Collaborative Visual SLAM Framework for a Multi-Robot System

Nived Chebrolu[1], David Marquez-Gamez[2] and Philippe Martinet[1]

***Abstract*— This paper presents a framework for collaborative visual SLAM using monocular cameras for a team of mobile robots. The robots perform SLAM individually using their on-board processors thereby estimating the seven degrees of freedom (including scale) for the motion of the camera and creating a map of the environment as a pose-graph of keyframes. Each robot communicates to a central server by sending local keyframe information. The central server merges them when a visual overlap is detected in the scene and creates a global map. In the background, the global map is continuously optimized using bundle adjustment techniques and the updated pose information is communicated back as feedback to the individual robots. We present some preliminary experimental results towards testing the framework with two mobile robots in an indoor environment.**

I. INTRODUCTION

Autonomous robots are increasingly being used for more and more complex problems each day such as exploration of large unstructured environments etc. In order to deal with these complex scenarios, a multi-robot system consisting of a team of robots (such as mobile robots, aerial vehicles etc) which are equipped with perception sensors is necessary. A multi-robot system extends the capability of a single robot by merging measurements from several team members and providing each robot with information beyond the range of their individual sensors. This facilitates more efficient usage of resources and achieves tasks which are not feasible for a single robot system.

Moreover the use of a multi-robot system allows parallel execution of tasks and also some degree of redundancy increasing both the efficiency and the robustness of the system. Consider a scenario of employing a multi-robot team for the purpose of mapping of a large unknown environment. The task can be divided among all the team members which can collaboratively build a global map reducing the overall execution time. This collective information including the relative positions of the robots can be used for making exploration strategies, path-planning and other higher level decision. However, in general the advantages of a collaborative system come at the cost of increased computations and communication load among robots.

In this work, we deal with a team of mobile robots each equipped with a monocular camera to perform SLAM. Using a monocular camera gives the advantage for the system to be used both for indoor/outdoor applications and also for scenes with large variations in depth. Typically these conditions impose severe restrictions on other vision sensors such as RGB-D cameras and stereo pairs. However, this makes running the visual SLAM process more challenging as the scale needs to be continuously estimated since no depth information is directly available.

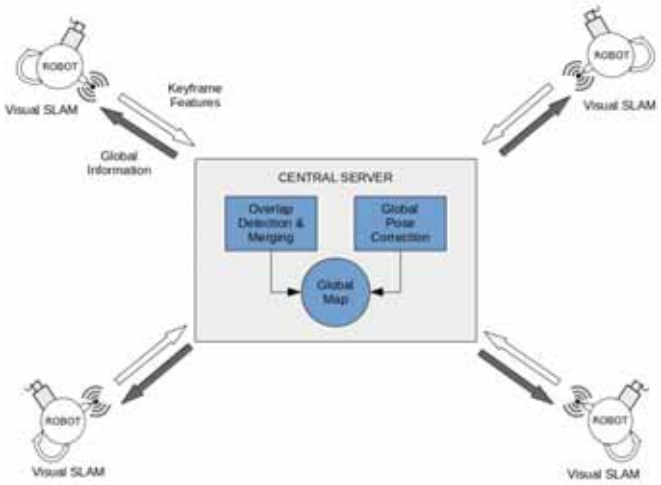

Fig. 1. System Architecture: Collaborative Visual SLAM

In this paper, we propose a framework for collaborative visual SLAM (as shown in Fig. 1) where:

- Each individual robot performs monocular visual SLAM and sends local keyframe information to a central server.
- The central server merges this information to create a global map and performs a pose correction using bundle adjustment.
- The updated pose is communicated back to individual robots as feedback thereby improving the local map and the localization estimate of the individual robots.

The organization of the paper is as following. The next section presents the related works. Then a general overview on our system is given in section III where every function is shortly described. Section IV presents the methodology where we detail each function. Finally experimental results are presented and analyzed in the section V.

