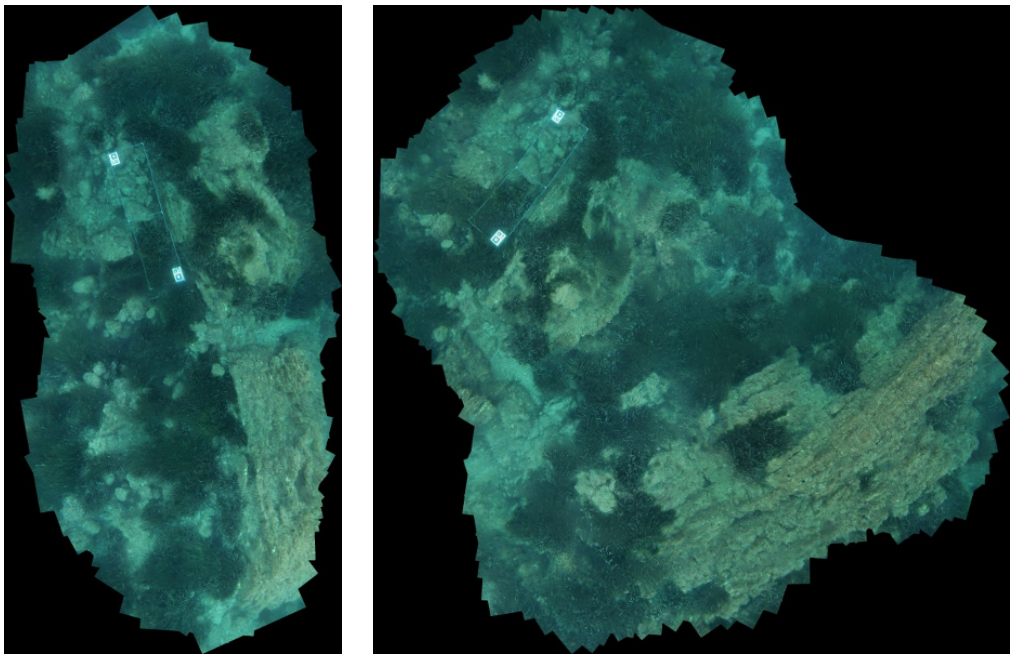


(**a**) (**b**)

Figure 7. (**a**) Photo-mosaic of *S*11. (**b**) Photo-mosaic of *S*12.

The resulting mosaics have associated an implicit trajectory which imposes the position of each image with respect to the origin of the mosaic system of coordinates. Due to the lack of any trajectory ground truth and the impossibility to get it, the mosaic is, to a certain extent, a qualitative reference to assess the quality of the resulting joined trajectories of Figure 6c,f, since BIMOS has already demonstrated its good performance in land and underwater. Notice how the mosaic of Figure 7b shows a montage very close to the SLAM trajectory of *S*11, and Figure 7a a mosaic fitting the SLAM

trajectory of $S12$. It can be assumed that the quality of the single and joined graphs obtained with BIMOS are similar to the quality of the mosaics.



(**a**) (**b**)

Figure 8. (**a**) Photo-mosaic of $S21$. (**b**) Photo-mosaic of $S22$.

If both mosaics are aligned by their left laterals, where the marker is located, something that fits very well with the global graph is obtained. The same applies to the mosaics of Figure 8. The structure containing the two markers is the point joining both trajectories. Then, the alignment of both figures by the markers gives a result that also fits perfectly with the global graph of Figure 6.

Figure 9a,b show the local SLAM trajectory of $S31$ and $S32$, and Figure 9c shows the global graph estimated after joining both trajectories and applying the Multi-robot graph SLAM approach.

An illustrative video of the whole process involving the three datasets can be seen in [57]. The video shows, at the beginning, some sequences grabbed underwater and used to test our approach. Afterwards, it shows the whole process for the three datasets exposed: (1) The local SLAM for both separated trajectories, (2) the moment when both sessions are joined and converted into a single global graph, and (3) how the graph continues growing in different directions, each one corresponding to each trajectory involved in the Multi-robot mission. As mentioned in previous sections, the joined graph is optimized every time a set of inter-session loop closings are confirmed.

