CI-Graph Simultaneous Localization and Mapping for Three-Dimensional Reconstruction of Large and Complex Environments Using a Multicamera System

Pedro Piniés, Lina María Paz, Dorian Gálvez-López, and Juan D. Tardós
Instituto de Investigación en Ingeniería de Aragón I3A, University of Zaragoza, María de Luna 1, 50018, Zaragoza, Spain
e-mail: ppinies@unizar.es, linapaz@unizar.es, dorian@unizar.es, tardos@unizar.es

Received 21 October 2009; accepted 3 June 2010

Submapping and graphical methods have been shown to be valuable approaches to simultaneous localization and mapping (SLAM), providing significant advantages over the classical extended Kalman filter (EKF) solution: they are faster and, when using *local coordinates*, produce more consistent estimates. The main contribution of this paper is *CI-Graph SLAM*, a novel algorithm that is able to efficiently map large environments by building a graph of submaps and a spanning tree of this graph with the following properties: (1) any pair of neighboring submaps in the spanning tree are *conditionally independent* and (2) the current submap is always up to date, containing the marginal probabilities of the submap variables given all previous measurements. Thanks to these properties, an old submap can be updated at any time by performing a single propagation from the current map to the old submap along the spanning tree. This operation is required only when a map is revisited, with a cost linear with the number of maps in the loop. At the end of the experiment the method performs a single propagation through the whole tree, recovering exactly the same marginals for all the map variables as the EKF–SLAM algorithm does, without ever needing to compute the whole covariance matrix. To evaluate CI-Graph performance in extremely loopy environments, the method was tested using a synthetic Manhattan world. The behavior of the algorithm in large real environments is shown using the public data sets from the RAWSEEDS project in which a robot equipped with a trinocular camera traversed indoor and outdoor environments with several loops and revisited areas. Loops are robustly closed using a novel technique that detects candidate loop closures using a visual vocabulary tree and filters them using temporal and geometric constraints. Our experiments show that when using frontal cameras, the technique outperforms FAB-MAP. The epipolar geometry of the loop-closing images is used to find feature matches that are imposed on the CI-Graph to correct the submap estimations along the loop.

1. INTRODUCTION

The goal of the current simultaneous localization and mapping (SLAM) research is to develop very efficient algorithms that will allow a robot to operate in increasingly bigger environments without losing accuracy. At the same time, vision sensors are gaining even more importance in present-day applications due to the richness of information they can provide and their attractive low cost.

To perform SLAM in large environments, submapping algorithms divide the whole map into groups of state vector variables (features and/or vehicle poses) that are processed separately. Previous submapping techniques are based on building local maps of limited size that are statistically *independent* (Huang, Wang, & Dissanayake, 2008; Paz, Tardós, & Neira, 2008b; Tardós, Neira, Newman, & Leonard, 2002; Williams, Dissanayake, & Durrant-Whyte, 2002). This requirement imposes important constraints to the submap structure. Valuable information present in a submap cannot be used to improve other submap estimates because otherwise the independence property could not be preserved. In addition, environment features observed in different maps have independent estimations in each map.

Instead of using independent submaps, our *CI-Graph* SLAM approach is based on building *conditionally independent* submaps (CI submaps) as proposed in Piniés and Tardós (2008). In that work we showed that CI submaps can share submap components and information and the whole map posterior can be obtained in a consistent manner by a single backpropagation of information from the current submap to the previous submaps. Although the technique was demonstrated in real environments, it was restricted to sequences of maps forming simple topologies such as single loops. The generalization to more complex topologies in which the CI property between maps still holds is not trivial.

The main contribution of this paper is *CI-Graph SLAM*, a novel algorithm that is able to efficiently map large environments with any topology by building a graph of submaps in which an edge between two submaps represents that they share some information. The core of the

algorithm consists of building a spanning tree of the graph where these two properties are guaranteed:

1. Any pair of neighboring submaps in the spanning tree are *conditionally independent*. We show that in the presence of loops in the graph, this can be achieved by including a copy of the variables that form the loop (the reobserved features or the robot pose) in all the submaps along the loop.
2. The submap where the robot is currently located is always maintained up to date, containing the marginal probabilities of the submap variables given all previous measurements. During exploration, this property is guaranteed by construction. When the robot revisits a map, the old map is updated before the robot switches to it.

Thanks to these two properties, an old submap can be updated at any time by performing a single propagation from the current map to the old submap along the path that joins them in the spanning tree, with a cost that is linear in the number of submaps in the path. In practice, this operation is required only when a map is revisited to guarantee that the second property still holds. To recover the marginals for all the map variables, it is enough to perform a single propagation though the whole spanning tree.

It is important to note that the method of decoupling the whole map in CI submaps and maintaining the spanning tree is general, not requiring any assumption about the underlying distributions or the estimation method used. In our implementation we perform the classical linear–Gaussian approximation and use extended Kalman filter (EKF)–SLAM for computing the submaps. As a result, our CI-Graph implementation when using absolute coordinates obtains exactly the same linear–Gaussian approximation of the marginals for all the map variables as the classical single-map EKF–SLAM algorithm does, without ever needing to compute the whole covariance matrix.

Given the importance of visual sensors, cameras were used during the experimental validation of the technique. To allow the system to work with an arbitrary number of cameras, we have designed an algorithm that treats each camera independently. The only information required is the intrinsic and extrinsic calibration of the camera array. In particular, the experiments were carried out in the context of the RAWSEEDS project with a trinocular camera mounted on a robot (RAWSEEDS, 2009). To detect that the robot is revisiting a previous mapped area, we have developed an appearance-based loop-closing algorithm based on a visual vocabulary tree (Nister & Stewenius, 2006). To avoid false positives, the candidate loops are filtered using a combination of temporal and geometric consistency checks. We have compared our approach with the FAB-MAP (Cummins & Newman, 2008) implementation made available by the authors, obtaining superior performance. We believe that the RAWSEEDS data sets are particularly challenging for FAB-MAP because of the use of forward-looking cameras instead of panoramic or lateral-looking cameras.

In Section 2 we give a summary of the related work. Section 3 is devoted to explain the CI-Graph SLAM algorithm, reviewing first the theory on which CI submaps are based. Section 4 shows a simulation experiment in a Manhattan-like world that allows us to test the computational properties of the algorithm in complex topologies. In Section 5 we present the algorithm implemented to work with an arbitrary number of cameras. Section 6 explains the appearance-based method used to detect loop closures and revisited areas. Section 7 shows the results obtained in real experiments using the trinocular system. Finally, in Section 8 we draw the conclusions and propose future work. A preliminary version of CI-Graph building two-dimensional (2D) maps from laser scans was presented in Piniés, Paz, and Tardós (2009). In this paper, apart from a more detailed presentation of the method and its properties, we demonstrate the technique performing three-dimensional (3D) visual SLAM in real large environments with multiple loops. We have added an algorithm that allows us to easily include any number of cameras in the system and a robust loop-closing algorithm based on visual appearance.

2. RELATED WORK

CI-Graph is a submapping algorithm designed to work efficiently in *large environments* without making any approximation besides the inherent EKF linearizations. Nevertheless, the optimistic effect of linearizations in the estimation of the map covariance (Castellanos, Martinez-Cantin, Tardós, & Neira, 2007; Julier & Uhlmann, 2001) is greatly reduced by representing each submap with respect to its own local reference. Although CI-Graph can deal with any kind of sensors, the union of CI-Graph with vision devices turns out to be natural and powerful for two reasons. First, sensors with partial observability require the integration of measurements taken from different robot poses to obtain an accurate estimation of a feature. Thus, the ability of the algorithm to share features and state components between maps makes it especially suitable to be applied in *visual SLAM*. Second, visual sensors provide very rich visual information that is extremely useful to recognize previously visited areas after long traversals of the environment. Using this information, loop-closing constraints can be applied to the system, increasing map accuracy. In the following sections we give a brief review of these three main topics: large-scale SLAM, visual SLAM, and relocation algorithms.

2.1. Large-Scale SLAM

Essential tasks in mobile robotics strongly rely not only on a precise estimation of the robot location but also on an accurate map estimate of the surrounding environment. SLAM

algorithms confront both problems in a single estimation process. The first consistent solution proposed was based on the EKF (Durrant-Whyte & Bailey, 2006; Smith, Self, & Cheeseman, 1988). However, a standard implementation of the algorithm suffers from memory and time complexities of $O(n^2)$ per step, where n is the total number of features stored in the map. To reduce the computational cost, new algorithms take advantage of the fact that SLAM is a sparse problem, i.e., from a given robot position only a limited number of features is visible. If all features were always visible, then no algorithm could overcome the computational complexity of the EKF solution because the linearized system to be solved would be full.

Submapping strategies have become interesting approaches because they work in small regions of the environment, reducing the computational cost of EKF and improving consistency. Under the assumption of white noise and if no information is shared between maps, submaps are statistically independent. This allows submaps to be consistently joined using the map joining algorithm (Tardós et al., 2002) or the equivalent constrained local submap filter (CLSF) (Williams et al., 2002) with joining cost $O(n^2)$. More recently, divide and conquer SLAM (Paz et al., 2008b) has been shown to provide a more efficient strategy to join local maps with amortized linear cost in exploration, outperforming past sequential methods. Despite its scalability, the main limitations of these techniques are their inability to share information between maps and a memory cost of $O(n^2)$.

There are submapping techniques that work on approximations trading off precision for complexity properties (Leonard & Feder, 2000). Some of these techniques combine submaps with a graph structure that represents adjacency relations between maps. In Atlas (Bosse, Newman, Leonard, Soika, Feiten, et al., 2003), and CTS (Leonard & Newman, 2003), for example, nodes of the graph correspond to submaps and links between nodes represent relative locations between adjacent submaps. However, in order to achieve high efficiency, they do not impose loop constraints to update the graph estimation. Hierarchical SLAM (Estrada, Neira, & Tardós, 2005) improves these approaches by introducing an optimization step along the cycles of the graph. Nevertheless, it remains an approximate algorithm because the optimized information is not transmitted back to the submaps.

In contrast to EKF-based approaches, there is a family of algorithms that considers the full SLAM problem in a smoothing and mapping (SAM) sense. Graph SLAM and square root SLAM (Dellaert & Kaess, 2006; Thrun, Burgard, & Fox, 2005) showed that the intrinsic structure of the problem can be modeled as a *sparse graph* (obtained from the sparse information matrix) when the state vector is augmented with the total trajectory. The sparse structure of these algorithms is based on the *conditional independence* of the features given the whole trajectory (Thrun et al., 2005) that in the case of an information form representation of the state produces a sparse information matrix (Eustice, Singh, & Leonard, 2006).

The main problem of these techniques is that their complexity continuously grows with the number of robot poses, even when the robot is revisiting a previously mapped area. Iterative algorithms (Kaess, 2008) have been developed to reduce this complexity.

Based on EKF, graphical SLAM (Folkesson & Christensen, 2006) builds a compressed graph of all robot and feature poses as nodes. However, special cases such as loop closings need particular manipulations. Treemap (Frese, 2006) is based on creating a *balanced binary tree* structure of the map. The technique uses a detailed graph granularity to construct the tree, in which leaf nodes represent each map entity (current robot and feature locations), resulting in a very complete but complex graph algorithm.

In this work we are interested in methods that do not use any approximations. We consider the SLAM problem as a Gaussian graph model that evolves over time. We propose *CI-Graph* SLAM based on Piniés and Tardós (2008), a submapping method that performs EKF updates, efficiently reducing its quadratic cost. Unlike other nonapproximated submapping approaches (Tardós et al., 2002; Williams et al., 2002), CI-Graph SLAM builds a spanning tree of *conditionally independent* submaps that allows us to transmit information between submaps in a consistent manner. Compared to batch algorithms (Dellaert & Kaess, 2006; Thrun et al., 2005), CI-Graph does not need to augment the state vector with the full trajectory. Instead, only robot poses corresponding to map transitions are considered. Graphically, we could say that the reduction in the computational cost in batch algorithms is based on conditioning *along* the whole robot trajectory, whereas in our algorithm the cost is reduced by conditioning *across* the border between neighboring maps. Compared to Treemap (Frese, 2006) or Thin Junction Tree (Paskin, 2003), in CI-Graph the nodes do not represent each element of the map but the CI submaps. This results in a smaller graph with a high level of abstraction of the map.

2.2. Visual SLAM

Visual SLAM research has been focused on the use of either monocular or stereo vision to obtain 3D information from the environment, usually represented as a sparse set of interest points. This visual framework allows the estimation of the trajectory of the camera in six degrees of freedom (DOF). Several monocular SLAM systems have been proposed, obtaining good results for small environments (Bailey, 2003; Davison, Reid, Molton, & Stasse, 2007; Deans & Hebert, 2000; Fitzgibbons & Nebot, 2002; Gil, Reinoso, Martínez-Mozos, Stachniss, & Burgard, 2006; Jensfelt, Kragic, Folkesson, & Bjorkman, 2006; Kwok & Dissanayake, 2004; Lemaire, Lacroix, & Sola, 2005; Sola,

Monin, Devy, & Lemaire, 2005). One of the main challenges when working with monocular systems is how to initialize a feature given the partiality of the bearing-only information provided by a single camera. The work done in Civera, Davison, and Montiel (2008) represents a remarkable solution to this problem with the introduction of an inverse depth (ID) parameterization to represent points. In this way, the information provided by a feature can be incorporated into the estimation from the first image in which the feature is observed. To compute an estimate, most of the previous systems are based on the implementation of a Bayesian filter and in particular on the EKF algorithm. A different and very interesting solution for monocular SLAM is presented in Klein and Murray (2007), where the authors adopt a more classical computer vision approach using as estimator engine a recursive structure from motion (SfM) algorithm. Under this scheme, the SLAM system is split into two separate threads. The first thread tracks the pose of the camera relative to the map, as if this map were certain, and the second thread creates a map with a subset of the frames (called keyframes). Features are included in the map using triangulation between two keyframes. To obtain the map, feature positions and keyframe poses are jointly optimized using bundle adjustment. This technique uses thousands of interest point correspondences between keyframes, obtaining highly accurate solutions of medium-size loops (around 150 keyframes). A recent version of the algorithm uses as well small line edges as features to improve camera tracking during fast motion (Klein & Murray, 2008).

The main advantage of stereo visual systems over monocular cameras is that the former can provide the scale of the estimation when the baseline between the cameras is known and points close to the camera are observed. Davison and Murray demonstrated the first active stereo visual SLAM system (Davison, 2003; Davison & Murray, 2002). More stereo SLAM implementations for small environments can be found in Hygounenc, Jung, Soueres, and Lacroix (2004), Iocchi, Konolige, and Bajracharya (2000), Jung and Lacroix (2003), Saez, Escolano, and Penalver (2005), and Se, Lowe, and Little (2002). Stereo vision can be greatly improved by combining monocular and stereo information, as is pointed out in Paz, Piniés, Tardós, and Neira (2008a) and Sola, Monin, Devy, and Vidal-Calleja (2008). Points far from the camera and parameterized using ID are useful to obtain accurate angular information, whereas close points provide translation and scale information through the stereo baseline, avoiding "scale-drift."

Visual SLAM in large environments has recently been tackled in different works (Clemente, Davison, Reid, Neira, & Tardós, 2007; Eade & Drummond, 2007; Konolige & Agrawal, 2008; Paz et al., 2008a; Piniés & Tardós, 2008), most of them based on submapping techniques in order to reduce the computational cost.

In Clemente et al. (2007), a hand-held monocular camera is used to build a set of statistically independent submaps. When a loop closure is detected, the relative position of the maps in the loop is optimized using the hierarchical SLAM framework. Because submaps are independent and the scale is not observable, each map converges to a different scale, introducing an extra difficulty in the loop-closing procedure. In addition, common features between adjacent submaps are duplicated and suboptimally estimated because their mutual information cannot be merged.