II. RELATED WORK

Traditionally, SLAM has been performed using range sensors like laser scanners, sonars or using stereo vision [1]. More recently, monocular cameras (bearing only sensors) are also being used as the primary vision sensor like in [2],[3],[4]. In the multi-robot context, this problem has been studied under the banner of multi-camera structure from motion (SfM) [5] or multi-camera SLAM [6].

In [7], the authors analyse the improvement in localization quality of cooperative multi-robot localization over single robot localization. [8] demonstrates that by incorporating relative bearing information of the cameras, the overall accuracy of the localization is strongly improved. In this

[1] Nived Chebrolu and Philippe Martinet are with Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN), Ecole Centrale de Nantes, France.nived.chebrolu@eleves.ec-nantes.fr, philippe.martinet@irccyn.ec-nantes.fr

[2] David Marquez-Gamez is with Robotics Division, IRT Jules Verne, Nantes, France. david.marquez@irt-jules-verne.fr

approach, an Extended Kalman Filter (EKF) is adopted to maintain the state containing configurations of all robots. In [9], the authors propose an interesting idea where two UAVs with monocular cameras act as a flexible stereo rig. With additional input from IMUs the relative poses are recoverd with absolute scale starting from an unknown initial configuration.

[10] deals with large-scale collaborative SLAM in an outdoor environment involving heterogeneous robots such as UAVs and ground mobile robots equipped with stereo cameras. It employs a global graph which maintains the relative relationships between a series of submaps built by each robot. The links between each submaps are created by events like robot rendezvous, scene feature matches or absolute localization information provided by GPS etc. These constraints allow the correction of the position estimates of submaps with respect to each other. [11] uses a multi-camera system to estimate the trajectory of moving objects in the scene along with building a 3D map of static objects. However, the system requires the image streams from all the cameras to be synchronized making it impractical to be used for real-time applications.

Several decentralized solutions have been proposed where data fusion is performed using only robots which are in direct communication range of each other. [12] proposes a method to efficiently distribute map information across a team of robots which is robust to node failures and changes in network topology. The proposed scheme consists of a local optimization module which executes single robot SLAM, a communication module which propagates the local graphs to other robots and a neighbourhood graph optimization module which combines all the local graphs into maps describing the neighbourhood of a robot. On the other hand, recently many centralized cloud based architectures for collaborative SLAM have been designed where the data intensive tasks can be mitigated to a powerful back-end cluster system [13], [14], [15]. This allows the use of small and energy efficient on-board processor to be placed on the robots while offloading major computations to the cloud.

In [16], the authors propose a centralized framework for a group of MAVs equipped with monocular cameras. Each MAV performs visual odometry on its on-board processor and sends keyframe information to a ground server where it is merged to realize a global map. In this paper, we present a similar framework however each robot is capable of performing complete SLAM individually using full image information instead of using only features. In addition, a feedback mechanism is put in place which corrects the local estimates continuously. Also the framework allows the robots to join asynchronously and no prior relative information is required.

III. SYSTEM OVERVIEW

Figure 2 illustrates the overall scheme of our collaborative SLAM system. Each mobile robot equipped with a monocular camera performs visual SLAM using its on-board computer. This provides each robot with an estimate of its pose and a 3D map of the environment in their respective co-ordinate frames. In our approach, we use a monocular SLAM algorithm based on direct image alignment which is able to estimate the seven DoF's including the scale of the scene.

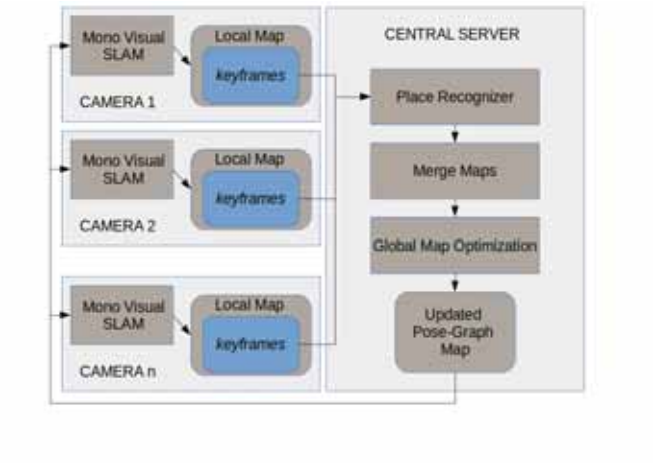

Fig. 2. Overall scheme of our collaborative SLAM system

The map of the environment is stored as a pose-graph of keyframes each consisting of a semi-dense depth map of the corresponding view. This function is based on LSD SLAM [3].

