SPM-SLAM: Simultaneous Localization and Mapping with Squared Planar Markers

Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez,
R. Medina-Carnicer

PII: S0031-3203(18)30322-4
DOI: https://doi.org/10.1016/j.patcog.2018.09.003
Reference: PR 6653

To appear in: *Pattern Recognition*

Received date: 5 January 2018
Revised date: 31 July 2018
Accepted date: 5 September 2018

Please cite this article as: Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, R. Medina-Carnicer, SPM-SLAM: Simultaneous Localization and Mapping with Squared Planar Markers, *Pattern Recognition* (2018), doi: https://doi.org/10.1016/j.patcog.2018.09.003

Highlights

- First system for Simultaneous Localization and Mapping in large environemnts from Squared Planar Markers
- Print markers with a regular printer and place them in environment. Use a camera an our system finds the camera position while estimating the pose of the markers at the same time.
- Real time operation on CPUs (150Hz) with a single execution thread.

SPM-SLAM: Simultaneous Localization and Mapping with Squared Planar Markers

Rafael Muñoz-Salinas[1,2,*], Manuel J. Marín-Jimenez[1,2], R. Medina-Carnicer[1,2]

Abstract

SLAM is generally addressed using natural landmarks such as keypoints or texture, but it poses some limitations, such as the need for enough textured environments and high computational demands. In some cases, it is preferable sacrificing the flexibility of such methods for an increase in speed and robustness by using artificial landmarks.

The recent work [1] proposes an off-line method to obtain a map of squared planar markers in large indoor environments. By freely distributing a set of markers printed on a piece of paper, the method estimates the marker poses from a set of images, given that at least two markers are visible in each image. Afterwards, camera localization can be done, in the correct scale. However, an off-line process has several limitations. First, errors can not be detected until the whole process is finished, e.g., an insufficient number of markers in the scene or markers not properly spotted in the capture stage. Second, the method is not incremental, so, in case of requiring the expansion of the map, it is necessary to repeat the whole process from start. Finally, the method can not be employed in real-time systems with limited computational resources such as mobile robots or UAVs.

To solve these limitations, this work proposes a real-time solution to the problems of simultaneously localizing the camera and building a map of planar markers. This paper contributes with a number of solutions to the problems arising when solving SLAM from squared planar markers, coining the term SPM-SLAM. The experiments carried out show that our method can be more robust, precise and fast, than visual SLAM methods based on keypoints or texture.

Keywords: Fiducial Markers, Marker Mapping, SLAM

1. Introduction

Simultaneous localization and mapping is the problem of creating a map of the environment and estimating the camera pose at the same time [2]. Sparse [3] and dense [4] solutions using natural features have attracted most of the research effort

*Corresponding author
Email addresses: rmsalinas@uco.es (Rafael Muñoz-Salinas), mjmarin@uco.es (Manuel J. Marín-Jimenez), rmedina@uco.es (R. Medina-Carnicer)
[1]Computing and Numerical Analysis Department, Edificio Einstein. Campus de Rabanales, Córdoba University, 14071, Córdoba, Spain, Tlfn:(+34)957212255
[2]Instituto Maimónides de Investigación en Biomedicina (IMIBIC). Avenida Menéndez Pidal s/n, 14004, Córdoba, Spain, Tlfn:(+34)957213861

reaching a high degree of performance. Nevertheless, they have a number of limitations in some realistic scenarios. First, they require a certain amount of texture, which in some indoor environments is not available (e.g., labs and corridors). Second, *bag of words* (BoW) [5] methods are employed to solve the relocalization problem. However, they have a limited performance under viewpoint changes and repetitive patterns that are common in certain environments. Third, when using a single camera, the map generated is scale agnostic and consequently can not be directly employed for navigation tasks. Four, it is well known that translational movement is required in order build the map when using a single camera.

In many cases, using artificial markers is an acceptable solution to robustly estimate the camera pose at a very reduced cost, using very few computational resources, and solving the above-mentioned problems. Approaches based on squared planar markers are very popular [6, 7, 8, 9, 10, 11, 12, 13, 14]. These markers are composed of an external black border and an internal code (most often binary) to uniquely identify them. Their main advantage is that the corners of a single marker can be employed for camera pose estimation. The general approach, however, is using a single marker, or at most, a small set of them for which their relative pose is known beforehand (e.g., a set of markers in a printed piece of paper). Of course, the range of applicability of such systems is limited to very small areas. Very few attempts have been made to create large-scale camera localization system based on squared planar markers that anyone can print and stick in the desired environment.

The main difficulty when dealing with squared planar markers is the ambiguity problem [15, 16, 17]. Although in theory, it should be possible to accurately obtain the pose from the four corners of a planar square, in practice, due to noise in the localization of the corners, two solutions appear, and in some cases, it is impossible to distinguish the correct one.

In the previous work [1], the authors of this paper proposed an off-line method to create a map of squared planar markers able to deal with the ambiguity problem. One only needs to print markers with a regular printer, place them in the desired environment so as to cover the working area, and taking pictures of them. Then, the method creates a map with the pose of the markers by analyzing the images. The method can employ an unlimited number of high-resolution images in order to achieve very high accuracy since no time restrictions are imposed (it is an off-line process). However, off-line processes have several limitations. First, errors cannot be detected until the whole process is finished, e.g., an insufficient number of markers in the scene or markers not properly spotted in the capture stage. Second, the method is not incremental, so, in case of requiring the expansion of the map, it is necessary to repeat the whole process from start. Finally, the method is not suitable for real-time applications, e.g., a mobile robot or a UAV with limited computational

Figure 1: SPM-SLAM: a method to build cost-effective and robust localization systems using squared markers. The printed markers are placed in arbitrary locations of the environment. Then, given a video sequence spotting the markers, the proposed system builds the map of squared planar markers and localizes the camera at the same time. Blue squares represent the map of markers, obtained from the images of the video sequence shown. The right image shows a three-dimensional representation of the markers found in blue, and in green the selected locations of the camera employed for optimization.

resources that has to create a map of the environment while navigating.

This paper proposes a complete system able to dynamically build a map of the markers and simultaneously localizing the camera pose from a video sequence. The proposed method uses only information from planar markers and is able to deal with the ambiguity problem. In our tests, the proposed method is able to run at more than 150 Hz using a single thread on a laptop computer. Figure 1 shows an example created in our lab, in which the pose of both the markers and the camera is obtained incrementally using the images of the video sequence recorded with a hand-held camera.

Four original contributions are made in this paper in order to build the proposed system. First, this is, up to our knowledge, the first real-time SPM-SLAM system able to deal with the ambiguity problem and to operate in large indoor environments. Second, we propose a method for map initialization from a set of ambiguously detected markers seen from at least two different locations. Third, we propose a method to estimate the pose of markers from ambiguous observation of them. Finally, we propose a method for loop closure detection and correction using squared planar markers. It is also worth mentioning that the proposed method is publicly available for evaluation purposes[3].

The rest of the paper is organized as follows. After presenting the related work in Sect. 2, we give in Sect. 3 an overview of the problem along with the main elements required to solve it. Sect. 4 describes the details of each module of the proposed system, and, in Sect. 5, a thorough experimental study is carried out on challenging scenes to validate our approach. Finally, the main conclusions of this work are summarized in Sect. 6.

[3] http://www.uco.es/investiga/grupos/ava/node/58

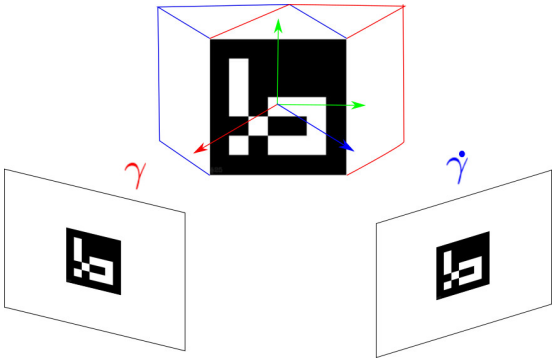

Figure 2: Pose ambiguity problem: the same observed projection can be obtained from two different camera poses γ and $\dot{\gamma}$.

2. Related works

Visual SLAM aims at solving the simultaneous localization and mapping problem using visual information exclusively. In [18], Klein and Murray presented their PTAM system, in which two different threads running in parallel perform tracking and mapping. The work showed the possibility of splitting the tasks into two different threads, achieving real-time performance. However, their keypoint descriptors did not consider the detection of large loops. The recent work of Mur-Artal *et al.* [3] presents a keyframe-based SLAM method using ORB keypoints [19]. Their approach operates in real-time and is able to detect the loop closure and correct the poses accordingly. Engel *et al.* [4] proposed a semi-dense monocular visual SLAM solution called LSD-SLAM. In their approach, scenes are reconstructed in a semi-dense fashion, by fusing spatiotemporally consistent edges. However, in order to solve the relocalization and loop-closure problems, they use keypoint features. Despite the good results achieved by these type of systems, they pose several drawbacks. Tracking typically fails with rapid motion, large changes in viewpoint, or significant appearance changes. In addition, the maps obtained are scale agnostic, so they can not be directly used for robot navigation. Finally, relocalization often fails in regions of the environment far from these where the map was created.

Square-based fiducial markers [6, 7, 8, 9, 10, 11] consist of an external black border and an internal code (most often binary) to uniquely identify each marker. The main advantage of such markers is that the camera pose can be estimated using its four corners, under some circumstances. While in theory, the pose of the camera can be uniquely estimated from the four corners, in practice, there is a rotation ambiguity that corresponds to an unknown reflection of the plane about the camera's z-axis [15, 16, 17]. The problem is shown in Figure 2. The marker is represented as the side of a cube, which can be in two different orientations (red and blue color) thus obtaining almost identical projections from two different camera locations γ and $\dot{\gamma}$. The methods proposed in [15, 16, 17] find the best solution by a careful analysis of the projections. In most cases, the reprojection error of one solution is much lower than the reprojection error of the other one. Then, no

ambiguity problem is observed and the correct solution is the one with lowest error. However, when imaging small planes or planes at a distance significantly larger than the camera's focal length, in the presence of noise, the reprojection error of both solutions is so similar that it is not possible to determine the correct one. In these cases, the observed size of the marker in the image becomes small and thus the error in the estimation of the corners becomes relatively large. Then, the reprojection errors of both solutions become similar and the correct one can not be distinguished.

Maybe because of this problem, very few attempts have been made to automatically create large maps of markers placed in arbitrary locations as we propose in this work. Davison *et al.* propose in [20] a method for monocular SLAM in which a squared marker is employed for initialization. Although they rely exclusively on natural features for tracking, they show that using a single squared marker for initialization is a good choice to obtain the map in the real scale.

Lim and Lee [21] present an approach to SLAM with planar markers. An Extended Kalman-Filter (EKF) is used to track a robot pose while navigating in an environment with some markers in it. As markers are found, they are added to the map considering the current robot pose along with the relative pose of the marker and the robot. Their approach, however, does not consider the ambiguity or the loop closure problems. A similar approach is presented in [22] for an autonomous blimp.

