Stereo CI-Graph SLAM

Benchmark Solutions

Authors: Lina M. Paz, Pedro Piniés,
Dorian Gálvez and Juan D. Tardós

date: end of month 35 (September 2009)

1 Introduction

In this work we describe our multicamera SLAM system which integrates a set of novel technologies allowing us to gather most of the information available in the images. We consider information from features both close and far from the cameras Paz et al. [2008]. Given a bunch of cameras and the rigid transformation between them, 3D information from nearby scene points can be obtained, at the same time each camera can also provide bearing only information from distant scene points. Both types of information are relevant to obtain good estimates of the translation and the attitude of the camera system. The first main contribution of the proposed method is that it can easily deal with any number of cameras since each camera is treated independently.

The second main contribution is a novel SLAM algorithm that allows us to efficiently build maps of large environments when the camera system follows complex trajectories. Our algorithm is able to operate in large environments by decomposing the whole map in local-maps of limited size. Instead of building independent submaps we build conditionally Independent submaps Piniés and Tardós [2008], which allow the system to share both camera velocity information and current feature information during local map initialization. This adds robustness to the system without sacrificing precision or consistency in any way. Finally, by using the CI-Graph algorithm Piniés et al. [2009] we can extend the properties of the CI-submaps to more complex robot trajectories and map topologies.

We validated the approach on the proposed indoor dataset Bicooca_2009-02-25b for benchmarking which has been described and tested in Deliverable 3.2 of the RAWSEEDS project. Due to the reported lack of texture in the corresponding dataset validation, we integrate odometry readings to drive the stereo system in those places where features can not be extracted from the images.

1.1 CI-Graph Algorithm Description

In order 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 $\mathcal{E}_\mathcal{G}$ are its undirected *edges*. 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 the other submap.

In addition, the algorithm builds a spanning tree $\mathcal{T}(\mathcal{N}, \mathcal{E}_\mathcal{T})$ of the graph $\mathcal{G}$, 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 $\mathcal{G}$ which 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, sharing some vehicle and feature states. Each edge in $\mathcal{E}_\mathcal{T}$ will be labeled with the corresponding shared states. Given any pair of submaps, $\mathbf{m}_i$ and $\mathbf{m}_j$, there is a unique path in $\mathcal{T}$ connecting them. This path allows us to transmit information from map to map without loosing the conditional independence property between submaps. In figure 1, spanning tree edges $\mathcal{E}_\mathcal{T}$ will be depicted using a continuous line while the remaining edges of $\mathcal{G}$, i.e. $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$, will be traced with a dashed line.

Two operational levels can be distinguished in the algorithm. Local operations that are only applied 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 corresponding to standard EKF-SLAM equations. Graph operations are more sporadic and can be considered as the interface between CI-submaps. In the following subsections, the graph operations are explained in detail as presented in Algorithm 1.

1.1.1 Starting a new submap

Suppose that robot is in submap $\mathbf{m}_i$ and we decide to start a new submap $\mathbf{m}_j$. The steps followed in the algorithm are:

- 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 $\mathbf{m}_j$

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 1. At time k_2, submaps $\mathbf{m}_1$ and $\mathbf{m}_2$ have been already explored and a new submap is being created $\mathbf{m}_3$. 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.