Each robot sends its keyframe information including its pose in the local co-ordinate frame to the central server. Here the *place recognizer* function constantly monitors all the keyframes to detect overlapping scenes from different robots. The overlap detection is performed in the appearance space by extracting visual features from each keyframe and comparing them in a fast manner using Bag of Words (BoW) technique [17].

Once an overlap is detected between two cameras, the map merging sequence is initiated. It involves computing an initial transformation estimate between the matched keyframes by using a RANSAC version of the traditional Horn's algorithm [18]. This estimate is used as a starting point to run an optimization algorithm which estimates the similarity transformation between the two matched keyframes. Finally, this estimate is refined by performing an iterative closest point algorithm as described in [19].

After computation of this transformation, the two corresponding maps are merged into a global map and a new constraint is added between the two matched keyframes.In parallel, a bundle adjustment procedure is run over the global graph and the updated poses of the keyframe graph are communicated back to the individual robots as feedback. This information is in turn used by each robot to improve its localization estimate and the local map.

IV. METHODOLOGY

In this section we detail each individual function in Fig. 2.

A. Visual SLAM

Each mobile robot performs an on-board monocular SLAM process which is able in real time to estimate its pose and create the map of the environment as a pose-graph of keyframes . The problem of scale drift is addressed by implicitly including it as a parameter in the overall optimization procedure. This function is based on LSD-SLAM [3]. The overall method consists of the following main components:

1) Tracking: The camera pose $\xi \in se(3)$ is estimated with respect to the current keyframe K_i which consists of the image (I_i), the depth map (D_i) and the depth map variance V_i. For each new image I_j, the relative pose $\xi_{ji} \in se(3)$ is computed by minimizing the photometric error:

$$\xi_{ji} = \underset{\xi}{\operatorname{argmin}} \sum_{p} \left\| r_p^2(p, \xi_{ji}) \right\| \tag{1}$$

where the photometric residual $r_p(p, \xi_{ji}) = I_i(p) - I_j(\omega(p, D_i(p), \xi_{ji}))$ and ω is a warping function which computes the location of a pixel from the first image in the second image given the relative transformation ξ_{ji}. Note that in the actual implementation, a variance normalized residual is minimized thereby implicitly including depth accuracy in the computation of ξ_{ji}. The optimization problem can be posed as a weighted least squares problem [20] which can be solved using the Gauss-Newton minimization method [21].

2) Depth Map Estimation: A semi-dense inverse depth map is continuously estimated for each new frame. The depth map is computed by making several stereo comparisons of varying baseline over consecutive frames of the input video. The variable base line allows for accurate estimation of both near and far regions of the image. The method maintains a probabilistic depth hypothesis for each pixel modelled by a gaussian distribution which is continuously refined using an filtering approach described in [22]. Finally when the camera moves far from the current keyframe, a new keyframe is created and its depth map is initialized by projecting points from the previous keyframe on it.

3) Map Management and Optimization: The frame to frame alignment method previously introduced in IV-A.1 inherently accumulates drift over time due to small errors in each estimate arising from sensor noise and other model inaccuracies.

To deal with this problem, the SLAM system maintains the map as a graph where each vertex is the pose of the keyframe and each edge represents the relative transformation between the corresponding keyframes. Each time a new keyframe is added to the map, new edges are created and finally when previously visited regions of the scene are encountered, additional edges (loop closures) are added which help in reducing the accumulated drift.