In Konolige and Agrawal (2008), following a spirit similar to that of bundle adjustment, the authors solve a reduced nonlinear system of camera pose constraints obtained from the actual large system of stereo observations. The system reduction is based on two operations. First, features are marginalized out from the system and their information is spread out among the corresponding camera frames, increasing their linkage. Second, as frames accumulate over extended trajectories, increasing complexity, the size of the system is again reduced using marginalization, keeping only a selected subset of frames called the *skeleton*. However, it is not clear how this subset is selected and what the influence is of the loss of information due to the marginalization. Using this technique, large loop closures can be solved quickly. Similarly, the works of Mei, Sibley, Cummins, Newman, and Reid (2009) and Sibley, Mei, Reid, and Newman (2009) relies on an adaptive bundle adjustment method in which poses and features are represented in relative references using a graph model. During loop closure events, the graph is traversed in a breadth-first search to look for nodes that should be adjusted. Nodes are traversed until the reprojection error reaches an imposed threshold. This approximation defines an active region that allows the method to achieve a solution in constant time with a map representation in relative coordinates. A questionable point remains in the representation of the total solution, which requires transformation of the relative representation into a global Euclidean reference with a cost that is far from being constant.

In Eade and Drummond (2007), a graph of statistically independent submaps is built using a monocular camera. As in Clemente et al. (2007), each submap converges to its own scale, but this situation is explicitly managed in the algorithm by working with similarity transformations instead of rigid transformations between maps. Similarly to Klein and Murray (2007), the camera pose is not estimated as part of the local state but is tracked independently using the map estimate. The uncertainty of each local map is stored in information form. Common features between adjacent maps are used to locally constrain the relative positions of the maps, whereas graphs cycles (loop closures) generate a second type of topological constraints. By combining local and topological constraints in a total cost function defined by the authors, the graph structure is optimized using a preconditioned gradient descent algorithm.

Although global edges in the optimization are not statistically independent, they are treated as such by the algorithm, producing, according to the authors, conservative results.

Finally, Paz et al. (2008a) and Piniés and Tardós (2008) present previous work with monocular and stereo systems in large environments. The estimation algorithm in these papers is based on conditionally independent submaps (CI submaps). The main advantage of CI submaps over independent ones is that common information between maps can be consistently shared and transmitted, taking into account all the information available about a feature. However, complex camera trajectories with several intersecting loops are not easily tackled by the algorithm. The work presented in the current paper overcomes this problem, extending the use of CI submaps to arbitrary map topologies.

2.3. Loop Closing and Relocation

In the visual SLAM context, relocation and loop closure detection are two of the most important topics that have been addressed in recent years. The former allows the system to carry out recovery actions after rough motions and blurring defects, which are very common causes for tracking failure. The second, loop closure detection, deals with the capability of a visual SLAM system to recognize a previously visited place. Both make part of the well-known loop closure problem. Recently, a comparison among the most relevant loop closure methods for visual SLAM was presented in Williams, Cummins, Neira, Newman, Reid, et al. (2009). Three paradigms were compared: image-to-image, which works only in the image space; map-to-map, which uses metric map information; and image-to-map, which uses visual and metric information to perform relocation. The main conclusion of the work suggests a fusion of image-to-image and relocation to deal with the drawbacks of the individual methods.

If we focus on loop closure detection, we find that appearance-based methods are particularly suitable for online place recognition due to the richness in information that cameras provide. These methods can be classified as image-to-image and commonly rely on the construction of visual vocabularies or bag of words. The work presented in Sivic and Zisserman (2003), for example, describes an approach to object and scene retrieval that searches for and localizes all the occurrences of a user-outlined object in a video. The object is represented by a set of viewpoint-invariant region descriptors or clusters that constitutes the visual words, so that recognition can proceed successfully despite changes in viewpoint, illumination, and partial occlusion. The temporal continuity of the video within a shot is used to track the clusters and reject unstable regions, reducing the effects of noise in the descriptors. The analogy with text retrieval is in the implementation, in which matches on descriptors are precomputed (using vector quantization), and inverted file systems and document rankings are used. The result is fast retrieval of a ranked list of key frames. In Cummins and Newman (2008) the authors propose a probabilistic framework for navigation using only appearance data. By learning a generative model of appearance, they can compute the probability that any two sets of observations originate from the same location; hence given a vocabulary and an approximate probabilistic model of observations, it is possible to compute a probability density function (pdf) over location from an observation. Although the algorithm complexity is linear in the number of places, learning a generative model is an offline process. In addition, every time the model has to be updated, the system experiences a high cost. The method provides high accuracy on recognizing a place but does not use metric information from the map to discard false positives on loop closure detections.

In the case of relocation for visual SLAM, the work of Williams, Klein, and Reid (2007) is one of the first that added a robust module to provide a real-time monocular SLAM system with a relocation tool based on fast keypoint learning (Lepetit & Fua, 2006): a descriptor (a Harris patch) is registered and learned at any viewpoint, scale, or orientation by performing classification with randomized trees. When the SLAM system has become lost, the randomized list classifier is used on each new frame to generate a tree of feature correspondence hypotheses detected in the image. Relocation is performed in an image-to-map framework in which the relative pose of the camera to the map is computed using the well-known random sample consensus (RANSAC) algorithm for three feature correspondences and their three-point pose (Fischler & Bolles, 1981). Despite its robustness, the method does not scale well to larger environments like the image-to-image method.

We find alternative works that deal with the full loop closure problem in a pose-to-pose fashion (Olson, 2009). A topological graph of robot poses is maintained so that when there is evidence of an unambiguous loop, nodes in the graph are merged only if the uncertainty between their poses is consistent. This prevents one from accepting false-positive cases. The idea of joining visual loop detection and relocation is explored in this work in a way similar to that of Eade and Drummond (2008), where the construction of a visual vocabulary and the solution of an epipolar geometry problem are fused in a single scheme. The method that we propose basically consists of three stages: first, we build an appearance-based representation (bag of words) of images acquired from one of the cameras; then, our system checks whether the current scene was seen before, so that a loop is detected. This is done during SLAM execution, in which the loop detection thread delivers candidate past keyframes to the active current submap. These candidates present high similarity with previous images of already visited submaps. After a loop is detected, the algorithm discards false positives and performs closure by verifying the

geometrical constraints imposed: features detected in candidate past keyframes are matched in the current frames using an active epipolar search based on the fundamental matrix calculation.

3. CI-GRAPH SLAM

The main contribution of this paper is a new algorithm that extends the properties of CI submaps to arbitrary trajectories with any number of loops. A detailed explanation of CI submaps was presented in Piniés and Tardós (2008). In the next section we give a brief review of CI submaps. In Section 3.2 we define our new CI-Graph technique. Finally in Section 3.3 we describe a practical implementation of our algorithm.

3.1. Conditionally Independent Submap Review

Figure 1 shows a Bayesian network that represents the stochastic dependencies between a pair of sequentially built CI submaps. The state vector of each submap is given by

$$\begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_C \end{bmatrix} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_C \end{bmatrix}, \tag{1}$$

where $\mathbf{x}_A$ represents state components that belong only to the first map, $\mathbf{x}_B$ is for elements exclusively included in the second submap, and $\mathbf{x}_C$ represents features and vehicle states that are shared between both. Notice that common elements $\mathbf{x}_C$ are replicated in each submap.

As can be seen in the figure, the only connection between the set of nodes $(\mathbf{x}_A, \mathbf{z}_a)$ and $(\mathbf{x}_B, \mathbf{z}_b)$ is through node $\mathbf{x}_C$, i.e., both subgraphs are *d-separated* given $\mathbf{x}_C$ (Bishop, 2006). This implies that nodes $\mathbf{x}_A$ and $\mathbf{z}_a$ are conditionally independent of nodes $\mathbf{x}_B$ and $\mathbf{z}_b$ given node $\mathbf{x}_C$. Intuitively this means that if $\mathbf{x}_C$ is known, submaps 1 and 2 do not carry any additional information about each other. This property is what we call the *submap conditional independence (CI) property*, which can be stated as

$$\begin{aligned} p(\mathbf{x}_A|\mathbf{x}_B, \mathbf{x}_C, \mathbf{z}_a, \mathbf{z}_b) &= p(\mathbf{x}_A|\mathbf{x}_C, \mathbf{z}_a), \\ p(\mathbf{x}_B|\mathbf{x}_A, \mathbf{x}_C, \mathbf{z}_a, \mathbf{z}_b) &= p(\mathbf{x}_B|\mathbf{x}_C, \mathbf{z}_b). \end{aligned} \tag{2}$$

The process of building two consecutive CI submaps can be summarized in four steps:

1. Build the first submap, obtaining
$$p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a). \tag{3}$$
2. Start the second map with the result of marginalizing out the noncommon elements:
$$p(\mathbf{x}_C|\mathbf{z}_a) = \int p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a)\, d\mathbf{x}_A. \tag{4}$$
3. Continue building the second submap, adding new features $\mathbf{x}_B$ and observations $\mathbf{z}_b$ to it, obtaining
$$p(\mathbf{x}_B, \mathbf{x}_C|\mathbf{z}_a, \mathbf{z}_b). \tag{5}$$
4. Update the first submap using the following *back-propagation* equations:
$$\begin{aligned} p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a, \mathbf{z}_b) &= p(\mathbf{x}_A|\mathbf{x}_C, \mathbf{z}_a, \mathbf{z}_b) p(\mathbf{x}_C|\mathbf{z}_a, \mathbf{z}_b) \\ &= p(\mathbf{x}_A|\mathbf{x}_C, \mathbf{z}_a) p(\mathbf{x}_C|\mathbf{z}_a, \mathbf{z}_b), \end{aligned} \tag{6}$$
where the second equality comes from submap CI property (2). Observe that the first factor can be obtained from the first submap (3) by conditioning, whereas the second factor is obtained from the second submap (5) by marginalization.

Note that the second map is initialized up to date by construction in the second step and is maintained updated during the third step. After applying the fourth step, both maps are up to date without having to join them. Observe as well that the fourth step (updating the first submap) can be applied at any moment. This allows us to schedule the back-propagation in moments with low CPU loads or to delay it until a loop closure is detected.

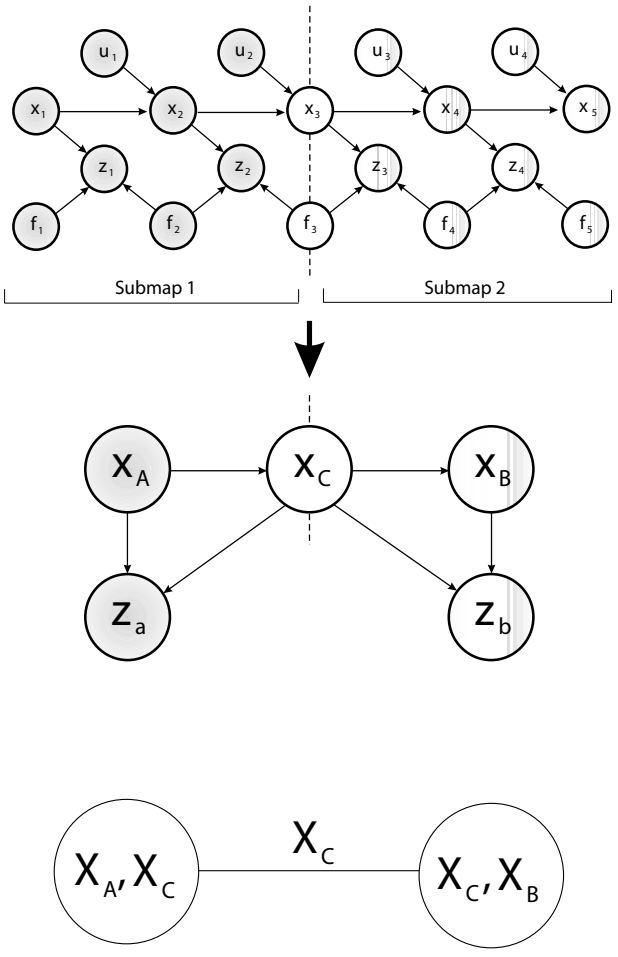


Figure 1. Bayesian network that describes the probabilistic dependencies between a pair of consecutive CI submaps (top and middle) and its corresponding CI graph (bottom).

3.2. Definition of CI-Graph

To work with complex topologies, the algorithm proposed is based on building an undirected graph of the CI submaps. An undirected graph is defined as a pair $\mathcal{G} = (\mathcal{N}, \mathcal{E}_\mathcal{G})$, where $\mathcal{N}$ are the *nodes* of $\mathcal{G}$ and $\mathcal{E}_\mathcal{G}$ its undirected *edges* (Cormen, Leiserson, & Rivest, 2001). In our graph, $\mathcal{N}$ is the set of CI submaps $\mathbf{m}_i$ with $i = 1 \dots N$. An edge connecting two nodes is created either because the robot makes a transition between the corresponding submaps or because being the robot in a submap, it observes a feature that belongs to another submap.

In addition, the algorithm builds a spanning tree $\mathcal{T}(\mathcal{N}, \mathcal{E}_\mathcal{T})$ of the graph $\mathcal{G}$ with certain properties, where $\mathcal{E}_\mathcal{T} \subset \mathcal{E}_\mathcal{G}$. A spanning tree $\mathcal{T}$ of a connected undirected graph $\mathcal{G}$ is defined as a subgraph of $\mathcal{G}$ that is a tree (it contains no cycles) and connects all the nodes. Our algorithm ensures that by construction, any pair of submaps $(\mathbf{m}_i, \mathbf{m}_j)$ that are adjacent in $\mathcal{T}$ have a conditionally independent structure as shown in Figure 1, sharing some vehicle and feature states. Each edge in $\mathcal{E}_\mathcal{T}$ will be labeled with the corresponding shared states (Figure 1, bottom). Given any pair of submaps, $\mathbf{m}_i$ and $\mathbf{m}_j$, there is a unique path in $\mathcal{T}$ connecting them. This path will allow us to transmit information from map to map without losing the conditional independence property between submaps. In all graph figures of the paper, spanning tree edges $\mathcal{E}_\mathcal{T}$ will be depicted using a continuous line and the remaining edges of $\mathcal{G}$, i.e., $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$, will be traced with a dashed line.

To perform exact inference in the graph, the corresponding spanning tree has to accomplish the following properties:

1. Any pair of neighboring submaps in the tree are *conditionally independent* given the common elements between them. This allows us to share elements between maps and to update each submap taking into account all observations without having to join them.
2. The submap where the robot is currently located is always maintained up to date, containing the marginal probabilities of the submap variables given all measurements. All previous maps are then updated backward from the current map using the backpropagation equations of CI submaps.

Let us now explain how both properties can be fulfilled during tree construction in more detail.

To accomplish property 1 in a pure exploratory trajectory, we have to follow only the steps explained in Section 3.1 for building consecutive conditionally independent submaps. However, when a loop-closing event occurs (a feature from a previous map is reobserved or the robot relocalizes in a map) this requirement is violated. To avoid this inconsistency, we define the following rule: *if two maps share an element in common, all maps in the unique path between them (we are in a tree) must have a copy of that element*.

Figure 2 depicts this process. At the top we can see the structure of three submaps before a loop-closing event takes place. At this moment the spanning tree built (which in this simple example it is just a single branch) perfectly accomplishes all the properties: all pairs of neighboring maps are conditionally independent given its common elements, and the current submap (submap 3) is up to date by construction. In the second row of the figure, the robot at position $\mathbf{x}_7$ closes a loop by observing feature f_1 through measurement $\mathbf{z}_7$. As can be observed, property 1 is not fulfilled anymore because there is an alternative path through feature f_1 between submaps 3 and 2, so they are not d-separated anymore. To rectify this problem, we apply our correction rule and include a copy of the common element f_1 between submaps 1 and 3 in all maps along the loop. Using this rule, we successfully restore the CI property as can be observed in the third row of Figure 2.