1.1.2 Re-observing 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 $\mathbf{m}_j$. The process followed is:

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

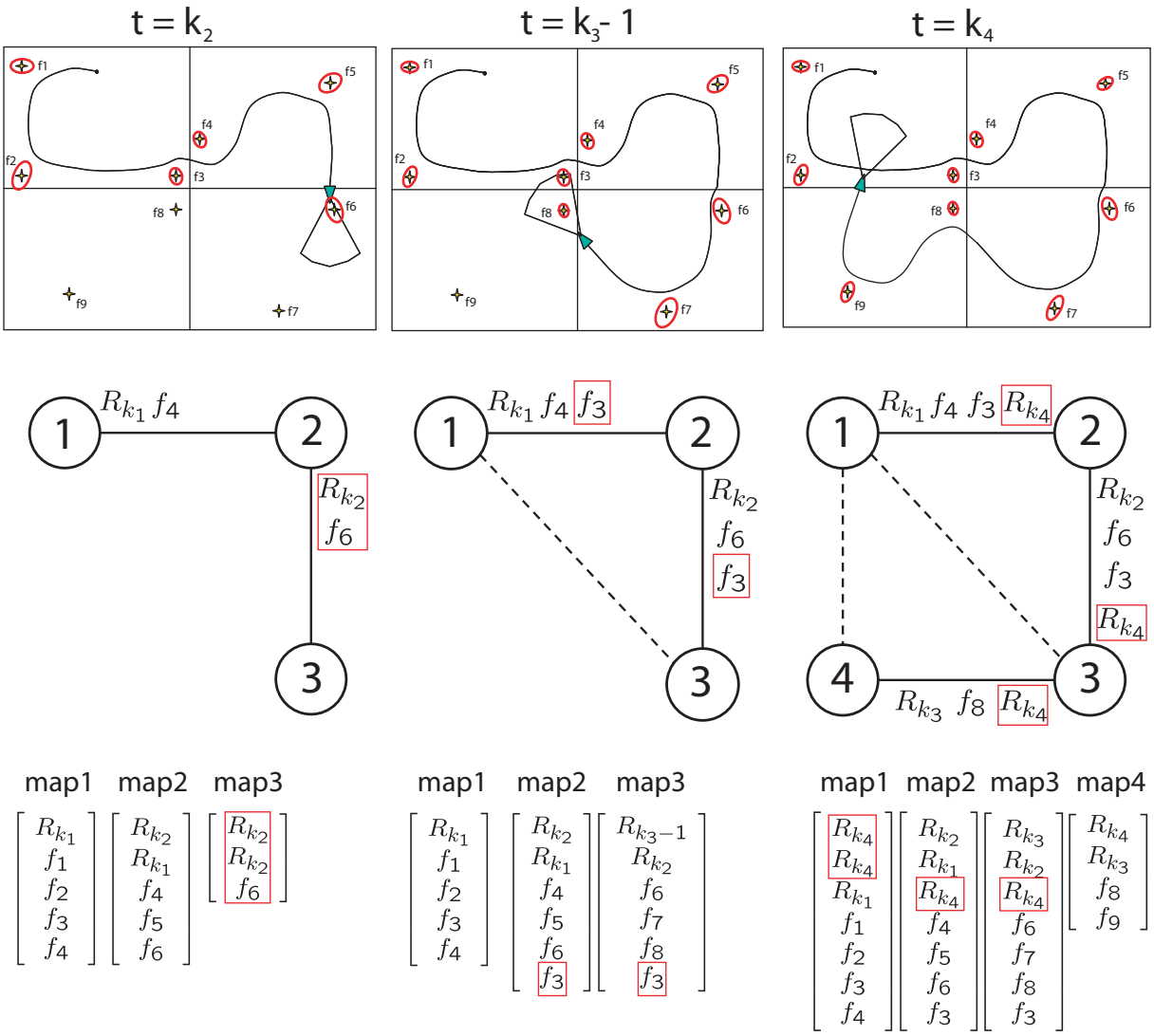

Figure 1: Example using CI-Graph SLAM. The figure is divided in three rows that show information about the state of a simulated experiment at three different instants of time (columns). In the first row, the map of the simulated environment with the current robot position is shown. In the second row, the graph of relations between submaps will be created according to the state of the estimation. In the last row we will show the state vectors of the estimated submaps at different moments of time.

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**
  $\mathbf{u}_{k-1}, \mathbf{Q}_{k-1} = getOdometry$
  $\mathbf{m}_i = ekfPrediction(\mathbf{m}_i, \mathbf{u}_{k-1}, \mathbf{Q}_{k-1})$
  $\mathbf{z}_k, \mathbf{R}_k = getObservations$
  $\mathcal{DA}_k = dataAssociation(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k)$
  if revisiting $\mathbf{m}_j$ **then**
    $\{$*Subsection 1.1.3*$\}$
    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 1.1.1*$\}$
    $addNode(\mathbf{m}_j, \mathcal{N})$
    $addEdge(\langle \mathbf{m}_i, \mathbf{m}_j \rangle, \mathcal{E}_\mathcal{T})$
    $copyRobot(\mathbf{m}_i, \mathbf{m}_j)$
    $copyActiveFeat(\mathbf{m}_i, \mathbf{m}_j)$
    $i = j$ $\{$*Map change*$\}$
  end if
  if reobserved $\mathbf{f} \notin \mathbf{m}_i$ & $\mathbf{f} \in \mathbf{m}_j$ **then**
    $\{$*Subsection 1.1.2*$\}$
    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
    $addEdge(\langle \mathbf{m}_j, \mathbf{m}_i \rangle, \mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T})$
  end if
  $\mathbf{m}_i = ekfUpdate(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k, \mathcal{DA}_k)$
  $\mathbf{m}_i = addNewFeatures(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k, \mathcal{DA}_k)$
end for
$\{$*Subsection 1.1.4*$\}$
$updateAllMaps(\mathbf{m}_i, \mathcal{T})$ $\{$*Updates* $\mathcal{T}$ *starting from* $\mathbf{m}_i\}$

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 *back-propagation* equations and the correlations with the elements of $\mathbf{m}_l$ are also calculated Piniés and Tardós [2008].

Figure 1 at time $k_3 - 1$ shows an example of this case. Feature f_3 that belongs to submap $\mathbf{m}_1$ is measured by the robot when it is traversing submap $\mathbf{m}_3$. Since 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 $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$.

1.1.3 Revisiting a previous submap

When the algorithm detects that the robot revisits an already traversed area $\mathbf{m}_j$, 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 $\mathbf{m}_j$
- Copy the current robot pose along all nodes of the path
- Add $\langle \mathbf{m}_i, \mathbf{m}_j \rangle$ to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$

As in the previous subsection, to update submaps in the path we use the *back-propagation* equations and to copy the current robot pose, correlations with submaps elements are calculated as well.

Figure 1 at time k_4 shows an example of this operation. When the robot makes a transition between submaps $\mathbf{m}_4$ and $\mathbf{m}_1$, 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.

1.1.4 Updating all maps from the current submap

Using the Graph operations just described, we can assure that the current submap is always updated with all available information. In addition, the CI property between submaps is preserved. An interesting property of the back-propagation equations is that they can be applied at any moment. They work correctly even if we back-propagate twice the same information. This allows us to schedule the back-propagation in moments with low CPU loads, or when graph operations are required. If the whole map has to be updated, the *back-propagation* equations are recursively applied starting from the current node and following the spanning tree $\mathcal{T}$.