The work of Klopschitz and Schmalstieg [23] shows a system for estimating the 3D location of fiducial markers in indoor environments. A video sequence of the environment is recorded and camera position estimated using (Structure from Motion) SfM (with natural keypoints). Once camera locations are accurately obtained, marker locations are obtained by triangulation. This approach does not deal with the dual problem of camera and marker localization jointly. Also, it assumes the correct functioning of an SfM method that, as we have already commented, is not always possible.

Karam *et al.* [24] propose the creation of a pose graph where nodes represent markers and edges the relative pose between them. The map is created in an on-line process, and edges are updated dynamically. Whenever a pair of markers are seen in a frame, their relative position is updated and if it is better than the previous one, replaced. For localization, their approach selects, from the set of visible markers at that time, the one whose path to an origin node is minimum. Their approach poses several problems. First, they do not account for the ambiguity problem. Second, they only consider one marker for localization from all visible ones. However, using all visible markers at the same time can lead to better localization results. Third, their experimental results conducted do not really prove the validity of their proposal in complex scenes.

The work of Neunert *et al.* [25] presents a monocular visual-inertial EKF-SLAM system based on artificial landmarks.

Again, an EKF is employed to do SLAM fusing information from the markers and an inertial measurement unit. As in previous cases, the marker ambiguity and loop closure problems are not considered in their work. Additionally, our method does not need the use of inertial sensors to solve the problem.

Finally, the authors of this work have proposed in [1] an off-line method to obtain a map of markers in large indoor environments. In our previous work, we deal with the ambiguity problem and propose a robust solution for marker mapping and localization. The main difference with this work is that [1] considers that all the video frames are all available at once, and solves the problem globally. Nonetheless, the off-line approach poses some limitations. For instance, errors in the video capture of the sequence, such as a missing marker, cannot be detected until it is post-processed. In contrast, this work proposes an online solution, in which the map is built incrementally as video frames are captured.

3. System overview

The problem to be solved in this paper is the following. Consider a video sequence recording an environment in which fiducial markers have been placed. The goal is to simultaneously determine the pose of the camera w.r.t. an arbitrary reference system (global reference system) and to create a map of the markers and their location in the global reference system. In our problem, only information from the fiducial markers is used.

This section provides an introduction to the system main elements. First, we provide a reference guide that will help the reader with the notation employed through this work (Sect. 3.1) and then we provide a brief explanation of the main mathematical concepts required for this work (Sect. 3.2). Afterward, we will describe the map (Sect. 3.3), which is the data structure employed to keep the operational information and we will continue (Sect. 3.4) providing a brief description of the system control loop.

3.1. Clarification on the notation

Below, we provide a summary of the most relevant terms employed along the paper:

- f^t: frame. Image acquired by a camera at the time instant t.
- frs: frame reference system. Reference system centered on the camera origin when the frame was acquired. Each frame has its own reference system.
- mrs: marker reference system. Reference system centered in a marker. Each marker has its own reference system.
- grs: global reference system. The common reference system w.r.t. which we desire to obtain all measures.

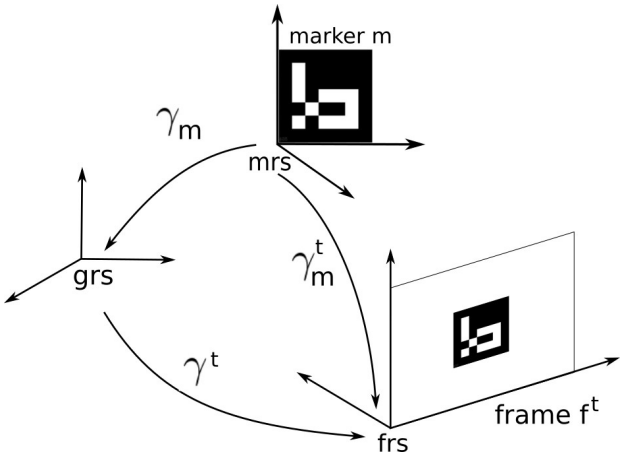

Figure 3: Main terms employed along the paper.

- γ_m: grs $\leftarrow$ mrs. Transform points from the reference system of marker m to the global reference system.
- γ^t: frs $\leftarrow$ grs. Transform points from the global reference system to the reference system of frame t.
- γ_m^t: frs $\leftarrow$ mrs. Transform points from the reference system of marker m to the reference system of frame t.

Please notice that, when using transformations γ, the superscript refers to frames, while the under-script refers to markers. Fig. 3 depicts the terms explained.

3.2. Initial concepts

Let us consider a three-dimensional point $p_a \in \mathbb{R}^3$ in an arbitrary reference system a. In order to express such point into another reference system b, it undergoes a rotation followed by a translation. Let us denote by $\gamma \in \mathbb{R}^6$ the three rotational and translational components $r = (r_x, r_y, r_z)$ and $t = (t_x, t_y, t_z)$:

$$\gamma = (r, t) \mid r, t \in \mathbb{R}^3 \tag{1}$$

Using Rodrigues' rotation formula, the rotation matrix $\mathbf{R}$ can be obtained from r as:

$$\mathbf{R} = \mathbf{I}_{3\times3} + \overline{\mathbf{r}} \sin\theta + \overline{\mathbf{r}}^2(1 - \cos\theta), \tag{2}$$

where $\mathbf{I}_{3\times3}$ is the identity matrix and $\overline{\mathbf{r}}$ denotes the antisymmetric matrix

$$\overline{\mathbf{r}} = \begin{bmatrix} 0 & -r_z/\theta & r_y/\theta \\ r_z/\theta & 0 & -r_x/\theta \\ -r_y/\theta & r_x/\theta & 0 \end{bmatrix}, \tag{3}$$

such that $\theta = ||r||_2$.

Then, in combination with t, the 4×4 matrix

$$\Gamma(\gamma) = \begin{bmatrix} \mathbf{R} & t^\top \\ \mathbf{0} & 1 \end{bmatrix} \tag{4}$$

can be used to transform a point in homogeneous coordinates from a to b as:

$$\begin{bmatrix} p_b^\top \\ 1 \end{bmatrix} = \Gamma(\gamma) \begin{bmatrix} p_a^\top \\ 1 \end{bmatrix} \tag{5}$$

To ease the notation through this paper, we will define the operator $(\cdot)$ to express the transform of Eq. 5:

$$p_b = \gamma \cdot p_a. \tag{6}$$

Likewise, we also define the operator $(\cdot)$ to concatenate transforms such as:

$$\gamma_{a,b} = \Gamma(\gamma_a)\Gamma(\gamma_b) = \gamma_a \cdot \gamma_b. \tag{7}$$

The basic idea is that we will operate along the paper with the 6-dof parameter γ as if it was the equivalent 4×4 matrix $\Gamma(\gamma)$.

A point p projects into the camera plane into a pixel $u \in \mathbb{R}^2$. Assuming that the camera parameters are known, the projection can be obtained as a function:

$$u = \Psi(\delta, \gamma, p), \tag{8}$$

where

$$\delta = (f_x, f_y, c_x, c_y, k_1, \ldots, k_n),$$

refers to the camera intrinsic parameters, comprised by the focal distances (f_x, f_y), optical center (c_x, c_y) and distortion parameters $(k_1, \ldots, k_n)$. The parameter γ represents camera pose from which frame was acquired, i.e., the transform that moves a point from an arbitrary reference system to the camera one.

Squared fiducial markers are comprised by an external black border and an inner binary code m that identify them.

Assuming that all markers have the same size s, their four corners can be expressed w.r.t. one of its corners as:

$$\begin{aligned} c_1 &= (\ 0,\ 0,\ 0\ ), \\ c_2 &= (\ 0,\ s,\ 0\ ), \\ c_3 &= (\ s,\ s,\ 0\ ), \\ c_4 &= (\ s,\ 0,\ 0\ ). \end{aligned} \tag{9}$$

We shall denote by

$$\omega_m^t = \{u_{m,l}^t \mid u \in \mathbb{R}^2, l = 1 \dots 4\} \tag{10}$$

the pixel locations in which the four corners of marker m (Eq. 9) are observed in frame f^t.

The reprojection error of a marker m in a frame f^t given the transform γ is calculated as:

$$e_m^t(\gamma) = \sum_{l=1}^{4} \left[\Psi(\delta, \gamma, c_l) - u_{m,l}^t\right]^2, \tag{11}$$

where Ψ is the projection function defined in Eq. 8, and γ is a transform moving the 3d corner locations from the mrs to the frs.

As already mentioned, when estimating the pose of a marker m from a single image, there is a rotation ambiguity that corresponds to an unknown reflection of the plane about the camera's z-axis. However, it is possible to obtain the two solutions [16, 17], let us denote them by

$$\theta_m^t = \{\gamma_m^t, \dot{\gamma}_m^t\}, \tag{12}$$

and their corresponding reprojection errors $e_m^t(\gamma_m^t)$ and $e_m^t(\dot{\gamma}_m^t)$. Assuming that the solutions are sorted such that $e_m^t(\gamma_m^t) < e_m^t(\dot{\gamma}_m^t)$, the ambiguity problem occurs when the ratio $e_m^t(\gamma_m^t)/e_m^t(\dot{\gamma}_m^t)$ is near one. In our work, we assume that when the error ratio is above a threshold τ_e, the solution γ_m^t is unambiguous and thus a reliable indication of the maker pose w.r.t. to the frame. Otherwise, the pose is considered ambiguous. This paper proposes methods to deal and overcome the ambiguity problem by using information from either multiple markers or multiple views.

3.3. System map

In order to store all the elements required to solve the SLAM problem, our system maintains the following map

$$\mathcal{W} = \{\mathcal{M}, \mathcal{F}, \mathcal{G}, \omega, \gamma^{t-1}\}.$$

$\mathcal{W}$ is composed of a set of markers $\mathcal{M}$, a set of keyframes $\mathcal{F}$, a connection graph $\mathcal{G}$, the frame index ω of the current keyframe f^ω, and the pose of the last frame observed γ^{t-1}.

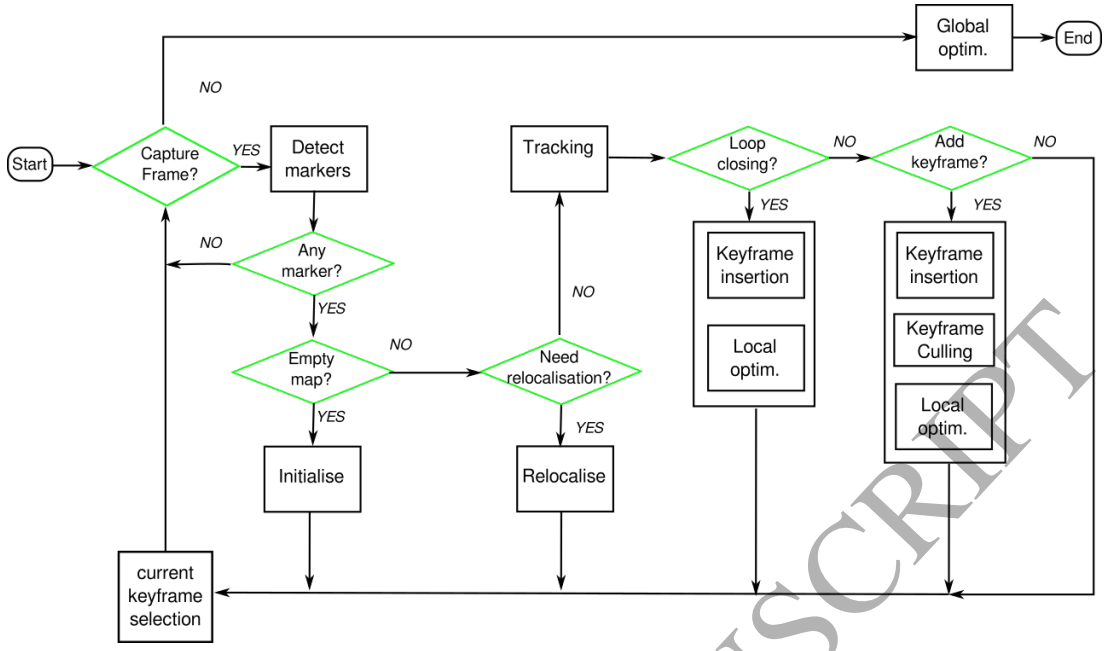

Figure 4: Pipeline for marker-based SLAM. See text for details.

