Universitat de les Illes Balears

Departament de Ciències Matemàtiques i Informàtica

Vision-Based Topological Mapping and Localization by means of Local Invariant Features and Map Refinement

E. GARCIA-FIDALGO and A. ORTIZ

July / 2013 TECHNICAL REPORT A-04-2013

Vision-Based Topological Mapping and Localization by means of Local Invariant Features and Map Refinement

Emilio Garcia-Fidalgo and Alberto Ortiz*

Abstract

An appearance-based approach for topological visual mapping and localization using local invariant features is proposed in this paper. To optimize running times, matchings between the current image and previous visited places are determined using an index based on a set of randomized KD-trees. A discrete Bayes filter is used for predicting loop candidates, whose observation model is a novel approach based on an efficient matching scheme between features. In order to avoid redundant information in the resulting maps, in this work, we also present a map refinement framework, which takes into account the visual information stored in the map for refining the final topology of the environment. These refined maps save storage space and improve the execution times of localizations tasks. The approach has been validated using image sequences from several environments.

1 Introduction

Mapping and localization are essential problems in mobile robotics. In order to solve them, several approaches have been proposed to perform both tasks at the same time, creating an incremental map of an unknown environment while localizing the robot within this map. These techniques are generically known as SLAM [1] (Simultaneous Localization and Mapping). In SLAM, loop closure detection is a key challenge to overcome. It implies the correct detection of previously seen places from sensor data. This allows generating consistent maps and reducing their uncertainty.

Ultrasounds and laser sensors have been used for years for SLAM and loop closure detection. Nevertheless, in the last decades there has been a significant increase in the number of visual solutions because of the low cost of cameras, the richness of the sensor data provided and the availability of cheap powerful computers. This naturally guides us to an appearance-based SLAM, where the environment is represented in a topological way by a graph. Each node of this graph

*E. Garcia-Fidalgo and A. Ortiz are with the Department of Mathematics and Computer Science, University of the Balearic Islands, 07122 Palma de Mallorca, Spain. {emilio.garcia, alberto.ortiz} at uib.es.

represents a distinctive visual location visited by the robot while the edges indicate connectivities between locations. Using this representation, the loop closure problem can be solved comparing images directly, avoiding to maintain and estimate the position of feature landmarks.

In the *Bag-of-Words* (BoW) [2] approach, local invariant features obtained from an image are quantized into a vector according to a visual vocabulary. This representation is one of the most used techniques in appearance-based SLAM. However, this method presents some drawbacks. On the one hand, the perceptual aliasing effect [3], where two different places can be perceived as the same, becomes likelier because of the quantization process. On the other hand, on most occasions, an offline training phase is required to build the visual vocabulary.

Topological maps obtained from visual information tend to contain spurious paths and nodes [4, 5]. This is because of images noise, partial invariance of image descriptors to viewpoint, scale and/or illumination changes, or due to the mapping algorithm itself. The final map obtained can be very large and can contain more nodes that are actually required to represent the environment, deriving in an increment of the storage needs and the computational requirements. We propose to refine the map as it is being built, instead of investing efforts in improving the mapping algorithm.

To cope with the aforementioned issues, this paper presents a complete visual mapping and localization framework based on raw local invariant features and a map refinement strategy. Our framework has been assessed using multiple indoor and outdoor datasets captured under different weather conditions and illumination changes. Very promising results have been obtained in all cases. As main contributions, we present a Bayesian framework for visual loop closure detection which uses local invariant features as image descriptor. It comprises a novel observation model which allows us to succeed in challenging loop closure situations. Using this algorithm as a key component, a topological mapping and localization framework is also proposed, which uses a map refinement strategy to remove the redundant paths that exist in the graph. This strategy is also presented in this work, introducing a novel map refinement theory based on the visual information obtained from the images of the environment. This refinement is done online each time a loop closure is detected.

The rest of the paper is organized as follows: Section 2 enumerates fundamental works related to loop closure detection and visual localization and mapping, Section 3 explains the basics of our algorithm, Section 4 shows how images are described and matched in our approach, Section 5 exposes a Bayesian loop closure algorithm using visual features, Section 6 presents our map refinement framework, Section 7 shows experimental results obtained from different datasets, and Section 8 concludes the paper.

2 Related Work

A high number of appearance-based localization and mapping solutions have been proposed along the last decade. Although many works assume the availability of omnidirectional images [6, 7, 8, 9], many others make use of monocular configurations [10, 11, 12, 13]. Our approach belongs to this latter class.

Although most works are based on either topological maps or metric maps, some authors have tried to make hybrid solutions, combining both paradigms in one. Zivkovic et al. [6] presented an algorithm for automatically generating hierarchical maps from images. A low-level map is built using SIFT features. Then they cluster nodes to construct a high-level representation. Later, [5] showed a navigation system based on a topological map which used the epipolar geometry to obtain a robust heading estimation.

Referring to the image description, the BoW approach has become quite popular. Cummins and Newman developed FAB-MAP [10], where a Chow-Liu tree is used for modelling the dependencies between visual words. Angeli et al. [11, 12] extended the BoW paradigm to incremental conditions and relied on Bayesian filtering to estimate the probability of loop closure. Their work was expanded constructing a complete topological SLAM system [4] and including robot odometry information [14]. Fraundorfer et al. [15] present a highly scalable vision-based localization and mapping method using image collections. Local geometric information is used to navigate in the topological map. Despite its well-known general performance, the BoW paradigm is more affected by perceptual aliasing [3]. For this reason, our loop closure detection algorithm follows an approach similar to [11, 12], but using local invariant features for image description and matching.

Other approaches make use of global descriptors, such as Gist [16]. Singh and Kosecka [17] computed Gist descriptors in omnidirectional images of urban environments for detecting loop closures. They presented a novel image matching strategy for panoramas. Bayes filtering is not considered in this work. Liu and Zhang [18] applied Principal Component Analysis (PCA) to Gist descriptors in order to compute the likelihood in a particle filter. This filter is used for detecting loop closures. Siagian and Itti [19] presented a biologically-inspired system to scene classification using Gist as image representation.

Rather than BoW or global descriptors, some authors have used local invariant features for visual localization and mapping as well as for loop closure detection [3, 20, 21, 13]. Zhang [3] presented a method for selecting a subset of Scale-Invariant Feature Transform (SIFT) [22] keypoints extracted from an image. These features are used for matching consecutive images. A location is represented by a set of features that can be matched consecutively in several images. The problem of this approach is that the number of features to manage increases while new images are added, and a linear search for matching becomes intractable. In order to overcome this issue, in this work, we index features using a set of randomized KD-Trees.

In a previous work [23], we developed an appearance-based mapping and localization method. This paper improves that work with a better observation model, with regard to loop closure detection, and with a novel map refinement strategy.

3 Algorithm Overview

The main goal of our approach is to construct a clean visual representation of the robot environment using a single monocular camera while localizing the robot within this map. In a real scenario, storing all images taken by the camera is impossible. We need to reduce the number of images to manage without losing visually distinct locations found in the robot environment. This subset of images are called *keyframes* [21]. Our approach is also based on the keyframe concept, as it is outlined in Fig. 1 and Alg. 1. In our map, each node represents a keyframe image, and each keyframe is represented by its SIFT [22] features. In order to select these keyframes, we discard: (a) images similar to the current location of the robot (keyframe), since they do not provide distinct visual information about the environment and therefore are redundant; and (b) robot camera turns, because they are noisy and can introduce errors in the mapping and localization processes. For the first case, SIFT features of the current image are matched applying the ratio test [22] to the features of the current location keyframe. If the number of matched features is higher than a threshold, the image is considered similar to the current location.

The same matching step is applied between the current image and the last received image in the sequence: if it is not possible to match a certain number of features, the image is considered a turn. If the image is not similar to the current location and is not a turn, it is considered useful and needs to be processed in order to determine whether a loop can be closed or whether a new keyframe has to be added to the map. Otherwise, the image is discarded. This keyframe selection policy is shown graphically in Fig. 2.

Our approach makes use of a discrete Bayes filter to detect loop closures. This filter is updated with every image irrespective of whether the image has been discarded or not. If a loop closure is not found, the current image is considered as a new keyframe and is added to the map as a new node. Otherwise, a link is created between the current location of the robot and the loop closure candidate and, then, a map refinement process runs, in order to determine if redundant paths have been added. As a consequence, a set of superfluous nodes can result to be detected. If this is the case, they are removed from the map, and the robot position within the map is updated accordingly. In order to avoid false loop closure detections between the current image and its neighbours in the sequence, new keyframes are not inserted directly as a loop closure hypothesis in the filter. They are inserted in a cache list and incorporated after a certain number of images have passed. The image description and matching process, the loop closure detection algorithm and the map refinement strategy are explained in detail in the following sections.

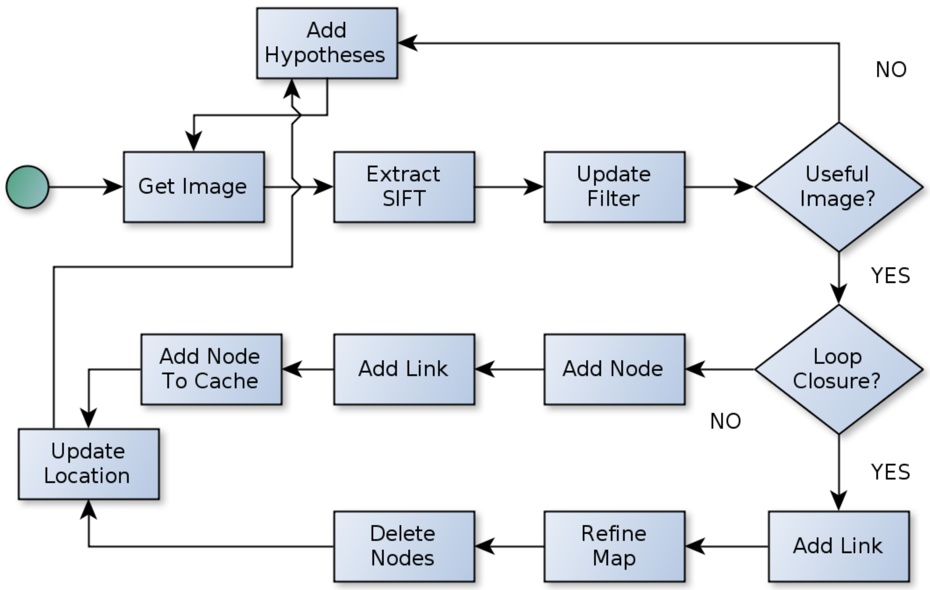


Figure 1: Overall algorithm diagram. See text for details.