1.2 Working with several cameras

In order to allow the system to easily scale with the number of cameras, each camera is treated independently in the algorithm. The only interaction between cameras is when a new feature is initialized. The feature is first initialized in one of the cameras. Since no depth information is available, this feature is included in the state vector using inverse depth parametrization Civera et al. [2008]. Using the known relative transformation between cameras, the recently introduced feature is predicted and searched in the images of the other cameras and correspondingly updated when found. The rigid transformation between the 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 in the map independently. In addition, to improve the computational cost of the algorithm, inverse depth features are transformed to 3D cartesian parametrization according to the parallax index explained in Civera et al. [2007].

1.3 Appearance-based loop closing

In order to close loops in the trajectory, a visual procedure is used. This method consists of three stages: first, one image per second is acquired from the stereo camera and converted into an appearance-based representation; then, it is checked if the current scene was seen before, so that a loop is detected, and finally, the loop is closed by obtaining a transformation between the robot's poses by solving a perspective-n-point problem.

1.3.1 Appearance-based representation

An appearance-based representation of an image is obtained by using a visual bag of words Sivic and Zisserman [2003]. This is a technique that represents an image by using a numeric vector created from the local features this contains. We use *SURF* points Bay et al. [2008] as image features. A *SURF* feature is a point in the image associated to a real 64-dimensional descriptor which summarizes the distribution of the intensity content within the point neighbourhood.

The bag of words technique 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 i-th 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 very suitable for managing big amounts of images; moreover, Nister and Stewenius [2006] presents a hierarchical version which 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. This way, the comparisons for converting an image descriptor into a visual word only need to be done in a branch and not in the whole discretized space, shrinking the search complexity logarithmically.

We use a hierarchical vocabulary with $k = 9$, $L = 6$ and the *kmeans++* algorithm Arthur and Vassilvitskii [2007] as clustering function. This vocabulary was created from a set of 1300 images obtained from the *Mixed / Bovisa_2008-09-01_Static* dataset. These images represent indoor and outdoor scenes, so that the vocabulary is generic enough to be used with any other dataset.

1.3.2 Loop detection

The loop detection procedure runs independently of the rest of the system, at a frequency of 1Hz. In order to detect a loop, one stereo pair is acquired at time t. The image from the left camera is converted into its vector representation, named v_t. This vector is compared with the set of all the vectors of the images obtained before, W, to check if any of them is similar enough to consider both scenes the same. If there is a satisfactory match with some vector $w_{t'} \in W$ acquired at time $t' \leq t - c$, a loop may be found. To confirm it, there must be consistency with the images previously matched. The constant c is the time interval that must pass to consider an already seen scene as revisited; it is set to 20 seconds. Finally, the current vector v_t is added to the list of already visited scene vectors W.

To make vector similarity comparisons faster, an inverted file is maintained. This file keeps a record of in which vectors each visual word is present. This way, when the vector v_t is going to be compared with vectors from W, comparisons are only made with those vectors $w_{t'} \in W$ which have at least one visual word in common with v_t. When the vector v_t is added to W, the inverted file is updated by including v_t in the lists of the visual words it contains.

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

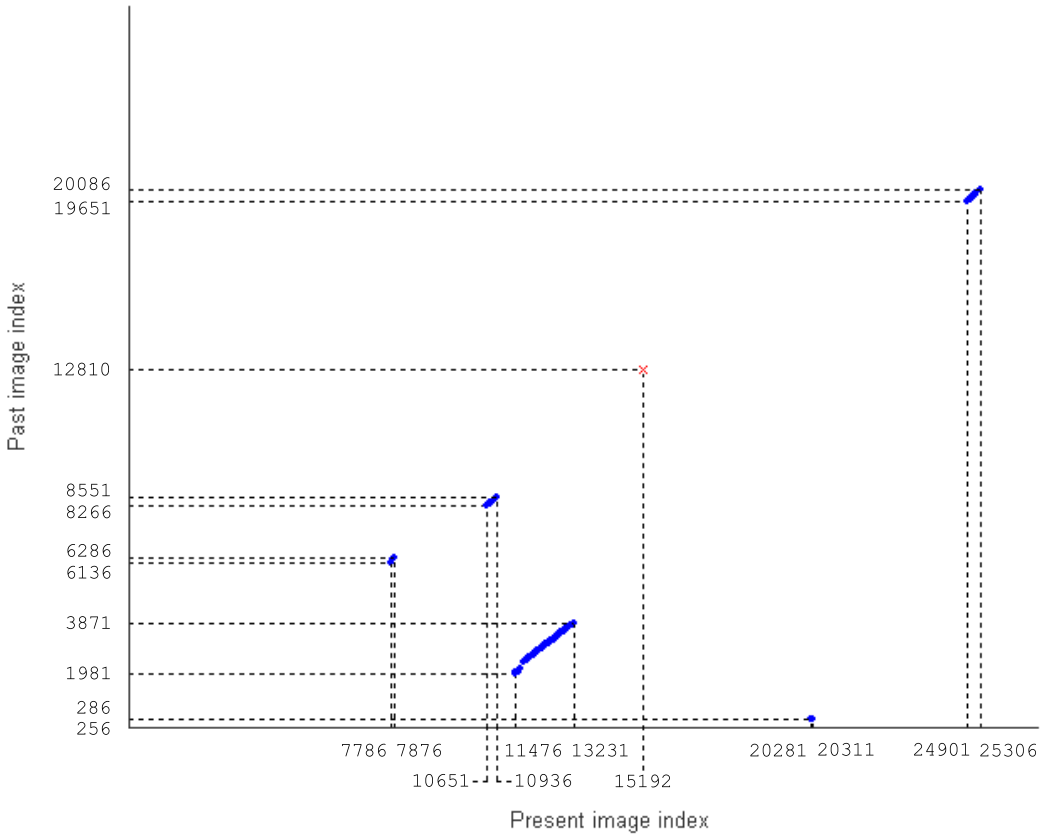

Figure 2: Indices of images matched by the loop detection algorithm (dots), and undetected loops (crosses).