The set

$$\mathcal{M} = \{m | m \in \mathcal{D}\},$$

represents the set of markers observed in the scene, where $\mathcal{D}$ is the set of valid marker identifiers, also called dictionary [10]. If the pose of marker m has been properly estimated, then γ_m represents the transformation moving the corners of the marker from the marker reference system (mrs) to the global reference system (grs). Nevertheless, the pose of a marker is not necessarily estimated the first time it is spotted. Sometimes, several observations are required in order to produce an estimate of its pose. So, if the pose of an observed marker has not been estimated, we set $\gamma_m = 0$ indicating that it is invalid. Along the video sequence, marker poses will be subject to continuous optimization in order to globally match all available observations.

The map also registers information about some of the visited frames, named keyframes. In particular, for each keyframe $f^i \in \mathcal{F}$, we keep the following information:

- First, the transform moving points from the grs to the frame reference system (frs) is denoted γ^i. The initial estimate of the frame pose is obtained using the information from the markers in $\mathcal{M}$. Along the video sequence, the poses are refined as part of an optimization process.

- Second,

$$\Lambda^i = \{m | m \in \mathcal{M}\}, \tag{13}$$

represents the set of visible markers in frame f^i.

The graph $\mathcal{G}$ represents relations between the observed keyframes. Nodes represent keyframes, while edges represent the total number of common markers visible in two keyframes, i.e., $|\Lambda^i \cap \Lambda^j|$ for keyframes f^i and f^j. There will only be an edge between two nodes if they have at least a common marker. The graph will be used both in the optimization process to accurately estimate marker and keyframe poses, and for the loop closure detection and correction.

The current keyframe f^ω is the element from $\mathcal{F}$ closest (in the 3D space) to the previous frame and it plays an important role when inserting new keyframes to the map, and in the process of loop closure detection. Finally, γ^{t-1} represents the pose of the last visited frame f^{t-1}.

3.4. Operational overview

This section aims at providing an outline of the main tasks involved in the proposed system. Figure 4 depicts the steps that are explained below.

In the beginning, the map is empty and the system is in *initialization* mode, looking for one or several markers to start the tracking process. Initialization can be done from either one or two frames using a novel method we propose (Sect. 4.1), which is one of the contributions of this paper. As a result of the initialization, the initial keyframe(s) are added to the map, as well as the initial marker(s).

For each new frame acquired f^t, markers are detected and *tracking* is performed. Tracking consists in estimating the current frame pose γ^t, by minimizing the reprojection error of the visible marker corners. To do so, the previous frame pose γ^{t-1} is used as the initial estimate. Indeed, tracking can only be done using the observed markers with a valid pose ($\gamma_m \neq 0$) in the map (Sect. 4.3).

Once γ^t is estimated, the map is updated. If f^t has no new markers, it is added to the map only if the 3D distance to the current keyframe f^ω is above a certain threshold (Sect. 4.4). However, if the frame contains new markers, both the frame and markers are added to the map. Nonetheless, if the pose of a new marker cannot be unambiguously determined, it is added but setting $\gamma_m = 0$, delaying the estimation of its pose to subsequent frames. This work also contributes by proposing a method to estimate the pose of markers from ambiguous estimations in an arbitrary number of frames (Sect. 4.5).

To avoid the unnecessary growth of the map, a keyframe *culling* process is run whenever a new keyframe is added to the map. The goal is to keep the minimal set of keyframes that allows a proper optimization of the reprojection error of the markers of the map (Sect. 4.6).

After keyframe culling, a *local optimization* process is run in order to adjust both the keyframe and marker poses by minimizing the reprojection error of the markers (Sect. 4.7). The basic idea is that the new observations just added (keyframes and markers) help to improve the keyframe poses. However, this is a local process since only those keyframes of the map that observe the markers in the current frame will be affected by this process.

An important aspect to consider while tracking is the detection of *loop closures* (Sect. 4.9). A loop closure takes place when a region of the map is revisited. In most cases, the error accumulated along the camera path makes impossible to obtain an accurate pose estimation. So, the closure must be properly detected and corrected before continuing the tracking process (Sect. 4.7). Then, the current keyframe can be added to the map and the local optimization process run again to improve the accuracy of the map. The method for loop closure detection and correction using markers (Sect. 4.9) in another contribution of this work.

The final step before capturing a new frame is the *selection of the current keyframe* f^{ω}. As already indicated, the current keyframe is the one nearest to f^{t-1} and is used for keyframe insertion and loop closure detection.

In case of not detecting any marker in the current frame, the system moves to the state of *relocalization*. In relocalization mode, the system waits until valid markers are observed to estimate the current pose. However, relocalization is a process prone to more errors than tracking, since no initial estimation is known from the previous frame. We propose a method to relocalize even from ambiguously detected markers in Sect. 4.8.

Finally, when there are no more frames to be processed, we perform a global optimization process (Sect. 4.10) that jointly optimizes keyframe and marker poses, along with the camera intrinsic parameters.

4. Detailed system description

This section explains in detail the different components involved in the whole process previously outlined.

4.1. Initialization

Initialization is done at the beginning of the video sequence and can not be completed until a set of restrictions are met. At the initialization stage, the global reference system is established and the first markers and keyframes added to the map.

We propose two initialization methods: *one-frame* and *two-frames* initialization. The former is able to initialize the system with only one frame if there is at least one marker unambiguously detected. The latter initializes the system even from ambiguously detected markers only. In order to be able to run both methods simultaneously, we proceed as follows. We store the first frame f^0 and run one-frame initialization. If it succeeds, then the system is initialized and we move to

the tracking state. If not, we process the next frame f^1 and run one-frame initialization in it. If it does succeed, we move to tracking state. If not, we try two-frames initialization using f^0 and f^1. If it does not succeed, we will try using f^0 with subsequent frames $f^2, f^3, \ldots$. If after n attempts, neither one-frame nor two-frames approaches succeed, the role of f^0 is replaced by f^n.

4.1.1. One-frame initialization

One-frame initialization is done when among the markers observed in a frame, the pose of at least one marker is detected unambiguously, i.e.

$$\exists m \in \Lambda^t \mid e(\gamma_m^t)/e(\dot{\gamma}_m^t) > \tau_e. \tag{14}$$

Then, the frame is considered to be the center of the global reference system and added to the map along with the observed markers. If two-frames initialization succeeds, the two frames are added to the map along with all the detected markers, and a connection between them is established in the graph (the weight of the edge equals the number of common markers in both frames). Then, the first frame is assumed to be the center of the grs, and the γ^t is established as the pose of the second frame, which is also the current keyframe f^ω. Below, we explain the method we propose for two-frames initialization.

4.1.2. Two-frames initialization

The proposed two-frames initialization method estimates the pose of both frames and markers even from markers ambiguously observed in the two frames. To do so, we first need to estimate the relative pose $\gamma^{1,0}$ between the two frames considered f^0 and f^1. Let us illustrate the proposed solution with the simplest example: a single marker m which is ambiguously detected in frames f^0 and f^1. Let be $\gamma_m^{1,0}$ the relative pose for the frames using the marker m. In that case, for the frame f^0 we obtain two marker poses:

$$\theta_m^0 = \{\gamma_m^0, \dot{\gamma}_m^0\},$$

and another two:

$$\theta_m^1 = \{\gamma_m^1, \dot{\gamma}_m^1\}$$

for frame f^1.

The relative pose between the two frames can be estimated from θ_m^0 and θ_m^1. In total, there are four possible solutions for $\gamma_m^{1,0} \in \mathcal{C}_m^{1,0}$ by combining the elements in θ_m^0 and θ_m^1:

$$\mathcal{C}_m^{1,0} = \{\gamma_m^1 \left(\gamma_m^0\right)^{-1}, \gamma_m^1 \left(\dot{\gamma}_m^0\right)^{-1}, \dot{\gamma}_m^1 \left(\gamma_m^0\right)^{-1}, \dot{\gamma}_m^1 \left(\dot{\gamma}_m^0\right)^{-1}\}. \tag{15}$$

If there is more than one marker, we shall denote

$$\mathcal{C}^{1,0} = \{\mathcal{C}_m^{1,0} \mid m \in \Lambda^1 \cup \Lambda^0\} \tag{16}$$

the set of possible transforms between the two frames. The best solution is the one that better explains the given observations, i.e., the solution that minimizes the reprojection error of the marker corners in both frames:

$$\begin{aligned}\beta(\gamma) = \sum_{m \in \Lambda^1 \cup \Lambda^0} & \min\left(e_m^1(\gamma\gamma_m^0), e_m^1(\gamma\dot{\gamma}_m^0)\right) + \\ & \min\left(e_m^0(\gamma^{-1}\gamma_m^1), e_m^0(\gamma^{-1}\dot{\gamma}_m^1)\right),\end{aligned} \tag{17}$$

$$\gamma^{1,0} = \underset{\gamma \in \mathcal{C}^{1,0}}{\operatorname{argmin}}\, \beta(\gamma). \tag{18}$$

Since it is not known yet which of the two solutions from θ_m^i is the good one, the min function is employed. In order to make the process robust, initialization is only accepted if the average reprojection error $\beta(\gamma^{1,0})/|\mathcal{C}^{1,0}|$ is small and the baseline between the frames is larger than τ_{bl}, which is a parameter of the system.