However in the case of monocular SLAM, the scale of the scene cannot be observed directly which over a long trajectory leads to a drift causing major errors in tracking. To take care of the scale parameter, the overall pose graph is constructed in a manner such that the mean inverse of each keyframe is one and instead the edges are represented as transformation $\xi_{ji} \in sim(3)$. This allows the integration of the scale parameter directly in the optimization problem. So, the scaled transformation between the keyframes is estimated by minimizing the error function:

$$E(\xi_{ji}) = \sum_{p} \left\| r_p^2(p, \xi_{ji}) + r_d^2(p, \xi_{ji}) \right\| \tag{2}$$

where the depth residual is $r_d(p, \xi_{ji}) = [p']_3 - D_j([p']_{(1,2)})$ and $p' = \omega(p, D_i(p), \xi_{ji})$

Finally, the overall map consisting of keyframe poses as vertices and $sim(3)$ constraints as edges, is continuously optimized in parallel using a general graph optimization framework like g2o [21]. This optimization over the graph reduces the drift both in scale and pose estimates.

B. Place Recognizer

This module runs continuously on the central server and is responsible to find scene overlap between different robots. Since the relative position of each robot is not known in the global coordinate frame at the beginning, the overlap is detected using the appearance space information only.

For every new keyframe image, visual features (e.g. SURF [23]) are computed which are view-point invariant. These features are then quantized with respect to a vocabulary and the resulting *visual words* description is stored. This *bag of words* (BoW) technique allows the scene to be represented as a collection of words which facilitates fast comparisons of feature descriptors.

We use the FAB-MAP method [24] to detect a scene overlap. This algorithm takes as input the BoW description of each image, compares it against all previously seen images. It gives as output the probability with which the current image matches any of the previously seen images. Moreover, it also computes the probability of the current image being a new one. These probabilities are calculated by solving a recursive Bayes estimation problem:

$$p(L_i|Z_k) = \frac{p(Z_k|L_i, Z_{k-1}) p(L_i|Z_{k-1})}{p(Z_k|Z_{k-1})} \tag{3}$$

where L_i is a scene (location) in the world, Z_k is an observation (visual words) at time k. In equation (3), $p(L_i|Z_{k-1})$ is the prior belief of our location, $p(Z_k|L_i, Z_{k-1})$ is the observation likelihood and $p(Z_k|Z_{k-1})$ is a normalizing term. The exact evaluation of these terms can be found in [24].

Traditionally, the FAB-MAP technique has been used to find loop closures over long trajectories. Instead in our application, we use it to find if a place has been visited by other robots in the team and in-effect creating a *virtual loop closure*. It is termed *virtual* since the loop closure is obtained as a result of the same place being visited by two different robots (as opposed to the same place being visited by the same robot). Finally, in order to avoid spurious matches, we only proceed for map merging if FAB-MAP reports a match over three consecutive images.

C. Map Merge

When the place recognizer module detects a scene overlap between robots and indicates a match point, the map merging procedure is initiated. The transformation between matching frames is done in three steps followed by an update to the global map.

1) Initial Transformation Estimate Using Horn's Method: For each keyframe image arriving at the central server, SURF features are computed and stored. Note that the same features were also used to compute a BoW representation required as an input to FAB-MAP.

The Horn's method describes a closed form solution using unit quaternions to compute the scaled transformation given three 3D point correspondences between two point clouds [18]. From the matching keyframe candidates proposed by FAB-MAP, 2D feature correspondences are extracted. Later the depth of each 2D feature is computed by taking an

average over the keyframe depth weighted by their variances in the descriptor neighbourhood. It should be noted that descriptors often end up in regions of discontinuities (such as corners or edges) and the averaging step may result in bad depth estimates. Finally, in order to deal with bad matches between features, we implement a robust RANSAC based version of the Horn's algorithm.

2) Refining Estimate Using $Sim3$ Tracker: As a second step, we use the tracking method based on minimizing the cost function as described in equation 2 to find an improved estimate for the scaled transformation. The estimate provided by Horn's method is used as a starting point for the tracker.