As this is a pure visual-based SLAM approach and no other sensorial input is included in the multi-localization process, this is firstly computed in pixel units. To find the relation between pixels and metric units, the known real dimensions of the markers used to establish the starting and end points of each trajectory (see pictures (a) and (e) of Figure 4) were related with the pixel dimensions of the markers in the images. These relations resulted in coefficients ranging between 0.0015 and 0.0019, depending on each video sequence.

Another way to verify the consistency of the resulting optimized global map is comparing the transform between two images (called I1 and I2) that close an inter-session loop, when it is obtained by two different means: (1) using Algorithm 1, and (2) running the next operation: $\ominus P1 \oplus P2$, where $P1$ is the global pose associated with the graph node corresponding to I1, and $P2$ is the global pose associated with the graph node corresponding to I2. The idea is illustrated in Figure 10: $P1$ and $P2$ are the global poses associated with two loop closing nodes, T_{I2}^{I1} is the direct transform between P1 and P2 obtained with the RANSAC-based algorithm applied directly on I1 and I2. In principle, if the graph

is consistent, T_{I2}^{I1} has to be equivalent to the inverse of $P1$ composed with $P2$, which is the transform between $I1$ and $I2$, but obtained composing the global poses between the respective nodes of the loop closing images.

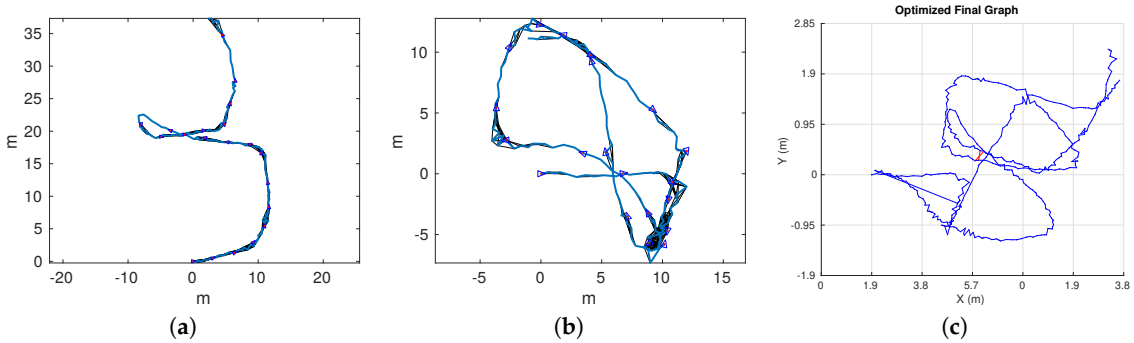


Figure 9. (**a**) Trajectory $S31$, (**b**) Trajectory $S32$, (**c**) Multi-robot final graph from $S31$ and $S32$.

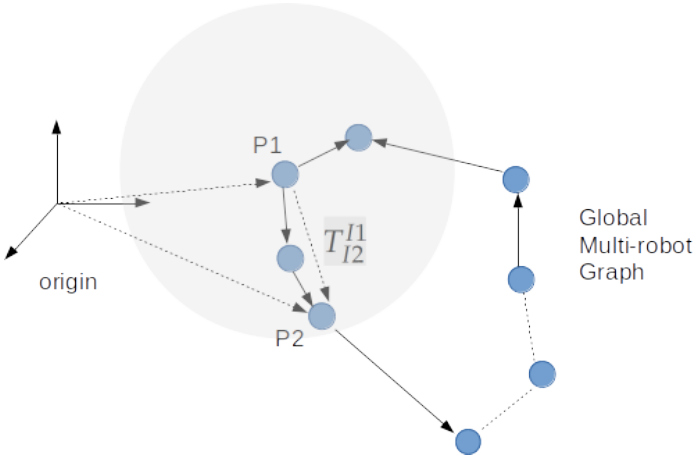


Figure 10. Transforms between nodes P1 and P2.