Once the pose between the frames is determined, the pose of the observed markers w.r.t the grs must be estimated. For each marker, four possible poses can be obtained given the values of θ_m^0 and θ_m^1. The best solution is the one minimizing the reprojection error of the marker corners in both frames:

$$\gamma_m = \begin{cases} \gamma_f & \text{if } \gamma_f \in \theta_m^0 \\ \gamma^{0,1}\gamma_f & \text{if } \gamma_f \in \theta_m^1 \end{cases} \tag{19}$$

$$\gamma_f = \underset{\gamma \in \theta_m^0 \cup \theta_m^1}{\operatorname{argmin}} \begin{cases} e_m^0(\gamma) + e_m^1(\gamma^{1,0}\gamma) & \text{if } \gamma \in \theta_m^0 \\ e_m^0(\gamma^{0,1}\gamma) + e_m^1(\gamma) & \text{if } \gamma \in \theta_m^1 \end{cases} \tag{20}$$

where $\gamma^{0,1} = \left(\gamma^{1,0}\right)^{-1}$,

4.2. The reference keyframe selection for the current frame

The selection of the current keyframe f^ω is the last step of the process. The goal is to find, amongst the keyframes, the nearest one to the current camera location. It will be used when adding new keyframes to avoid adding frames too close to each other, and also in the process of loop closure detection. It is computed as:

$$\omega = \underset{j \in \mathcal{F}}{\operatorname{argmin}}\ d(f^j, f^t), \tag{21}$$

where d denotes the translational distance between two frame poses:

$$d(f^i, f^j) = ||(t_x^i, t_y^i, t_z^i) - (t_x^j, t_y^j, t_z^j)||_2.$$

4.3. Tracking

Tracking is dealt as a non-linear optimization process minimizing the reprojection error of the observed markers in the new frame f^t, using the previous pose γ^{t-1} as starting point.

Tracking consists in estimating the pose that transforms a point from the grs to the frs of frame f^t, i.e. γ^t, assuming that the pose of the markers is fixed. In essence, the problem reduces to minimizing the squared reprojection error of the marker corners (Eq. 11) so as to obtain the value of γ^t. Nevertheless, please remember that not all the detected markers can be employed for tracking. Only those with a valid pose on the map (i.e., $\gamma_m \neq 0$) are eligible for tracking purposes. In addition, as we will explain later in Sect. 4.9, the markers that cause loop closures will not be employed for tracking purposes. Thus, only a subset of the detected markers, denoted by $\mathcal{L}'(t)$ (see Eq. 36 in Sect. 4.9.1) are employed for tracking.

As a consequence, the reprojection error of the markers employed for tracking is expressed as:

$$\mathbf{E}(t) = \sum_{m \in \mathcal{L}'(t)} e_m^t(\gamma^t \cdot \gamma_m). \tag{22}$$

The Levenberg–Marquardt's (LM) algorithm [26] is used for minimization.

4.4. Addition of new keyframes and markers to the map

Once γ^t has been estimated, we must decide if the frame should be added to the map as a new keyframe. The key idea is that keyframes are added only if they add information to the system. Thus, the following rules apply:

1. If the frame has, at least, one new marker, the frame and the markers are added to the map. Markers unambiguously detected are added with their estimated location, whereas ambiguously detected markers are added setting their pose to zero, i.e., $\gamma_m = 0$. Later, it will be possible to estimate their pose from multiple views (Sect. 4.5).
2. Else,
   (a) If there is a marker in $\mathcal{M}$ with zero pose ($\gamma_m = 0$) whose pose is unambiguously detected in this frame, then add the frame and set the marker pose.
   (b) Else, the frame is added only if the distance to the current keyframe f^ω is larger than the threshold τ_{bl}. This ensures a minimum baseline between frames for triangulation and optimization (Sects. 4.5 and 4.7).

4.5. Marker pose estimation

As previously indicated, when a marker m is first spotted in a frame f^t, it is added to the map. If its location w.r.t to the frame γ_m^t can be unambiguously determined, then its pose in the grs is obtained as:

$$\gamma_m = \left(\gamma^t\right)^{-1} \cdot \gamma_m^t \tag{23}$$

Otherwise, its pose is set to zero until it can be estimated.

This work proposes a method to estimate the pose of a marker from a set of ambiguous observations (see Section 2 for a description of the ambiguity problem). Let $(\gamma^1, \ldots, \gamma^n)$ be the poses of the keyframes in which marker m has been observed so far. Remember that these are known poses obtained using other markers (not the marker m) and that they express the transform from the grs to the frs (Fig. 3).

Consider now the case of the marker m when ambiguously observed in a keyframe f^i. The markrer pose wrt the grs system γ_m is unknown as well as γ_m^i, its position wrt to the frame. Since the marker m has been ambiguously detected, there are two possibilities for the true pose γ_m^i, let us denote them γ_m^i and $\dot{\gamma}_m^i$. Since the marker is observed in n keyframes, and two different marker poses are obtained for each keyframe, there is a total of $n \times 2$ possible poses for the marker m in all keyframes it is observed, that we shall denote by:

$$\Theta_m = \{\gamma_m^1, \dot{\gamma}_m^1, \ldots, \gamma_m^n, \dot{\gamma}_m^n\}. \tag{24}$$

See Figure 5 for a visual explanation of the problem for the limited case of two keyframes.

Let us also define $I(\alpha)$ as the function that given a pose $\alpha \in \Theta_m$, returns the frame it belongs to:

$$I(\alpha) = i, \quad if \alpha \in \Theta_m \wedge \alpha = (\gamma_m^i \vee \dot{\gamma}_m^i). \tag{25}$$

Our proposal is to choose $\gamma_m^* \in \Theta_m$ that minimizes the reprojection error of the marker in all n frames:

$$\gamma_m^* = \underset{\alpha \in \Theta_m}{\operatorname{argmin}} \sum_{i=1}^{n} e_m^i \left(\gamma^i \left(\gamma^{I(\alpha)} \right)^{-1} \alpha \right), \tag{26}$$

where $\gamma^{I(\alpha)}$ is the pose of the keyframe $I(\alpha)$. Then, the initial marker pose can be obtained from its ambiguous observation as:

$$\gamma_m = \left(\gamma^{I(\gamma_m^*)} \right)^{-1} \gamma_m^*. \tag{27}$$

In order for Equation 26 to be useful, the distance between the keyframes considered must be large enough. Imagine the degenerate case of two keyframes f^i and f^j with zero or infinitesimal displacement between them. Then, the estimated

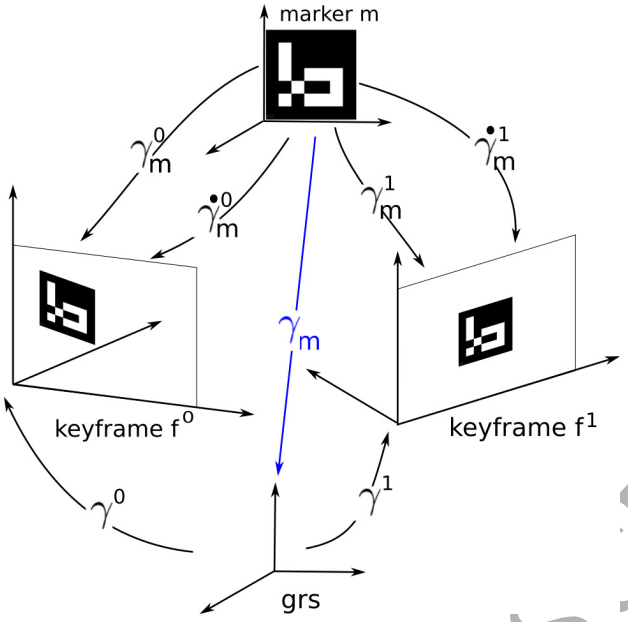

Figure 5: Initial pose estimation γ_m of a marker m ambiguosly detected in two keyframes f^0 and f^1 with known poses γ^0 and γ^1. Due to ambiguity, there are two possible poses for each keyframe-marker pair. In total, it makes the set of poses $\{\gamma_m^0, \dot{\gamma}_m^0, \gamma_m^1, \dot{\gamma}_m^1\}$ from which the best estimation γ_m is obtained using Eq.27.

poses of the markers in both keyframes are likely to be equal and thus it would be impossible to resolve the marker pose ambiguity. To prevent this case, keyframes inserted in the map must have a minimum distance τ_{bl} (as already explained in Sect. 4.4). Finally, it is necessary to establish the minimum number of keyframes in which markers must be observed: our experience indicates that three is a good value.

4.6. Keyframe culling

In order to avoid an unlimited growth of the map, every time a keyframe is added, a culling procedure is run to remove redundant keyframes. This helps to keep the processing time constrained when global and local optimizations are done (Sect. 4.7 and 4.10). The key idea is that for each marker $m \in \mathcal{M}$, we select a set of keyframes from $\mathcal{F}$ in which m is seen from a wide variety of poses. Therefore, the number of keyframes is reduced without significantly compromising the results of the optimization processes.

Let us denote

$$\Omega^m = \{i \mid f^i \in \mathcal{F}\} \tag{28}$$

to the set of keyframes selected for marker m, considering that the size of the set $N_c \in |\Omega^m|$ is given. Our goal can be described as selecting from $\mathcal{F}$, the N_c most distant keyframes. We proceed as follows: first, we select the two most separated keyframes from $\mathcal{F}$ in which the marker appears and add it to Ω^m. Then, we keep adding the farthest keyframe

from the elements in Ω^m until the desired number of keyframes is reached. The process is repeated independently for each keyframe obtaining:

$$\Omega = \bigcup_{m \in \mathcal{M}} \Omega^m. \tag{29}$$

Finally, the keyframes of the map not in Ω are removed from the map.

4.7. Local optimization

Whenever a new keyframe is added to the map, the pose of its neighbor keyframes, as well as the pose of the markers observed in them, is updated to account for the new observation.

Let us denote by

$$\mathcal{G}(f^t) = \{i \mid f^i \in \mathcal{F}, \Lambda^i \cap \Lambda^t \neq \emptyset\}, \tag{30}$$

the set of keyframes connected to a keyframe f^t according to the graph $\mathcal{G}$, and by

$$\mathcal{G}_m(f^t) = \bigcup_{i \in \mathcal{G}(f^t)} \Lambda^i, \tag{31}$$

to the markers visible from the keyframes in $\mathcal{G}(f^t)$ (see equation 13).

The goal of the local optimization is to jointly optimize the poses of the keyframes and markers in $\mathcal{G}(f^t)$ and $\mathcal{G}_m(f^t)$ respectively. However, since the markers in $\mathcal{G}_m(f^t)$ might also be seen from other keyframes not included in $\mathcal{G}(f^t)$, those must also be considered. Let us denote the whole set of keyframes in which marker m is visible as

$$\mathcal{V}(m) = \{i \mid f^i \in \mathcal{F}, m \in \Lambda^i\}. \tag{32}$$

Then, the total reprojection error of all markers and keyframes included in the local optimization is expressed as:

$$\mathbf{e}(\gamma^1, \ldots, \gamma^a, \gamma_1, \ldots, \gamma_b) = \sum_{m \in \mathcal{G}_m(f^t)} \sum_{i \in \mathcal{V}(m)} e_m^t(\gamma^i \cdot \gamma_m), \tag{33}$$

where γ^i represents the pose of the keyframes in $\mathcal{G}(f^t)$, γ_m the poses of the markers in $\mathcal{G}_m(f^t)$, and a, b represent the cardinality of these sets.

As in the previous case, Eq. 33 is minimized using the LM algorithm [26], but in this case, exploiting the sparseness of the data. It is clear that since not every marker projects in every keyframe, the Jacobian matrix employed by the LM algorithm is sparse and thus using a sparse approach reduces considerably the computing time.

Finally, it must be indicated that, cannot avoid infinite solutions, the pose of the first keyframe f^0 is never subject to optimization. Otherwise, the whole reference system could be moved arbitrarily generating infinite solutions for the optimization process.

4.8. Relocalization

When no valid markers are visible in the current frame, its pose cannot be estimated and the system enters in relocalization mode. Later, when a frame with valid markers is available again, the system proceeds to relocalization.