Algorithm 1 Appearance-Based Mapping and Localization

```
/* Variables */
$I = \{I_0, \dots, I_{N-1}\}$: Sequence of N input images.
$G$: Graph representing the environment topology.
$k$: Current keyframe index.
$F_t$: Set of SIFT features obtained from image $I_t$.
$c$: Candidate keyframe index for closing a loop.
$M^i_j$: Matchings between images $I_i$ and $I_j$.
$C$: List of nodes not inserted in the filter as hypothesis yet.
$l$: Boolean variable indicating whether a loop has been detected or not.

$k = 0$
$F_0$ = describe($I_0$)
addToCache($C, 0$)
for $t = 1$ to $N - 1$ do /* While there are images */
    $F_t$ = describe($I_t$)
    updateFilter($F_t$)
    $M^t_k$ = match($F_t, F_k$)
    $M^t_{t-1}$ = match($F_t, F_{t-1}$)
    if useful($M^t_k, M^t_{t-1}$) then /* Useful Image */
        $l, c$ = detectLoopClosure($G, F_t$)
        if $l$ then /* Loop Closure Detected */
            addLink($G, k, c$)
            refineMap($G$)
            $k = c$
        else /* New node */
            addNode($G, t$)
            addLink($G, k, t$)
            addToCache($C, t$)
            $k = t$
        end if
    end if
    releaseHypotheses($C$)
end for
```

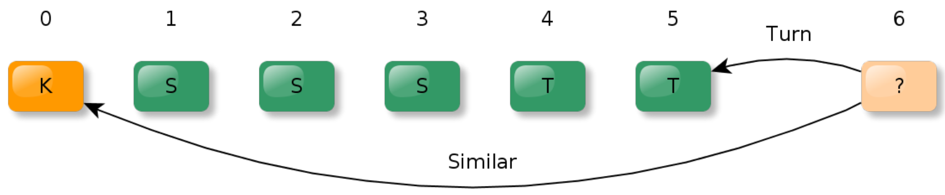


Figure 2: Image selection policy. The current image taken by the camera (6) is matched with the image that represents the current keyframe (0) and the last received image in the sequence (5) in order to determine if it is a useful frame. K represents the current location (keyframe), S and T represent images discarded because they are, respectively, similar enough to the current location or camera turns.

4 Image Description and Matching

As commented above, in our approach each image is described using the SIFT [22] algorithm, where interest points are defined as maxima and minima of a difference of Gaussians function applied, in scale space, to a series of resampled images. Each feature is then described defining a histogram of gradient orientations around the point at the selected scale, resulting in a 128-dimensional descriptor. In this work, these descriptors are compared using Euclidean distance.

The loop closure detection algorithm, as we will see shortly, needs to match efficiently the features of the current image with features of all previously considered keyframes, in order to determine whether it is a revisited place. Therefore, a method for an efficient nearest neighbour search is needed to match these high-dimensional descriptors. Tree structures have been widely used to this end, since they reduce the search complexity from linear to logarithmic. To the same purpose, we maintain a set of randomized KD-trees containing all the SIFT descriptors of previously seen keyframes. An inverted index, which maps each feature to the image where it was found, is also created. Given a query descriptor, these structures allow us to obtain, traversing the tree just once, the top K nearest neighbours keypoints among all keyframes.

5 Probabilistic Loop Closure Detection

Given a new image, a discrete Bayes filter is used to detect loop closure candidates. This filter estimates the probability that the current image closes a loop with an already seen location, ensuring temporal coherency between consecutive predictions. Given the current image I_t at time t, we denote z_t as the set of SIFT descriptors extracted from this image. These are the observations in our filter. We also denote L_i^t as the event that image I_t closes a loop with image I_i, where $i < t$. Using these definitions, we want to detect the image of the map I_c whose index satisfies:

$$c = \underset{i=0,\dots,t-p}{\arg\max} \{ P \left(L_i^t | z_{0:t} \right) \}, \tag{1}$$

where $P\left(L_i^t|z_{0:t}\right)$ is the full posterior probability at time t given all previous observations up to time t. As in [12], the most recent p images are not included as hypotheses in the computation of the posterior since I_t is expected to be very similar to its neighbours and then false loop closure detections will be found. This parameter p delays the publication of hypotheses and needs to be set according to the frame rate or the velocity of the camera.

Separating the current observation from the previous ones, the posterior can be rewritten as:

$$P\left(L_i^t|z_{0:t}\right) = P\left(L_i^t|z_t, z_{0:t-1}\right) \ , \tag{2}$$

and then, using conditional probability properties, the next equality holds:

$$P\left(L_i^t|z_t, z_{0:t-1}\right) P\left(z_t|z_{0:t-1}\right) = P\left(z_t|L_i^t, z_{0:t-1}\right) P\left(L_i^t|z_{0:t-1}\right) \ , \tag{3}$$

from where we can isolate our final goal to obtain:

$$P\left(L_i^t|z_t, z_{0:t-1}\right) = \frac{P\left(z_t|L_i^t, z_{0:t-1}\right) P\left(L_i^t|z_{0:t-1}\right)}{P\left(z_t|z_{0:t-1}\right)} \ . \tag{4}$$

$P\left(z_t|z_{0:t-1}\right)$ is independent of L_i^t, so it can be seen as a normalizing factor. Under this premise and the Markov assumption, the posterior is defined as:

$$P\left(L_i^t|z_{0:t}\right) = \eta P\left(z_t|L_i^t\right) P\left(L_i^t|z_{0:t-1}\right) \ , \tag{5}$$

where η represents the normalizing factor, $P\left(z_t|L_i^t\right)$ is the observation likelihood and $P\left(L_i^t|z_{0:t-1}\right)$ is the probability distribution after a prediction step. Decomposing the right side of (5) using the Law of Total Probability, the full posterior can be written as:

$$P\left(L_i^t|z_{0:t}\right) = \eta P\left(z_t|L_i^t\right) \sum_{j=0}^{t-p} P\left(L_i^t|L_j^{t-1}\right) P\left(L_j^{t-1}|z_{0:t-1}\right) \ , \tag{6}$$

where $P\left(L_j^{t-1}|z_{0:t-1}\right)$ is the posterior distribution computed at the previous time instant and $P\left(L_i^t|L_j^{t-1}\right)$ is the transition model.

Unlike [12], we do not model explicitly the probability of no loop closure in the posterior. If the loop closure probability of I_t with I_c ($P\left(L_c^t|z_{0:t}\right)$) is not high enough, we discard L_c^t.

5.1 Transition Model

Before updating the filter using the current observation, the loop closure probability at time t is predicted from $P\left(L_j^{t-1}|z_{0:t-1}\right)$ according to an evolution model. The probability of loop closure with an image I_j at time $t-1$ is diffused over its neighbours following a discretized Gaussian-like function centered at j. In more detail, 90% of the total probability is distributed among j and exactly four of its neighbours ($j-2, j-1, j, j+1, j+2$) using coefficients (0.1, 0.2, 0.4, 0.2, 0.1), i.e. $0.9 \times$ (0.1, 0.2, 0.4, 0.2, 0.1). The remaining 10% is shared uniformly across the rest of loop closure hypotheses according to $\frac{0.1}{\max\{0,t-p-5\}+1}$. This implies that there is always a small probability of jumping between hypotheses far away in time, improving the sensitivity of the filter when the robot revisits old places.

5.2 Observation Model

Once the prediction step has been performed, the current observation needs to be included in the filter. We want to compute the most likely locations given the current image I_t and its keypoint descriptors z_t, but we want to avoid comparing I_t with each previous keyframe image, since this is not tractable. To this end, the structures described in section 4 are used. Note that if the robot has revisited the same place several times and the current image I_t closes this loop again, each descriptor in z_t can be close to descriptors from different previous images in the Euclidean space. This fact is taken into account in the computation of our likelihood.

For each hypothesis i in the filter, a score $s\left(z_t, z_i\right)$ is computed. This score represents the likelihood that the current image I_t closes the loop with image I_i given their descriptors, z_t and z_i respectively. Initially, these scores are set to 0 for all frames from 0 to $t-p$. For each descriptor in z_t, the K closest descriptors among the previous keyframe images are retrieved without taking into account the p immediately previous frames, and each of them, denoted by n, adds a weight w_n to the score of the image where it belongs to. This value is normalized using the total distance of the K candidates retrieved:

$$w_n = 1 - \frac{d_n}{\sum\limits_{k \in K} d_k}, \forall n \in K, \tag{7}$$

where d is the Euclidean distance between the considered query descriptor in z_t and the nearest neighbour descriptor found in the tree structure. This value is accumulated to a score according to:

$$s\left(z_t, z_{j(n)}\right) = s\left(z_t, z_{j(n)}\right) + w_n, \forall n \in K, \tag{8}$$

being $j(n)$ the index of the image where the candidate descriptor n was extracted. The computation of the scores is finished when all descriptors in z_t have been processed. Then, the likelihood function is calculated according to the following rule (similarly to [12]):

$$P\left(z_t | L_i^t\right) = \begin{cases} \frac{s(z_t, z_i) - s_\sigma}{s_\mu} & \text{if } s\left(z_t, z_i\right) \geq s_\mu + s_\sigma \\ 1 & \text{otherwise} \end{cases}, \tag{9}$$

being respectively s_μ and s_σ the mean and the standard deviation of the set of scores. Only the most likely locations given the current observation z_t update their posterior. After incorporating the observation to our filter, the full posterior is normalized in order to obtain a probability function.

5.3 Selection of a Loop Closure Candidate

Algorithm 2 Visual Loop Closure Detection

```
/* Variables */
$B$: Discrete Bayes filter.
$F_t$: Set of SIFT features obtained from image $I_t$.
$c$: Candidate image index for closing a loop.
$P_c$: Probability of candidate image index for closing a loop.
$n_{hyp}$: Number of hypotheses in the Bayes filter.
$E^i_j$: Set of matchings surviving the epipolarity constraint-based filter.
$L$: Output boolean variable for indicating the existence of a loop.
$L_{im}$: Output integer with the index of the image loop closure.

/* Thresholds */
$T_{loop}$: Minimum probability to consider a loop candidate.
$T_{ep}$: Minimum number of surviving matchings after epipolar geometry validation.
$T_{hyp}$: Minimum number of hypotheses for considering loop candidates.

$c$, $P_c$ = getCandidate($B$) /* Getting the best loop candidate */
**if** $P_c > T_{loop}$ **and** $n_{hyp} > T_{hyp}$ **then**
    $E^t_c$ = epipolarGeometry($F_t$, $F_c$)
    **if** length($E^t_c$) $> T_{ep}$ **then**
        $L =$ True; $L_{im} = c$
    **else**
        $L =$ False; $L_{im} = -1$
    **end if**
**else**
    $L =$ False; $L_{im} = -1$
**end if**
```

In order to select a final candidate, we do not search for high peaks in the posterior distribution, because loop closure probabilities are usually diffused between neighbouring images. This is due to visual similarities between consecutive keyframes in the sequence. Instead, for each location in the filter, we add the probabilities along a defined neighbourhood. This neighbourhood is the same as defined in section 5.1: frames ($j-2, j-1, j, j+1, j+2$) for image j.

The image I_j with the highest sum of probabilities in its neighbourhood is selected as a loop closure candidate. If this probability is below a threshold T_{loop}, the loop closure hypothesis is not accepted. Otherwise, an epipolarity analysis between I_t and I_j is performed in order to validate if they can come from the same scene after a camera rotation and/or translation. Using a RANSAC procedure, the matchings that do not fulfill the epipolar constraint are discarded. If the number of surviving matchings is above a threshold T_{ep}, the loop closure hypothesis is accepted; otherwise, it is definitely rejected.