[2006]:

$$s(v_t, w_{t'}) = 1 - \frac{|| \frac{v_t}{|| v_t ||} - \frac{w_{t'}}{|| w_{t'} ||} ||}{2} \quad (1)$$

We use the L_1-norm to compute this score, so that it is defined between 0 (completely different words) and 1 (perfect match). Vectors from matches $< v_t,\ w_{t'} >$ whose score is above a threshold $\lambda = 0.037$ are likely to come from images that represent the same scene, so they are considered loop candidates. The rest of the matches are discarded.

We impose a temporal constraint to reliably detect loops and to avoid mismatches. A loop in a place visited at time t and t'_0 is detected if there is a match $< v_t,\ w_{t'_0} >$, as well as previous matches $< v_{t-1},\ w_{t'_1} >$, ..., $< v_{t-N+1},\ w_{t'_{N-1}} >$ such that $\max(|t'_0 - t'_1|, \ldots, |t'_{N-2} - t'_{N-1}|) \leq 2$ seconds. The constant N is the minimum time duration of the loop trajectory to be found, and is set to 3 seconds.

Figure 2 shows the images from the left hand side of the stereo camera matched by the loop detection algorithm on the *Indoor / Bicocca_2009-02-25b* dataset. Each group of close dots represents each one of the five loops detected in the whole trajectory. Figure 3 shows two of the images matched in the first loop. There is a sixth loop in this dataset, marked with a cross in the graph, which cannot be detected. This is due to a limitation of this appearance-based approach: it cannot handle places seen from very different points of view. Figure 4 illustrates this difficulty with the non-matched images from the missed loop.

(a) Input image (7786)

(b) Matched image (6136)

Figure 3: Example of a successful loop detection

(a) Input image (15192)

(b) Expected match (12810)

Figure 4: The appearance-based approach cannot detect a loop with these images, acquired in very close positions but from different points of view.

1.3.3 Loop closing

Once a loop is detected at time t and we know that the current place was previously visited at time t', the loop is closed by finding a transformation between the current robot's pose and the one at time t'.

For that, the two images from the current stereo pair, I_t^1 and I_t^2, and one of the images from the stereo pair acquired at t', $I_{t'}$, are searched for *SURF* features. Those features which are not present in the three images are removed. The rest of the features are be reconstructed in the 3D space by using stereo triangulation and their pixel coordinates in I_t^1 and I_t^2. This way, we obtain the set of points in the space in the current camera reference, C_t. Given those points and their projections in $I_{t'}$, we can find the transformation ${}^{C_t}T_{C_{t'}}$ by solving the perspective-n-point (PnP) problem Moreno-Noguer et al. [2007]. This problem consists of estimating the pose of a calibrated camera from n 3D-to-2D point correspondences. After obtaining ${}^{C_t}T_{C_{t'}}$, the constant transformation between the robot and the stereo camera can be applied to find the final transformation between robot's poses at t and t', and successfully close the loop.