Since relocalization consists in estimating the frame pose without prior information, it is prone to more errors than the tracking process. When tracking, the initial pose employed for optimization γ^{t-1} is close to the real one. Thus, marker pose ambiguity is not a problem when optimizing. However, when relocalizing without knowledge of the previous pose, we are subject to the ambiguity problem.

Imagine the simplest case of being in relocalization mode and detecting a single marker. If the marker is unambiguously detected there would be no problem. However, it would be very risky to relocalize from this single marker if its pose was ambiguously determined.

In this work we employ the robust method for relocalization from a set of ambiguously detected markers proposed in [1]. Let us consider the simplest case of two markers i and j already in the map with known poses that are ambiguously detected in the frame f^t. The set of marker poses obtained in frame f^t would be

$$\Theta^t = \{\gamma_i^t, \dot{\gamma}_i^t, \gamma_j^t, \dot{\gamma}_j^t\}.$$

Let us imagine that γ_i^t is the best estimation. Then, we could use it to estimate the pose of the frame f^t as:

$$\mathcal{E}(\gamma_i^t) = \gamma_i^t \gamma_i^{-1},$$

considering that γ_i is already known. Then, the reprojection error of both markers

$$\sum_{k \in \{i,j\}} e_k^t(\mathcal{E}\left(\gamma_i^t\right)\gamma_k)\,.$$

should be minimal.

In general, the relocalization problem from markers reduces to finding the transform from Θ^t minimizing the reprojection error of the valid markers observed in f^t:

$$\underset{\gamma \in \Theta^t}{\operatorname{argmin}} \sum_{k \in \Lambda^t | \gamma_k \neq 0} e_i^t(\mathcal{E}\left(\gamma\right)\gamma_k)\,. \tag{34}$$

The only case in which Eq. 34 cannot, is if f^t contains a single valid marker which is ambiguously detected. In the rest of possible cases, the proposed method can on, e.g., one marker unambiguously detected, two markers ambiguously detected, one marker unambiguously detected and two markers ambiguously detected, etc.

The pose estimated in Eq. 34 is an initial solution that is further refined using Eq. 22, as in the tracking process.

4.9. Loop closure detection and correction

When a region of the map is revisited after a long trajectory, there is an inevitable drift in the camera pose estimation. Then, loop closure consists in detecting this situation and correcting the drift by distributing error along the loop.

It must be noticed that loop closure in our problem cannot be handled as in keypoint-based SLAM. When using keypoints, loop closure can be detected after tracking using several approaches such as [27, 28, 29, 30]. Once detected, the keypoints in the current frame are matched against the keypoints in the starting loop keyframe.

We would like to stress that while revisiting a place does not affect the tracking performance of traditional SLAM techniques, it can be a cause of failure in SPM-SLAM if it is not managed properly. Imagine a scenario with ten markers $\{0, \ldots, 9\}$ placed around a room. In the initial frame, the marker 0 is spotted, and then, the camera moves around the room discovering the sequence of markers 1 to 9. Eventually, the camera approaches the initial location. Suppose that in frame f^{t-1}, only the marker 9 is visible, and so γ^{t-1} is computed using it. Then, in the next frame, both markers 0 and 9 are visible at the same time. Because of the drift, the estimated pose of marker 9 may be far from the real one. So, if the pose of f^t is computed using only marker 9, it differs from the pose computed from marker 0. Even more, the frame pose obtained using both markers simultaneously would be incorrect too.

So, in our problem, the loop closure first affects tracking, making impossible to obtain a reliable frame pose estimation. That is the reason why, when tracking is done, not all known markers can be used (Sect. 4.3). In addition, it must be indicated that the local optimization method (Sect. 4.7) cannot be applied to solve the drift in many cases. If the drift is too large, the LM algorithm, which is a local optimization method, is unable to obtain good solutions. Therefore, it is required to do first a deeper correction, and then to employ the local optimization.

4.9.1. Detection

A loop closure is detected when, in the current frame f^t, there is a marker that is already in $\mathcal{M}$ with a valid pose, but that is not visible in any of the neighbors (in the graph $\mathcal{G}$) of the current keyframe f^ω. Let us denote by

$$\mathcal{L}(t) = \{i \in \Lambda^t \mid i \notin \mathcal{G}_m(f^\omega), \gamma_i \neq 0\}, \tag{35}$$

the set of markers in f^t that causes the loop closure detection, where $\mathcal{G}_m(f^\omega)$ (previously defined in Eq. 31) is the set of keyframes sharing markers with the keyframe f^ω.

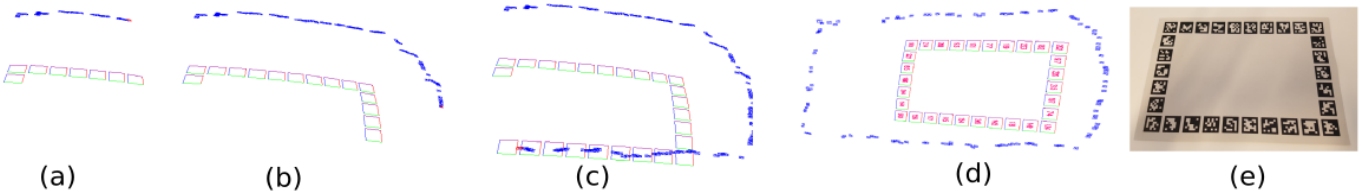

Figure 6: First test video sequence and results. The images (a-e) show the sequence in which the markers printed on a piece of paper are reconstructed. Blue squares represent the camera locations while the multicolored squares are the markers. The video sequence starts at the upper left corner of the piece of paper and moves along the border. (f) shows a photo of the printed piece of paper. (Best viewed in color)

When a loop closure is detected, two poses for f^t can be estimated: the one obtained using the markers in $\mathcal{L}(t)$, and another one using the rest of the detected markers:

$$\mathcal{L}'(t) = \Lambda^t \setminus \mathcal{L}(t). \tag{36}$$

The former is the one that we should have if there had been no drift, and we shall denote it $\dot{\gamma}^t$. The latter, is the pose obtained by tracking γ^t (Eq. 22) and is the nearest to the previous frame f^{t-1}. The goal of loop closure is to adjust the pose of all keyframes and markers so that both poses (γ^t and $\dot{\gamma}^t$) become equal.

To compute $\dot{\gamma}^t$ using the markers in $\mathcal{L}(t)$ the method explained in Sect. 4.8 is employed to have an estimation. If no reliable pose can be obtained with this method, it means that there is only one marker in $\mathcal{L}(t)$ and that its pose is ambiguously detected in f^t. Then, we assume that there are two possible solutions for $\dot{\gamma}^t$, namely $\dot{\gamma}^t_1, \dot{\gamma}^t_2$, and we will apply the correction method using both and then select as the optimal solution the one providing less error. Therefore, our method can do loop closure even with a single marker ambiguously detected.

The loop closure correction method works in two stages. First, a spanning tree containing the keyframes that form the loop is created and its poses corrected. Second, the corrected keyframe poses are employed to correct the marker locations, and then, an optimization aimed at reducing the reprojection error is done.

4.9.2. Correction

Let us assume, for the sake of clarity, that $\mathcal{L}(t)$ has only one marker m. We select from $\mathcal{F}$ the keyframe in which m was first seen f^c. Then, a minimum spanning tree $\mathcal{T}$ is created with its root at f^c using Kruskal's algorithm [31]. If the size of $\mathcal{L}(t)$ is greater than one, we select as f^c the oldest keyframe in which a marker from $\mathcal{L}(t)$ is seen.

The spanning tree $\mathcal{T}$ is used to create the pose graph $\mathcal{P} = \{V, E\}$. The nodes

$$V = \{i \mid i \in \mathcal{F}\}$$

are the keyframes included in $\mathcal{T}$, and the edges

$$E = \{(i,j) \mid i,j \in \mathcal{F}\}$$

represent connections between them. We also add f^t as a node in the graph, as well as the edges (t,c) and (t,ω) to close the loop.

Given the initial keyframe poses, stored in the map, the relative transformation $\hat{\gamma}^{j,i}$ between connected keyframes of the graph are obtained as:

$$\hat{\gamma}^{j,i} = \gamma^j \cdot (\gamma^i)^{-1}, \tag{37}$$

and stored for later use. The only exception is the edge connecting the new frame to the closing keyframe $\hat{\gamma}^{c,t}$, which is computed by using the expected pose $\dot{\gamma}^t$ instead of γ^t:

$$\hat{\gamma}^{c,t} = \gamma^c \cdot (\dot{\gamma}^t)^{-1}.$$

The value $\hat{\gamma}^{c,t}$ represents the transformation that one should obtain if there had been no drift, however the observed transformation is in fact $\gamma^c \cdot (\gamma^t)^{-1}$, so there is a difference between them:

$$\Delta^{c,t} = \hat{\gamma}^{c,t} \cdot \gamma^t \cdot (\gamma^c)^{-1}. \tag{38}$$

The basic idea is to spread this difference among all graph connections by making small adjustments in the keyframe poses. Then, the error along all the edges of the graph is expressed as:

$$\Delta(\mathcal{P}) = \sum_{(i,j)\in E} ||\hat{\gamma}^{j,i} \cdot \gamma^i \cdot (\gamma^j)^{-1}||_2^2. \tag{39}$$

The sparse LM algorithm is used to minimize Eq. 39. It is important to stress that the relative transformations $\hat{\gamma}^{j,i}$ are computed once before the optimization and remain fixed during it. However, the values γ^i are changed during optimization. Another important aspect that must be indicated is that γ^c is not subject to optimization (it remains fixed), for two reasons. First, it could generate infinite solutions since all keyframes could undergo a global transformation that does not affect the error function. Second, the keyframe f^c may have connections with other keyframes not in the pose graph $\mathcal{P}$. By freezing the pose γ^c, we ensure that the loop of keyframes optimized remains attached to the rest of map keyframes.

In contrast to [32], we do not need to introduce the scale factor into the optimization process as we know the actual size of each marker and it can be reliably estimated.

Once the poses of the frames have been corrected, marker poses must be corrected accordingly. Let us denote by $\Delta\gamma^i$ the transform applied to frame f^i to move it the corrected pose. Imagine the simplest case of the marker m to be observed only in one frame f^i. Then, the corrected pose of the marker should be

$$\gamma'_m = \Delta\gamma^i \cdot \gamma_m.$$

If the marker is observed from more than one keyframe, we can average the corrections to obtain an initial estimate of the pose:

$$\gamma'_m = \left(\frac{1}{|\mathcal{V}(m)|} \sum_{i \in \mathcal{V}(m)} \Delta\gamma^i \right) \gamma_m. \tag{40}$$

Up to this point, the corrected keymarker and frame locations can be used as a good starting solution for local optimization already explained in Sect. 4.7.