Finally, another threshold T_{hyp} is defined to ensure a minimum number of hypotheses in the filter, so that loop closure candidates are meaningful. This step counteracts the fact that first images inserted in the filter tend to attain a high probability of loop closure after the normalization step, what leads to incorrect detections. The approach is outlined in Alg. 2.

6 Map Refinement

Visual topological maps tend to contain redundant nodes and paths due to several reasons. On the one hand, sometimes the current image acquired by the robot is blurred, what makes difficult to identify loop closures at the right time and therefore new nodes are added to the map. The loop closure is identified once the image stream becomes stable again. The net result is that a redundant path has been generated because of these noisy images. On the other hand, the Bayes filter does not detect a revisited place instantaneously, but needs some frames to become aware of the loop closure: along these frames, the posterior moves from one keyframe (hypothesis) to another, while a new path containing the unmatched frames is created. In order to correct these problems and to maintain the map structure as simple as possible in storage and computational terms, in this work, we present a map refinement framework based on the visual information obtained from each node of the environment.

Our method is executed each time a loop closure is detected. The idea is to refine the local area of the map around the loop-closing node, since the redundant paths are generated within its neighbourhood. To this end, the k-neighbourhood of the loop-closing node is obtained. This is the set of nodes from which we can reach the loop-closing node in k steps or less, where k has been set experimentally to 10. For each element in this set, all paths to the loop-closing node are obtained using an adjacency list. If there is only one path between the nodes, there are no redundant paths and this route is left unaltered. Otherwise, a further analysis of the different paths is performed. To this end, a path P between nodes i and j is defined as:

$$P_j^i = \{N_0, N_1, \ldots, N_n\}, 0 \leq n \leq k+1, \tag{10}$$

being N_0 the starting node of P_j^i and N_n the loop-closing node. We define the *erasability* of a path as:

$$M(P_j^i) = (deg^-(N_i) = 1) \wedge (deg^+(N_i) = 1), \forall i \in \{1, \ldots, n-1\}, \tag{11}$$

where deg^- and deg^+ are, respectively, the input degree and the output degree of a vertex. Therefore, a path is classified as *erasable* if (11) holds for each inner node of the path. Otherwise, the path is classified as *non-erasable*. The meaning of an erasable path in our context is that the route can be deleted without breaking the topology of the environment. Examples of erasable and non-erasable paths are shown in Fig. 3.

Once all paths have been classified according to their erasability, a decision about which ones can be deleted is made, taking into account that real alternative paths have to be preserved, since they correspond to parts of the environment. To this end, we propose to generate a model path taking into account the visual features of all paths and comparing each erasable path against this model to verify if this is a redundant or a real path. In order to create the model path, a k-means clustering process with 100 centroids is performed using the SIFT features of the

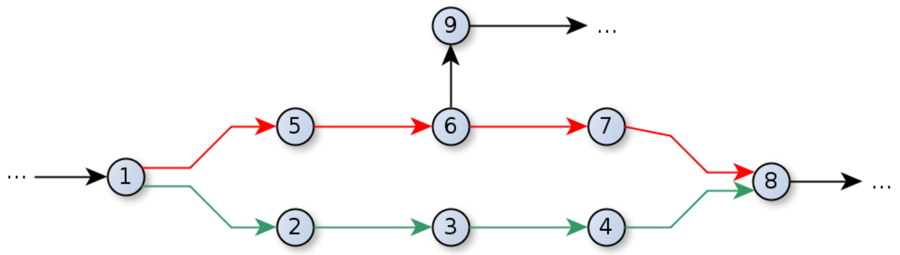


Figure 3: Example of erasability. Two paths exist between node 1 and node 8. The red path (up) is classified as *non-erasable*, since the inner node 6 does not hold condition (11). This path can not be removed without losing the path starting at node 9. The green path (bottom) is classified as *erasable* and is a candidate to be removed.

keyframes that are included in all paths between the corresponding nodes. This gives us a set of reference virtual descriptors representing all the paths as a whole. Then, k-means is also used for quantizing the descriptors of the nodes of each erasable path, obtaining representatives for each route. A set of randomized KD-trees is created using the reference virtual descriptors. The virtual descriptors of each path are matched against the model using these trees in order to obtain a distance between each erasable path and the model path. This distance is computed as the average distance of the matched centroids.

For each erasable path between the nodes, if the distance is below a threshold, this path is quite similar to the others and then it can be considered as redundant. If there is at least one route non-erasable between the nodes, the inner nodes belonging to these paths are deleted. Otherwise, if all paths are classified as erasable, the most different path (higher distance) is left unaltered, and the remaining ones are removed. The full algorithm for map refinement is outlined in Alg. 3.

Fig. 4 shows several examples of situations that our map refinement strategy is able to overcome. In (a), (b) and (c), the removed paths have been selected because the distances to the model path are lower than the others. In (d), since there exists a non-erasable path and the distances of the other paths to the model are lower than T_p, both erasable routes are deleted. In (e), any of the routes can be deleted since they are non-erasable paths. In (f), there exists a non-erasable path and then, the erasable one could be deleted. However, in this case, the distance of the path to the model is higher than T_p, indicating that this is a real alternative path, and therefore the path can not be removed.

7 Experimental Results

In this section, we will show several experimental results for assessing our solution from different points of view. This section is organized as follows: first, the loop closure detection algorithm is evaluated irrespective of the mapping and localization process; then, results for the full approach for mapping and localization

Algorithm 3 Map Refinement

```
/* Variables */
$n_l$: Input loop closing node.
$N$: K-neighbourhood of a node.
$P$: The set of paths between two nodes.
$M$: Array to store the erasability of each path in a set.
$D$: Array to store the distances between each path in a set and a reference model.
$R_m$: Reference model path. It is computed using k-means clustering.
$P_{max}$: Path with maximum distance to the $R_m$.
$k$: Maximum number of steps to consider a node as a neighbour.

/* Thresholds */
$T_p$ : Maximum distance to consider two paths similar to one another.

$N$ = getKNeighbours($n_l$, $k$)
for all $n_n \in N$ do
    $P$ = getPaths($n_n, n_l$)
    if length($P$) $> 1$ then
        $M$ = array(length($P$))
        $D$ = array(length($P$))
        $R_m$ = computeModelPath(P)
        for $p \in$ indexOf($P$) do
            $M[p]$ = computeErasability($P[p]$)
            $R_p$ = computePathDescriptor($P[p]$)
            $D[p]$ = distance($R_m$, $R_p$)
        end for
        $P_{max}$ = computeMaxPath($D$)
        for $p \in$ indexOf($P$) do
            if $M[p]$ and $D[p] < T_p$ and (existNonErasable($P$) or $P[p] \neq P_{max}$) then
                deletePath($P[p]$)
            end if
        end for
    end if
end for
```

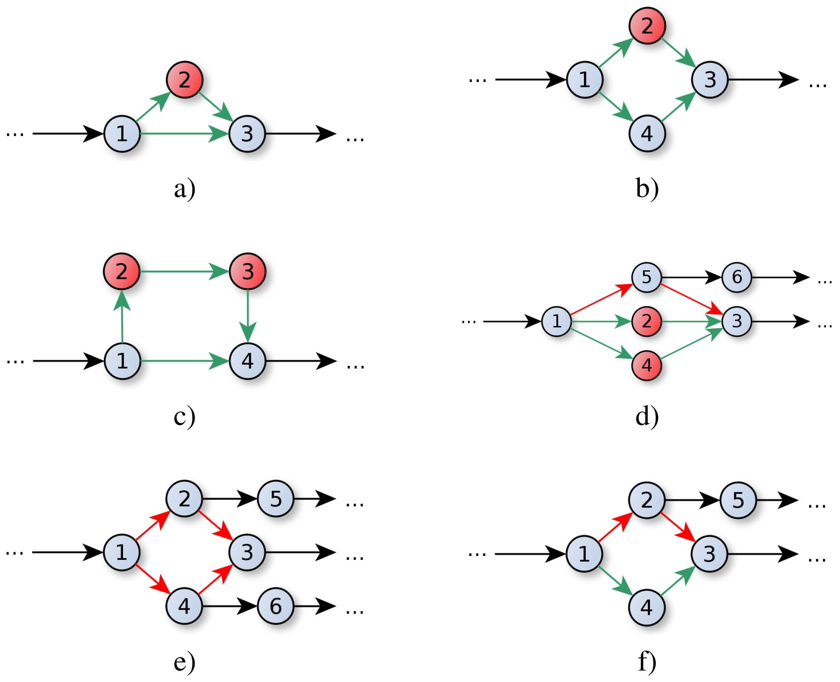


Figure 4: Examples of situations solved by our map refinement strategy. Green and red paths are, respectively, *erasable* and *non-erasable* paths. Red nodes indicate that they will be removed by our approach. When there are several erasable paths, the decision is taken according to the distance of the paths to the model path, as explained in the text.

algorithm are shown; finally, experiments for validating our map refinement algorithm are reported.

7.1 Loop Closure Detection

Several experiments have been carried out in order to validate the suitability of our framework for loop closure detection tasks. Datasets from indoor and outdoor environments have been processed, providing results under different environmental conditions. In the following, the main features of each dataset are presented first. Next, results from particular cases are highlighted. Finally, global results and considerations for the whole set of sequences are reported.

In more detail, the *Lip6Indoor* dataset comprises 388 images of two loops along the corridors of a research building, what leads to a strong perceptual aliasing conditions during the loop closure analysis. The *Lip6Outdoor* is a longer dataset of 1063 images that completes a large loop outdoors under sunny weather conditions. Both datasets are publicly available[1].

The *UIBSmallLoop* and *UIBLargeLoop* datasets have been recorded by ourselves around the Anselm Turmeda building at our University campus. They consist of 388 and 997 images, respectively, taken under bad weather conditions, for verifying the performance of our approach under these situations. Finally, the *UIBIndoor* dataset, also recorded by ourselves inside the Anselm Turmeda building, presents an indoor environment which means a number of difficulties for loop closure. First of all, the camera velocity is not constant. This is due to the fact that we needed to climb up and down the stairs during the recording. This difficulty enables us to validate the ability of the filter to self-adapt under camera speed changes. Besides, when the camera is at the stairs, a number of images of white walls result, what gives rise to the detection of very few features. Moreover, the dataset presents some parts where illumination changes, leading on some occasions to overexposed images. Some examples of these problems are shown in Fig. 5.

Fig. 6 illustrates the performance of the likelihood function for detecting loop closures through the *Lip6Indoor* dataset. The right picture shows the likelihood function values for every pair of frames I_i and I_j while the left picture is the ground truth (only the lower triangles are shown). As can be seen, our likelihood presents high values for real loop closures, which are shown as diagonals in the images. There are more noise in the likelihood at the beginning of the sequence because there are less images in the trees, which implies that nearest neighbours for each descriptor are shared between a minor number of images. This effect decreases along the sequence.

Fig. 7 shows the suitability of the Bayes framework in a loop closure detection situation. In this case, the camera has visited twice the same place. When it returns to this place again, two high peaks corresponding to the previous visits can be seen in the likelihood, representing possible loop candidates for the current image.