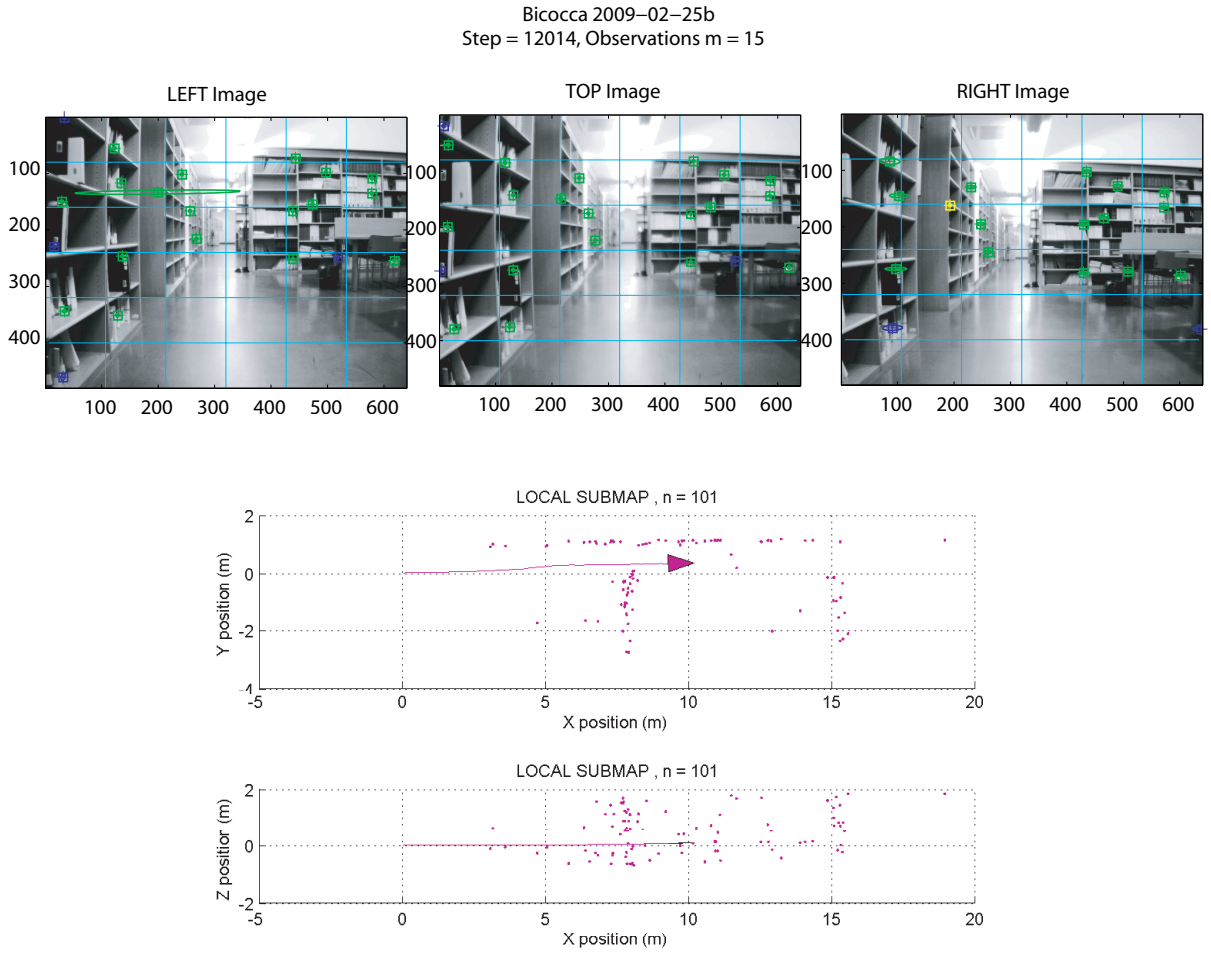

Figure 5: SLAM system performing trinocular tracking (top). Top and lateral views of the local submap reconstruction (bottom).

1.4 Stereo SLAM results

In order to obtain a benchmark solution we ran our Stereo CI-Graph SLAM system on dataset Biccoca_2009-02-25b. The dataset consists of 26335 trinocular image frames collected during 30 minutes at 15 FPS. Figure 5 shows an example of the system performance when building a local map along the library. We can see how features in the map are predicted and search over right, left and top images in order to update the state vector. A reconstruction is also shown both in top and lateral view for the resulting submap.

Figure 6 shows the results obtained after running our loop closing approach. Pairings between past and current images are highlighted with red points and green crosses respectively on the odometry path. In order to overcome the lack of texture in critical parts of environment, the odometry readings were used along with the CI-Graph method over the full path. A total map of the dataset is shown in figure 7 where each local submap is represented in absolute coordinates before applying the loop closing constraints. The estimated trajectory is also presented in Figure 8.

We compared our estimation with the provided Ground Truth solution when this was available (see Fig. 10 and Fig. 9). Figure 11 shows the Absolute Trajectory Error (ATE) evaluation where our Stereo SLAM produces an error of $1.38119m$ in position and $0.04012rad$ in orientation with a

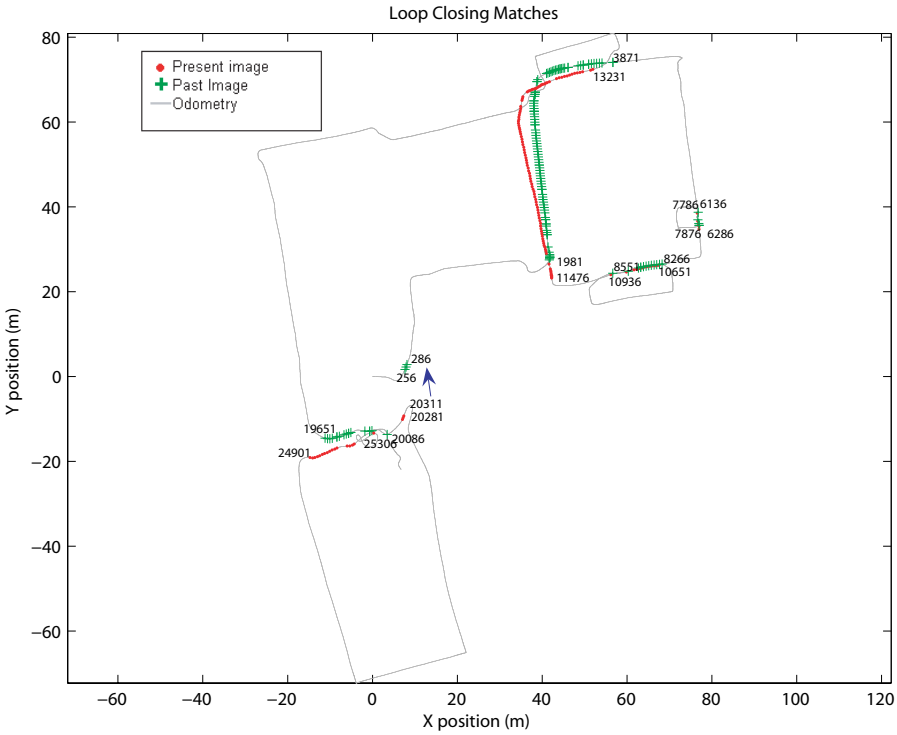

Figure 6: Obtained loop closing matches. Results are shown along the odometry path

maximum error of $1.92394m$ and $0.14556rad$ correspondingly. The error distribution is shown in Figure 13 considering a 3σ error bound for the uncertainty. In addition the Relative Pose Error (RPE) was computed producing a Mean Square Error of $3.5468m^2$ in translation and $0.6002rad^2$ in orientation (see Fig. 12).