Property 2 is guaranteed by construction in exploratory trajectories. However, when the robot revisits a previous map, this requirement is not accomplished anymore because the revisited map is out of date. To fulfill property 2 in this case, we define the following rule: *when the robot revisits a previous map i from the current map j, we first make an update of all maps in the path between j and i (i included) using the backpropagation equations (6)*. By this rule we guarantee that the revisited map, which is going to become the current map, is up to date. Finally we copy the robot along the path between both maps in order to accomplish property 1 and switch to the revisited map.

Observe that the method proposed is independent of both the particular distribution of the probability densities and the optimization or estimation method implemented to obtain a model of the environment. The decomposition in submaps performed in CI-Graph depends only on general probabilistic concepts and the intrinsic structure of the SLAM problem. The actual implementation of the algorithm developed in this paper is based on a filter paradigm and, in particular, on the EKF. This means that we are intrinsically making a linear–Gaussian approximation of the corresponding submap marginals. The corresponding backpropagation equations for Gaussian distributions can be found in Appendix A. To reduce the effect of linearization errors and obtain better consistency properties, we implement the CI submaps using local coordinates. The needed transformation is explained in Appendix B.

3.3. CI-Graph Algorithm Description

Two operational levels can be distinguished in the algorithm: local operations that are applied only to the current submap $\mathbf{m}_i$ and graph operations that are performed through the graph involving at least two submaps. Most of the time, the operations carried out when the robot moves inside a CI submap are local operations

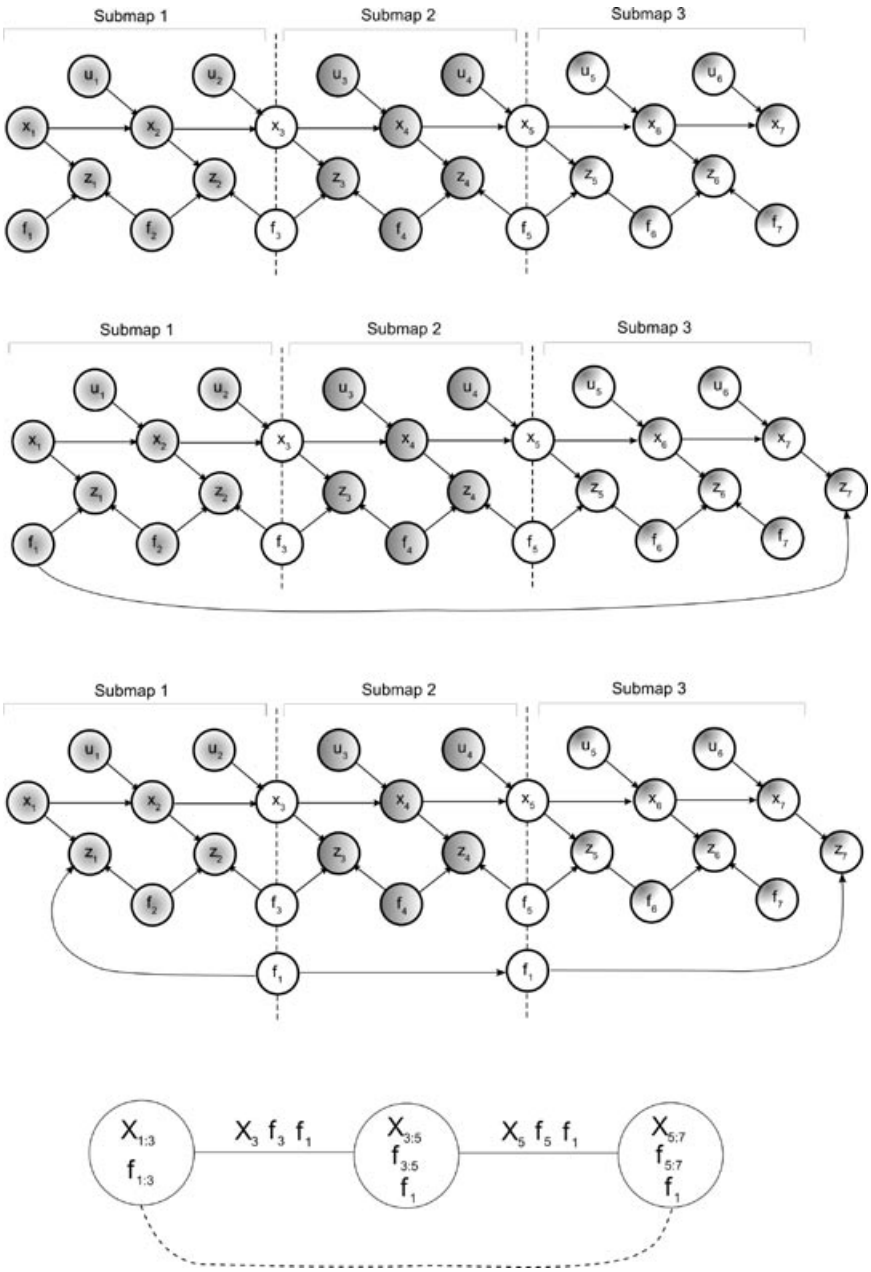


Figure 2. Loop-closing example. Row 1: before closing the loop, maps accomplish the required properties by construction. Row 2: when a previous feature is reobserved, the CI requirement is violated because there is a new path through f_1 that connects submaps 2 and 3. Row 3: by copying the common feature f_1 along the path from submap 1 to submap 3, the CI property is restored. Row 4: CI-Graph and spanning tree of the three submaps after loop closing.

corresponding to standard EKF–SLAM equations. Graph operations are more sporadic and can be considered as the interface between CI submaps. In the following sections, the graph operations are explained in detail as presented in Algorithm 1.

3.3.1. Starting a New Submap

Suppose that the robot is in submap $\mathbf{m}_i$. In our current implementation, the decision to start a new submap $\mathbf{m}_j$ is based on both a maximum number of features per map and a threshold on the maximum angle

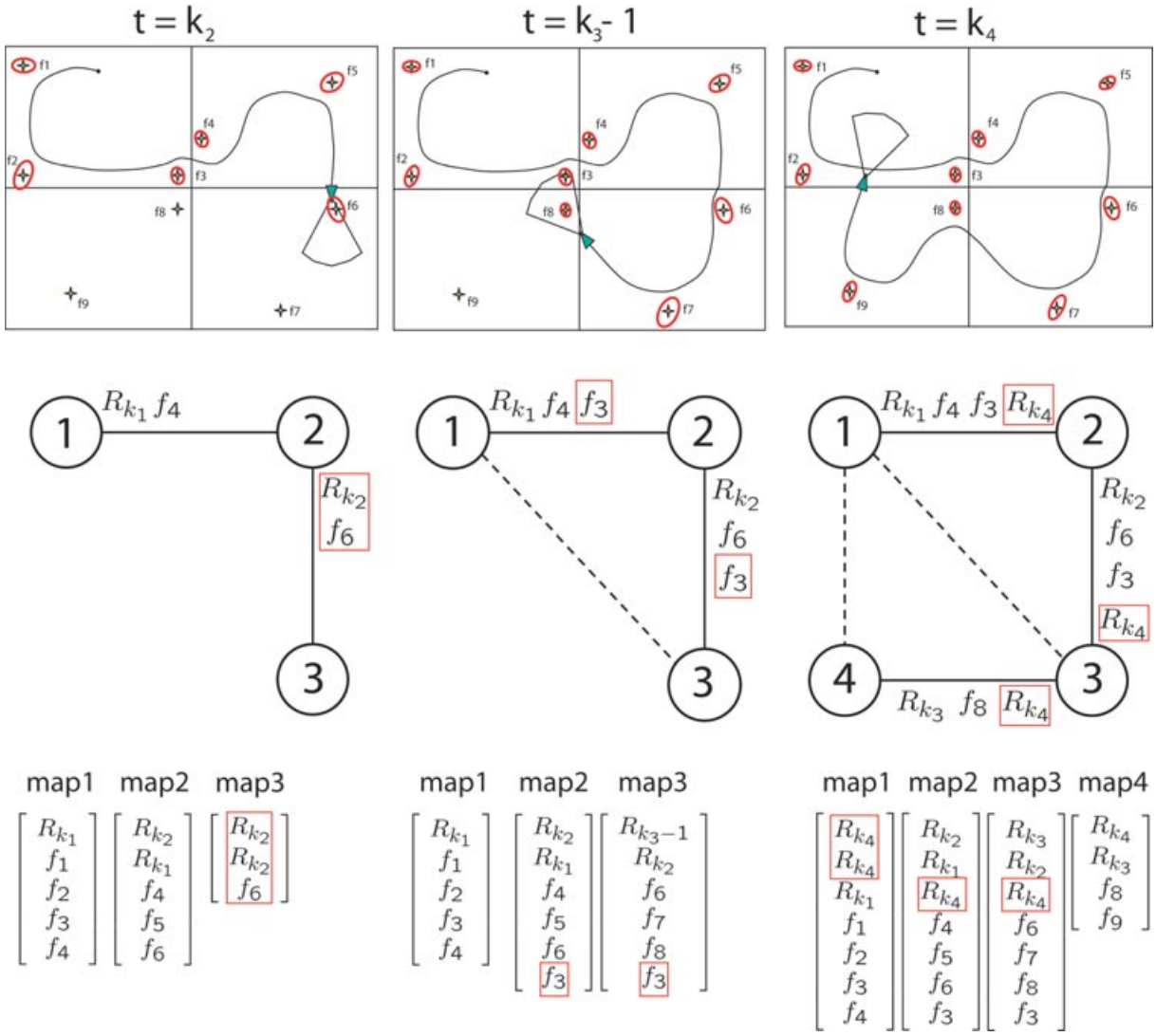


Figure 3. Example using CI-Graph SLAM. The figure is divided into three rows that show information about the state of a simulated experiment at three different instants of time (columns). In the first row, the map of the simulated environment with the current robot position is shown. In the second row, the graph of relations between submaps is created according to the state of the estimation. In the last row we show the state vectors of the estimated submaps at different moments of time. The red boxes highlight which variables are introduced in the common part between maps at each time stamp.

uncertainty allowed. The steps in the algorithm are as follows:

- Add $\mathbf{m}_j$ to $\mathcal{N}$.
- Add edge $\langle \mathbf{m}_i, \mathbf{m}_j \rangle$ to $\mathcal{E}_\mathcal{T}$.
- Copy robot pose and last seen features from $\mathbf{m}_i$ to $\mathbf{m}_j$ (properties 1 and 2 are accomplished by construction).

In fact, the robot pose is copied twice in submap $\mathbf{m}_j$. The first copy will represent the current robot position, which changes as the robot moves through the new map. The second copy will represent the initial position of the robot when it entered the map. This initial pose remains fixed as a common element with map $\mathbf{m}_i$.

An example can be seen in Figure 3. At time k_2, submaps $\mathbf{m}_1$ and $\mathbf{m}_2$ have already been explored and a new submap $\mathbf{m}_3$ is being created. Nodes $\mathbf{m}_1$ and $\mathbf{m}_2$ share in common a robot position R_{k_1} and a feature f_4. Submap 3 is initialized with robot R_{k_2} and feature f_6 from submap 2.

Algorithm 1 CI-Graph SLAM

$\mathbf{z}_0, \mathbf{R}_0 = getObservations$
$\mathbf{m}_0 = initMap(\mathbf{z}_0, \mathbf{R}_0)$
$[\mathcal{G}, \mathcal{T}] = initGraph(\mathbf{m}_0)$ $\{\mathcal{G}(\mathcal{N} = \mathbf{m}_0, \mathcal{E}_\mathcal{G} = \emptyset)\}$
$i = 0$ {i for current submap}
for $k = 1$ to steps **do**
  $\mathbf{u}_{k-1}, \mathbf{Q}_{k-1} = getOdometry$
  $\mathbf{m}_i = ekfPrediction(\mathbf{m}_i, \mathbf{u}_{k-1}, \mathbf{Q}_{k-1})$
  $\mathbf{z}_k, \mathbf{R}_k = getObservations$
  $\mathcal{D}A_k = dataAssociation(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k)$
  loopClosing = *False*
  if reobserved $\mathbf{f} \notin \mathbf{m}_i$ & $\mathbf{f} \in \mathbf{m}_j$ **then**
    {*Subsection 3.3.2*}
    for $\langle \mathbf{m}_k, \mathbf{m}_l \rangle$ in path($\mathbf{m}_j, \mathbf{m}_i$) **do**
      $copyFeat(\mathbf{f}, \mathbf{m}_k, \mathbf{m}_l)$
    end for
    $addEdge(\langle \mathbf{m}_j, \mathbf{m}_i \rangle, \mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T})$
    loopClosing = $True$
  end if
  $\mathbf{m}_i = ekfUpdate(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k, \mathcal{D}A_k)$
  if loopClosing & revisiting $\mathbf{m}_j$ **then**
    {*Subsection 3.3.3*}
    for $\langle \mathbf{m}_k, \mathbf{m}_l \rangle$ in $path(\mathbf{m}_i, \mathbf{m}_j)$ **do**
      $backPropagation(\mathbf{m}_k, \mathbf{m}_l)$
      $copyRobot(\mathbf{m}_k, \mathbf{m}_l)$
    end for
    $addEdge(\langle \mathbf{m}_i, \mathbf{m}_j \rangle, \mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T})$
    $i = j$ {*Map change*}
  else if newMap $\mathbf{m}_j$ **then**
    {*Subsection 3.3.1*}
    $addNode(\mathbf{m}_j, \mathcal{N})$
    $addEdge(\langle \mathbf{m}_i, \mathbf{m}_j \rangle, \mathcal{E}_\mathcal{T})$
    $copyRobot(\mathbf{m}_i, \mathbf{m}_j)$
    $copyActiveFeat(\mathbf{m}_i, \mathbf{m}_j)$
    $i = j$ {*Map change*}
  end if
  $\mathbf{m}_i = addNewFeatures(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k, \mathcal{D}A_k)$
end for
{*Subsection 3.3.4*}
$updateAllMaps(\mathbf{m}_i, \mathcal{T})$ {*Updates* $\mathcal{T}$ *starting from* $\mathbf{m}_i$}

3.3.2. Reobserving a Feature from a Different Map

This situation occurs when the robot is at submap $\mathbf{m}_i$ and observes *for the first time* a feature that is already included in a previous submap $\mathbf{m}_j$. A loop closure algorithm running in parallel flags CI-Graph when a previous image presents high similarity with the current image (see Section 6). The process is as follows:

- Copy the feature from $\mathbf{m}_j$ to $\mathbf{m}_i$ along all nodes of the path in $\mathcal{T}$ (to restore property 1).
- Add $\langle \mathbf{m}_j, \mathbf{m}_i \rangle$ to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$.

If $\langle \mathbf{m}_k, \mathbf{m}_l \rangle \in \mathcal{T}$ represents an edge in the path, to copy the feature from $\mathbf{m}_k$ to $\mathbf{m}_l$, the feature is first updated with the information contained in $\mathbf{m}_l$ using *backpropagation* equations (A.10)–(A.13) and the correlations with the elements of $\mathbf{m}_l$ are calculated with Eq. (A.15). After the reobserved features are fetched, the update step of the EKF closes the loop in the current submap. Observe that new information obtained after the loop-closing event is deferred and not transmitted to the rest of the loop. A backpropagation takes place only when the robot revisits a previous map (property 2).