[1]http://cogrob.ensta-paristech.fr/loopclosure.html

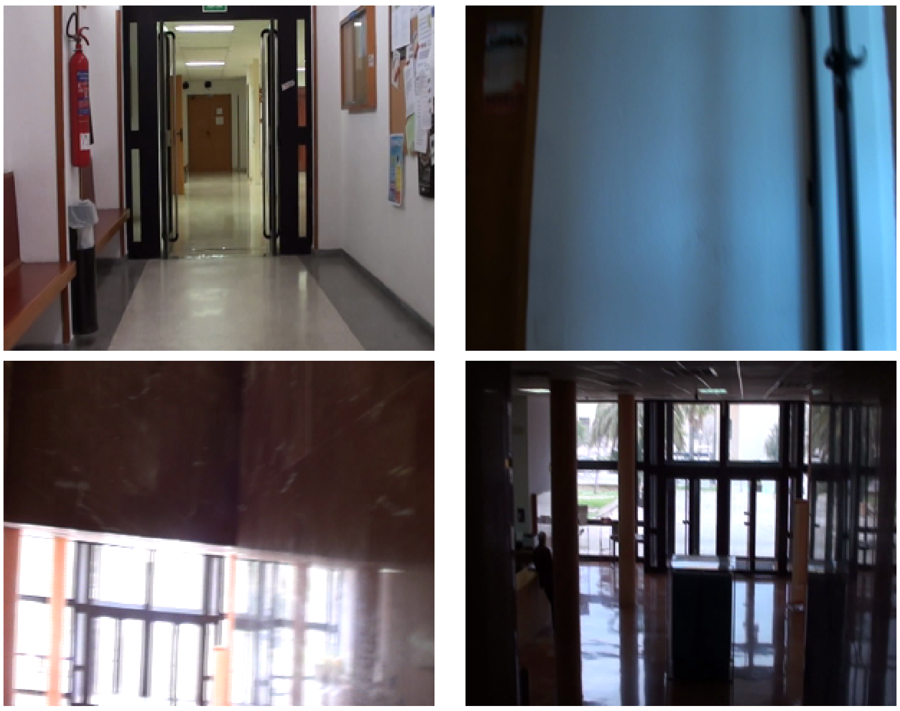


Figure 5: Examples of images from the *UIBIndoor* environment. (Top, Left) First image in the sequence. (Top, Right) Image taken from the stairs. (Bottom, Left) Overexposed image. (Bottom, Right) Image after camera stabilization.

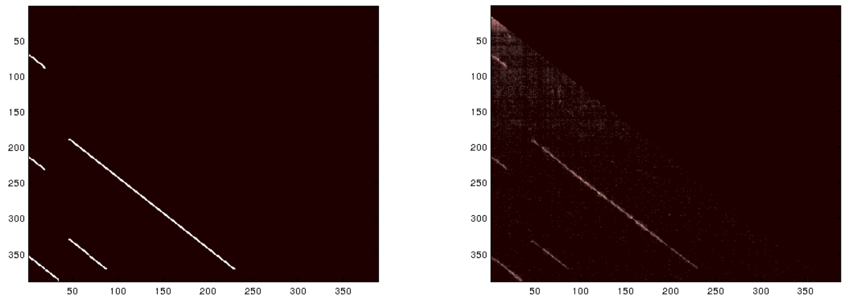


Figure 6: (Left) Ground truth loop closure matrix for the *Lip6Indoor* dataset. (Right) Likelihood matrix computed using our approach.

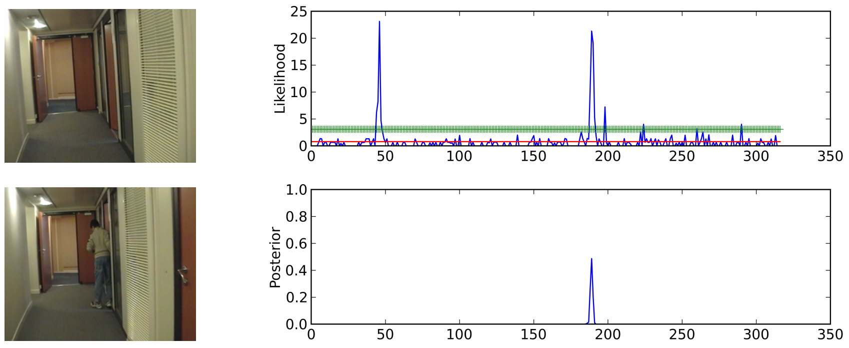


Figure 7: Example of loop closure detection visiting several times the same place in the *Lip6Indoor* dataset. Image 331 (Top, Left) closes a loop with image 189 (Bottom, Left) and image 48 (not shown). As can be seen in (Top, Right), current likelihood presents two strong peaks corresponding to each candidate. After the normalization step, the posterior (Bottom, Right), shows a single peak in the last loop candidate. Red and green lines show respectively s_μ and $s_\mu + s_\sigma$ values.

After the prediction, update and normalization steps, the posterior presents only one single peak at the second candidate image, i.e. the filter ensures temporal coherency between predictions.

Fig. 8 shows an example of situation where a loop is detected despite there is a person in the image who was not in the previous visit, what proves the ability of the filter to detect loops when the appearance of the environment changes. The likelihood function exhibits a clear single peak for the expected loop candidate. After normalizing the posterior, our approach accepts the loop closure since the epipolar constraint between the two images is satisfied. Our approach is also able to detect loop closures under camera rotations. An example can be found in Fig. 9.

If an overexposed image or with not enough features arrives at the filter, the full posterior does not present high peaks and a false negative is generated. When the image stream becomes stable, the algorithm reacts and starts detecting loop closures again. This shows that our approach is able to manage these challenging kinds of situations. Fig. 10 shows an example of loop closure detection from the *UIBIndoor* dataset. An example of loop closure detection for one of the outdoor UIB datasets can be found in Fig 11.

In order to obtain global performance measures, each dataset has been provided with a ground truth, which indicates, for each image in the sequence, which images can be considered as a loop closure with it. The assessment has been performed against this ground truth counting for each sequence the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), where positive is meant for detection of loop closure. Then, several metrics are computed:

- *Precision.* Ratio between real loop closures and total amount of loop clo-

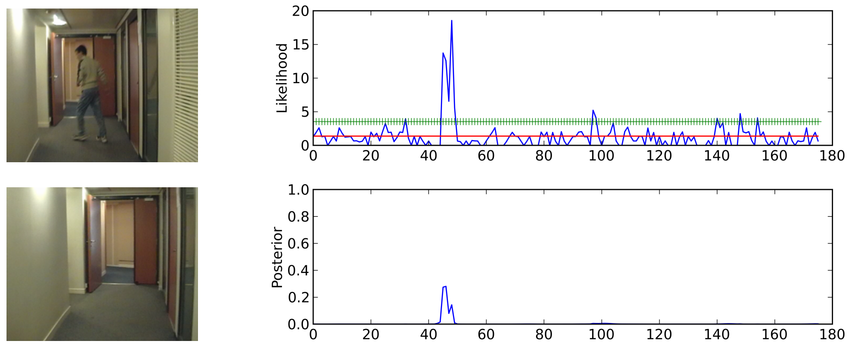


Figure 8: Example of loop closure detection with changes in the environment in the *Lip6Indoor* dataset. Image 190 (Top, Left) closes a loop with image 47 (Bottom, Left). Likelihood (Top, Right) presents a high peak despite there is a person in the current image. (Bottom, Right) is the final posterior. Red and green lines show respectively s_μ and $s_\mu + s_\sigma$ values.

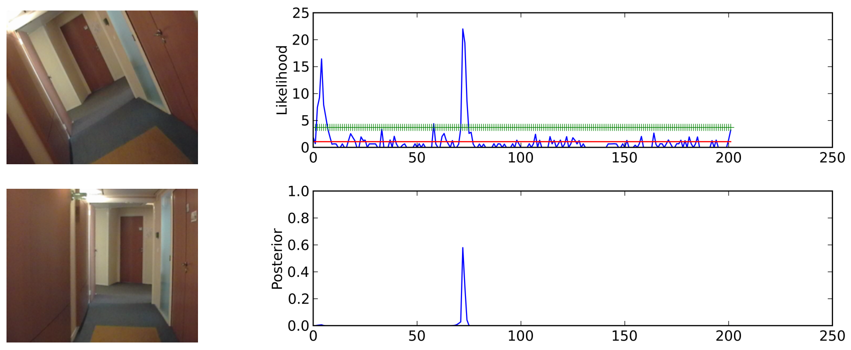


Figure 9: Example of loop closure detection under camera rotations. Despite there is a camera rotation, image 216 (Top, Left) closes a loop with image 72 (Bottom, Left). The likelihood (Top, Right) presents two high peaks since it is the third time the camera visits this place. (Bottom, Right) shows the final posterior, proving that the filter ensures the temporal coherency between loop detections. Red and green lines show respectively s_μ and $s_\mu + s_\sigma$ values.

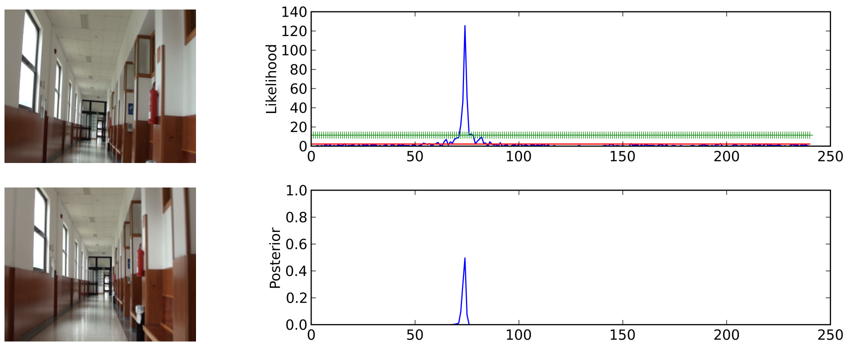


Figure 10: Example of loop closure detection from the *UIBIndoor* dataset. Image 260 (Top, Left) closes a loop with image 73 (Bottom, Left). (Top, Right) Likelihood given the current image. (Bottom, Right) Full posterior after the normalization step. Red and green lines show respectively s_μ and $s_\mu + s_\sigma$ values.

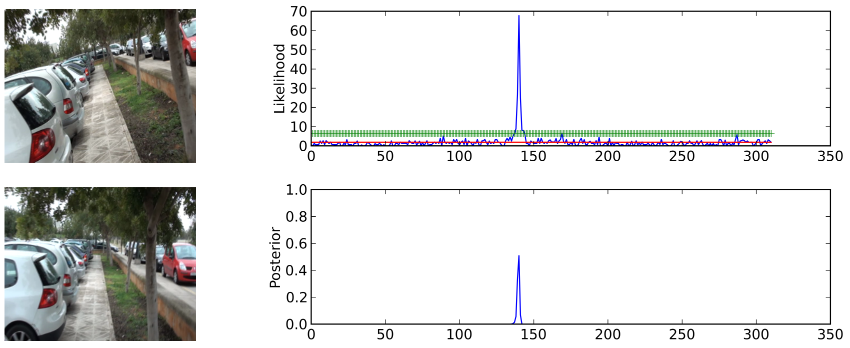


Figure 11: Example of loop closure detection under bad weather conditions and camera rotations for the *UIBSmallLoop* dataset. Image 330 (Top, Left) closes a loop with image 139 (Bottom, Left). (Top, Right) Likelihood given the current image. (Bottom, Right) Full posterior after the normalization step. Red and green lines show respectively s_μ and $s_\mu + s_\sigma$ values.