Tables 1 and 2 show some samples of quantitative results of intersession loop closings with the corresponding transforms calculated by both aforementioned ways. The first to forth columns show, respectively, the number of both images that close an inter-session loop and the graph nodes which each image is associated with. Column I1 contains images of the sessions $S11$ and $S21$, and column I2 contains images of sessions $S12$ and $S22$. Fifth and sixth columns indicate the 2D transform, in translation and rotation (x, y, θ), computed indirectly through the graph and directly using RANSAC (Algorithm 1). The units of these transforms are expressed in pixels and radians. The seventh column indicates the difference between both transforms, in module (meters) and orientation (radians). These samples indicate that: (a) for $S11$ and $S12$, the difference of transforms ranges between 1.8 cm and 0.15 mm in module and between 0.05 rad. (2.86°) and 0.0089 rad. (0.5°) in orientation, and (b) the difference for $S21$ and $S22$ ranges between 1.02 cm and 14.8 cm in module and between 0.0085 rad. (0.5°) and 0.21 rad. (12.03°) in orientation. These differences are totally acceptable, taking into account that there are errors inherent to the RANSAC transform estimation process due to the possible (and usual) presence of any inconsistent inlier, and differences (or errors) due to the successive graph optimizations, which also cause subsequent readjustments of all node poses.

Table 1. Comparison of Transforms between images of *S*11 and *S*12.

I1	I2	Node I1	Node I2	Graph Transform	Ransac Transform	dif:(mod., yaw)
56	280	115	113	[3.04;−4.05;0.1711]	[3.12;−3.99;0.18]	(0.00015 m, 0.0089 rad)
64	287	131	126	[12.10;41.17;0.0633]	[21.71;33.31;−0.03]	(0,018 m, 0.01 rad)
152	420	307	393	[37.09;31.95;−0.34]	[36.65;31.47;−0.29]	(0,00097 m, 0.05 rad)

Table 2. Comparison of Transforms between images of *S*21 and *S*22.

I1	I2	Node I1	Node I2	Graph Transform	Ransac Transform	dif:(mod., yaw)
335	1181	672	602	[30.43;59.40;2.42]	[−18.61;−1.69;2.61]	(0.148 m, 0.2 rad)
603	1190	941	621	[1.19;−30.22;−0.024]	[11.77;19.69;0.1959]	(0.097 m, 0.21 rad)
877	1211	1215	662	[−6.34;−92.66;−0.196]	[24.69;−50.403;−0.314]	(0.099 m, 0.12 rad)
871	1211	1209	662	$[-20.65;-84.08;-3.69 \times 10^{-4}]$	[14.75;−43.66;−0.086]	(0.0102 m, 0.085 rad)

3.3. Some Considerations of the Data Reduction

The length of the HALOC global descriptors used in the aforementioned tests is 384 floats. That is, a total of 1536 bytes per image, considering that in $C++$ a float needs 4 bytes for memory storage. All this, regardless the image resolution and the amount of SIFT features per image. That means that no matter how big is the image and how many visual features are being detected per image, that the length of the hash maintains invariable. Conversely, the size of a color image with a very reduced resolution of 320×240 pixels would be $320 \times 240 \times 3 = 230{,}400$ bytes. The save on memory space for data storage is clearly reduced when using the HALOC hash instead of the original images. The set of image features must be stored for every image, in any case, since they are needed in the later processes of loop closing confirmation. However, the computational cost of comparing two hashes to retrieve the best candidates for loop closing just calculating the L1-norm of two vectors is much lower than finding the best candidates with a brute-force recursive feature double-matching with RANSAC. From this point on, one can think of applying additional strategies to limit the communication between robots, complementing, for instance the solution proposed in [35], where the images are sent only among robots that view, simultaneously, a common point. In this case, we introduce an additional layer to compress the information to be exchanged, since instead of sending JPEG images, robots would send their respective hashes.

3.4. Sources Availability

The source code for the odometry computation has been developed in Matlab and it is available at [58]. The source code of the HALOC library is available at [59], for its $C++$ version, and in [60], for its version in Python.

The sources for the local SLAM, the Map Joining and the later Multi-Robot Graph SLAM have been developed also in Matlab, and they are available for the community at [61]. The pose-based graph management has been programmed using the Matlab library for localization and pose estimation especially addressed to mobile autonomous vehicles [62].