Figure 3 at time $k_3 - 1$ shows an example of this case. Feature f_3, which belongs to submap $\mathbf{m}_1$, is measured by the robot when it is traversing submap $\mathbf{m}_3$. Because edge $\langle \mathbf{m}_1, \mathbf{m}_3 \rangle \notin \mathcal{T}$, f_3 is transmitted along the path $\langle \mathbf{m}_1, \mathbf{m}_2 \rangle$, $\langle \mathbf{m}_2, \mathbf{m}_3 \rangle$ that connects both nodes. Observe that the feature is replicated in all intermediate nodes. Finally, edge $\langle \mathbf{m}_1, \mathbf{m}_3 \rangle$ is included in $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$.

3.3.3. Revisiting a Previous Submap

After closing a loop in map $\mathbf{m}_i$ by reobserving a feature from a different map $\mathbf{m}_j$, the algorithm starts revisiting $\mathbf{m}_j$ if the updated robot position is inside this map. In this case, the transition from the current submap $\mathbf{m}_i$ to $\mathbf{m}_j$ is as follows:

- Update all nodes in the path from $\mathbf{m}_i$ to $\mathbf{m}_j$ (to accomplish property 2).
- Copy the current robot pose along all nodes of the path (to accomplish property 1).
- Add $\langle \mathbf{m}_i, \mathbf{m}_j \rangle$ to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$.

As in the preceding section, to update submaps in the path we use the *backpropagation* equations (A.10)–(A.13), and to copy the current robot pose correlations with submaps, elements are calculated with Eq. (A.15).

Figure 3 at time k_4 shows an example of this operation. When the robot makes a transition between submaps $\mathbf{m}_4$ and $\mathbf{m}_1$, the current robot position R_{k_4} is replicated along all nodes that are in the path, i.e., along $\mathbf{m}_3$, $\mathbf{m}_2$, and $\mathbf{m}_1$. Finally, edge $\langle \mathbf{m}_4, \mathbf{m}_1 \rangle$ is added to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$ and submap $\mathbf{m}_1$ becomes the current map.

3.3.4. Updating All Maps from the Current Submap

Using the graph operations just described, we can ensure that the current submap is always updated with all available information (property 2). In addition, the CI property between submaps is preserved (property 1). An interesting characteristic of the backpropagation equations is that they can be applied at any moment. They work correctly even if we backpropagate twice the same information: the terms inside the parentheses in Eqs. (A.12) and (A.13) will be zero and the maps will remain unchanged. This allows us to schedule the backpropagation in moments with low CPU loads or when graph operations are required. If the whole

map has to be updated, the *backpropagation* equations are recursively applied, starting from the current node and following the spanning tree.

3.4. Relationship to Junction Trees

Given a graph with cycles that represents the relationship between a set of variables, a junction tree is a tree having two properties (Wainwright & Jordan, 2008):

- Covering: Every clique in the graph is contained in at least one of the tree nodes.
- Running intersection: for any two clique nodes $C1$ and $C2$, all nodes on the unique path joining them contain the intersection $C1 \cap C2$.

A junction tree allows *exact* belief propagation by passing forward and backward messages along the tree instead of using approximate loopy belief propagation in the original graph. This technique, along with some approximations to reduce the number of edges in the graph, was used for SLAM in Paskin (2003).

Our technique, although developed using only the concept of conditional independence between submaps, is related to junction trees. In fact, it can be seen that our spanning tree *is a* junction tree, because all the cliques between the SLAM variables are contained in a local map, and the technique used for restoring the conditional independence after a loop closure ensures the running intersection property. However, there are two main differences. First, junction trees are typically obtained through a process of moralization and triangulation of the original graph, whereas in our approach the spanning tree corresponds to submaps and is directly obtained by the CI-Graph algorithm. Second, general junction trees compute potentials for each tree node that are updated using forward and backward propagations along the tree, whereas in our technique the current map always contains the marginals of the submap variables given all previous information. This ensures that an old submap can be updated when desired by a single one-way propagation from the current map to the old submap along the spanning tree. This operation is needed only when the robot revisits a map and at the end of the experiment, to recover the marginals of the whole map.

4. SIMULATION RESULTS

CI-Graph SLAM was tested using a simulated environment that emulates a Manhattan world, similar to the one proposed in Dellaert and Kaess (2006), with 2,420 point features lying on the walls of a 11×11 matrix of building blocks. For this $2D$ example, the total space is divided into submaps using a grid cell. When the robot crosses the border between two cells for the first time, a new submap is initialized. If the arriving cell has already been traversed, we consider that a previous submap is revisited.

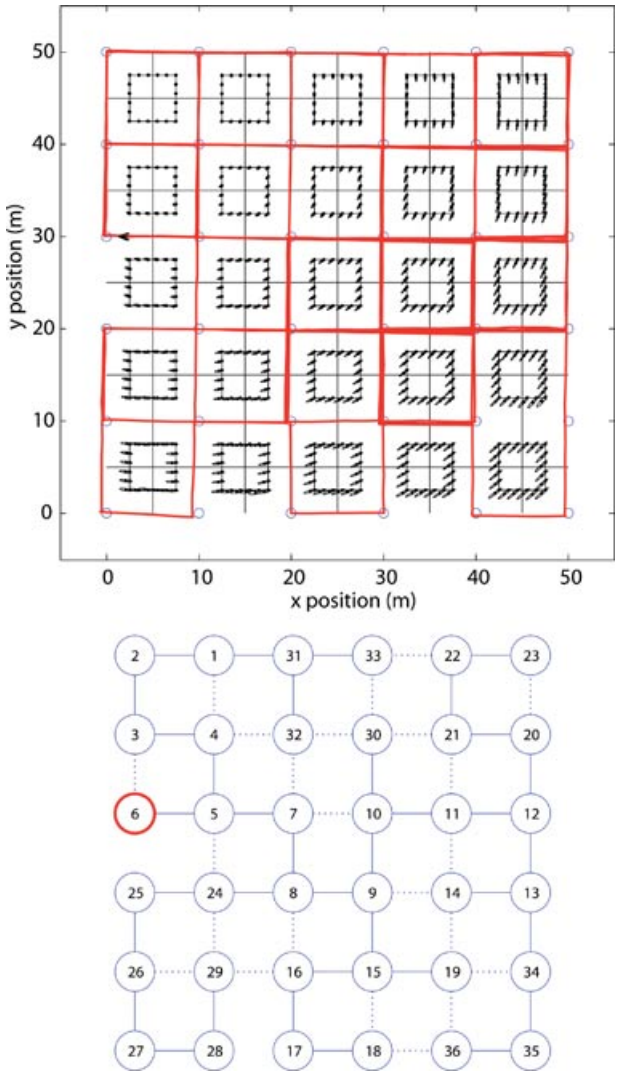


Figure 4. CI-Graph SLAM execution on a simulated environment of a 5×5 matrix of building blocks of a Manhattan world (top). The environment is divided up into 36 submaps (nodes) using a cell grid. The darker red line represents the estimated trajectory. Ellipses also show the estimated feature uncertainties. The final CI-Graph contains direct links (continuous lines) that form the spanning tree between nodes (bottom). Indirect links are shown as dashed lines such that they represent mutual information seen between adjacent nodes. Thus, there exists a path formed by direct links through which the information can be transmitted.

In the Manhattan environment, the vehicle performs a randomly chosen trajectory of 1,600 steps of 1 m. Figure 4, top, shows a smaller 5×5 example to give the reader an idea of the experiment. Each time the vehicle reaches a block corner (circles), a control input is applied randomly. This kind of motion allows the robot to perform

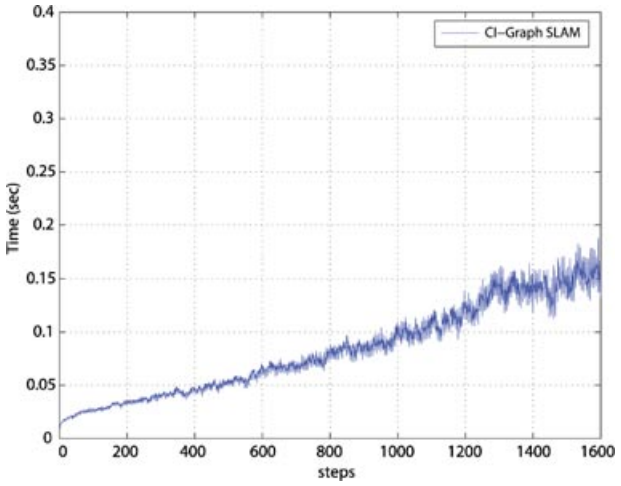


Figure 5. Average running time per step for 300 random runs in 11×11 Manhattan world experiment. The size of the mapped environment ranges from $n = 947$ to $n = 2{,}199$ point features. Also, in each run, the submap size does not exceed 200 features on average.

any trajectory: the robot can move from one map to its neighbor and perform any large loops. The motion model noises are assumed to be Gaussian with, respectively, $\sigma_x = \sigma_y = 0.05$ m and $\sigma_\theta = 0.3$ deg standard deviation. As the robot moves, the graph of CI submaps is created on the fly. Figure 4, bottom, shows an example of the resulting spanning tree with nodes numerated in the order they were created.

In this simulated experiment, Monte Carlo runs are particularly suitable to evaluate the CI-Graph SLAM efficiency. We ran 300 samples of our algorithm implemented in MATLAB on an Intel Core 2 Quad at 2.83 GHz. Note that each sample represents a different random trajectory and so a different spanning tree. For the same reason, the number of total mapped features varies although the environment remains unmodified. Figure 5 shows the mean running time per step. We can see that for this extremely loopy environment, the algorithm presents a close-to-linear running time.

5. WORKING WITH MULTIPLE CAMERAS

The performance of CI-Graph SLAM in 3D environments is evaluated using a trinocular camera mounted on a mobile robot. To make the algorithm more general, we implemented two alternative motion models depending on whether odometry information is available. When no odometry is available, we suppose that the camera moves smoothly in three dimensions and we apply a constant velocity model in which 12 parameters are used to represent the camera location in the state vector: three Euclidean coordinates to represent the camera pose, three Euler angles for the camera orientation, and six parameters for linear and angular velocities. When odometry readings are available, we switch to a simpler model. In such a case, the state vector is reduced to six parameters, avoiding the estimation of both linear and angular velocities.

Regarding the map, we maintain the same scheme used in our previous work in which the state vector contains feature points represented either as ID (Civera et al., 2008) or as 3D Euclidean features. The reason is that the range of depth values that can be extracted with an array of cameras is limited: for close objects observable from more than one camera, depth and bearing information can be successfully obtained and the scale of the environment observed. On the contrary, the depth of far points cannot be computed straightforwardly and observations taken from distant camera locations are required to triangulate. Nevertheless, distant points still provide useful bearing information to compute the attitude of the system.

In Paz et al. (2008a), we took advantage of the disparity information provided by the stereo system in order to disambiguate between close and distant regions from the camera. Therefore, features were initialized as 3D Euclidean points using disparity if a Gaussian test to evaluate the linearity of the reconstruction was accomplished. Otherwise, features were initialized as ID. In our current work, on the contrary, each camera of the multicamera array is treated independently by the algorithm, allowing the system to easily scale with an arbitrary number of cameras. This means that we do not make any assumption about the existence of disparity information.

The only interaction between cameras is when a new feature is initialized. One camera is selected as the reference camera to initialize new features; in our implementation this is the right-hand camera of the trinocular system. Because no depth information is available, this feature is first included in the state vector using ID parameterization. Using the known extrinsic calibration between cameras, the recently introduced feature is predicted in the other camera images, where we perform an active search over the uncertainty region yielded by the image projection of the innovation covariance. The rigid transformation between cameras allows us to obtain the depth information of nearby features. For the rest of the steps of the SLAM algorithm, each camera predicts and updates features independently.

We briefly summarize the distinct processes associated with the construction of a local submap in the multicamera framework.

5.1. State Representation

We describe the map state vector as composed of the reference camera location $\mathbf{x}_r$ together with feature poses $\mathbf{x}_{f_{1:n}}$, both with respect to a local map base reference B that is usually chosen as the initial camera location. Because no depth information is available when a point is initialized,

all map features $\mathbf{x}_{\text{ID}}$ are first coded using the *ID parameterization* (the process is detailed in Section 5.2):

$$\mathbf{x}_B = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_{f_{1:n}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_{\text{ID}} \end{bmatrix}. \tag{7}$$

To be coherent with our real experiments when odometry readings are available, the camera location is given by six elements: three Euclidean coordinates for camera pose $\mathbf{r}$ and three Euler angles for camera attitude ψ. To propagate the camera motion, we use relative transformations with a zero mean Gaussian noise associated with the six camera parameters. Equation (8) resumes the propagation of the motion on the camera location using homogeneous compositions (Castellanos & Tardós, 1999):

$$\hat{\mathbf{x}}_{r_{t+1}} = \hat{\mathbf{x}}_{r_t} \oplus \hat{\mathbf{x}}_{r_t r_{t+1}} = \begin{bmatrix} \mathbf{r}_{t+1} \\ \Psi_{t+1} \end{bmatrix}, \tag{8}$$

$$\hat{\mathbf{x}}_{r_t r_{t+1}} = \mathbf{x}_{r_t r_{t+1}} + \nu, \tag{9}$$

$$\nu \approx \mathcal{N}(0, \mathbf{Q}); \tag{10}$$

$$\mathbf{Q} = \text{diag}(\sigma_x, \sigma_y, \sigma_z, \sigma_\varphi, \sigma_\phi, \sigma_\theta). \tag{11}$$

Given the camera location $\mathbf{x}_{r_i}$ at instant i, an ID point seen from this position is defined as

$$\mathbf{x}_{\text{ID}} = \begin{bmatrix} \mathbf{r}_i \\ \theta_i \\ \phi_i \\ \rho_i \end{bmatrix}, \tag{12}$$

where $\mathbf{r}_i$ represents the optical center of the camera from which the feature was first observed, θ_i (azimuth) and ϕ_i (elevation) define the direction of the ray passing through the image point, and $\rho_i = 1/d_i$ is the ID (Civera et al., 2007).

5.2. Selection, Management, and Tracking of Map Features

We use the Shi–Tomasi detector to select trackable image points from 640×480 images. For each image point, we assign a 15×15 surrounding patch as descriptor. We use a uniform distribution of image points by splitting the image of the reference camera (right camera) in a regular grid. Then, the point with the best Shi–Tomasi response per grid cell is selected as shown in Figure 6. In this way, we guarantee that only very informative points are observed from the scene, which will increase their tracking stability. During the following steps, those cells that become and remain empty for a given time are monitored as candidates to initialize new features. The approach is accompanied by a feature management strategy that deletes nonpersistent features to avoid an unnecessary growth in population.