Dataset	#Imgs	Size	TP	TN	FP	FN	Pr	Re	Acc
Lip6Indoor	388	240×192	191	151	0	31	100	86	91
Lip6Outdoor	1063	240×192	551	435	0	52	100	91	95
UIBSmallLoop	388	300×240	194	172	0	2	100	99	99
UIBLargeLoop	997	300×240	439	491	0	47	100	90	95
UIBIndoor	384	300×240	157	177	0	30	100	84	92
	3220		1532	1426	0	162	100[a]	90[a]	94.4[a]

Table 1: Results for the five datasets. Precision (Pr), Recall (Re) and Accuracy (Acc) columns are expressed as percentages. See text for details. [a]These results are computed as the average of all values.

sures detected ($\frac{TP}{TP+FP}$).

- *Recall*. Ratio between real loop closures and total amount of loop closures existing in the sequence ($\frac{TP}{TP+FN}$).
- *Accuracy*. Percentage of correctly classified (true positive or true negative) images ($\frac{TP+TN}{TP+TN+FP+FN}$).

The results for each sequence are shown in Table 1. As can be seen, no false positives resulted in any case. This is essential, since false positives can induce errors in mapping and localization tasks. As a consequence, the classifier always reaches 100% in precision for all datasets.

In all experiments, we obtain a high rate of correct detections (TP and TN). False negatives are due to, on the one hand, the sensitivity of the filter. When an old place is revisited, the likelihood associated to that hypothesis needs to be higher than the other likelihood values during several consecutive images in order to increment the posterior for this hypothesis. This introduces a delay in the loop closure detection, deriving in false negatives. This sensitivity can be tuned by modifying the transition model of the filter. However, a higher sensitivity can introduce loop detection errors, i.e. false positives. On the other hand, false negatives are also due to camera rotations. When the camera is turning around a corner, it is difficult to find and match features in the image, which prevents the hypothesis from satisfying the epipolar constraint and leads to the loop closure hypothesis to be rejected, despite the posterior for this image is higher than T_{loop}. In spite of the difficulties of the *UIBIndoor* dataset, our approach is able to succeed, as can be seen in Table 1.

The paths followed by the camera in the UIB datasets are shown in Fig 12 and Fig 13. Whenever the camera explores new places, no loop closures are found. When a place is revisited, the algorithm starts to find loop closures. Several images are usually needed until closing the loop, due to the filter inertia. These images correspond to the false negatives which are found.

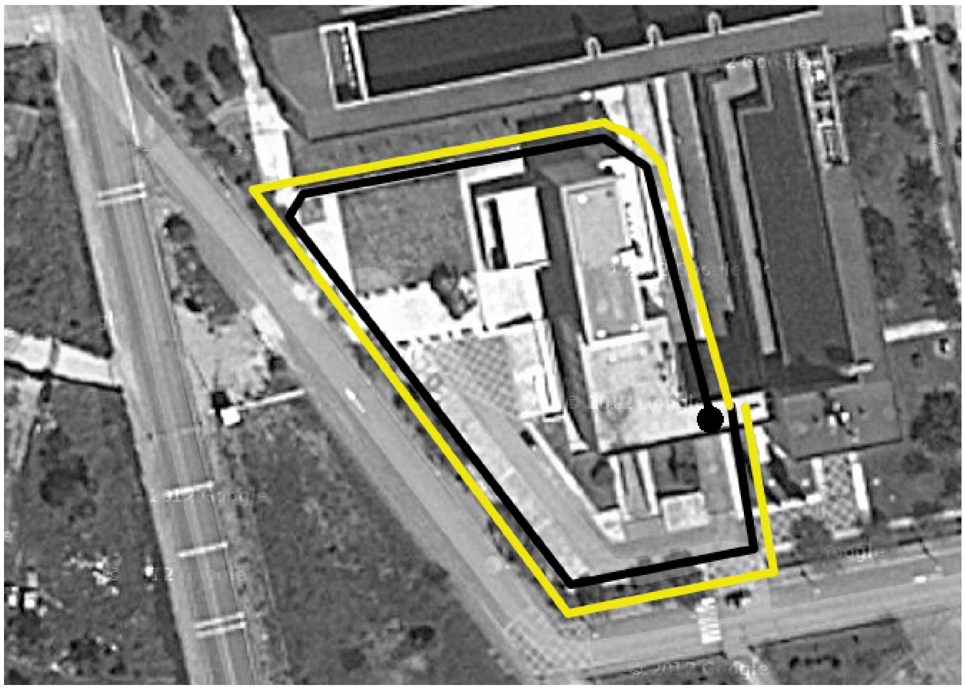


Figure 12: Path followed by the camera during the *UIBSmallLoop* experiment. The black point indicates the beginning of the sequence, the black lines show no loop closure detections (highest posterior probability is under T_{loop}) and the yellow lines represent loop closure detections (highest probability is above T_{loop} and the epipolar constraint is satisfied).

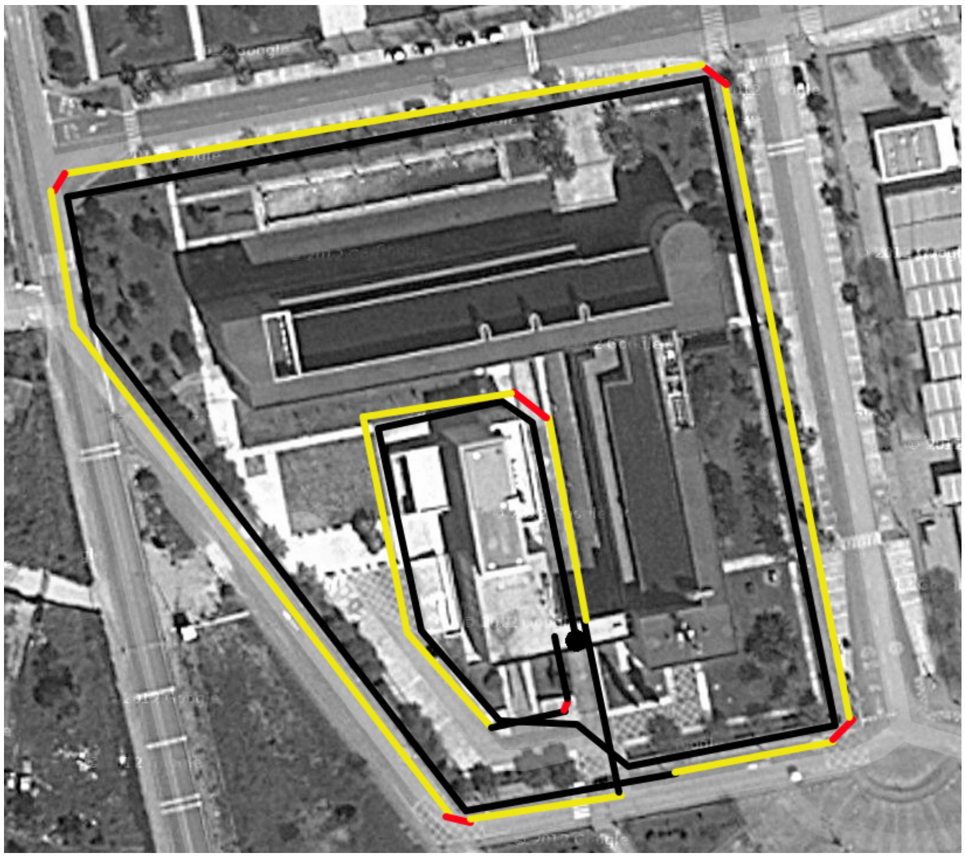


Figure 13: Path followed by the camera during the *UIBLargeLoop* experiment. The black point indicates the beginning of the sequence, the black lines show no loop closure detections (highest posterior probability is under T_{loop}), the red lines show rejected hypoteses (no epipolar geometry is satisfied) and the yellow lines represent loop closure detections (highest probability is above T_{loop} and the epipolar constraint is satisfied).

7.2 Mapping and Localization

The same sequences used in the previous experiments have also been used to validate our framework regarding mapping and localization. A real map of the environment and the topological map generated by our approach are shown for each sequence. The main zones of these maps have been labelled with letters to simplify the identification of each part in the topological structure, since topological maps do not preserve the shape. The results are shown in Fig. 14 to 19.

As can be seen, maps generated by our framework represent topologically the real scenario of the robot. Connections between each part of the topological map are the same than in the real environment. The maps do not present redundant paths or spurious nodes between locations, saving storage space for the map and improving the computational efficiency of the localization process, since less nodes represent the same visual environment. Our map refinement strategy help us to clean the final structure, correcting the problems generated by the blurred images and the delays inherent to the loop closure detection process.

Although this work focuses on mapping and localization, our maps could also be used for navigating if they are tagged with metric information in order to know the relationships between the locations and the actions to perform for reaching one node from another.

Maps are mainly created during the first exploration of the environment. Revisiting a place normally turns into reassigning the current location of the robot to an existing node of the map. However, sometimes maps are completed with new nodes corresponding to images which are visually in-between two nodes. Generally, they provide unregistered information about the robot scenario, as can be seen for example in Fig. 17.

To finish, it is typical that a few nodes at the beginning of the sequence do not close any loop, generating a short tail in the map. This is due to the prediction of the Bayes filter, which tends to move the probability away from the beginning of the sequence, producing that the first loop is closed with the subsequent frames. Notice that this, however, does not affect the final result of the localization process.