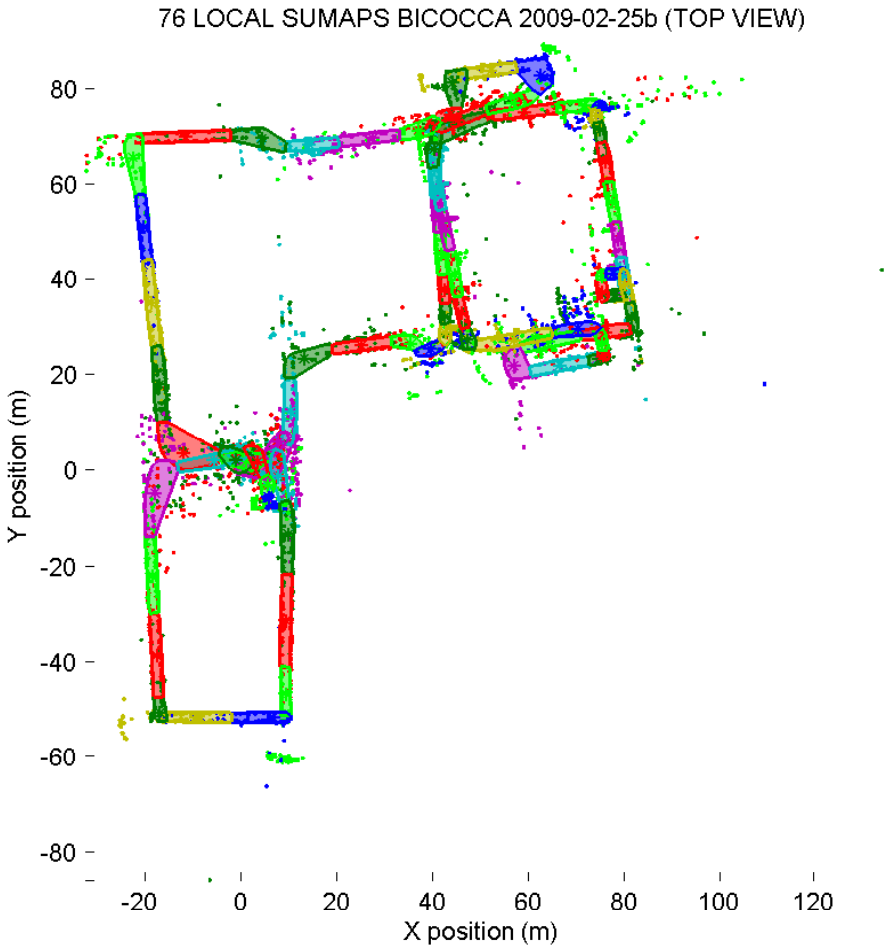

Figure 7: Map result with Trinocular sequence and odometry on Bicocca_2009-02-25b.

References

Arthur, D. and Vassilvitskii, S. [2007], k-means++: the advantages of careful seeding, *in* 'SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1027–1035.

Bay, H., Ess, A., Tuytelaars, T. and Gool, L. V. [2008], 'Surf: Speeded up robust features', *Computer Vision and Image Understanding (CVIU)* **110**(3), 346–359.

Civera, J., Davison, A. J. and Montiel, J. M. M. [2007], Inverse depth to depth conversion for monocular SLAM, *in* 'IEEE International Conference on Robotics and Automation, 2007', pp. 2778–2783.

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

Moreno-Noguer, F., Lepetit, V. and Fua, P. [2007], 'Accurate non-iterative o(n) solution to the pnp problem', *Computer Vision, IEEE International Conference on* **0**, 1–8.

Nister, D. and Stewenius, H. [2006], Scalable recognition with a vocabulary tree, *in* 'Computer