3) Correction using ICP: A final correction is made using the iterative closest point (ICP) algorithm, a technique from point cloud registration literature, which tries to find a transformation that minimizes the distance between a set of corresponding points in two clouds. We use an augmented version of ICP which also includes surface normal and tangent information to improve the estimate as described in [19]. Note that both the $Sim3$ tracker and the ICP procedure use a gradient approach to find the solution. In theory both these methods could be used in any order. However, the ICP procedure was found to be most accurate starting from a better estimation of the scale factor. Therefore, it was decided to perform the ICP step after the tracking step.

4) Global Map Update: Once the transformation is computed, new similarity constraints are added between the matching keyframes. The corresponding local maps are transformed into the global coordinate frame considering one of the two as reference (if its the first map merge) or using the existing reference otherwise. After the new constraints have been added a bundle adjustment step is performed over the merged graph.

D. Overall Feedback System

Each time different robots visit the same place in the environment, new constraints are created in the global graph. While the mobile robots move in the environment, they may cross each others path multiple times resulting in *virtual loop closures*. These loop closure constraints help in reducing the overall drift.

Finally, the central server communicates the updated pose graph to individual robots which can then use this information to update their localization estimate and the local maps.

This overall feedback mechanism facilitates the extension of sensing capability of an individual robot beyond the direct reach of their respective on-board sensors. In a sense, each robot in the team is able to "look" beyond what they can directly see and thus taking advantage of the collaborative system.

V. EXPERIMENTAL RESULTS

In this section we present preliminary results with the aim of validating the concepts presented before. The experiments presented are not intended to be particularly challenging examples, they are simply used to take the reader through the functionality of the system.

The experiments were performed using two *Turtlebots*, each equipped with a *uEye* monocular camera attached with a wide-angle lens ($\sim$130°Field of View) and a Core 2 Duo laptop. The images are captured at 30 Hz with a resolution of 640 x 480 pixels. The experiments were conducted in an industrial-like indoor environment approximately 20m x 20m. The two robots start exploring the environment asynchronously. Moreover, their starting positions are not known to the central server. The robots traverse through regions with large variations in scene scales. The depth of theses scenes range from 1m to 15m. Finally, the communication between the robots and the central server is carried through the standard Wi-Fi protocol.

Figures 3 and 4 shows results from the monocular SLAM process running on the local computers of the two robots $R1$ and $R2$ respectively. The three columns in these figures show the images captured by the camera, the trajectory of the robot and the corresponding map built at three different instants (corresponding to the three rows). The trajectory of the robot $R1$ is illustrated in red and that of robot $R2$ in blue. Each pyramid in the map represents a keyframe location and the lines joining these keyframes represent the constraints. The green pyramid depicts the current keyframe for both the cameras.

At instant 1, we see that robot $R1$ has completed a small square loop. During this trajectory, it makes some loop closures as well. Later robot $R2$ starts exploring some other part of the environment. At instant 2, the robot $R2$ completes a loop closure. As a result, we see that the overall trajectory of robot $R2$ has been optimized and the scale factor is corrected as well. At instant 3, robot $R1$ enters a region previously visited by robot $R2$. At this time, the visual place recognition system triggers a merge between the two maps. The relative transformation between these two views is computed in three stages as described in section IV.

In this example, robot $R1$'s origin is considered as the reference coordinate frame. All the keyframes of robot $R2$ are transformed with respect to this origin. The global trajectory of the two robots and the joint map is shown in figure 5 . Finally, after searching for additional constraints and optimizing the global map using bundle adjustment, the updated keyframe poses are sent back to the two robots. This updated keyframe pose is then used by robots $R1$ and $R2$ to correct the localization estimate and the local map.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a framework for collaborative visual SLAM for a team of mobile robots using a centralized approach. Each robot is able to individually perform monocular SLAM using its camera and on-board computer. The central server continuously receives local keyframe information from individual robots over Wi-Fi. Keyframes from all the robots are merged at this server and an optimization procedure is followed which minimizes the overall pose and mapping error. The updated pose information is sent back to the individual robots incorporating a feedback mechanism. No prior information regarding the relative position of the robots or the initial configuration is required. The system allows the robots to join and leave the team asynchronously.

The framework can be extended to work with different type of cameras by including the corresponding projection functions. In addition to the centralized framework, it would be interesting to add robot-to-robot communication. In this case, the robots can also exchange pose and map information with each other when they are in direct communication range.