4. Discussion, Conclusions and Future Work

This paper presents a new approach to visual SLAM for Multi-robot configurations, based on joining, in a single pose-based graph, several trajectories of different robots which operate simultaneously in a common area of interests. The system finds loop closings between images of different robot trajectories by means of a hash-based methodology (HALOC), and uses them to add additional constraints to the global graph. As exposed in the text, the use of HALOC clearly guarantees an important reduction in storage space, amounts of data to be transferred and time dedicated for loop closing detection, especially in centralized multi-robot configurations. Using HALOC also assures a

proper graph optimization since it has already been proved underwater, showing excellent results in terms of success ratios in loop closing detection.

The strategy for map joining comes from [46], but adapted to a Multi-robot configuration, which differs from a multi-session case in some aspects. This strategy is simple, easy to replicate, effective, and, more importantly, as flexible as possible to modulate the moment for map joining depending on the mission conditions and convenience, which implies a trade off between the accuracy of the local maps before they are joined, or the need for joining as soon as possible all the trajectories to centralize the global multiple localization of all the robotic team in a single agent.

Preliminary experiments permitted to show how the application of the new approach for joining, online, multiple ongoing sessions was perfectly feasible, suggesting a certain consistency and reliability in the results, from a qualitative point of view.

Although, until now, we focused our efforts exclusively in the estimation of the camera pose and trajectory, this algorithm has been designed to be applied on board a vehicle. Therefore, one priority ongoing task is testing this algorithm in a team of real vehicles operating in the sea. To this end, a ROS [63] wrapper in C++ is currently being developed and tested.

We have now focused our efforts exclusively in the estimation of the camera pose and trajectory assuming that navigation and control are solved issues. In fact, most of the existing research on SLAM makes the same assumption. However, the continuous re-estimation of the vehicle poses thanks to the SLAM algorithm surely affects the control of the vehicles, because the control modules are fed with the poses and velocities. In addition, changes in control affect, in turn, the vehicle navigation. At the moment the SLAM algorithm is completely decoupled from the control module, but once it is installed on a vehicle, the vehicles velocity and pose provided by our SLAM modules, together with the mission goal points, will be input in the navigation and control subsystems. Another possible line of research that is also under consideration is to make the goal points also depend on SLAM in order to add exploration to the AUV capabilities.

Other future work includes:

(1) Extending the tests to additional environments with longer trajectories. (2) Extending the assessment of the approach by means of evaluating the performance of the SLAM pose corrections in the presence of additive Gaussian noise in the visual odometry, and all evaluation techniques employed in [46]. (3) Comparing with other Multi-robot software packages still not tested underwater, such as DSLAM.

Matlab sources are available in a public repository giving the chance to the scientific community of testing, replicating, and also improving them.

Author Contributions: Both authors contributed equally to this work, including the Conceptualization, theoretical methodology, the implementation of the software, the validation with the datasets obtained in the sea, writing the original draft, and the final supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by Ministry of Economy and Competitiveness under contract DPI2017-86372-C3-3-R (AEI,FEDER,UE).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Durrant-Whyte, H.; Bailey, T. Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms. *Robot. Autom. Mag.* **2006**, *2*, 99–110. [CrossRef]
2. Burguera, A.; González, Y.; Oliver, G. Underwater SLAM with Robocentric Trajectory Using a Mechanically Scanned Imaging Sonar. In Proceedings of the International Conference on Intelligent Robotis and Systems (IROS), San Francisco, CA, USA, 25–30 September 2011; pp. 3577–3582.
3. Ferri, G.; Djapic, V. Adaptive Mission Planning for Cooperative Autonomous Maritime Vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5586–5592.