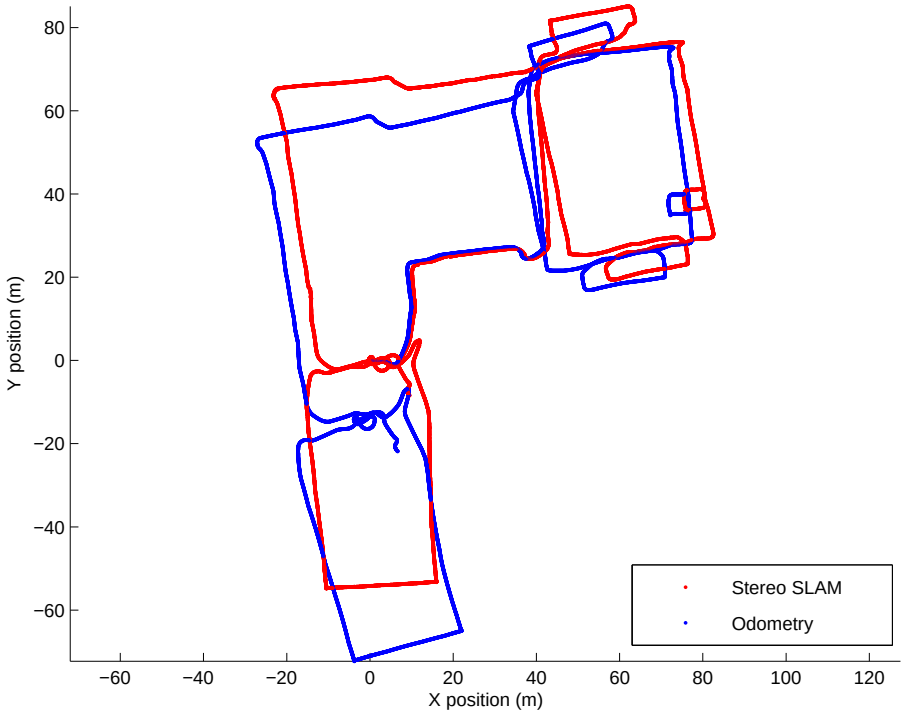

Figure 8: Estimated trajectory with Stereo SLAM

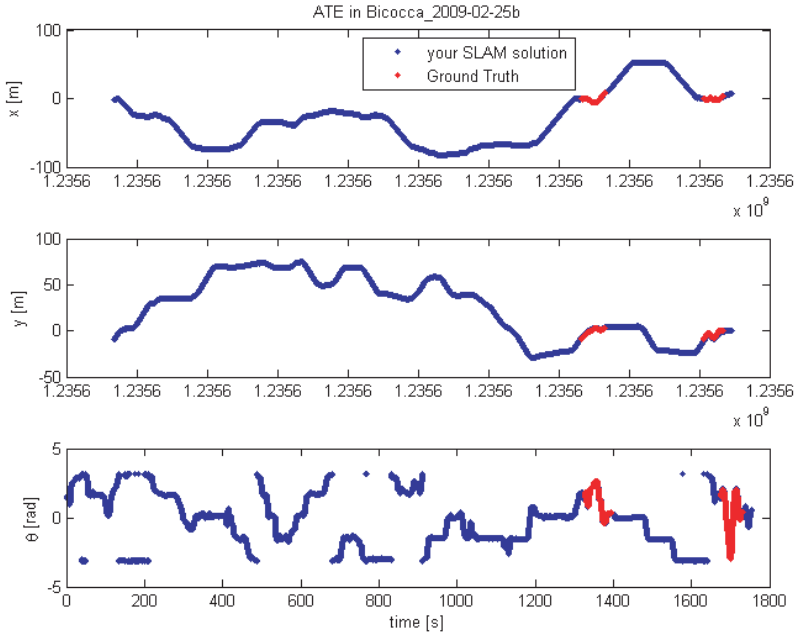

Figure 9: Comparison between Ground Truth Position and Stereo SLAM before closing the loop. The comparison has been done when Ground Truth is available.

Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on', Vol. 2, pp. 2161–2168.

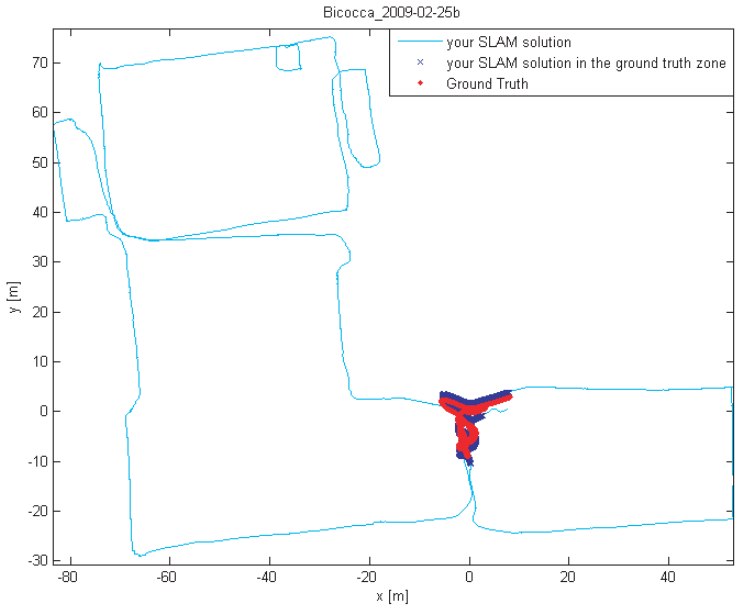

Figure 10: Full estimated trajectory and Ground Truth.