Once an image point is selected, it must be introduced as a new feature into the state vector. This adding process is described as follows:

1. **Adding a new feature as an ID point**. At instant t, a feature $\mathbf{x}_{f_i}$ is first initialized in the reference camera. Because no depth information is available, this feature is included in the map state vector using ID parameterization (Civera et al., 2008). In our application we chose the right camera as the main reference frame r.
2. **Predicting map features on the multicamera array**. For each camera c in the trinocular array, we predict initialized map features and look for image pairings that will be used to update the map estimate using the EKF. The extrinsic calibration that relates the camera c to the reference camera r is used as a perfect pose measurement. This relative pose $\mathbf{x}_{rc}$ is required to predict the map features by using the measurement equation:

$$\begin{aligned} \mathbf{z}^c_{\text{ID}} &= \mathbf{h}^c_{\text{ID}}(\mathbf{x}_r, \mathbf{x}_{rc}, \mathbf{x}_{\text{ID}}) + \upsilon \\ &= \text{projection}[\ominus(\mathbf{x}_r \oplus \mathbf{x}_{rc}), \mathbf{x}_{\text{ID}}] + \upsilon. \end{aligned} \tag{13}$$

   For each projected feature, we perform a standard active search to find a match $\mathbf{z}$. In this process, normalized cross correlation is carried out with the corresponding feature patch. When all match candidates for each image feature are found, the randomized joint compatible (RJC) algorithm is applied to calculate a consistent pairing hypothesis $\mathcal{H}_{rc}$. The resulting innovation $\upsilon = (\mathbf{z}^c_{\text{ID}} - \mathbf{h}^c_{\text{ID}})_{\mathcal{H}}$ allows us to update the map state vector.
3. **Transforming points from ID to 3D Euclidean coding**. To reduce the memory and computational complexity, ID features are transformed into 3D Cartesian parameterization [Eq. (14)] whenever possible according to the parallax index explained in Civera, Davison, and Montiel (2007). The measurement equation used to predict the 3D point in the current location of the camera is given in Eq. (15):

$$\mathbf{x}_{\text{3D}} = \text{ID to 3D}(\mathbf{x}_{\text{ID}}) = \mathbf{r}_i + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i), \tag{14}$$

$$\begin{aligned} \mathbf{z}^c_{\text{3D}} &= \mathbf{h}^c_{\text{3D}}(\mathbf{x}_r, \mathbf{x}_{rc}, \mathbf{x}_{\text{3D}}) + \upsilon \\ &= \text{projection}[\ominus(\mathbf{x}_r \oplus \mathbf{x}_{rc}) \oplus \mathbf{x}_{\text{3D}}] + \upsilon. \end{aligned} \tag{15}$$

To track map features at each subsequent instant t, we apply a process similar to the one described in steps (2) and (3). The difference is that the features must also be predicted on the reference camera r. All the details about how to calculate the measurement equations for ID and 3D points are available in Paz (2008).

Figure 6 shows an example of the system when building a local map along a university library of the indoor experiment from which the data set is obtained. We can see how features in the map are predicted and searched over right, left, and top images in order to update the state vector. A reconstruction is also shown in both top and lateral views for the resulting submap.

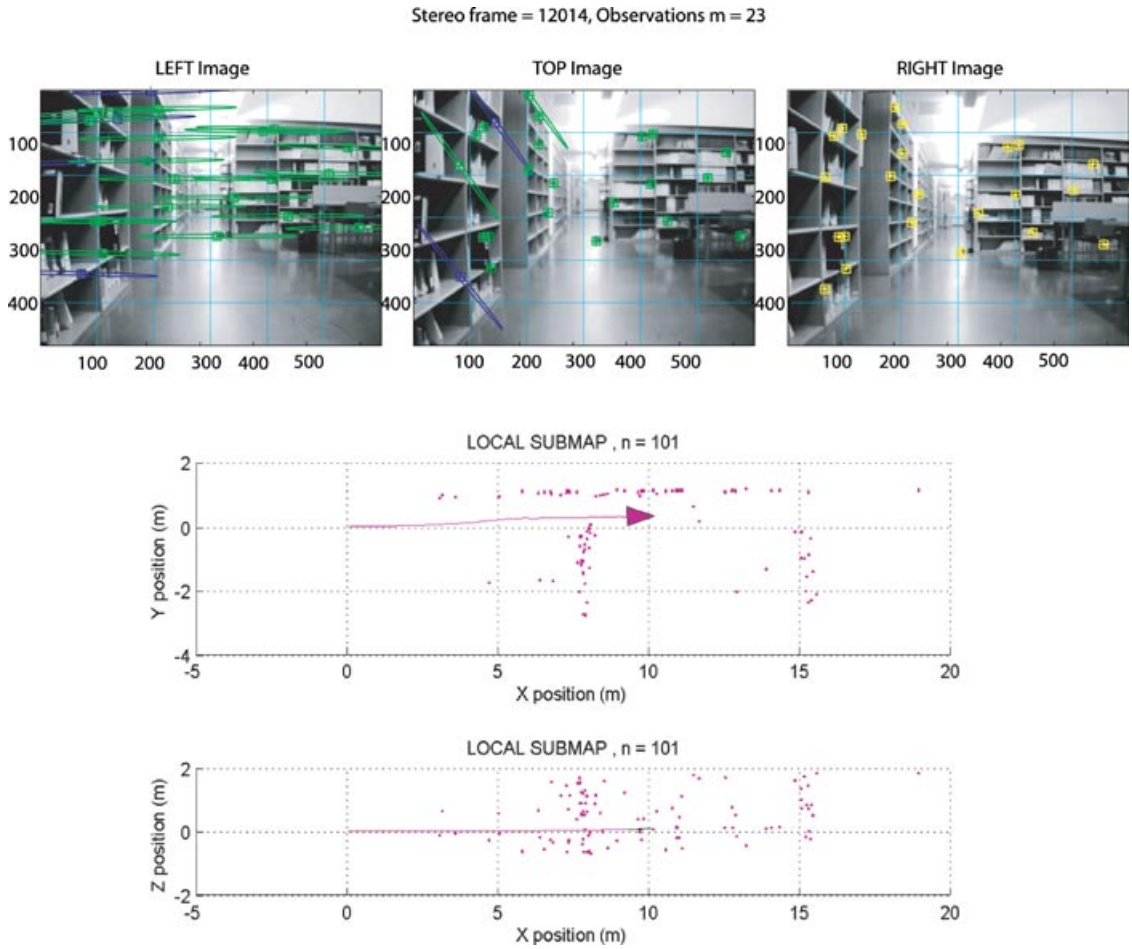


Figure 6. Top: SLAM system performing trinocular tracking. Large ellipses are produced after projecting recently initialized map features using the corresponding extrinsic calibration. Bottom: top and lateral views of the local submap reconstruction.

6. APPEARANCE-BASED LOOP CLOSING

We use an algorithm based on appearance to solve the loop closure problem because this approach has proved to be very successful (Cummins & Newman, 2008; Williams et al., 2009). This method consists of three stages: First, an image is acquired from one of the cameras, and its appearance-based representation is obtained. Then, in posterior steps we periodically check whether the current scene was seen before, so that a loop can be detected. Finally, when a loop is detected, the current image is actively searched for features from the old image to close the loop. These processes are detailed in the following sections.

6.1. Appearance-Based Representation

To obtain an appearance-based representation of an image, we build a visual bag of words in an offline scheme training (Sivic & Zisserman, 2003). This technique represents an image by using a numeric vector that is created from local features. In our implementation we use the *SURF* detector (Bay, Ess, Tuytelaars, & Van Gool, 2008) to obtain invariant image features. Each SURF feature has associated a real 64-dimensional descriptor that summarizes the distribution of the intensity content within the point neighborhood.

The bag-of-words method consists of clustering the image descriptor space (the 64-dimensional SURF space, in our case) into a fixed number C of clusters. The centers of the resulting clusters are named *visual words*; after clustering, a *visual vocabulary* is obtained. Now, a set of image features can be represented in the visual vocabulary by means of a vector v of length C. For that, each feature is associated to its closest visual word; then, each component v_i is set to a value in accordance with the relevance of the ith word in the vocabulary and the given set or 0 if that word is not associated to any of the image descriptors. In general, the more a word appears in the data used to create the visual vocabulary, the lower its relevance is. The vector v is the *bag-of-words* representation of the given set of image

descriptors. This way, the appearance of an image can be simply described by a numeric vector.

This method is suitable for managing big amounts of images; moreover, Nister and Stewenius (2006) present a hierarchical version that improves efficiency. In this version, the descriptor space clustering is done hierarchically, obtaining a visual vocabulary arranged in a tree structure, with a branching factor k and L depth levels. Then, the comparisons for converting an image descriptor into a visual word need to be done only in a branch and not in the whole discretized space, reducing the search complexity to logarithmic. We use an implementation of this technique.

6.2. Loop Detection

To detect a loop, one of the captured images is analyzed every second (1 Hz). An image obtained at time t is converted into a vector v_t according to its bag-of-words representation. This vector is compared with the set of all the vectors of the images obtained before, $\mathcal{W}$, to check whether any of them is similar enough to consider both scenes the same. If there is a satisfactory match with some vector $w_{t'} \in \mathcal{W}$ acquired at time $t' \leq t - T$, a loop may be found. To confirm this, there must be consistency with the images previously matched. The parameter T is the minimum length of the loop; matches occurring in the last T seconds are ignored. This is necessary because of the overlap between consecutive images. Its value depends on the expected variability of the scenes. We use $T = 20$ s for indoor routes and $T = 30$ s for outdoor environments. Note that if T is very high, small loops may be missed. Finally, the current vector v_t is added to $\mathcal{W}$, and features from its image are stored in case they can be used in a future loop-closing event.

To make vector similarity comparisons faster, an inverted index is maintained (Nister & Stewenius, 2006). This index stores in which vectors each visual word is present. This way, when the vector v_t is compared with vectors from $\mathcal{W}$, comparisons are made only with those vectors $w_{t'} \in \mathcal{W}$ that have at least one visual word in common with v_t. When the vector v_t is added to $\mathcal{W}$, the inverted index is updated by including v_t in the lists of the visual words it contains.

The similarity between two vectors is scored when they are compared. This score grows as the resemblance between the two images is higher. Given two image vectors v_t and $w_{t'} \in \mathcal{W}$, the score of its match is related to the normalized distance between the two vectors (Nister & Stewenius, 2006):

$$s(v_t, w_{t'}) = 1 - \frac{\left\| \frac{v_t}{|| v_t ||} - \frac{w_{t'}}{|| w_{t'} ||} \right\|}{2}. \tag{16}$$

We use the L_1 norm to compute this score, so that it is defined between 0 (completely different words) and 1 (perfect match). When v_t is compared against all the vectors $w_{t'} \in \mathcal{W}$, a list of matches $\langle v_t, w_{t'} \rangle$ associated to their scores $s(v_t, w_{t'})$ is obtained. Low-score matches are discarded from this list. The score range depends on the number of features each image contains. For this reason, a score is considered low by comparing it to the maximum expected score for a certain image (denoted λ_t). Because our images are taken from a video sequence, we approximate λ_t with $s(v_t, v_{t-1})$. If images separated by 1 s are not similar (e.g., if the robot is turning), this approximation is not reliable and λ_t is small. Therefore, we remove the matches $\langle v_t, w_{t'} \rangle$ with $\lambda_t < 0.1$ or whose score $s(v_t, w_{t'})$ does not achieve $\alpha\lambda_t$. Here, α is the minimum confidence expected for a loop closure candidate and determines the precision and recall of the method. We can select the confidence level according to the environment. For example, indoor scenes tend to show repetitive structures, causing perceptual aliasing. In this case, a stricter value may be desirable.

To reliably detect loops and to avoid mismatches, we impose temporal and geometrical constraints. To accept a match found by the bag-of-words technique between the current image and an old image, we require that the images corresponding to the last N seconds also find matches with old images separated by time gaps not bigger than G seconds. This gap implicitly defines the expected accuracy of bag-of-words matches. The constant N defines how long the scenes of a loop closure must overlap to accept it. High values increase the detection reliability but prevent some loops from being found. Small values, on the contrary, allow us to find more loops but are also prone to produce false positives. After some testing, we have fixed the values $G = 2$ and $N = 3$ s for all our experiments.

The geometric consistency of the loop is verified by computing the fundamental matrices $\mathcal{F}_l$ and $\mathcal{F}_r$ between the current images from the left and right cameras and the matched image. The eight-point algorithm (Hartley, 1997) is used on the correspondences of SURF points obtained from these images, filtering the outliers with RANSAC. If there are enough inliers and the computation of the fundamental matrices is well conditioned, the loop detection is accepted.

6.3. Loop Closing

Although loop detection is performed using SURF features, the building of local maps use instead Harris corners, because they can be tracked with better accuracy (Tuytelaars & Mikolajczyk, 2008). When a candidate loop between robot poses $\mathbf{x}_t$ and $\mathbf{x}_{t'}$ is detected, the fundamental matrices $\mathcal{F}_l$ and $\mathcal{F}_r$ between the images are passed to the CI-Graph SLAM algorithm. This allows us to search for correspondences for the old Harris features along the corresponding epipolar lines of the current left and right camera images. To find robust matches, searched features must be present in both images and consistent positions. This condition is checked by using the known fundamental matrix between the cameras in the trinocular rig. If only a few matchings are found, the loop is considered wrong and it is finally discarded. On the contrary, when enough robust features

are available, these yield data association between features taken at times t' and t, enabling CI-Graph to close the loop in the current submap by applying the steps described in Section 3.3.2.

7. EXPERIMENTS AND RESULTS

The core of our CI-Graph algorithm for multicamera SLAM was implemented in MATLAB running on an Intel Core 2 Quad at 2.83 GHz. The visual relocation algorithm was implemented in C++ using the OpenCV library and run in a separate thread. The proposed techniques were tested in two real experiments with image sequences gathered in indoor and outdoor environments (RAWSEEDS 2009). These public data sets have recently been proposed as benchmarks for SLAM algorithms.

The indoor data set consists of 26,335 trinocular image frames (79,005 images to be processed) collected during 30 min at 15 frames per second along a path of some 760 m inside a university building (*Bicocca data set*) where around 100 m corresponds to fragments of revisited trajectory. The use of this data set is convenient to evaluate our algorithms because an indoor ground truth (GT) solution is available. The nature of the GT solution relies on a fixed system formed by a bunch of cameras located in the area where the vehicle starts its trajectory and that is several times revisited. From this initial GT solution an *extended GT* solution on the full trajectory is obtained using the pose-graph–based optimization algorithm developed by Grisetti, Stachniss, and Burgard (2009). Laser scans are used to build the pose-based graph in which manual inspection and selection of scans are performed to guarantee a highly accurate solution that matches the initial GT. The details about this *extended GT* are public and available at RAWSEEDS (2009). We consider this data set particularly challenging due to the intrinsic difficulty of extracting features in several places with lack of distinctive texture. In addition, the vehicle performs rough rotations in very narrow corners. In this data set it is possible to identify around six loops.

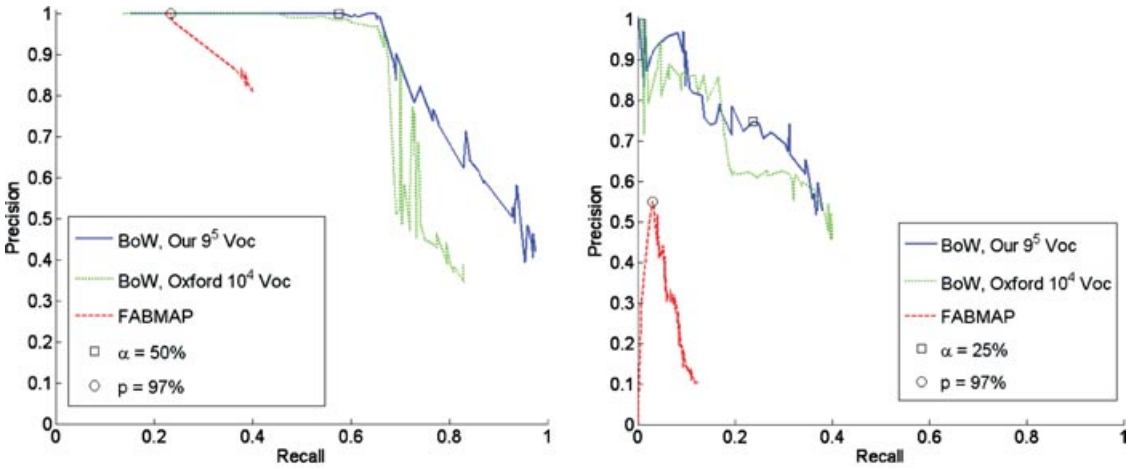
The outdoor data set consists of 34,173 trinocular frames (102,519 images to be processed), acquired in 38 min. During this time the vehicle travels across the surroundings of a university campus (*Bovisa data set*) describing a trajectory of 1.365 km with nine main loops identified (five loops smaller than 50 m and four loops larger than 100 m). This data set is also evaluated against the partial GT available from a global positioning system (GPS) device with a precision of 0.9 m. The difficulty of this data set stems from the little variability of the image background. This causes most of the features to lie in distant objects. In addition, part of the trajectory is performed in a rough terrain.