Fig. 3. Monocular SLAM Process on robot $R1$ at three different instants. Left: Image captured by the camera. Middle:Trajectory built by the robot. Right: Map built by the robot.

REFERENCES

[1] S. Thrun *et al.*, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, pp. 1–35.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *CoRR*, vol. abs/1502.00956, 2015.

[3] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[5] M. Kaess and F. Dellaert, "Probabilistic structure matching for visual slam with a multi-camera rig," *Comput. Vis. Image Underst.*, vol. 114, no. 2, pp. 286–296.

[6] J. Sola, A. Monin, and M. Devy, "Bicamslam : Two times mono is more than stereo," in *IEEE Int. Conf. on Robotics Automation*, 2007.

[7] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization." *Auton. Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[8] A. Martinelli, F. Pont, and R. Siegwart, "Multi-robot localization using relative observations," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, April 2005, pp. 2797–2802.

[9] M. W. Achtelik, S. Weiss, M. Chli, F. Dellaert, and R. Siegwart, "Collaborative stereo," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 2242–2248.

[10] T. A. Vidal-Calleja, C. Berger, J. Sol, and S. Lacroix, "Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 654 – 674, 2011.

[11] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 2, pp. 354–366, Feb 2013.

[12] A. Cunningham, B. Paluri, and F. Dellaert, "Ddf-sam: Fully distributed slam using constrained factor graphs." in *IROS*. IEEE, 2010, pp. 3025–3030.

[13] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3084–3089.

[14] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The roboearth cloud engine," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 438–444.

[15] "C2tam: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401 – 413, 2014.

[16] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular slam with multiple micro aerial vehicles," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 3962–3970.

[17] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV '03, 2003, pp. 1470–.

[18] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[19] J. Serafin and G. Grisetti, "Using augmented measurements to improve the convergence of icp," in *Proc. of the 4th International Conference on Simulation, Modeling and Programming for Autonomous Robots. (SIMPAR 2014)*, 2014.

[20] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, 2013, pp. 3748–3754.