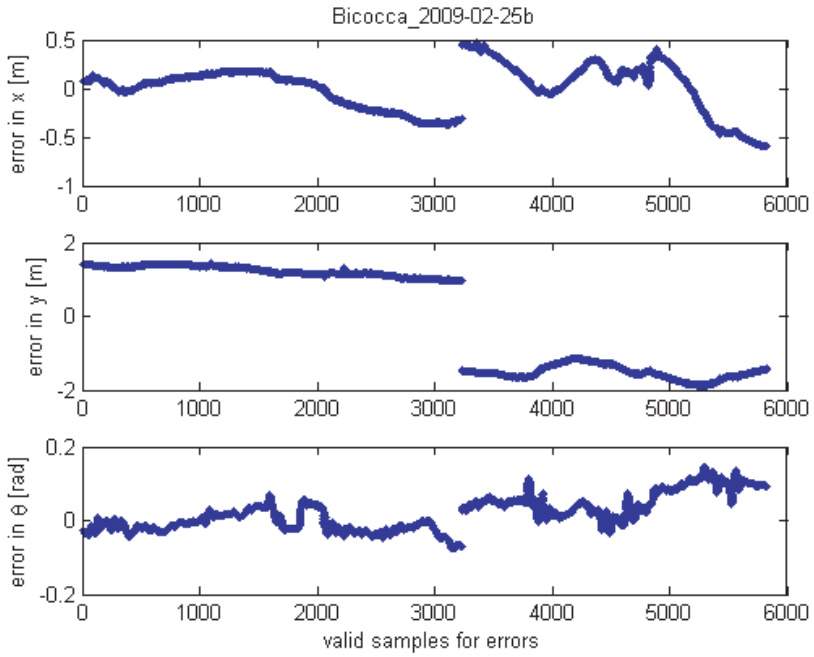

Figure 11: Position and Angular Root Square Error in the available samples.

Paz, L. M., Piniés, P., Tardós, J. D. and Neira, J. [2008], 'Large Scale 6DOF SLAM with Stereo in Hand', *Transactions on Robotics* **24**(5), 946–957.

Piniés, P., Paz, L. M. and Tardós, J. D. [2009], 'CI-Graph: an efficient approach for Large Scale SLAM ', *Proc. of the IEEE Int. Conf. Robotics and Automation (ICRA'09)* pp. 3913–3920.

Piniés, P. and Tardós, J. D. [2008], 'Large Scale SLAM Building Conditionally Independent Local

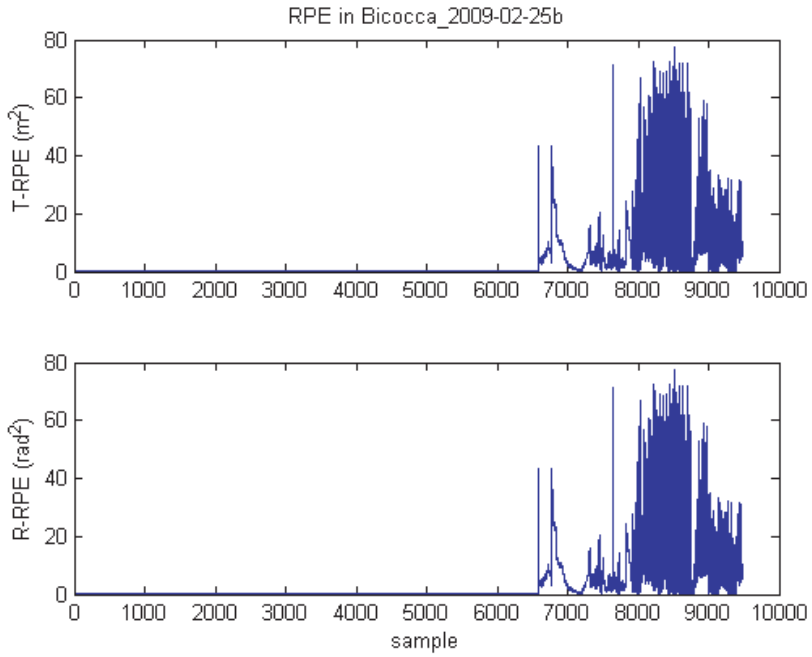

Figure 12: Position and Angular Root Square Error in the available samples.

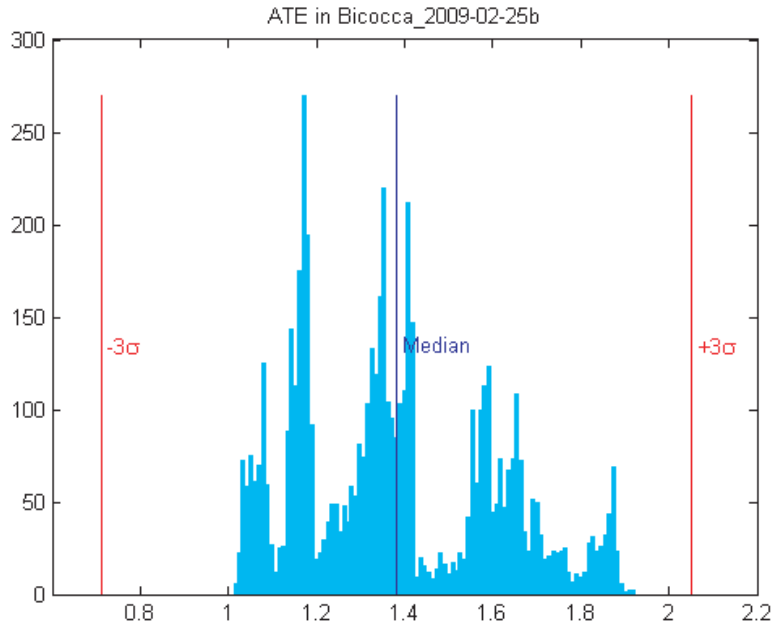

Figure 13: Error Distribution.

Maps: Application to Monocular Vision', *Transactions on Robotics* **24**(5), 1094–1106.

Sivic, J. and Zisserman, A. [2003], Video Google: A text retrieval approach to object matching in videos, *in* 'Proceedings of the International Conference on Computer Vision', Vol. 2, pp. 1470–1477.