4. Conte, G.; Scaradozzi, D.; Mannocchi, D.; Raspa, P.; Panebianco, L.; Screpanti, L. Development and Experimental Tests of a ROS Multi-agent Structure for Autonomous Surface Vehicles. *J. Intell. Robot. Syst.* **2018**, *92*, 705–718. [CrossRef]
5. Davison, A.; Calway, A.; Mayol, W. Visual SLAM. *IEEE Trans. Robot.* **2007**, *24*, 1088–1093. [CrossRef]
6. McDonald, J.; Kaess, M.; Cadena, C.; Neira, J.; Leonard, J.J. Real-time 6-DOF Multi-session Visual SLAM over Large-Scale Environments. *Robot. Auton. Syst.* **2013**, *61*, 1144–1158. [CrossRef]
7. Abdulgalil, M.; Nasr, M.; Elalfy, M.; Khamis, A.; Karray, F. Multi-Robot SLAM: An Overview and Quantitative Evaluation of MRGS ROS Framework for MR-SLAM. In *International Conference on Robot Intelligence Technology and Applications*; Springer: Cham, Switzerland, 2019; Volume 751, pp. 165–183.
8. Mahdoui, N.; Frémont, V.; Natalizio, E. Communicating Multi-UAV System for Cooperative SLAM-based Exploration. *J. Intell. Robot. Syst.* **2019**, *46*, 1–19. [CrossRef]
9. Shkurti, F.; Chang, W.; Henderson, P.; Islam, M.; Gamboa, J.; Li, J.; Manderson, T.; Xu, A.; Dudek, G.; Sattar, J. Underwater Multi-robot Convoying Using Visual Tracking by Detection. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 4189–4196.
10. Allotta, B.; Costanzi, R.; Ridolfi, A.; Colombo, C.; Bellavia, F.; Fanfani, M.; Pazzaglia, F.; Salvetti, O.; Moroni, D.; Pascali, M.A.; et al. The ARROWS Project: Adapting and Developing Robotics Technologies for Underwater Archaeology. *IFAC-PapersOnLine* **2015**, *48*, 194–199. [CrossRef]
11. Howard, A. Multi-Robot Simultaneous Localization and Mapping Using Particle Filters. *Int. J. Robot. Res.* **2006**, *25*, 1243–1256. [CrossRef]
12. Kaess, M.; Ranganathan, A.; Dellaert, F. iSAM: Incremental Smoothing and Mapping. *IEEE Trans. Robot. (TRO)* **2008**, *24*, 1365–1378. [CrossRef]
13. Kim, B.; Kaess, M.; Fletcher, L.; Leonard, J.; Bachrach, A.; Roy, N.; Teller, S. Multiple Relative Pose Graphs for Robust Coopetative Mapping. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010..
14. Schuster, M.J.; Schmid, K.; Brand, C.; Beetz, M. Distributed stereo vision-based 6D localization and mapping for multi-robot teams. *J. Field Robot.* **2019**, *36*, 305–332. [CrossRef]
15. Saeedi, S.; Trentini, M.; Seto, M.; Li, H. Multiple-Robot Simultaneous Localization and Mapping: A Review. *J. Field Robot.* **2016**, *33*, 3–46. [CrossRef]
16. Angeli, A.; Doncieux, S.; Meyer, J.; Filliat, D. Real-Time Visual Loop-Closure Detection. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA, 19–23 May 2008; pp. 1842 –1847.
17. Kim, A.; Eustice, R.M. Real-Time Visual SLAM for Autonomous Underwater Hull Inspection Using Visual Saliency. *IEEE Trans. Robot.* **2013**, *29*, 719–733. [CrossRef]
18. Monga, V.; Evans, B.L. Perceptual Image Hashing Via Feature Points: Performance Evaluation and Tradeoffs. *IEEE Trans. Image Process.* **2006**, *15*, 3452–3465. [CrossRef]
19. Liu, Y.; Zhang, H. Visual Loop Closure Detection with a Compact Image Descriptor. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 7–12 October 2012; pp. 1051–1056.
20. Arandjelovic, R.; Zisserman, A. All About VLAD. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013; pp. 1578–1585.
21. Bonin-Font, F.; Negre, P.; Burguera, A.; Oliver, G. LSH for Loop Closing Detection in Underwater Visual SLAM. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–4.
22. Negre Carrasco, P.L.; Bonin-Font, F.; Oliver-Codina, G. Global Image Signature for Visual Loop-closure Detection. *Autonom. Robots* **2016**, *40*, 1403–1417. [CrossRef]
23. Jain, U.; Namboodiri, V.; Pandey, G. Compact Environment-Invariant Codes for Robust Visual Place Recognition. In Proceedings of the 14th Conference on Computer and Robot Vision (CRV), Edmonton, AB, Canada, 17–19 May 2017; pp. 40–47.
24. Glover, A.; Maddern, W.; Warren, M.; Reid, S.; Milford, M.; Wyeth, G. Openfabmap: An Open Source Toolbox for Appearance-based Loop Closure Detection. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN, USA, 14–18 May 2011.