7.3 Map Refinement

The main goal of this last section is to verify the quality of the refined maps, in terms of storage space, computational times and usefulness/efficiency. We want to assess that the generated maps are representative of the environment and can be used for localization without compromising the original performance. To this end, we compare the maps of the five datasets used in this work with and without refinement. These maps are shown in Fig. 20 to 24. As can be seen, the original maps contain spurious nodes and alternative redundant paths between nodes, incrementing the execution time for mapping and localization processes, since more nodes need to be considered at each step. The refined maps shown in section 7.2

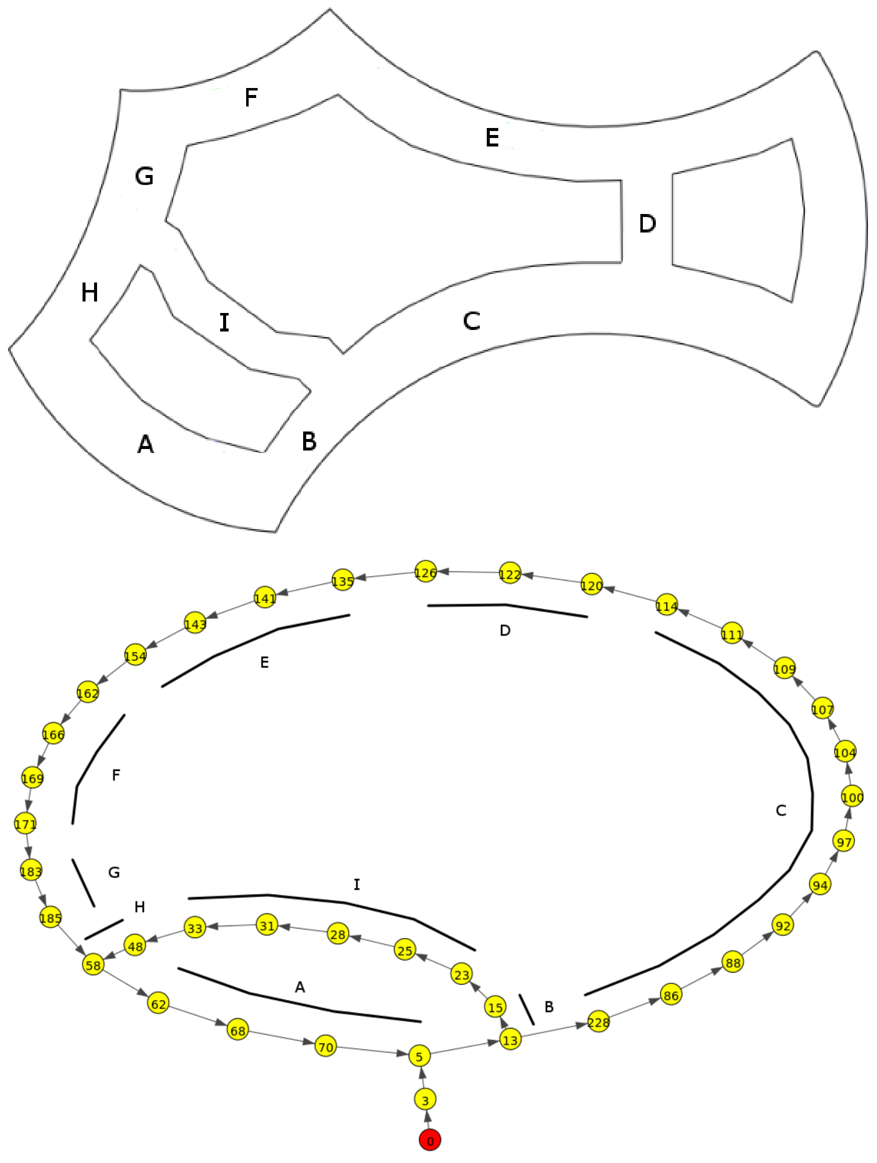
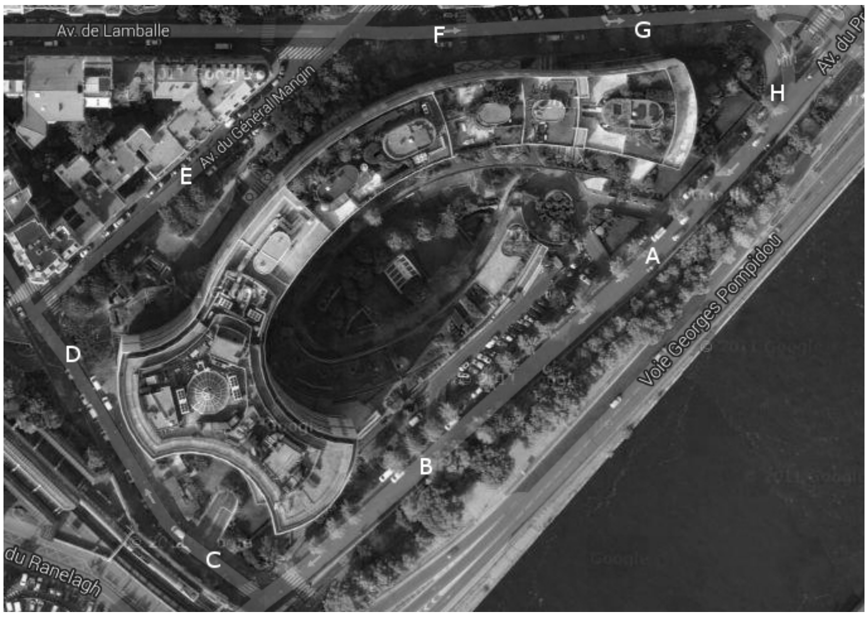


Figure 14: (Top) Reference map for the *Lip6Indoor* dataset. (Bottom) Topological map generated using our approach. Each part of the map is identified with a letter in both maps. The red node is the beginning of the sequence. The reference image has been obtained from http://cogrob.ensta-paristech.fr/loopclosure.html.



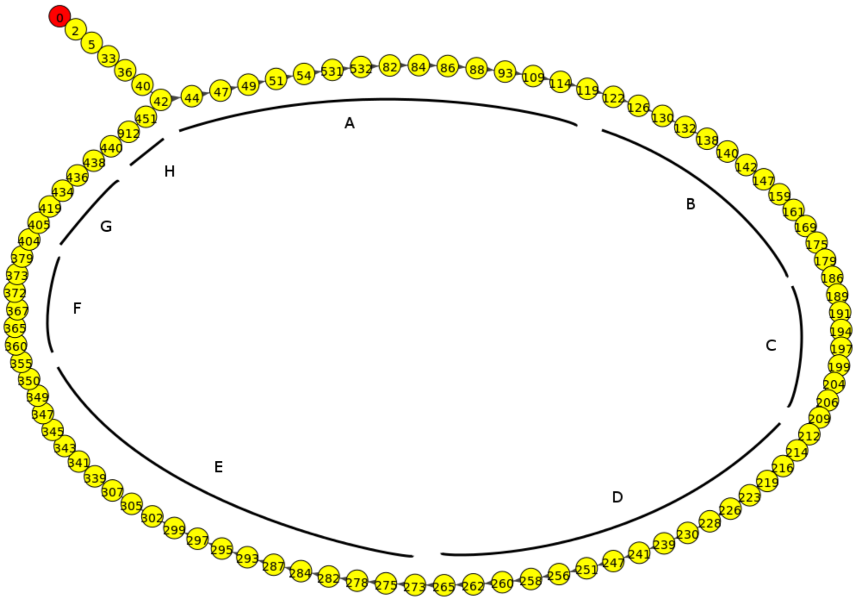


Figure 15: (Top) Reference map for the *Lip6Outdoor* dataset. (Bottom) Topological map generated using our approach. Each part of the map is identified with a letter in both maps. The red node is the beginning of the sequence.

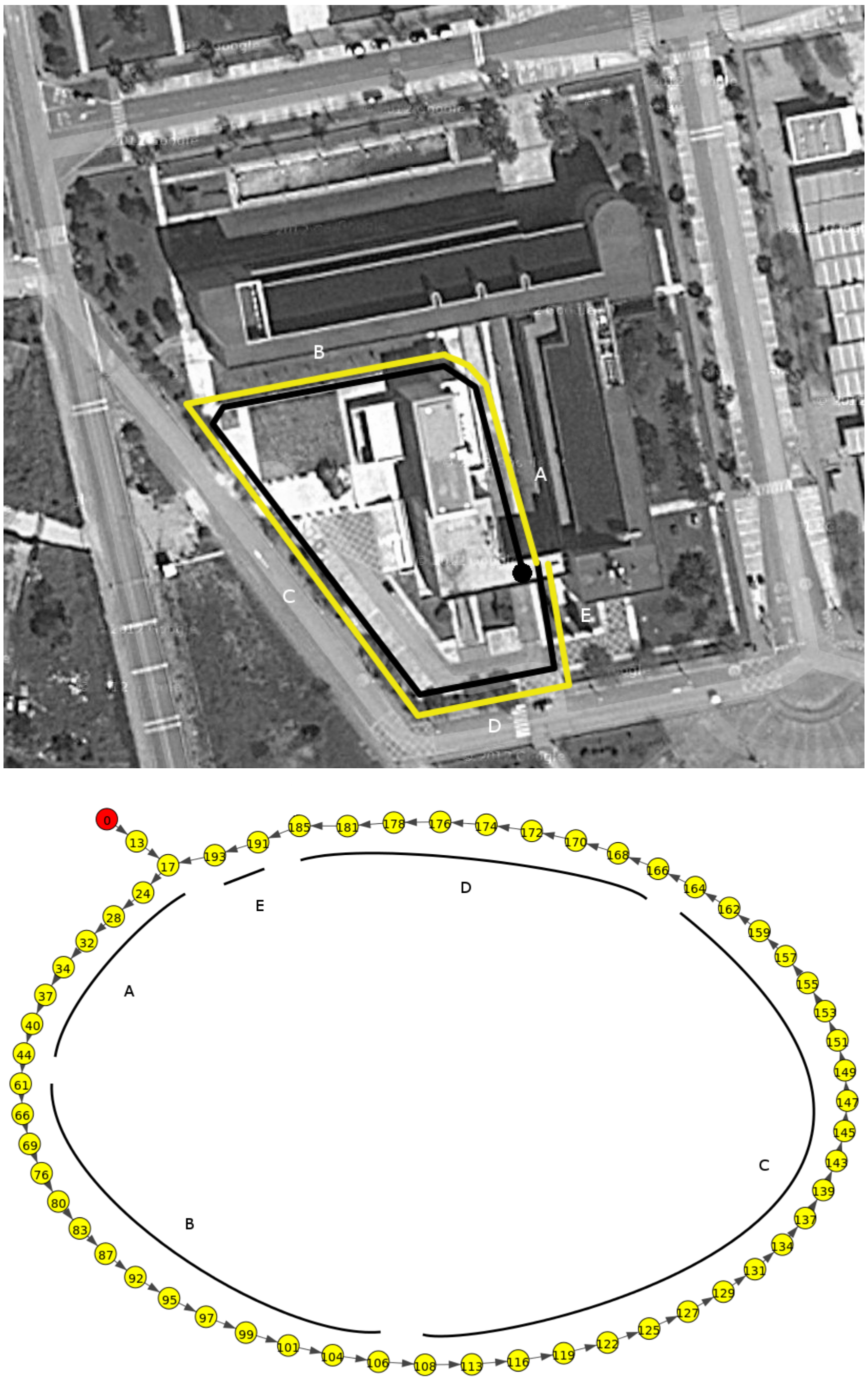


Figure 16: (Top) Reference map for the *UIBSmallLoop* dataset. (Bottom) Topological map generated using our approach. Each part of the map is identified with a letter in both maps. The red node is the beginning of the sequence.

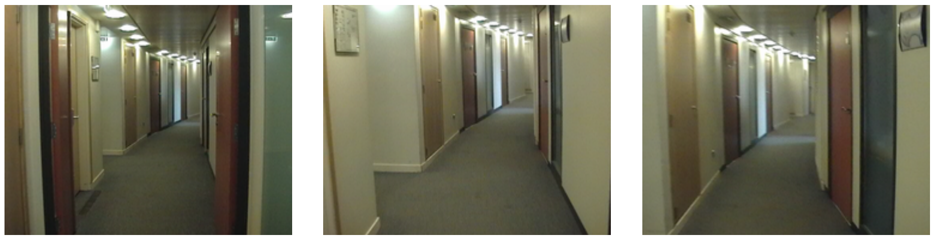


Figure 17: Example of adding intermediate nodes in the *Lip6Indoor* dataset. Images 13 (Left) and 86 (Right) were added to the map at the first loop. Image 228 (Center), was added to the map the next time the camera visits the same place. As can be seen, image 228 is visually in-between the left and right images.

present better the environment.