4.10. Global optimization

As the final step of our processing, when there are no more frames to be processed, we perform a global optimization. This process is similar to the local one (Sect. 4.7) with two exceptions. First, all keyframes and poses are included for optimization. Second, the camera intrinsic parameters are also included. Thus, the final global optimization consists in minimizing the function:

$$\mathbf{e}(\gamma^1, \ldots, \gamma^a, \gamma_1, \ldots, \gamma_b, \delta) = \sum_{m \in \mathcal{M}} \sum_{i \in \mathcal{F}} e^i_m(\gamma^i \cdot \gamma_m) \tag{41}$$

where $a = |\mathcal{M}|$ and $b = |\mathcal{F}|$. Again, the equation is solved using the sparse LM algorithm. The inclusion of the camera parameters in the optimization allows to obtain more precise results.

5. Experiments and results

This section explains the experiments carried out to validate our proposal using five different tests. All the tests were run on a $i7$ Intel computer with 8GB of RAM, running our code on a single thread. The ArUco library [6, 10] was employed for marker detection on the recorded video sequences and also for calibrating the employed cameras.

Two different measures have been employed to evaluate the quality of the proposed approach: the accuracy in the estimation of the marker poses, and the accuracy in the estimation of the frame poses. The first one can be evaluated by calculating Absolute Corner Error (ACE) [1], computed as the mean error between the estimated three-dimensional marker corner locations and the ground truth ones. In order to do this, it is necessary to transform the estimated corners to the ground truth reference system, which can be done using Horn's method [33]. The accuracy of the estimated frame poses

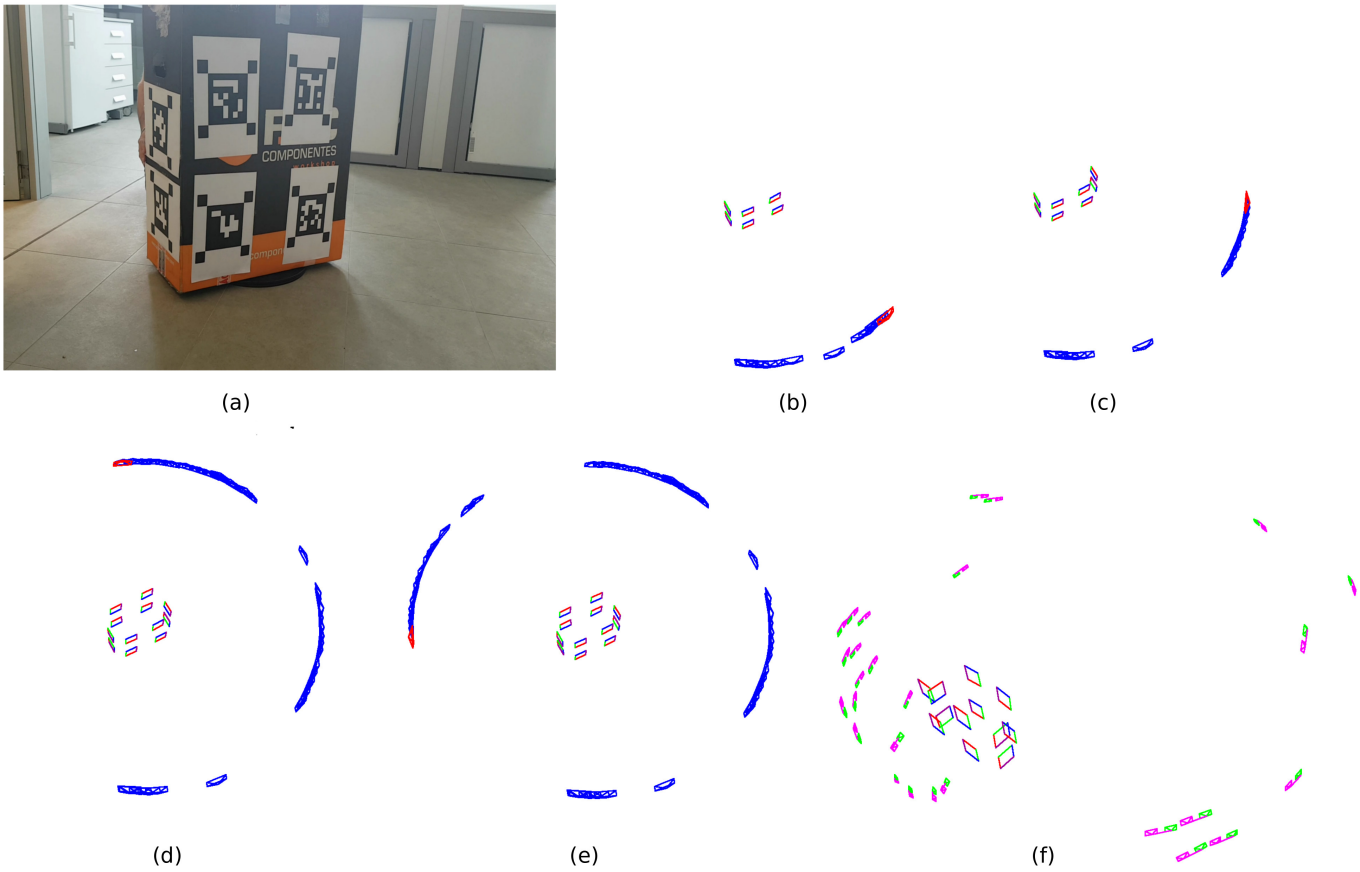

Figure 7: Calibration object test. (a) Image of the calibration object in which markers have been attached. (b-e) Images showing the SLAM process. The object is on a rotating platform. (f) Three-dimensional reconstruction of the calibration object and the positions of the stereo camera employed to evaluate the precision. Green and pink pyramids represent the position of the stereo cameras at the locations employed for testing (Best viewed in color).

is obtained using the Absolute Trajectory Error (ATE) measure, which calculates the mean error between the translation components of the frame poses estimated and the ground truth ones.

The proposed method has been compared with three other approaches. First, the off-line fiducial mapper method proposed in [1]. As already mentioned, the main difference is that the method proposed in this paper is a continuous mapping process, while [1] is an off-line process. Additionally, we have compared our system with two of the most popular SLAM methods: LSD-SLAM [4] and ORB-SLAM[4] [3]. The first method is based on semi-dense stereo matching, while the second one relies on ORB keypoints, however, both are capable of managing loop closures.

[4]We employed the latest version of the software ORB-SLAM2.

Regarding the parameters of the proposed method, we have set the value $\tau_e = 3$ (Eq. 14), which according to our experience is a good choice. This parameter is relevant in relocalization, keyframe insertion, and loop closure detection stages. The minimum possible value for this parameter is 1, but this allows ambiguous poses to be considered. On the other hand, large values involve a more conservative mode, in which only markers very reliably observed (very near) are employed. The other parameter to be considered is τ_{bl} (Sect. 4.1.2), which is the minimum distance between frames in the map. We have observed that a value of 7 mm performs well in most of our experiments.

Before continuing, we would like to indicate that the proposed system is publicly available for free usage [5].

5.1. Printed marker map

The first test is aimed at analyzing the precision of the proposed method in estimating the three-dimensional position of markers and the effectiveness of the loop closure. For that purpose, we have printed in an A4 piece of paper a set of markers as shown in Fig. 6(e). In this case, the exact location of the markers is known. To test our system, we recorded a video sequence of size 1920×1080 pixels moving around the border of the piece of paper so as to force a loop-closure. The sequence of markers reconstructed is shown in Figures 6(a-d). When analyzing the error, we obtained an ACE of 3.2×10^{-2} millimeters. The same video sequence was processed with the off-line mapper [1], obtaining a similar error of 2.9×10^{-2} millimeters.

5.2. Calibration object

In this second test, we employ the proposed system to create an arbitrary calibration object. The calibration object can be used either to calibrate a camera or to estimate its position. In this case, we used a cardboard box with markers attached on its sides. The positions of the markers are unknown and were estimated with the proposed method. To that end, the box is placed on a rotating platform and a video sequence is recorded with the camera being static looking at the box. Figure 7(a-e) shows the video sequence of the reconstruction.

In order to evaluate the accuracy of the calibration object, we used an indirect method. Instead of calculating the error of the marker positions, we calculated the error that the object induces in camera pose estimation. To do so, we employed a stereo camera consisting of two PtGrey FLEA3 units placed in parallel. In order to calibrate the cameras and estimate the displacement between them, the OpenCV toolbox [34] was employed. A chessboard pattern was captured 20 times at varying distances, and both intrinsic and extrinsic parameters of the two cameras estimated. According to the OpenCV tool, the average distance between them was 9.82 cm.

[5] `http://www.uco.es/investiga/grupos/ava/node/58`

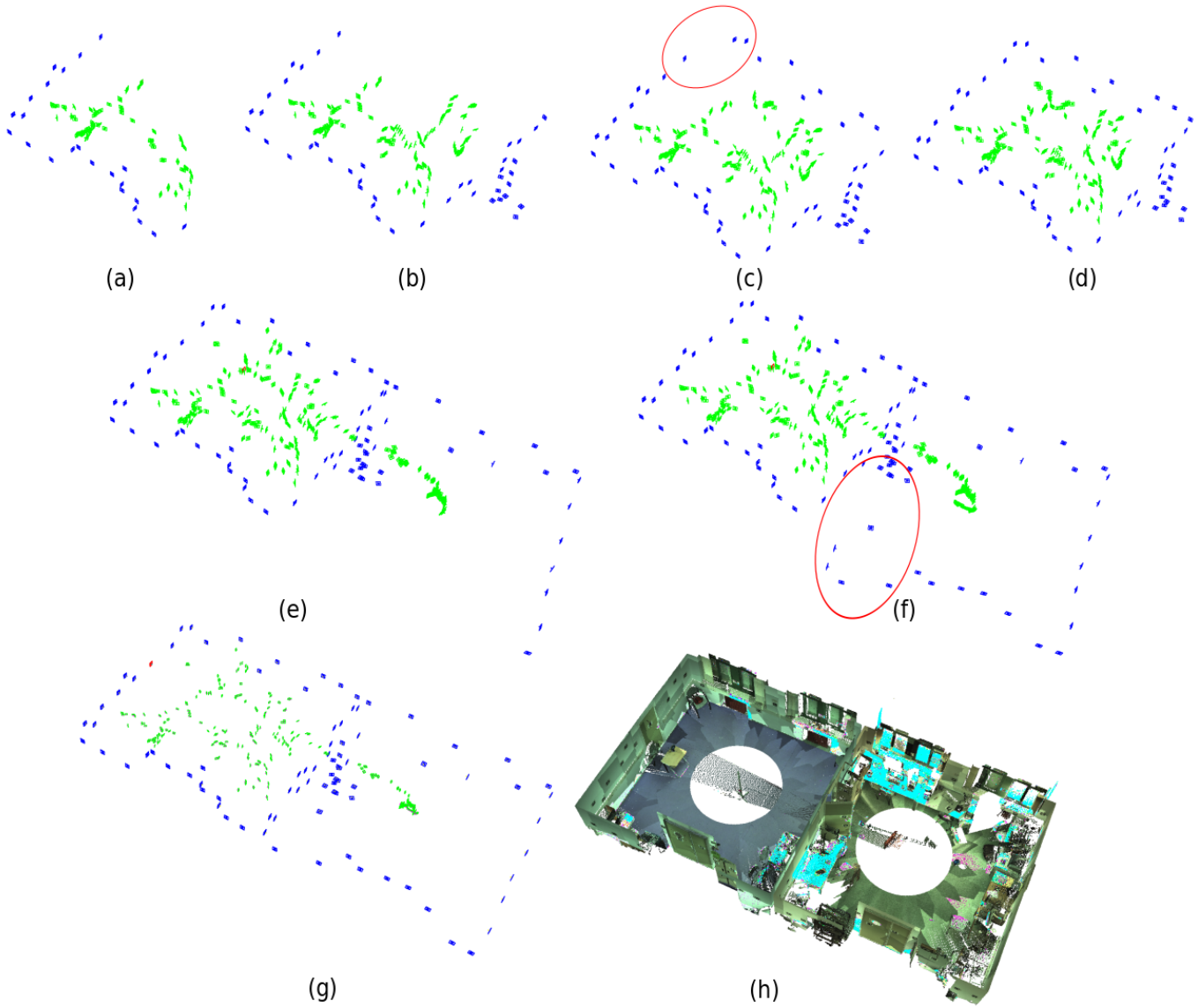

Figure 8: Full reconstruction of the laboratory. Images (a) to (g) show the status of the reconstruction in frames 4540, 6900, 8900, 9000, 12000, 12500 and 12900. Loop closure is done after frames (c) and (f). Red ellipses show the region of the map with the largest error before the loop closure. It can be observed in frames (d) and (g) that the correction of our system is successful. Image (h) shows the ground truth 3D reconstruction obtained with a Leica Laser Scanner. The average error obtained in this sequence is 0.021 meters.