7.1. Appearance-Based Loop Detection

To evaluate the efficacy of our appearance-based method for loop closure, we built a hierarchical vocabulary with branching factor $k = 9$ and tree depth $L = 5$ and using the k means++ clustering algorithm (Arthur & Vassilvitskii, 2007). This vocabulary was created from an independently selected set of 1,300 images from an indoor–outdoor RAWSEEDS data set. The resulting vocabulary is finally composed of 57,855 words. The proposed loop detection method (hereafter BoW) is compared with FAB-MAP (Cummins & Newman, 2008), a well-known state-of-the-art technique, thanks to an implementation made available by the authors. FAB-MAP has proved to be very successful in detecting loops in long trajectories for which it delivers very few false positives. The FAB-MAP implementation available online provides two vocabularies: one was created from an indoor image sequence and consists of 10,000 words; the other contains 10,987 words calculated from an outdoor image sequence. Given the current image, FAB-MAP returns the probabilities of being in one of the previous locations or in a new one. To select the matched image, we take the case with the highest probability, if it is above a certain threshold p, to avoid mismatches. As with BoW, matches that are close in time are ignored. Because the vocabulary we built for our system differs from those used by FAB-MAP, in both size and origin of training images, we also built a second vocabulary. This is used to make a comparison between results yielded by BoW and FAB-MAP when they use vocabularies of similar characteristics. The size of this second vocabulary was $k = 10$, $L = 4$, resulting in 10,000 words, and it was created from 1,074 images collected in New College, Oxford, publicly available thanks to the authors of FAB-MAP.

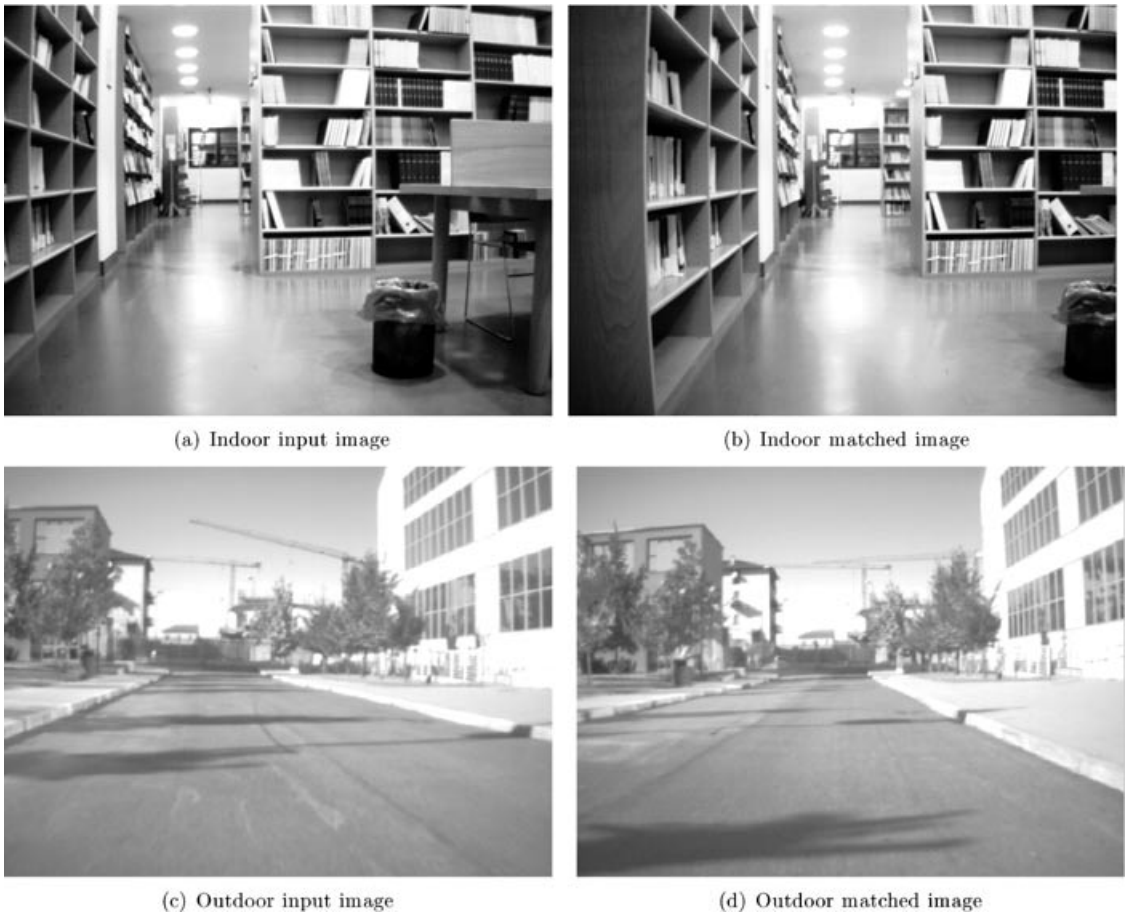
We ran both BoW and FAB-MAP in the indoor and outdoor data sets. To compare their results, we do not apply any epipolar constraint to the calculated matches at this moment. Figure 7 shows the precision-recall curves obtained by varying the parameters α for BoW (defined in Section 6.2) and p for FAB-MAP. We also show the curve when using our second vocabulary. In both cases, the shape of the BoW precision-recall curves is similar using either our RAWSEEDS vocabulary ($k = 9$, $L = 5$) or the Oxford one ($k = 10$, $L = 4$). We can also see that BoW outperforms FAB-MAP in these data sets when using the Oxford vocabulary. This suggests that the size of the vocabularies and the origin of the training images are not decisive when comparing BoW and FAB-MAP. We see that in the indoor data set both methods attain 100% precision (all the loops present in the trajectory are found), but this decreases quicker with FAB-MAP. The outdoor case is more challenging due to lighting conditions and the little variability of the background. Here, the performance of FAB-MAP drops, as shown in Figure 7(b). This leads us to guess that FAB-MAP is affected by the configuration of our cameras.

The FAB-MAP model considers the environment as "a collection of discrete and disjoint locations." For that reason, in Cummins and Newman (2008), images were taken perpendicular to the motion of the robot, so that the overlap was negligible.



(a) Indoor data set (b) Outdoor data set

Figure 7. Precision-recall curves of our BoW method for detecting loop closures, with two vocabularies, and FAB-MAP in both data sets. The chosen working values of α and p of each method are also highlighted.



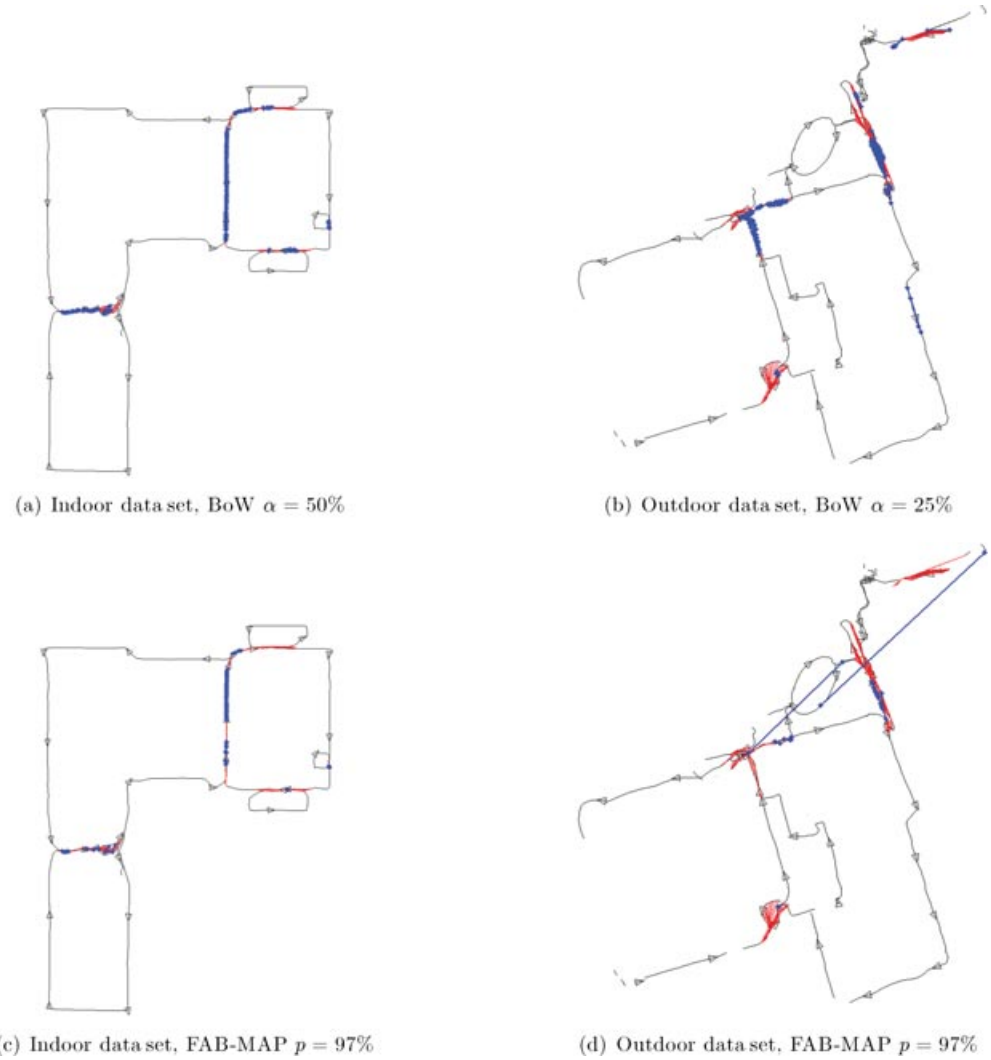
(a) Indoor input image (b) Indoor matched image

(c) Outdoor input image (d) Outdoor matched image

Figure 8. Examples of successful loop detections.

(a) Input image (b) Discarded match

Figure 9. Match correctly discarded by BoW in the indoor data set with $\alpha = 50\%$. A strict value of α allows rejection of several perceptual aliasing cases like this one.



(a) Indoor data set, BoW $\alpha = 50\%$ (b) Outdoor data set, BoW $\alpha = 25\%$

(c) Indoor data set, FAB-MAP $p = 97\%$ (d) Outdoor data set, FAB-MAP $p = 97\%$

Figure 10. Loops detected by our BoW method with $\alpha = 50\%$ and 25% and FAB-MAP with $p = 97\%$ in both data sets. Black lines and triangles denote the actual trajectory of the robot; light red lines, loops from GT, and dark blue lines, places matched.

(a) Input image (b) Matched image

Figure 11. Example of mismatch obtained by FAB-MAP in the outdoor data set with $p = 97\%$.

However, because our three cameras face forward, far-off objects (e.g., buildings) persist for many frames, thus making scenes overlap and be less discriminative.

It is easier for BoW to overcome those cases because our matching acceptance threshold ($\alpha\lambda_t$) is different for each scene and takes into account the similarity between consecutive frames.

These may be the reasons why performance of FAB-MAP is below the performance of BoW with our two vocabularies in Figure 7.

In our final system we use BoW with the RAWSEEDS 9^5 vocabulary, and we set different values of α for each data set. Figure 8 shows two examples of loops correctly detected in both data sets. In the outdoor data set, $\alpha = 25\%$ exhibits a good precision-recall balance. Indoors, a stricter value $\alpha = 50\%$ is required to avoid mismatches caused by perceptual aliasing in several similar-looking corridors, as shown in Figure 9. With these configurations and applying the epipolar constraint, we detect the loops shown in Figure 10. GT matches are marked in red, whereas the detected loop closure positions are highlighted with blue lines on the trajectory path. We also present the matches obtained by FAB-MAP with the working value $p = 97\%$, where it attains its higher precision rate (see Figure 7). We can see that BoW yields no mismatches in the indoor data set and that most of the loop closures are detected. Outdoors, the largest loops are found several times, although recall is smaller. There are also two matches between close positions, with no loop, on the right-hand side of the map [see Figure 10(b)]. When the loop-closing algorithm delivers the fundamental matrices to CI-Graph, the process performed to select robust features (detailed in Section 6.3) makes it able to discard those mismatched loops. In general, BoW detects more loop closures than FAB-MAP. Figure 11 shows one of the mismatches present in Figure 10(d).

7.2. Running Time Evaluation

We ran CI-Graph SLAM on both data sets, obtaining 86 indoor and 117 outdoor CI submaps, respectively. To control the growth of the state vector, we imposed a bound between 100 (indoor) and 150 (outdoor) features per each local submap. Additionally, we fixed a maximum uncertainty bound of 3 deg (indoor) and 5 deg (outdoor) on the main angle of rotation, yaw, so that a new map is also initialized when the estimated uncertainty on this variable goes beyond the bound, to reduce the effect of linearization errors. Each time that BoW detects a loop closure, the geometrical constraints given by the fundamental matrices are applied on the current images. If the matching process obtains eight successful pairings, the algorithm closes the loop by copying the reobserved features in the current submap along the tree path. Afterward, if the robot pose is inside the region of the old map, the robot switches to it. A total map obtained for each data set is shown in Figure 12, where each local submap is drawn in absolute coordinates.

Figure 13, second row, details the cost per step to carry out each of the main processes involved in the CI-Graph algorithm. For example, the corresponding indoor time plot (left) shows that performing EKF updates, data association, and addition of new features during the local submap building process (dark solid line) requires most of the time less than 2 s. A similar behavior is observed for the outdoor data set when building a local submap. When revisiting a previously built submap, some old features

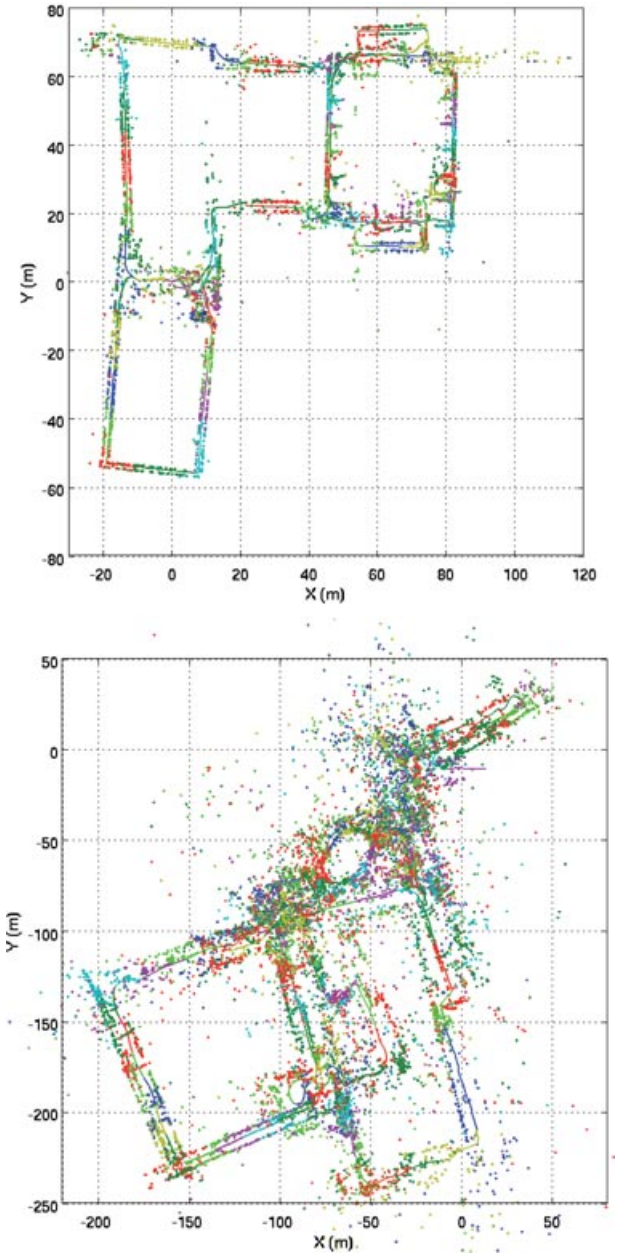


Figure 12. Final maps obtained after running CI-Graph SLAM. A total of 86 local submaps were built when the vehicle traversed the indoor environment (top). In contrast, 117 local submaps were delivered by CI-Graph SLAM when running on a larger outdoor experiment (bottom).

are reobserved but we allow the initialization of new features in the environment. This increases temporarily the cost of updating the local state vector until the vehicle exits from the submap. This effect can be seen in the time plots. Also, we can see that CI-Graph cost (blue stems) grows with the number of maps in the path that is traversed in the tree to bring common features. Notice that updating all submaps with a final backpropagation step requires 7.28 s (indoors) and 12.4 s (outdoors) in our MATLAB implementation.