An additional experiment has been performed in order to verify whether refined maps can be employed for localization with a performance similar to the original ones. First, we create the map of the environment and, for each image, the assigned keyframe is stored. After that, the sequence is processed again using the localization filter to determine, for each image, the closest location in the map. If this location is the same as the one stored during the mapping process, this image is considered as a correct localization (CL). For each sequence, we have also obtained the total mapping and localization times, as well as the number of nodes generated in the graph. These values have been measured for each sequence with and without the refinement step. The results are shown in Table 2. As it is shown, the map refinement step leads to less nodes than without. Despite the correct localization rate is slightly lower for some environments, refining the map improves the computational speed of the mapping and localizations processes. This effect is more significant the longer the sequence is, as is the case of the *Lip6Outdoor* and the *UIBLargeLoop* datasets. The *UIBSmallLoop* dataset presents small differences between the two versions of the map. This is because the resulting structure in the maps are practically the same, resulting into similar processing times. From the table we can also observe that, in general, the outdoors environments are more affected by the refinement step, since the correct localization rates are lower for these cases. In general terms, we can argue that the map refinement strategy proposed in this work can be used for saving space in memory and for improving the speed of the mapping and localization tasks without compromising the performance.

8 Conclusions and Future Work

In this work, a complete appearance-based mapping and localization framework based on local invariant features has been presented. When a new useful image is acquired, a discrete Bayes filter is used to select a loop closure candidate and decide whether this frame is a loop closure or else is a new node to be added to the map. This probabilistic filter presents a novel observation model based in an efficient

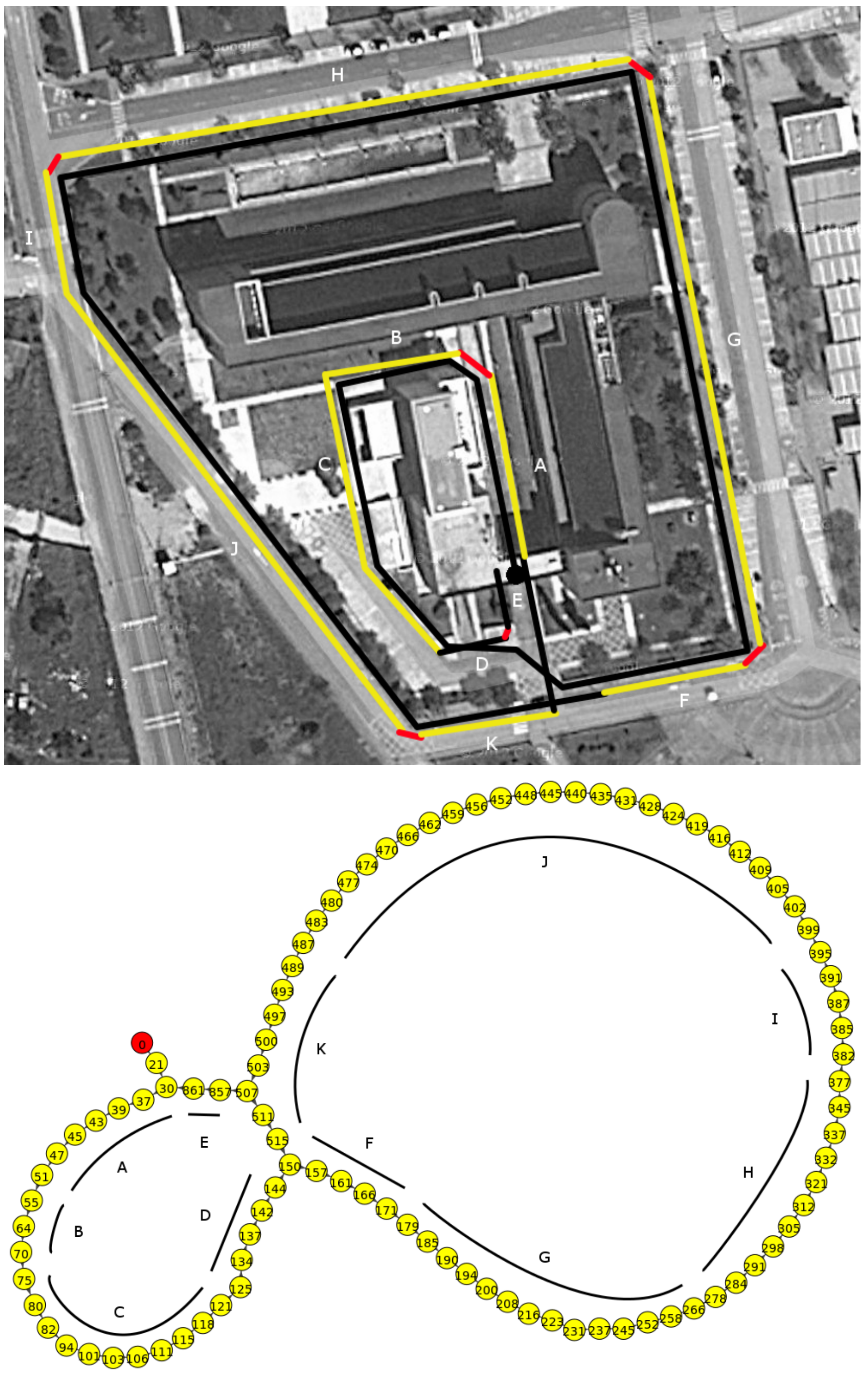


Figure 18: (Top) Reference map for the *UIBLargeLoop* dataset. (Bottom) Topological map generated using our approach. Each part of the map is identified with a letter in both maps. The red node is the beginning of the sequence.

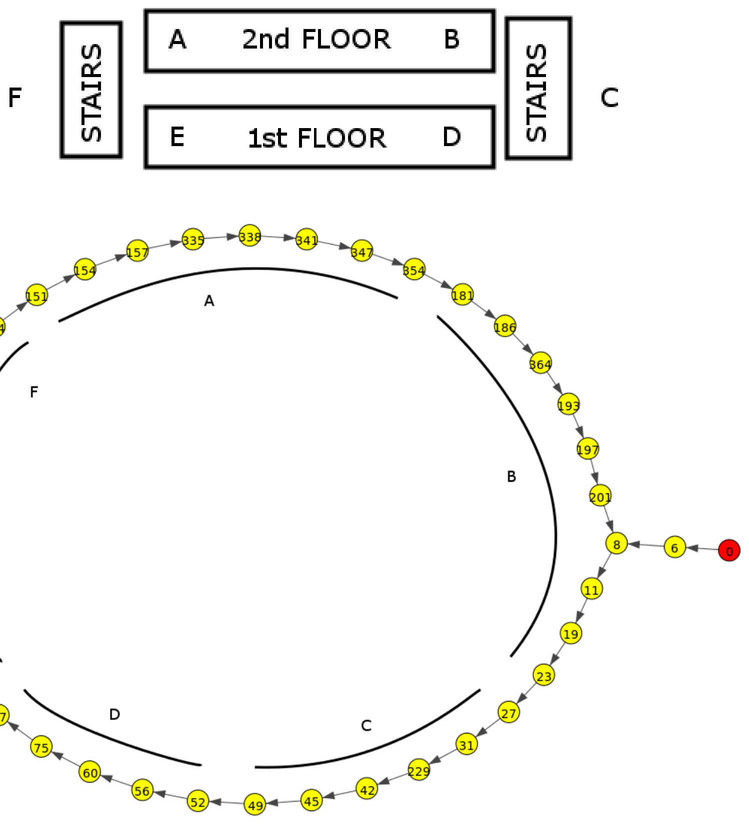


Figure 19: (Top) Reference map for the *UIBIndoor* dataset. (Bottom) Topological map generated using our approach. Each part of the map is identified with a letter in both maps. The red node is the beginning of the sequence.

	With Map Refinement				Without Map Refinement			
Dataset	N	M	L	%CL	N	M	L	%CL
Lip6Indoor	40	137.37	6.27	64	62	155.36	9.08	63
Lip6Outdoor	103	1005.52	29.05	63	141	1150.87	42.95	61
UIBSmallLoop	59	152.2	8.25	75	61	155.3	8.57	77
UIBLargeLoop	100	728.94	44.29	74	111	798.53	60.98	78
UIBIndoor	40	118.4	16.2	76	59	190.39	24.64	73

Table 2: Results for the map refinement experiment. *N*: number of nodes; *M*: mapping time in seconds; *L*: localization time in seconds; *%CL*: ratio between correct localizations and total number of elements. See text for details.

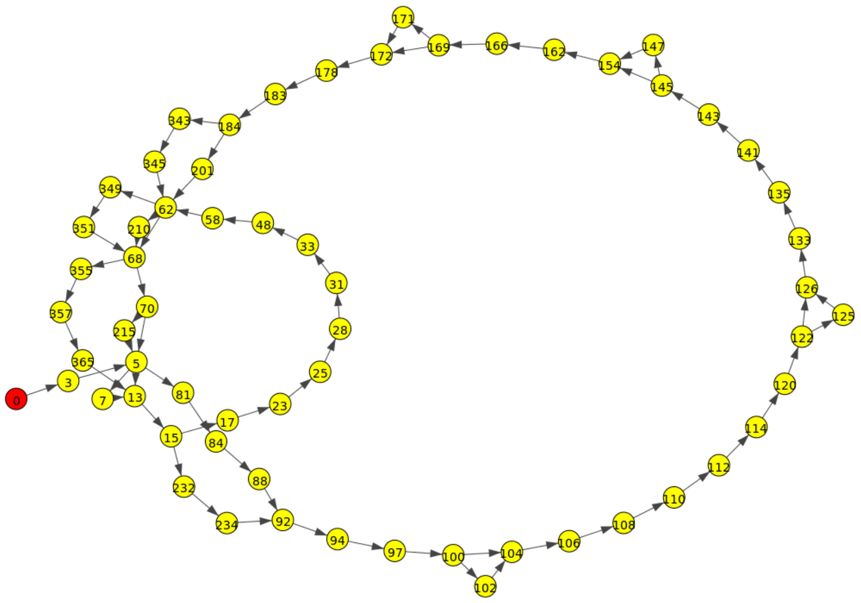


Figure 20: Map of the *Lip6Indoor* dataset obtained without using the map refinement strategy. The red node indicates the beginning of the sequence.

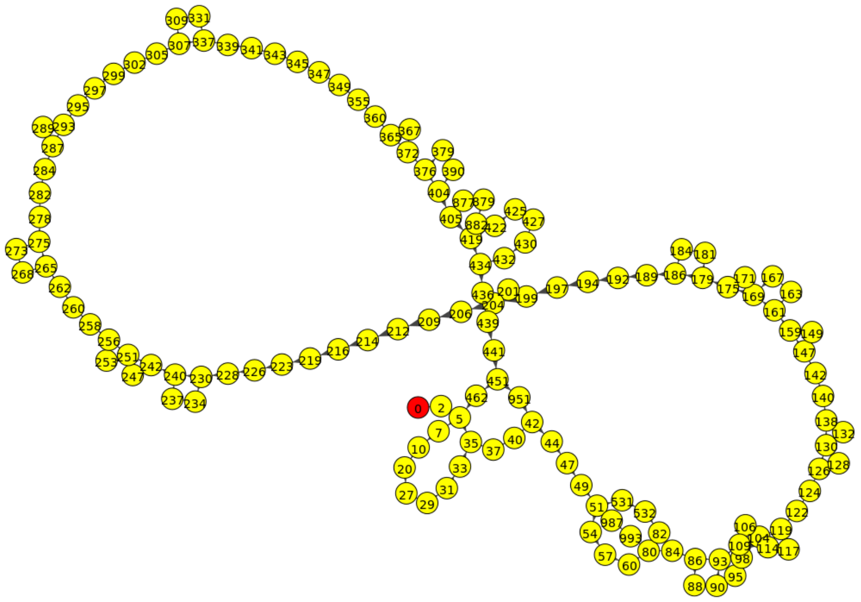


Figure 21: Map of the *Lip6Outdoor* dataset obtained without using the map refinement strategy. The red node indicates the beginning of the sequence.