Our calibration object was then used to estimate the relative position of the cameras in order to analyze the difference with respect to the OpenCV estimation. We employed the intrinsic parameters obtained by OpenCV, and only the extrinsic parameters were calculated. A total of 30 positions were used for evaluation, shown in Figure 7(f). The difference between our estimation and the OpenCV one was 1.23 ± 0.7 mm. The same experiment was done using the off-line mapper [1], obtaining a similar difference of 1.33 ± 0.9 mm.

5.3. Full lab sequence

For this test, we have placed a total of 90 markers along our laboratory, which is comprised of two rooms connected by a door. Each room has an approximated dimension of 7×7 squared meters. The goal of this test is to analyze the capability of the proposed method to reconstruct the three-dimensional position of markers placed in a large environment. The laboratory was scanned using a Leica 3D laser scanner (see Fig 8h) that provided a 3D point cloud with a very high accuracy. Then, we manually selected in the point cloud the points that belong to the corners of the markers. These are the ground truth corner locations employed to compute the ACE in this experiment.

We recorded a video sequence of 12.480 frames of size 1920×1080 pixels moving along the two rooms of the laboratory. The reconstruction sequence can be observed in Figs. 8(a-g) where the blue squares represent the markers placed on the walls $\mathcal{M}$, and the green squares the keyframe $\mathcal{F}$ stored in the map. The sequence starts recording the left room, moving around it until it is completely observed. In Fig. 8(c) one can observe that a loop is about to be closed. We have marked with a red ellipse the region of the closure. Then, Fig. 8(d) shows the result after the loop closure correction. Once the left room is completely observed, we cross the door and place the camera in the center of the right room Fig. 8(e). Then, we move the camera until all markers in the room are spotted. Again, we have marked in Fig. 8(f) the region affected by a large drift that is corrected using the loop closure method explained. The final reconstruction obtained is shown in Fig. 8(g).

This sequence is especially interesting since we have ground-truth information and thus we can evaluate the different parameters of our proposal.

The main parameters we are interested on are: the number of frames per marker kept in the map $N_c = |\Omega^m|$ (Eq. 28), and the image size. Both parameters influence the computing speed and the reconstruction error. Table 1 shows the different evaluations conducted. We have analyzed the video sequence for all combinations of the parameter $N_c = \{5, 10, 15\}$ with the image sizes $\{640 \times 480, 800 \times 600, 1920 \times 1080\}$. The column FPS[1] indicates the frames per second of our method when the time required to detect markers is not considered. In other words, it only considers the

Table 1: Metrics computed in the two-labs sequence. We tested our system using several values for the parameter $N_c = |\Omega^m|$ and the input image sizes. FPS[1] indicates the frames per second of our proposal discounting the time required for detecting the markers. FPS[2] represents the total frames per second including marker detection.

N_c	Image Size	FPS[1]	FPS[2]	ACE
5	640 × 480	576	236	0.034
5	800 × 600	518	189	0.023
5	1920 × 1080	412	71	0.023
10	640 × 480	253	155	0.021
10	800 × 600	251	136	0.023
10	1920 × 1080	221	61	0.023
15	640 × 480	148	108	0.021
15	800 × 600	150	101	0.021
15	1920 × 1080	140	53	0.022

amount of time required by our proposal. Column FPS[2] shows the total computing time considering the detection of the markers using the ArUco library.

As can be observed, apart from the first case, the rest of the tests obtained a very similar ACE. It is true that the lowest resolutions seem to obtain slightly better precisions in most cases. We believe that the margin of error in the ground truth estimation is in the range of millimeters and the error differences observed in these cases cannot be considered of relevance. Another possibility is that ArUco corner estimation performs better in low-resolution images.

In any case, we consider that the second row is probably the best trade-off between speed and precision since the full system can run at nearly 200 FPS using a single thread.

5.4. Comparison with other SLAM approaches

In the following experiments we evaluate the proposed method with the keypoints-based methods ORB-SLAM2 [3] and LSD-SLAM [4], along with the previously OffLine proposed method [1]. Eight video sequences were recorded in one of the rooms of our lab in which an Optitrack motion capture system is installed. It is comprised of six cameras and is able to track the position of an object at 100 Hz. The videos were recorded using a PtGrey FLEA3 camera, which captures images of 1920 × 1080 pixels at a frame rate of 60 Hz.

Three of the video sequences (Seq1, Seq2, and Seq3) point towards the wall and the camera starts and end at the same

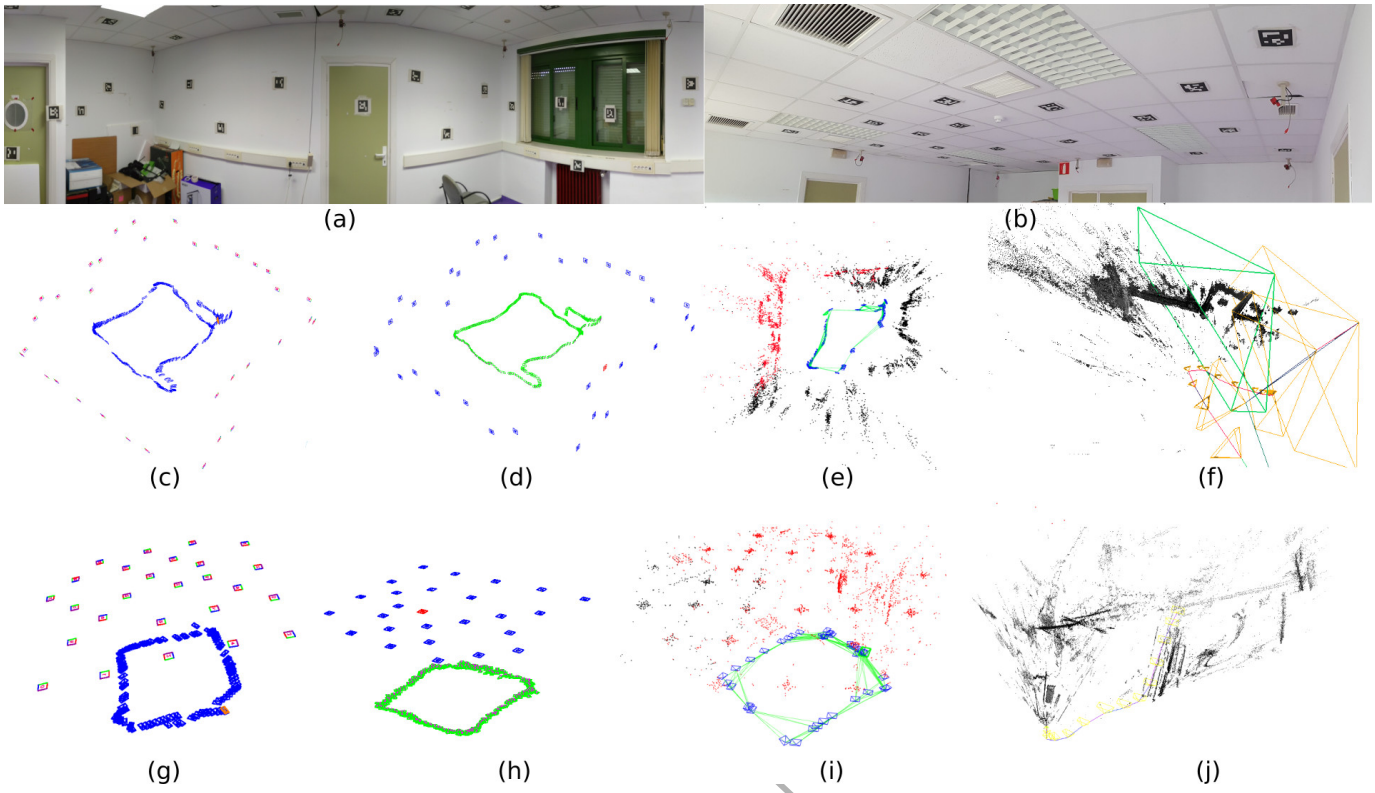

Figure 9: a) Panoramic view of the lab walls showing the markers on the wall. b) Panoramic view of the markers placed in the ceiling. c) Reconstruction obtained by our method for Seq1. Keyframes presented in blue and markers as rectangles. d) Results obtained by the Offline method [1] in Seq1. e) Results of the ORB-SLAM2 method for Seq1. The method fails at some point of the trajectory and is not able to properly recover the structure of the scene even when loop closure is detected. f) Results of LSD-SLAM for Seq1. The method completely fails to reconstruct the trajectory. g,h,i,j) Reconstruction results of the methods for Seq4, with markers in the ceiling. In that case, ORB-SLAM2 obtains better results than in the previous environment.

location, in order to facilitate the detection of the loop closure. In these sequences, the markers placed on walls are used for tracking (Fig. 9a). The sequences (Seq4, Seq5, and Seq6) shows another interesting use case in which the markers are placed in the ceiling (Fig. 9b). As previously, the sequence starts and ends at the same location to facilitate the detection of the loop closure. Finally, in the sequences Seq7 and Seq8, the camera is moved around pointing at the walls, and the sequences finish looking at the same region of the room where the sequence starts so that the closure of the loop could be detected. Nonetheless, although pointing at the same spot, the initial and final locations were separated by two meters approximately. In the first frames of the sequences, translational movements are performed so as to allow ORB-SLAM2 and LSD-SLAM methods to initialize.