The loop-closing thread runs in real time, at 0.57 s per frame on average. The maximum peak was 1.25 s on the data sets tested. This time includes extracting SURFs, inspecting the bag-of-words tree, applying the loop detection algorithm, and calculating the fundamental matrices. The most demanding step is the SURF extraction, that takes about 0.41 s per stereo image on average, whereas calculating the fundamental matrices takes 0.13 s. We use the OpenCV implementation (OpenCV, 2009) for these purposes. On the other hand, converting SURF features into words and maintaining the inverted index can be done in 28 ms with our own implementation of the hierarchical vocabulary of Nister and Stewenius (2006).

7.3. Accuracy Evaluation

The accuracy of CI-Graph SLAM is evaluated by comparing the filtered trajectory to the GT solution whenever this is available. That is, the errors at instant t are calculated by comparing the GT with the solution given by the EKF at that instant. This means that the trajectory error gets reduced drastically after a loop-closing event, as can be observed for the indoor experiment in Figure 13 at $t \approx$ 1,400 s. Figure 13, first row, shows the CI-Graph trajectory estimate (blue solid line), the odometry (green dashed line), and the GT solution (red line). They are aligned using a nonlinear optimization algorithm that calculates the most reliable transformation between the GT solution and CI-Graph trajectory. Time-stamp-solution interpolation is also performed to generate the same number of pose samples in both pose sets. We compute the absolute error produced by CI-Graph in the translation and orientation components as shown in Figure 13, third row, except for the outdoor experiment in which the GPS does not provide GT for heading. In this figure, some jumps are noticeable when the robots improves its pose estimation after a loop closure. For better illustration, all position errors are accumulated in a histogram (see Figure 13, fourth row) identifying the 2σ error bounds. According to the results, CI-Graph delivers a good precision for the indoor data set, achieving a mean absolute error of 1.64 m and ∼1.66 deg. In the outdoor experiment, after traversing 1.36 km, CI-Graph delivers a mean absolute error of 6.96 m and a maximum error of ≈17 m. These errors are mainly due to a loop closure that is missed at the end of the experiment.

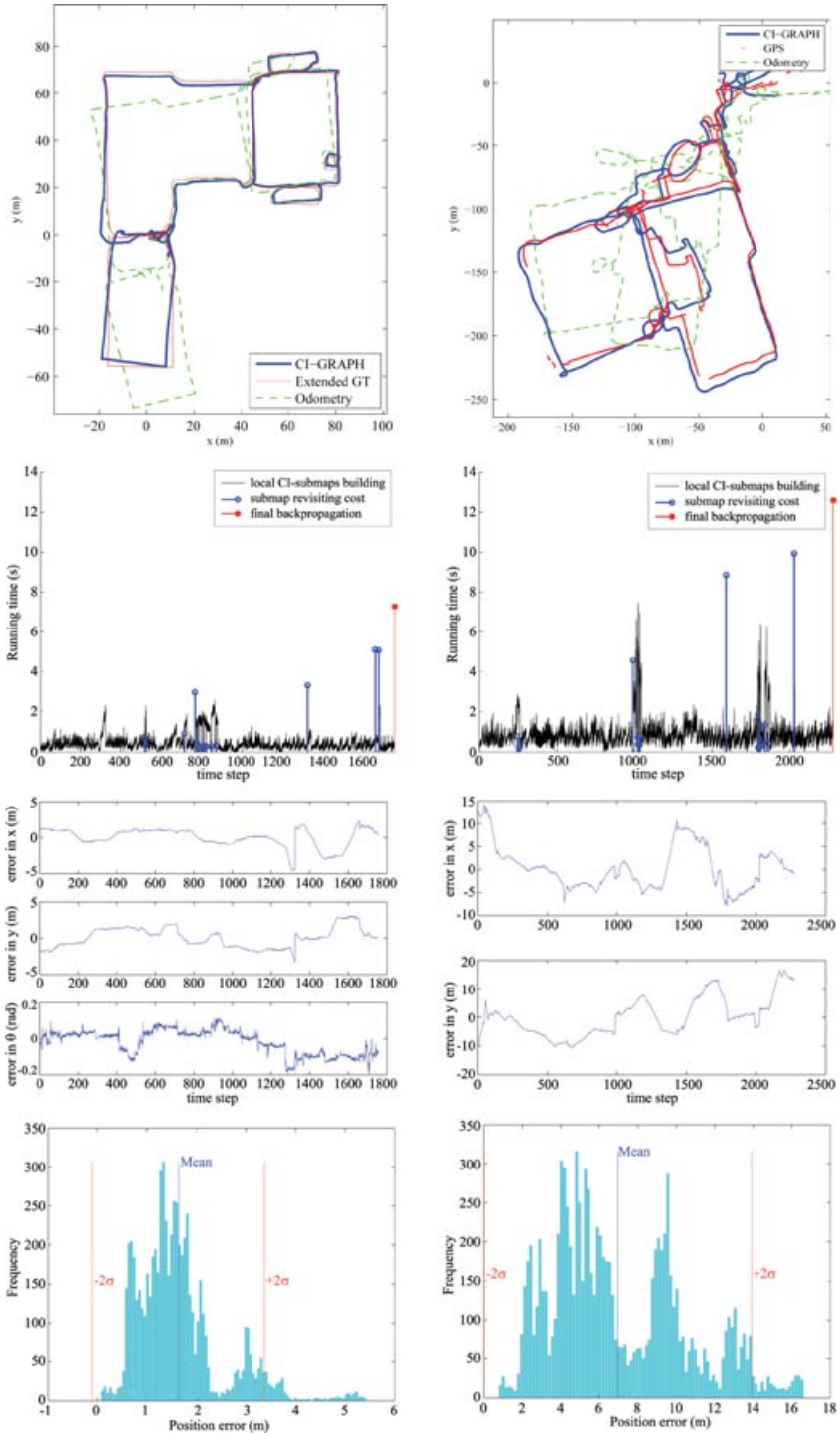


Figure 13. CI-Graph results for real indoor (left) and outdoor (right) experiments. First row: CI-Graph estimate of the filtered trajectory compared with GT and odometry. Second row: running times in each time step required to perform all the local submap operations (black solid line); blue stems represent the time of revisiting a previously built map, and the last red stem represents the time of updating all the CI submaps, except the current one, which is always up to date. Third row: absolute robot pose error. Fourth row: robot pose error histograms with 2σ error bounds.

8. CONCLUSIONS

In this paper we have presented CI-Graph, a novel submapping algorithm that allows us to efficiently solve complex map/trajectory topologies, reducing the computational cost of EKF–SLAM without introducing approximations other than the inherent linear–Gaussian approximation of the EKF. CI-Graph models the SLAM process as a Gaussian graph that evolves over time. Nodes of the graph correspond to CI submaps, and links between nodes reveal submap relations due to either robot transitions or covisible features. By building a spanning tree of the graph, we have shown that information can be shared and transmitted from map to map without losing the conditional independence property between submaps.

One of the advantages of using CI-Graph with respect to other approaches is its ability to reduce memory requirements when exploring an environment as it does not need to maintain all covariance matrix entries (correlation terms in EKF–SLAM). We have also shown with simulations the efficiency of CI-Graph SLAM to perform updates in extremely loopy environments. In the presented simulations, we have obtained an average cost per step close to linear time.

Our experimental results using a trinocular camera have shown that CI submaps are very adequate for visual SLAM. Sharing feature and camera states between CI submaps gives much better results than techniques using independent submaps, which start each new submap from scratch. The algorithm implemented treats each camera independently, which allows us to easily process as many cameras as required.

We have also presented a novel technique based on a bag of words, which runs independently of the rest of the system, to detect potential loop-closing candidates. Our results have shown that for vocabularies of similar origin and size, our technique outperforms FAB-MAP. We conjecture that this is probably due to the use of frontal cameras in our experiments.

Finally, we want to emphasize that the decomposition in submaps performed in CI-Graph is based on the intrinsic structure of the SLAM problem, independently of the underlying probability distributions or the estimation technique used. In future work, we plan to extend the CI-Graph framework to nonlinear optimization techniques such as bundle adjustment, given their superior accuracy (Strasdat, Montiel, & Davison, 2010) and increasing importance in current algorithms.

9. APPENDIX A: GAUSSIAN SUBMAPS

In this Appendix we describe the case when the probability densities of the CI submaps are Gaussians represented in covariance form. Suppose that we have built two submaps:

$$p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a) = \mathcal{N}\left(\begin{bmatrix}\hat{\mathbf{x}}_{A_a}\\ \hat{\mathbf{x}}_{C_a}\end{bmatrix}, \begin{bmatrix}P_{A_a} & P_{AC_a}\\ P_{CA_a} & P_{C_a}\end{bmatrix}\right), \quad (A.1)$$

$$p(\mathbf{x}_C, \mathbf{x}_B|\mathbf{z}_a, \mathbf{z}_b) = \mathcal{N}\left(\begin{bmatrix}\hat{\mathbf{x}}_{C_{ab}}\\ \hat{\mathbf{x}}_{B_{ab}}\end{bmatrix}, \begin{bmatrix}P_{C_{ab}} & P_{CB_{ab}}\\ P_{BC_{ab}} & P_{B_{ab}}\end{bmatrix}\right), \quad (A.2)$$

where uppercase subindexes are for state vector components and lowercase subindexes describe which observations $\mathbf{z}$ were used to obtain the estimate. For example, in the first submap, common elements $\mathbf{x}_C$ were estimated using only observations $\mathbf{z}_a$; hence, the mean and covariance estimates are denoted by $\hat{\mathbf{x}}_{C_a}$ and P_{C_a}, respectively.

The joined global map would be represented by

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C|\mathbf{z}_a, \mathbf{z}_b) = \mathcal{N}\left(\begin{bmatrix}\hat{\mathbf{x}}_{A_{ab}}\\ \hat{\mathbf{x}}_{C_{ab}}\\ \hat{\mathbf{x}}_{B_{ab}}\end{bmatrix}, \begin{bmatrix}P_{A_{ab}} & P_{AC_{ab}} & P_{AB_{ab}}\\ P_{CA_{ab}} & P_{C_{ab}} & P_{CB_{ab}}\\ P_{BA_{ab}} & P_{BC_{ab}} & P_{B_{ab}}\end{bmatrix}\right). \quad (A.3)$$

9.1. Backpropagation

From the submap CI property, we know that

$$p(\mathbf{x}_A|\mathbf{z}_a, \mathbf{z}_b, \mathbf{x}_C) = p(\mathbf{x}_A|\mathbf{z}_a, \mathbf{x}_C) = \mathcal{N}(\hat{\mathbf{x}}_{A|C}, P_{A|C}). \quad (A.4)$$

The conditional distribution $p(\mathbf{x}_A|\mathbf{z}_a, \mathbf{z}_b, \mathbf{x}_C)$ can be obtained from the global map by marginalizing out $\mathbf{x}_B$ and conditioning on $\mathbf{x}_C$, obtaining

$$\hat{\mathbf{x}}_{A|C} = \hat{\mathbf{x}}_{A_{ab}} + P_{AC_{ab}} P_{C_{ab}}^{-1}(\mathbf{x}_C - \hat{\mathbf{x}}_{C_{ab}}), \quad (A.5)$$

$$P_{A|C} = P_{A_{ab}} - P_{AC_{ab}} P_{C_{ab}}^{-1} P_{CA_{ab}}. \quad (A.6)$$

The conditional probability $p(\mathbf{x}_A|\mathbf{z}_a, \mathbf{x}_C)$ can also be obtained from the first map by conditioning on $\mathbf{x}_C$, which gives

$$\hat{\mathbf{x}}_{A|C} = \hat{\mathbf{x}}_{A_a} + P_{AC_a} P_{C_a}^{-1}(\mathbf{x}_C - \hat{\mathbf{x}}_{C_a}), \quad (A.7)$$

$$P_{A|C} = P_{A_a} - P_{AC_a} P_{C_a}^{-1} P_{CA_a}. \quad (A.8)$$

Because Eqs. (A.5) and (A.7) must be equal for all $\mathbf{x}_C$, this means that

$$P_{AC_a} P_{C_a}^{-1} = P_{AC_{ab}} P_{C_{ab}}^{-1}, \quad (A.9)$$

we will call this equality K.

From K we can calculate the value of $P_{AC_{ab}}$ and from Eqs. (A.5) and (A.7) the updated value of the mean $\hat{\mathbf{x}}_{A_{ab}}$. Finally, from previous results and by equating Eqs. (A.6) and (A.8), we can calculate the updated value of $P_{A_{ab}}$. As a result, we obtain the following backpropagation equations:

$$K = P_{AC_a} P_{C_a}^{-1} = P_{AC_{ab}} P_{C_{ab}}^{-1} \quad (A.10)$$

$$P_{AC_{ab}} = K P_{C_{ab}}, \quad (A.11)$$

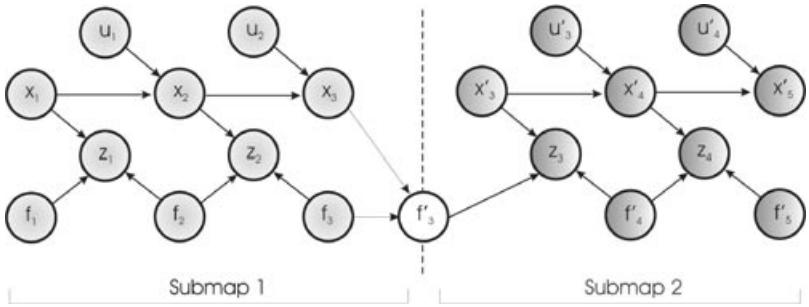


Figure B.1. Bayesian network that illustrates the process followed to generate a local map with its own base reference.

$$P_{A_{ab}} = P_{A_a} + K(P_{CA_{ab}} - P_{CA_a}), \tag{A.12}$$

$$\hat{\mathbf{x}}_{A_{ab}} = \hat{\mathbf{x}}_{A_a} + K(\hat{\mathbf{x}}_{C_{ab}} - \hat{\mathbf{x}}_{C_a}), \tag{A.13}$$

which allows us to obtain an update of $\mathbf{x}_A$, P_A and P_{AC} without having to join the maps or calculate P_{AB}.