25. Cieslewski, T.; Choudhary, S.; Scaramuzza, D. Data-Efficient Decentralized Visual SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2466–2473.
26. Arandjelović, R.; Gronat, P.; Torii, A.; Pajdla, T.; Sivic, J. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 1437–1451. [CrossRef] [PubMed]
27. Mur-Artal, R.; Montiel, J.M.; Tardos, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
28. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An Efficient Alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
29. Zhang, P.; Wang, H.; Ding, B.; Shang, S. Cloud-Based Framework for Scalable and Real-Time Multi-Robot SLAM. In Proceedings of the 2018 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 2–7 July 2018; pp. 147–154.
30. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. The KITTI Vision Benchmark Suite. Available online: http://www.cvlibs.net/datasets/kitti/ (accessed on 7 May 2020).
31. Karrer, M.; Schmuck, P.; Chli, M. CVI-SLAM—Collaborative Visual-Inertial SLAM. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2762–2769. [CrossRef]
32. Elibol, A.; Kim, J.; Gracias, N.; García, R. Efficient Image Mosaicing for Multi-robot Visual Underwater Mapping. *Pattern Recognit. Lett.* **2014**, *46*, 20–26. [CrossRef]
33. Young-Hoo, K.; Steven, L. Applicability of Localized-calibration Methods in Underwater Motion Analysis. In Proceedings of the XVIII International Symposium on Biomechanics in Sports; 2000. Available online: https://ojs.ub.uni-konstanz.de/cpa/article/view/2530 (accessed on 7 May 2020).
34. Hans-Gerd, M. *New Developments in Multimedia Photogrammetry*; Wichmann Verlag: Karlsruhe, Germany, 1995.
35. Pfingsthorn, M.; Birk, A.; Bülow, H. An Efficient Strategy for Data Exchange in Multi-robot Mapping Under Underwater Communication Constraints. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4886–4893.
36. Paull, L.; Huang, G.; Seto, M.; Leonard, J. Communication-constrained Multi-AUV Cooperative SLAM. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 509–516.
37. Jaume I University of Castellón, IRSLab; University of Girona, CIRS; University of Balearic Islands. TWIN roBOTs for Cooperative Underwater Intervention Missions. 2019. Available online: http://www.irs.uji.es/twinbot/twinbot.html (accessed on 7 May 2020).
38. Burguera, A.; Bonin-Font, F.; Oliver, G. Trajectory-based visual localization in underwater surveying missions. *Sensors* **2015**, *15*, 1708–1735. [CrossRef]
39. Zhao, B.; Hu, T.; Zhang, D.; Shen, L.; Ma, Z.; Kong, W. 2D Monocular Visual Odometry Using Mobile-phone Sensors. In Proceedings of the 34th Chinese Control Conference (CCC), Hangzhou, China, 28–30 July 2015; pp. 5919–5924.
40. SRV Group. Systems, Robotics and Vision Group, University of the Balearic Islands. Available online: http://srv.uib.es/projects/ (accessed on 7 May 2020).
41. Negre Carrasco, P.L.; Bonin-Font, F.; Oliver, G. Cluster-based Loop Closing Detection for Underwater SLAM in Feature-poor Regions. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 2589–2595. [CrossRef]
42. Peralta, G.; Bonin-Font, F.; Caiti, A. Real-time Hash-based Loop Closure Detection in Underwater Multi-Session Visual SLAM. In Proceedings of the Ocean 2019, Marseille, France, 17–20 June 2019.
43. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Burgard, K.K.W. G2o: A General Framework for Graph Optimization. *ICRA. IEEE* **2011**, 3607–3613.
44. Newman, P.; Leonard, J.; Rikoski, R. Towards Constant-Time SLAM on an Autonomous Underwater Vehicle Using Synthetic Aperture Sonar. In Proceedings of the Eleventh International Symposium on Robotics Research, Sienna, Italy, 19–22 October 2003; Volume 15, pp. 409–420.
45. Muller, M.; Surmann, H.; Pervolz, K.; May, S. The Accuracy of 6D SLAM Using the AIS 3D Laser Scanner. In Proceedings of the 2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Heidelberg, Germany, 3–6 September 2006; pp. 389–394.