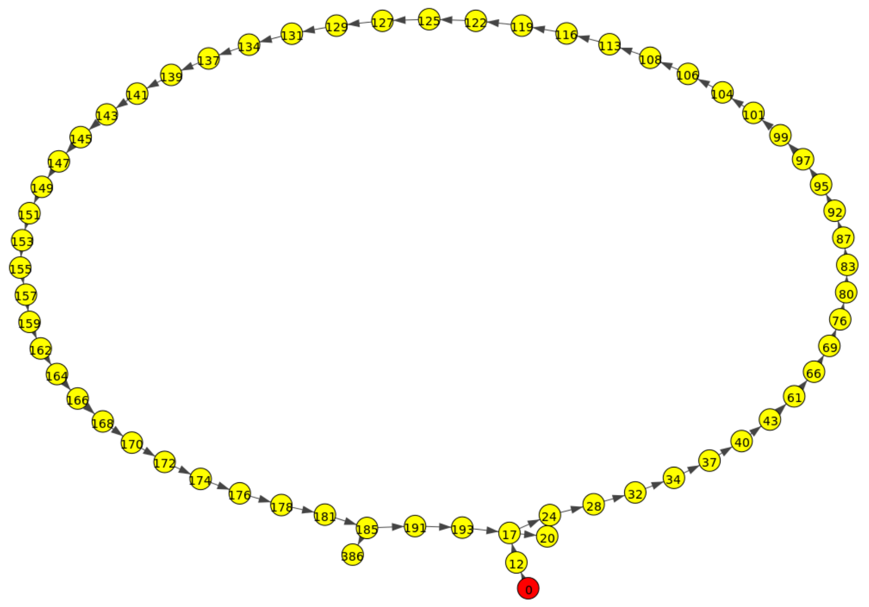


Figure 22: Map of the *UIBSmallLoop* dataset obtained without using the map refinement strategy. The red node indicates the beginning of the sequence.

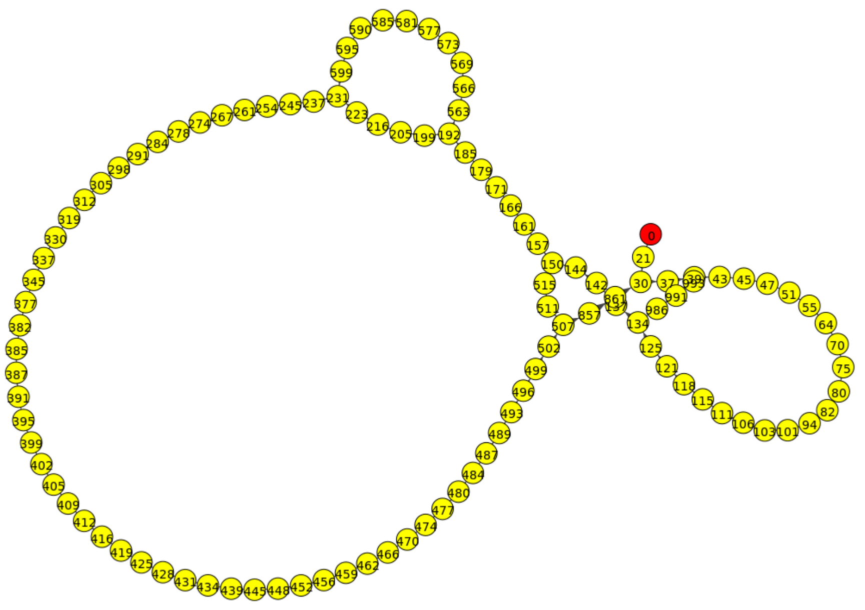


Figure 23: Map of the *UIBLargeLoop* dataset obtained without using the map refinement strategy. The red node indicates the beginning of the sequence.

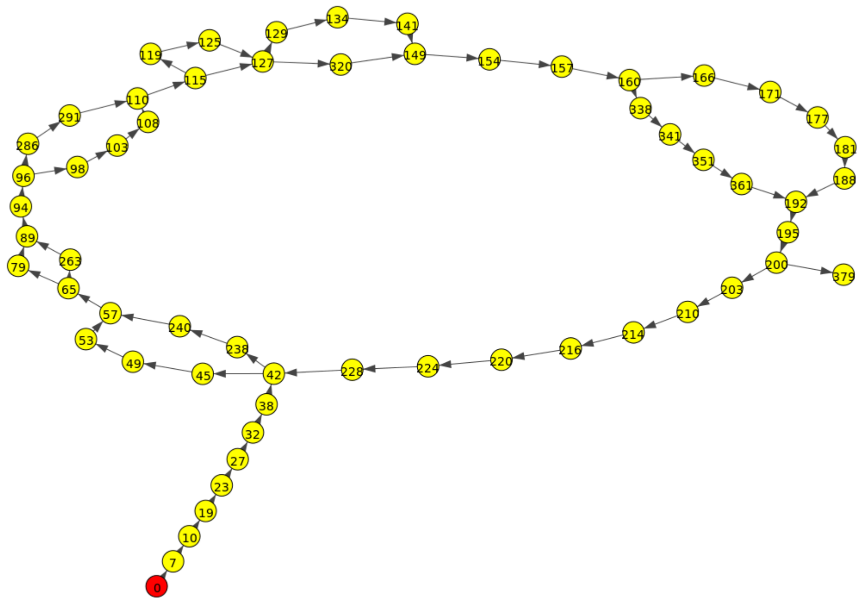


Figure 24: Map of the *UIBIndoor* dataset obtained without using the map refinement strategy. The red node indicates the beginning of the sequence.

matching scheme between the current image and the features of the current nodes in the map, using an index based on a set of randomized KD-trees. As a result, a topological map of the environment is obtained, which represents the scenario of the robot as a graph.

Using probabilistic filters for mapping and localization tasks usually produces spurious nodes and redundant paths over the graph. This is due to imperfections in the acquired images and the delays introduced by the filter. A key contribution of this work is a map refinement strategy for solving these problems, producing cleaner maps and saving storage space and computation resources for the mapping and localization tasks. This framework is executed each time a loop is closed and a predefined neighbourhood is refined in each step. The final decision of deleting nodes is taken according to the visual features of each path, avoiding to delete real paths in the environment.

In order to validate our solution, results from an extensive set of experiments, using datasets from different environments, have been reported. These results provide promising results, showing that our mapping and localization approach using a map refinement phase can be used for generating topological maps of the environment that, if they are provided with odometry information, can also be used for navigating in the current scenario.

Referring to future work, we intend to explore: (a) the use of other kinds of image descriptors based on local invariant features, such as e.g. binary descriptors, since they can improve our approach in computational terms; (b) the execution of the Bayes filter in a GPU to further the speed up the loop closure detection; and (c) the use of the full algorithm for larger mapping and localization environments.

Acknowledgements

This work is supported by the European Social Fund through the grant FPI11-43123621R (Conselleria d'Educacio, Cultura i Universitats, Govern de les Illes Balears) and by FP7 project INCASS (GA 605200).

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," in *International Conference on Computer Vision*, pp. 1470–1477, 2003.

[3] H. Zhang, "BoRF: Loop-Closure Detection with Scale Invariant Visual Features," in *International Conference on Robotics and Automation*, pp. 3125–3130, 2011.

[4] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Incremental Vision-Based Topological SLAM," in *International Conference on Intelligent Robots and Systems*, pp. 22–26, 2008.

[5] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose, "Navigation Using an Appearance Based Topological Map," in *International Conference on Robotics and Automation*, no. April, pp. 3927–3932, 2007.

[6] Z. Zivkovic, B. Bakker, and B. Krose, "Hierarchical Map Building Using Visual Landmarks and Geometric Constraints," in *International Conference on Intelligent Robots and Systems*, pp. 2480–2485, 2005.

[7] I. Ulrich and I. Nourbakhsh, "Appearance-Based Place Recognition for Topological Localization," in *International Conference on Robotics and Automation.*, pp. 1023–1029, 2000.

[8] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool, "Markerless Computer Vision Based Localization using Automatically Generated Topological Maps," in *European Navigation Conference*, pp. 235–243, 2004.

[9] D. G. Sabatta, "Vision-based Topological Map Building and Localisation using Persistent Features," in *Robotics and Mechatronics Symposium*, pp. 1–6, 2008.

[10] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[11] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Real-Time Visual Loop-Closure Detection," in *International Conference on Robotics and Automation*, pp. 1842–1847, 2008.

[12] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "A Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[13] H. Zhang, "Indexing Visual Features: Real-Time Loop Closure Detection Using a Tree Structure," in *International Conference on Robotics and Automation*, pp. 3613–3618, 2012.

[14] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Visual Topological SLAM and Global Localization," in *International Conference on Robotics and Automation*, pp. 4300–4305, 2009.

[15] F. Fraundorfer, C. Engels, and D. Nister, "Topological Mapping, Localization and Navigation Using Image Collections," in *International Conference on Intelligent Robots and Systems*, pp. 3872–3877, 2007.

[16] A. Oliva and A. Torralba, "Modeling the Shape of the Scene : A Holistic Representation of the Spatial Envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[17] G. Singh and J. Kosecka, “Visual Loop Closing using Gist Descriptors in Manhattan World,” in *Omnidirectional Robot Vision workshop, held with IEEE ICRA*, 2010.

[18] Y. Liu and H. Zhang, “Visual Loop Closure Detection with a Compact Image Descriptor,” in *International Conference on Intelligent Robots and Systems*, pp. 1051–1056, 2012.

[19] C. Siagian and L. Itti, “Rapid Biologically-Inspired Scene Classification using Features Shared with Visual Attention,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 300–12, 2007.

[20] A. Kawewong, N. Tongprasit, S. Tungruamsub, and O. Hasegawa, “Online and Incremental Appearance-Based SLAM in Highly Dynamic Environments,” *International Journal of Robotics Research*, vol. 30, no. 1, pp. 33–55, 2011.

[21] H. Zhang, B. Li, and D. Yang, “Keyframe Detection for Appearance-Based Visual SLAM,” in *International Conference on Intelligent Robots and Systems*, pp. 2071–2076, 2010.

[22] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] E. Garcia-Fidalgo and A. Ortiz, “Probabilistic Appearance-Based Mapping and Localization Using Visual Features,” in *Pattern Recognition and Image Analysis*, vol. 7887 of *Lecture Notes in Computer Science*, pp. 277–285, 2013.