9.2. Computing the Correlation between Submaps

If we want to calculate the correlation term $P_{AB_{ab}}$, we can do the following. We first obtain the expression of the covariance of $p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{z}_a, \mathbf{z}_b, \mathbf{x}_C)$ by conditioning the global map on $\mathbf{x}_C$:

$$\begin{bmatrix} P_{A_{ab}} - P_{AC_{ab}} P_{C_{ab}}^{-1} P_{CA_{ab}} & P_{AB_{ab}} - P_{AC_{ab}} P_{C_{ab}}^{-1} P_{CB_{ab}} \\ P_{BA_{ab}} - P_{BC_{ab}} P_{C_{ab}}^{-1} P_{CA_{ab}} & P_{B_{ab}} - P_{BC_{ab}} P_{C_{ab}}^{-1} P_{CB_{ab}} \end{bmatrix}. \tag{A.14}$$

Owing to the submap CI property, we know that $\mathbf{x}_A$ and $\mathbf{x}_B$ are conditionally independent given $\mathbf{x}_C$ and, therefore, the correlation term in Eq. (A.14) must be zero, which gives the following expression for the correlation term:

$$\begin{aligned} P_{AB_{ab}} &= P_{AC_{ab}} P_{C_{ab}}^{-1} P_{CB_{ab}} \\ &= K P_{CB_{ab}}. \end{aligned} \tag{A.15}$$

10. APPENDIX B: IMPLEMENTATION OF LOCAL MAPS

In this Appendix we describe how to build sequences of conditionally independent local maps, each with its own local base reference. Let us return to the example of Figure 1. With absolute maps, the last vehicle position $\mathbf{x}_3$ and feature $\mathbf{f}_3$ were chosen to initialize submap 2 in order to represent both maps with respect to the same reference and take advantage of the available estimation of the vehicle and the feature. Instead, we now want to represent submap 2 with respect to a local reference given by the current vehicle position $\mathbf{x}_3$ and still use the information about feature $\mathbf{f}_3$ in submap 2. For doing so in a consistent way, a copy of feature $\mathbf{f}_3$ expressed in the new reference must be calculated and included in submap 1. In the following, a prime will be used to denote entities relative to the new base reference:

$$\mathbf{f}'_3 = \ominus \mathbf{x}_3 \oplus \mathbf{f}_3. \tag{B.1}$$

After this process, the pdf that describes submap 1 is

$$p(\mathbf{x}_{1:3}, \mathbf{f}_{1:3}, \mathbf{f}'_3|\mathbf{z}_{1:2}, \mathbf{u}_{1:2}). \tag{B.2}$$

The new local map will start with robot position $\mathbf{x}'_3$ being exactly zero. Obviously this variable is completely independent of submap 1. By marginalizing Eq. (B.2), we obtain the pdf that describes the initial state of submap 2:

$$p(\mathbf{f}'_3|\mathbf{z}_{1:2}, \mathbf{u}_{1:2}). \tag{B.3}$$

Once the vehicle has traversed the second submap and has incorporated all observations gathered in it, the pdf associated with the final estimate of submap 2 is

$$p(\mathbf{x}'_{4:5}, \mathbf{f}'_{3:5}|\mathbf{z}_{1:4}, \mathbf{u}_{1:4}). \tag{B.4}$$

Figure B.1 shows the Bayesian network that corresponds to the new algorithm. As can be seen, the structure of the network is the same as in Figure 1, bottom. The only difference is that the part shared by both maps $\mathbf{x}_C$ corresponds in this case to the local representation of feature $\mathbf{f}'_3$. As a consequence, the submap CI property is valid for local submaps as well as for absolute submaps.

ACKNOWLEDGMENTS

This research has been funded in part by the European Union under projects RAWSEEDS FP6-IST-045144 and RoboEarth FP7-ICT-248942, the Dirección General de Investigación of Spain under projects DPI2009-13710 and DPI2009-07130, DGA (CONSI+D)-CAI under Programa Europa, and the Ministerio de Educación of Spain with scholarship FPU-AP2008-02272.

REFERENCES

Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In SODA '07: Proceedings of the Eighteenth Annual ACM–SIAM Symposium on Discrete Algorithms, New Orleans, LA (pp. 1027–1035). Philadelphia, PA: Society for Industrial and Applied Mathematics.

Bailey, T. (2003, September). Constrained initialisation for bearing-only SLAM. In Proceedings IEEE International

Conference Robotics and Automation, ICRA, Taipei, Taiwan (vol. 2, pp. 1966–1971).

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). SURF: Speeded up robust features. Computer Vision and Image Understanding, 110(3), 346–359.

Bishop, C. M. (2006). Pattern recognition and machine learning, 2nd ed. New York: Springer.

Bosse, M., Newman, P. M., Leonard, J. J., Soika, M., Feiten, W., & Teller, S. (2003, September). An Atlas framework for scalable mapping. In Proceedings IEEE International Conference Robotics and Automation, ICRA, Taipei, Taiwan (pp. 1899–1906).

Castellanos, J., Martinez-Cantin, R., Tardós, J., & Neira, J. (2007). Robocentric map joining: Improving the consistency of EKF–SLAM. Robotics and Autonomous Systems, 55(1), 21–29.

Castellanos, J. A., & Tardós, J. D. (1999). Mobile robot localization and map building: A multisensor fusion approach. Boston, MA: Kluwer Academic Publishers.

Civera, J., Davison, A. J., & Montiel, J. M. M. (2007, April). Inverse depth to depth conversion for monocular SLAM. In Proceedings IEEE International Conference Robotics and Automation, ICRA, Rome, Italy (pp. 2778–2783).

Civera, J., Davison, A. J., & Montiel, J. M. M. (2008). Inverse depth parametrization for monocular SLAM. IEEE Transactions on Robotics, 24(5), 932–945.

Clemente, L., Davison, A. J., Reid, I. D., Neira, J., & Tardós, J. D. (2007, June). Mapping large loops with a single handheld camera. In Proceedings Robotics: Science and Systems, Atlanta, GA.

Cormen, T. H., Leiserson, C., & Rivest, R. L. (2001). Introduction to algorithms, 2nd ed. Cambridge, MA: MIT Press.

Cummins, M., & Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. International Journal of Robotics Research, 27(6), 647–665.

Davison, A. J. (2003, October). Real-time simultaneous localisation and mapping with a single camera. In Proceedings International Conference Computer Vision, Nice, France (p. 1403).

Davison, A. J., & Murray, D. W. (2002). Simultaneous localization and map-building using active vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7), 865–880.

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

Deans, M., & Hebert, M. (2000, December). Experimental comparison of techniques for localization and mapping using a bearing-only sensor. In S. S. D. Rus (Ed.), International Symposium on Experimental Robotics (ISER'00). Lecture Notes in Control and Information Science, Honolulu, HI (vol. 271, pp. 395–404). Springer-Verlag.

Dellaert, F., & Kaess, M. (2006). Square root SAM: Simultaneous localization and mapping via square root information smoothing. International Journal of Robotics Research, 25(12), 1181–1203.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. IEEE Robotics & Automation Magazine, 13(2), 99–110.

Eade, E., & Drummond, T. (2007, October). Monocular SLAM as a graph of coalesced observations. In Proceedings 11th IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil (pp. 469–476).

Eade, E., & Drummond, T. (2008, September). Unified loop closing and recovery for real time monocular SLAM. In British Machine Vision Conference, BMVC 08, Leeds, UK.

Estrada, C., Neira, J., & Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. IEEE Transactions on Robotics, 21(4), 588–596.

Eustice, R. M., Singh, H., & Leonard, J. J. (2006). Exactly sparse delayed-state filters for view-based SLAM. IEEE Transactions on Robotics, 22(6), 1100–1114.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381–395.

Fitzgibbons, T., & Nebot, E. (2002, November). Bearing only SLAM using colour-based feature tracking. In 2002 Australasian Conference on Robotics and Automation, Auckland, New Zealand.

Folkesson, J., & Christensen, H. (2006). Graphical SLAM for outdoor applications. Journal of Field Robotics, 23(1), 51–70.

Frese, U. (2006). Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. Autonomous Robots, 21(2), 103–122.

Gil, A., Reinoso, O., Martínez-Mozos, O., Stachniss, C., & Burgard, W. (2006, October). Improving data association in vision-based SLAM. In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06), Beijing, China.

Grisetti, G., Stachniss, C., & Burgard, W. (2009). Non-linear constraint network optimization for efficient map learning. IEEE Transactions on Intelligent Transportation Systems, 10(3), 428–439.

Hartley, R. I. (1997). In defense of the eight-point algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6), 580–593.

Huang, S., Wang, Z., & Dissanayake, G. (2008). Sparse local submap joining filters for building large-scale maps. IEEE Transactions on Robotics, 24(5), 1121–1130.

Hygounenc, E., Jung, I., Soueres, P., & Lacroix, S. (2004). The autonomous blimp project of LAAS–CNRS: Achievements in flight control and terrain mapping. International Journal of Robotics Research, 23(4), 473–511.

Iocchi, L., Konolige, K., & Bajracharya, M. (2000, December). Visually realistic mapping of a planar environment with stereo. In Proceedings International Symposium on Experimental Robotics (ISER'00), Honolulu, HI.

Jensfelt, P., Kragic, D., Folkesson, J., & Bjorkman, M. (2006, May). A framework for vision based bearing only 3D SLAM. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'06), Orlando, FL (pp. 1945–1950).

Julier, S. J., & Uhlmann, J. K. (2001, March). A counter example to the theory of simultaneous localization and map building. In Proceedings IEEE International Conference Robotics and Automation, ICRA, Seoul, Korea (pp. 4238–4243).

Jung, I., & Lacroix, S. (2003, October). High resolution terrain mapping using low altitude aerial stereo imagery. In Proceedings of the 9th International Conference on Computer Vision, Nice, France (pp. 946–951).

Kaess, M. (2008). Incremental smoothing and mapping. Ph.D. thesis, Georgia Institute of Technology.

Klein, G., & Murray, D. W. (2007, November). Parallel tracking and mapping for small AR workspaces. In Proceedings Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan (pp. 1–10).

Klein, G., & Murray, D. W. (2008, October). Improving the agility of keyframe-based SLAM. In Proceedings 10th European Conference on Computer Vision, Marseille, France (pp. 802–815).

Konolige, K., & Agrawal, M. (2008). FrameSLAM: From bundle adjustment to real-time visual mapping. IEEE Transactions on Robotics, 24(5), 1066–1077.

Kwok, N., & Dissanayake, G. (2004, October). An efficient multiple hypothesis filter for bearing-only SLAM. In Proceedings IEEE/RJS International Conference on Intelligent Robots and Systems, Sendai, Japan (vol. 1).

Lemaire, T., Lacroix, S., & Sola, J. (2005, August). A practical 3D bearing-only SLAM algorithm. In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05), Edmonton, Canada (pp. 2449–2454).

Leonard, J., & Newman, P. (2003, August). Consistent, convergent and constant-time SLAM. In International Joint Conference Artificial Intelligence, Acapulco, Mexico (pp. 1143–1150).

Leonard, J. J., & Feder, H. J. S. (2000, October). A computationally efficient method for large-scale concurrent mapping and localization. In D. Koditschek and J. Hollerbach (Eds.), Robotics research: The Ninth International Symposium, Snowbird, UT (pp. 169–176). Springer Verlag.

Lepetit, V., & Fua, P. (2006). Keypoint recognition using randomized trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(9), 1465–1479.

Mei, C., Sibley, G., Cummins, M., Newman, P., & Reid, I. (2009, September). A constant time efficient stereo SLAM system. In British Machine Vision Conference, BMVC 09, London, UK.

Nister, D., & Stewenius, H. (2006, June). Scalable recognition with a vocabulary tree. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York (vol. 2, pp. 2161–2168).

Olson, E. (2009). Recognizing places using spectrally clustered local matches. Robotics and Autonomous Systems, 57(12), 1157–1172.

OpenCV (2009). OpenCV 2.0: Open Source Computer Vision Library. http://opencv.willowgarage.com/wiki/.

Paskin, M. A. (2003, August). Thin junction tree filters for simultaneous localization and mapping. In International Joint Conference Artificial Intelligence, Acapulco, Mexico (pp. 1157–1164).

Paz, L. M. (2008). Divide and conquer: EKF SLAM in $O(n)$. Ph.D. thesis, Universidad de Zaragoza.

Paz, L. M., Piniés, P., Tardós, J. D., & Neira, J. (2008a). Large scale 6DOF SLAM with stereo in hand. IEEE Transactions on Robotics, 24(5), 946–957.

Paz, L. M., Tardós, J. D., & Neira, J. (2008b). Divide and conquer: EKF SLAM in $O(n)$. Transactions on Robotics, 24(5), 1107–1121.

Piniés, P., Paz, L. M., & Tardós, J. D. (2009, May). CI-Graph: An efficient approach for large scale SLAM . In Proceedings of the IEEE International Conference Robotics and Automation (ICRA'09), Kobe, Japan (pp. 3913–3920).

Piniés, P., & Tardós, J. D. (2008). Large scale SLAM building conditionally independent local maps: Application to monocular vision. Transactions on Robotics, 24(5), 1094–1106.

RAWSEEDS (2009). Robotics advancement through Web-publishing of sensorial and elaborated extensive data sets (project FP6-IST-045144). http://www.rawseeds.org/rs/datasets.

Saez, J., Escolano, F., & Penalver, A. (2005, June). First steps towards stereo-based 6DOF SLAM for the visually impaired. In Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops-Volume 03, San Diego, CA, Washington, DC: IEEE Computer Society.

Se, S., Lowe, D., & Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. International Journal of Robotics Research, 21(8), 735–758.

Sibley, G., Mei, C., Reid, I., & Newman, P. (2009, June). Adaptive relative bundle adjustment. In Proceedings Robotics: Science and Systems, Seattle, WA (vol. 5). Cambridge, MA: MIT Press.

Sivic, J., & Zisserman, A. (2003, October). Video Google: A text retrieval approach to object matching in videos. In Proceedings of the International Conference on Computer Vision, Nice, France (vol. 2, pp. 1470–1477).

Smith, R., Self, M., & Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In O. Faugeras and G. Giralt (Eds.), Robotics research, The Fourth International Symposium, Santa Clara, CA (pp. 467–474). Cambridge, MA: MIT Press.

Sola, J., Monin, A., Devy, M., & Lemaire, T. (2005, August). Undelayed initialization in bearing only SLAM. In Proceedings IEEE/RJS International Conference on Intelligent Robots and Systems, Edmonton, Canada (pp. 2499–2504).

Sola, J., Monin, A., Devy, M., & Vidal-Calleja, T. (2008). Fusing monocular information in multicamera SLAM. IEEE Transactions on Robotics, 24(5), 958–968.

Strasdat, H., Montiel, J. M. M., & Davison, A. J. (2010, May). Real-time monocular SLAM: Why filter? In 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK.

Tardós, J. D., Neira, J., Newman, P. M., & Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. International Journal of Robotics Research, 21(4), 311–330.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. Cambridge, MA: MIT Press.

Tuytelaars, T., & Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. Foundations and Trends in Computer Graphics and Vision, 3(3), 177–280.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning, 1(1–2), 1–305.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., & Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. Robotics and Autonomous Systems, 57(12), 1188–1197.

Williams, B., Klein, G., & Reid, I. (2007, October). Real-time SLAM relocalisation. In Proceedings International Conference on Computer Vision, Rio de Janeiro, Brazil.

Williams, S. B., Dissanayake, G., & Durrant-Whyte, H. (2002, May). An efficient approach to the simultaneous localisation and mapping problem. In Proceedings IEEE International Conference Robotics and Automation ICRA, Washington, DC (vol. 1, pp. 406–411).