[21] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proceedings*

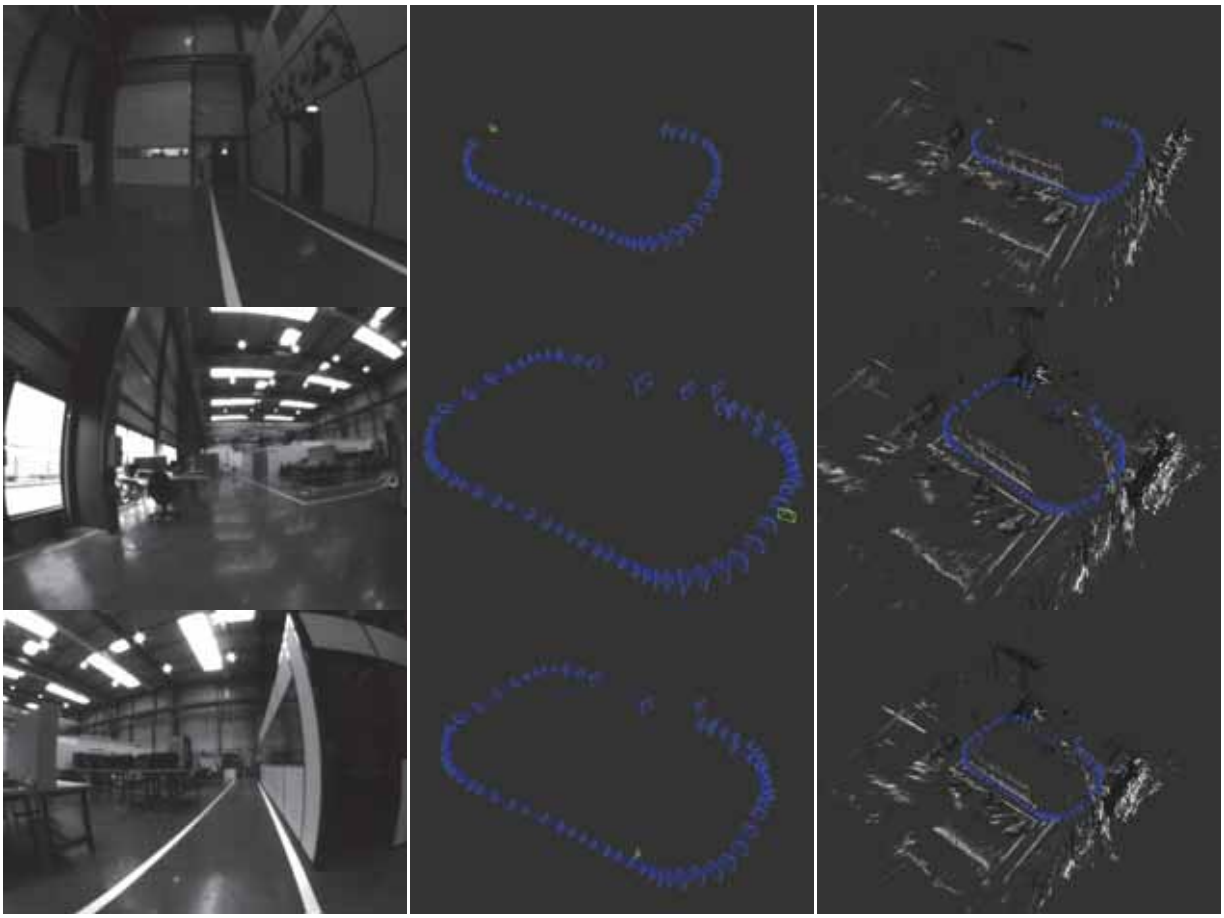

Fig. 4. Monocular SLAM Process on robot $R2$ at three different instants. Left: Image captured by the camera. Middle:Trajectory built by the robot. Right: Map built by the robot.

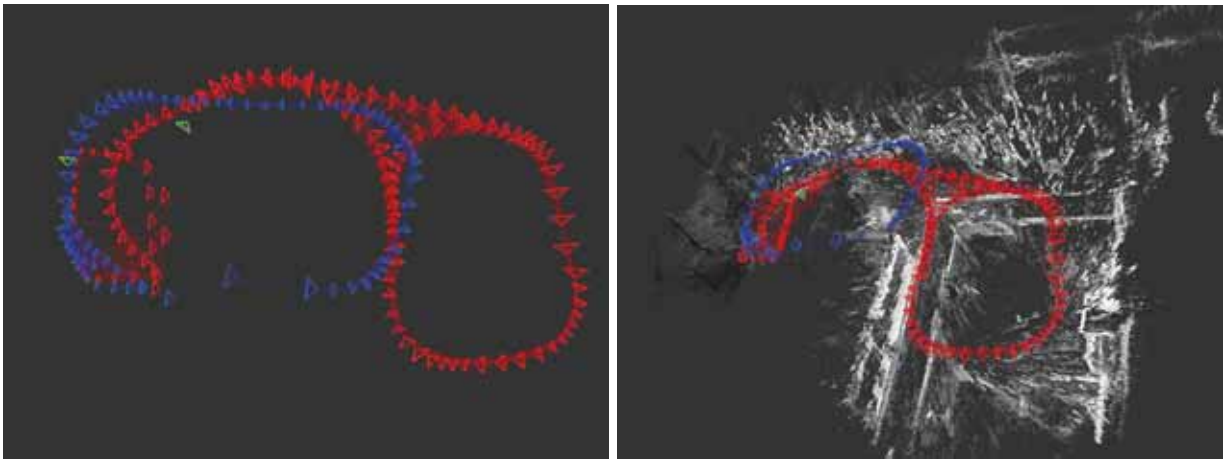

Fig. 5. Global map computed at the central server . *Left:* Trajectory of the two robots in the merged map. *Right:* Depth map associated with the keyframes from both robots.

of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 2011, pp. 3607–3613.

[22] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, 2013, pp. 1449–1456.

[23] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.

[24] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665.