For the ORB-SLAM2 and LSD-SLAM methods, the image resolution was reduced to 640×480 as usually for these methods. Since these SLAM methods run several threads simultaneously, their results may vary from one execution to

Table 2: Results obtained by the different tested methods for the eight sequences registered with a motion capture system. Absolute Trajectory Error is measured in centimeters. t_{rel} is a relative measure, and r_{rel} is (deg/25cm)

Seq.	SPM-SLAM			OffLine [1]			LSD-SLAM [4]			ORB-SLAM2 [3]		
	ATE	t_{rel}	r_{rel}	**ATE**	t_{rel}	r_{rel}	**ATE**	t_{rel}	r_{rel}	**ATE**	t_{rel}	r_{rel}
Seq 1	5.68	1.42	0.20	7.19	1.50	0.21	133.78	3.63	2.09	44.4	1.10	0.40
Seq 2	4.71	1.60	0.83	4.36	1.66	0.88	72.88	1.02	2.19	38.1	1.10	0.88
Seq 3	5.90	1.52	1.21	31.05	2.29	1.45	226.16	1.90	2.32	76.2	1.06	1.36
Seq 4	1.64	0.87	1.22	5.28	0.92	1.23	235.65	1.12	2.16	1.89	0.79	1.20
Seq 5	1.78	1.24	1.42	2.08	1.36	1.60	236.42	1.18	2.08	61.97	1.30	1.17
Seq 6	1.64	1.34	1.61	1.52	1.47	1.67	25.03	1.01	1.87	110.00	1.40	2.27
Seq 7	4.97	1.35	1.05	4.82	1.39	1.17	50.88	0.94	1.57	123.91	0.89	1.20
Seq 8	6.54	1.15	1.23	87.63	4.15	2.34	94.48	3.46	2.30	12.43	0.81	1.25

another. LSD-SLAM allows avoiding that problem by setting a parameter that forces a frame to be procesed before the next one is employed. To overcome the problem in ORB-SLAM2, we feed the images to the system with enough delay so that all optimization processes can be finished before the next image is given to the system. This is achieved by sleeping the camera thread 400 ms between the capture of each image.

The quantitative results obtained by the different methods are shown in Table 2. For each method we have computed, the absolute trajectory error (ATE), the average relative translation error t_{rel} and the relative rotational error r_{rel}, as proposed in [35], at the distance of 25 cm. As it can be observed, the proposed method obtain good results in all sequences. The worst results are obtained by the LSD-SLAM method, which fails in all sequences. We believe that the reason is that the rolling-shutter camera employed. The ORB-SLAM2 method systematically fails in all sequences but Seq4. The OffLine method [1] previously proposed fails in two of the sequences (Seq3, and Seq8). Finally, it is also worth noticing that the tracking results for the sequences in which the markers are in the ceiling obtain higher accuracy.

Figures 9(c-f) shows the results of the different methods for Seq1. The video sequences, the results obtained, code and scripts to reproduce the experiments of this section are publicly available [6].

5.5. Rotational movement

This final test aims at analyzing the capability of our method to work under mostly rotational movements. It is indeed a problematic case for monocular visual SLAM methods since some degree of translation is required to reliably compute

[6] http://www.uco.es/investiga/grupos/ava/node/58

the essential or homography matrices. To test the system, we recorded a video sequence in the first room of the lab using a camera of resolution 1280×960 pixels. The camera was set in the center of the room and a person was spinning while pointing at the walls with the camera. The reconstruction obtained is shown in Figure 1. In order to reconstruct the scene, we set the minimum baseline between frames $\tau_{bl} = 0.03$ m. Otherwise, very few frames are added to the map. The reconstruction ACE in this test is 3.1 cm. When the sequence was processed with the ORB-SLAM and LSD-SLAM methods, they were unable to initialize. The same sequence on with the method proposed in [1] which obtained an ACE of 2.9 cm. It must be noticed, however, that this is an off-line method and that the number of frames employed for optimization is not restricted as in our case. This is the reason why it outperforms the method proposed here in most cases.

6. Conclusions and future work

This work has proposed a system to simultaneous map and localizing from a set of squared planar markers (SPM-SLAM). Using as input a video sequence showing markers placed at arbitrary locations of the environment, the proposed system incrementally builds a precise map of the markers with correct knowledge of the scale and determines at the same time the camera pose. The proposed system allows to create cost-effective and large-scale tracking systems able to operate in real-time with minimal computational requirements.

This paper has contributed proposing a number of techniques to deal with the inherent ambiguity that occurs when using planar markers. First, we have proposed a method to initialize the map from a set of images with ambiguous markers only. Second, we have proposed a method to estimate the pose of markers from ambiguous observation given that the frame poses are known. Finally, we have proposed a method to detect and solve the loop closure problem from squared planar markers.

The experiments conducted have shown the effectiveness of the proposed system testing it in several situations. In the most complex scenario, our laboratory, two rooms of 49 squared meters each, have been properly mapped. The speed of our system reaches 150 Hz using a single thread, so it is especially attractive as an effective localization system in many robotic tasks requiring low computational resources. Additionally, the proposed system also removes the restriction of monocular SLAM systems that fail under rotational movements.

The proposed system is publicly available[7] for free usage. As future work, our plan is to mix natural features with

[7] http://www.uco.es/investiga/grupos/ava/node/58

fiducial markers. The basic idea is to be able to use the markers whenever they are seen but to avoid an excessive amount of them by also using keypoints. Another possible future work is to develop a model that analytically describes the type of errors present in squared planar tracking in order to assists placing the markers.

Acknowledgment

This project has been funded under projects TIN2016-75279-P and IFI16/00033 (ISCIII) of Spain Ministry of Economy, Industry, and Competitiveness, and FEDER.

References

[1] R. Muñoz-Salinas, M. J. Marín-Jimenez, E. Yeguas-Bolivar, R. Medina-Carnicer, Mapping and localization from planar markers, Pattern Recognition 73 (Supplement C) (2018) 158 – 171.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Transactions on Robotics 32 (6) (2016) 1309–1332.

[3] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, Orb-slam: A versatile and accurate monocular slam system, IEEE Transactions on Robotics 31 (5) (2015) 1147–1163.

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

[5] D. Galvez-López, J. D. Tardos, Bags of binary words for fast place recognition in image sequences, IEEE Transactions on Robotics 28 (5) (2012) 1188–1197.

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, M. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition 47 (6) (2014) 2280 – 2292.

[7] M. Fiala, Designing highly reliable fiducial markers, IEEE Trans. Pattern Anal. Mach. Intell. 32 (7) (2010) 1317–1324.

[8] H. Kato, M. Billinghurst, Marker tracking and hmd calibration for a video-based augmented reality conferencing system, in: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, IEEE Computer Society, 1999, pp. 85–94.

[9] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, W. Purgathofer, The studierstube augmented reality project, Presence: Teleoper. Virtual Environ. 11 (1) (2002) 33–54.

[10] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, R. Medina-Carnicer, Generation of fiducial marker dictionaries using mixed integer linear programming, Pattern Recognition 51 (2016) 481 – 491.

[11] M. Fiala, Comparing ARTag and ARToolKit Plus fiducial marker systems, in: IEEE International Workshop on Haptic Audio Visual Environments and their Applications, 2005, pp. 147–152.

[12] E. Olson, AprilTag: A robust and flexible visual fiducial system, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 3400–3407.

[13] J. Wang, E. Olson, Apriltag 2: Efficient and robust fiducial detection, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4193–4198. doi:10.1109/IROS.2016.7759617.

[14] J. DeGol, T. Bretl, D. Hoiem, Chromatag: A colored marker and fast detection algorithm, in: ICCV, 2017.

[15] D. Oberkampf, D. F. DeMenthon, L. S. Davis, Iterative pose estimation using coplanar feature points, Computer Vision and Image Understanding 63 (3) (1996) 495 – 511.

[16] G. Schweighofer, A. Pinz, Robust pose estimation from a planar target, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (12) (2006) 2024–2030.

[17] T. Collins, A. Bartoli, Infinitesimal plane-based pose estimation, International Journal of Computer Vision 109 (3) (2014) 252–286.

[18] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, in: Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on, 2007, pp. 225–234.

[19] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, in: 2011 International Conference on Computer Vision, 2011, pp. 2564–2571.

[20] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, Monoslam: Real-time single camera slam, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6) (2007) 1052–1067.

[21] H. Lim, Y. S. Lee, Real-time single camera slam using fiducial markers, in: ICCAS-SICE, 2009, 2009, pp. 177–182.

[22] T. Yamada, T. Yairi, S. H. Bener, K. Machida, A study on slam for indoor blimp with visual markers, in: ICCAS-SICE, 2009, IEEE, 2009, pp. 647–652.

[23] M. Klopschitz, D. Schmalstieg, Automatic reconstruction of widearea fiducial marker models, in: In ISMAR, IEEE Computer Society, 2007, pp. 1–4.

[24] K. Shaya, A. Mavrinac, J. L. A. Herrera, X. Chen, A self-localization system with global error reduction and online map-building capabilities, in: Intelligent Robotics and Applications, Springer, 2012, pp. 13–22.

[25] M. Neunert, M. Blösch, J. Buchli, An open source, fiducial based, visual-inertial state estimation system, arXiv preprint arXiv:1507.02081.

[26] K. Madsen, H. B. Nielsen, O. Tingleff, Methods for non-linear least squares problems (2nd ed.) (2004).

[27] J. Sivic, A. Zisserman, Efficient visual search of videos cast as text retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (4) (2009) 591–606.

[28] Y. Feng, Y. Wu, L. Fan, Real-time SLAM relocalization with online learning of binary feature indexing, Machine Vision and Applications 28 (8) (2017) 953–963.

[29] Y. Feng, L. Fan, Y. Wu, Fast localization in large-scale environments using supervised indexing of binary features, IEEE Transactions on Image Processing 25 (1) (2016) 343–358.

[30] L. Liu, H. Li, Y. Dai, Efficient global 2d-3d matching for camera localization in a large-scale 3d map, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2391–2400.

[31] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings of the American Mathematical Society 7 (1) (1956) 48–50.

[32] H. Strasdat, J. Montiel, A. J. Davison, Scale drift-aware large scale monocular slam, Robotics: Science and Systems VI.

[33] B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, J. Opt. Soc. Am. A 4 (4) (1987) 629–642.

[34] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision in C++ with the OpenCV Library, 2nd Edition, O'Reilly Media, Inc., 2013.

[35] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

R. Muñoz–Salinas received the Bachelor degree in Computer Science from the University of Granada (Spain) and the Ph.D. degree from the University of Granada in 2006. Since then, he has been working with the Department of Computing and Numerical Analysis of Cordoba University, currently he is a lecturer. His research is focused mainly on Computer Vision, Soft Computing techniques applied to Robotics and Human–Robot Interaction.

M. J. Marín–Jiménez received his B.Sc., M.Sc. degrees from the University of Granada, Spain, in 2003, and Ph.D. degree from the University of Granada, Spain in 2010. He has worked, as a visiting student, at the Computer Vision Center of Barcelona (Spain), Vislab-ISR/IST of Lisboa (Portugal) and the Visual Geometry Group of Oxford (UK). Currently, he works as an assistant professor at the University of Cordoba (Spain). His research interests include object detection, human-centric video understanding and machine learning.

R. Medina-Carnicer received the Bachelor degree in Mathematics from University of Sevilla (Spain) and the Ph.D. in Computer Science from the Polytechnic University of Madrid (Spain) in 1992. Since 1993 he has been a lecturer of Computer Vision at Cordoba University (Spain). His research is focused on edge detection, evaluation of computer vision algorithms, 3-D vision and pattern recognition.