46. Burguera, A.B.; Bonin-Font, F. A Trajectory-Based Approach to Multi-Session Underwater Visual SLAM Using Global Image Signatures. *J. Mar. Sci. Eng.* **2019**, *7*, 278. [CrossRef]
47. Smith, R.; Cheeseman, P.; Self, M. A Stochastic Map for Uncertain Spatial Relationships. In Proceedings of the 4th International Symposium on Robotics Research, Siena, Italy, 19–22 October 1987; pp. 467–474.
48. Kanzow, C.; Yamashita, N.; Fukushima, M. Levenberg-Marquardt Methods for Constrained Nonlinear Equations with Strong Local Convergence Properties. *J. Comput. Appl. Math.* **2002**, *172*, 375–397. [CrossRef]
49. Gavin, H. The Levenberg-Marquardt Method for Nonlinear Least Squares Curve-Fitting Problems. 2019. Available online: http://people.duke.edu/~hpgavin/ce281/lm.pdf (accessed on 10 May 2002).
50. Smith, R.; Self, M.; Cheeseman, P. A Stochastic Map for Uncertain Spatial Relationships. *Comput. Sci.* **1988**. Available online: https://www.semanticscholar.org/paper/A-stochastic-map-for-uncertain-spatial-Smith-Self/76a6c5352a0fbc3fec5395f1501b58bd6566d214 (accessed on 5 May 2002)
51. Grisetti, G.; Kuemmerle, R.; Stachniss, C.; Burgard, W. A Tutorial on Graph-Based SLAM. *Intell. Transp. Syst. Mag. IEEE* **2010**, *2*, 31–43. [CrossRef]
52. Carreras, M.; Hernandez, J.; Vidal, E.; Palomeras, N.; Ribas, D.; Ridao, P. Sparus II AUV—A Hovering Vehicle for Seabed Inspection. *IEEE J. Ocean. Eng.* **2018**, *43*, 344–355. [CrossRef]
53. Guerrero, E.; Bonin-Font, F.; Negre, P.L.; Massot, M.; Oliver, G. USBL Integration and Assessment in a Multisensor Navigation Approach for AUVs. *IFAC-PapersOnLine* **2017**, *50*, 7905–7910.
54. García, E.; Ortiz, A.; Bonnín, F.; Company, J.P. Fast Image Mosaicing using Incremental Bags of Binary Words. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016.
55. Bonin-Font, F.; Massot, M.; Codina, G.O. Towards Visual Detection, Mapping and Quantification of Posidonia Oceanica using a Lightweight AUV. *IFAC-PapersOnLine* **2016**, 49, 500–505. [CrossRef]
56. García, E.; Ortiz, A. Hierarchical Place Recognition for Topological Mapping. *IEEE Trans. Robot.* **2017**, *33*, 1061–1074. [CrossRef]
57. Bonin-Font, F. Multi-Robot Visual Graph SLAM–An Ilustrative Video. Available online: https://www.youtube.com/watch?v=c7faAm2HIpc (accessed on 16 May 2020).
58. Burguera, A. Visual Odometry Sources. 2015. Available online: https://github.com/aburguera/VISUAL_ODOMETRY_2D (accessed on 16 May 2020).
59. Negre Carrasco, P.L. ROS C++ Library for Hash-Based Loop Closure (HALOC). Available online: https://github.com/srv/libhaloc (accessed on 2 June 2015).
60. Bonin-Font, F. Python Library for Hash-Based Loop Closure (HALOC). Available online: https://github.com/srv/HALOC-Python (accessed on 27 March 2019).
61. Bonin-Font, F.; Burguera, A. Sources of Multi-Robot Visual Graph SLAM with a Sample Dataset. Available online: https://github.com/srv/Multi-Robot-Visual-Graph-SLAM (accessed on 7 May 2020).
62. Matlab. Localization and Pose Estimation: Pose Graph. Available online: https://es.mathworks.com/help/nav/ref/posegraph.html (accessed on 1 January 2019).
63. Quigley, M.; Conley, K.; Gerkey, B.P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An Open-Source Robot Operating System. In *ICRA Workshop on Open Source Software*; 2009. Available online: https://blog.acolyer.org/2015/11/02/ros-an-open-source-robot-operating-system/ (accessed on 1 May 2020).