On the Use of Binary Feature Descriptors for Loop Closure Detection

Emilio Garcia-Fidalgo and Alberto Ortiz
Department of Mathematics and Computer Science
University of the Balearic Islands
07122 Palma de Mallorca, Spain
{emilio.garcia, alberto.ortiz}@uib.es

Abstract

We propose an appearance-based loop closure detection algorithm based on binary features and a Bag-of-Words scheme. Unlike other approaches that build the visual dictionary offline, we introduce an indexing method for binary features, which, in combination with an inverted index, enable us to obtain loop closure candidates in an online manner. These structures are used in a discrete Bayes filter to select final loop candidates and to ensure temporal coherency between predictions. Our approach is validated using two publicly available datasets of outdoor environments and compared with the state-of-the-art FAB-MAP algorithm, showing very promising results and demonstrating that binary features can be used for visual loop closure detection.

1 Introduction

Simultaneous Localization and Mapping (SLAM) [8] algorithms perform mapping and localization tasks at the same time, creating an incremental map of an unknown environment while localizing the robot within this map. These techniques are essential in order to achieve autonomy in mobile platforms. In SLAM, loop closure detection is a key challenge to overcome which entails the correct identification of previously seen places from sensor data, allowing the generation of consistent maps. Several kinds of sensors have been used for years for 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.

Many appearance-based algorithms for loop closure detection developed recently ([11], [2], [5], [12]) are based on the Bag-Of-Words (BoW) approach [23, 19]. A BoW is a sparse vector representation of an image which is created quantizing the detected local features according to a set of representative features called *visual words*, which conform the *visual vocabulary*. This quantization takes place mapping each descriptor of the image onto the nearest image word in the dictionary. Next, the image is represented by a histogram of occurrences of each visual word in the image, reducing the total set of feature descriptors to a vector of integers. The visual dictionaries can be generated offline or online. As a main limitation, the offline approaches need a training phase, where sometimes millions of descriptors need to be clustered. This can take hours, depending on the number of training descriptors and the clustering technique used. Furthermore, the robot can operate in an environment with a totally different appearance from the training set employed for generating the dictionary, which implies that it is not representative of the scenario, augmenting the number of false detections. An alternative is to build the visual dictionary in an incremental manner, while the robot is navigating across the environment.

SIFT [16] and SURF [3] are the most commonly used features in this context, due to their invariance properties to illumination, scale and rotation changes. However, the detection and description of these features are computationally expensive. Recently there has been a growing interest in the use of binary descriptors, such as BRIEF [4], ORB [21], BRISK [15] and FREAK [1]. These features present advantages over the real-valued descriptors since they are faster to compute and require less storage space [12, 9]. Binary features are compared using the Hamming distance, which can be efficiently computed by means of a bitwise XOR operation and bit summation. Modern computers provide hardware support for executing these operations quickly. Despite these benefits, binary features have not been much used for visual loop closure detection.

In this paper, we present a method for computing a binary vocabulary that can be built online, avoiding a training phase and making use of binary features. This binary vocabulary is employed in a probabilistic loop closure detection algorithm based on a Bayes filter. The proposed indexing scheme enables us to match binary descriptors in an efficient manner, speeding up the loop closure detection process. We perform several experiments with two public data sets and compare our approach with the state-of-the-art FAB-MAP [5] algorithm, showing very promising results and demonstrating that these kind of features can be used in an online approach for detecting loop closures.

2 Related Work

The BoW algorithm was first applied to visual search techniques in the seminal work of Sivic an Zisserman [23], where this model was employed to detect similar scenes in video sequences. The SIFT descriptors extracted from a set of training images were clustered using the k-means algorithm, generating a visual vocabulary. When a query input frame was received, its descriptors were quantized using these visual words. As a result, the image was described by a list of integers specifying the number of occurrences of each visual word in the image. Using inverted files, a scoring process based on Term Frequency Inverse Document Frequency (TF-IDF) weighting was performed in order to select similar previous images. The main drawback of this approach was the high computational cost, since a linear search was used in order to find the nearest reference descriptor in the visual dictionary. Nister and Stewenius [19] improved this indexing scheme proposing a vocabulary tree based on a hierarchical k-means approach. This hierarchical quantization allowed to use a larger vocabulary size leading to a better recognition performance. Using this approach, Fraundorfer et al. [11] presented a highly scalable vision-based localization and mapping method using image collections. They used local geometric information to navigate within the topological map.

Probably the most popular solution based on the offline BoW approach is Fast Appearance-Based Mapping (FAB-MAP) [5], where a Chow-Liu tree approximates the co-occurrences between the visual words in the vocabulary. This approximation permitted the authors to compute efficiently an observation likelihood which was used in a Bayes filter for predicting loop closure candidates. Initially, this likelihood was computed for each image candidate in the filter, resulting into computational problems. To speed up this process, their work was improved in [6], introducing a probabilistic bail-out test based on the use of concentration inequalities for rapidly identifying promising loop closure hypotheses, and in [7], adapting the probabilistic model to be used with an inverted index similar to typical image search engines. Conversely, some authors have tried to detect loop closures using online visual vocabularies. Angeli et al. [2], using the online visual dictionary proposed by Filliat [10], extended the BoW paradigm to incremental conditions and relied on Bayesian filtering to estimate the probability of loop closure. Nicosevici and Garcia [18] presented an online visual vocabulary building method based on agglomerative clustering. They used this algorithm for mapping underwater environments.

All approaches exposed so far make use of SIFT [16] or SURF [3] features. The use of binary descriptors can speed up the loop closure detection process, since they are computed efficiently. The only approach found that creates a binary vocabulary is the work developed by Galvez-Lopez and Tardos [12]. They adapted the hierarchical BoW model proposed by Nister [19] to be used with key points detected with FAST [20] and described with the BRIEF [4] algorithm. Other novelties of their work included a direct index to obtain correspondences between images in an efficient way and matching images in groups to increase the accuracy of the loop closure detection process. In contrast to this approach, our approach builds a visual vocabulary online and relies on a Bayes filter to detect loop closures.

For searching words in a big visual dictionary, a linear search is not practical. This problem is solved using an approximate matching algorithm, which usually employs hierarchical structures, such as kd-trees or hierarchical k-means trees, to speed up the process. These structures are not suitable for binary descriptors since they assume that each dimension of the vector can be continuously averaged. Typical matching techniques for binary descriptors include hashing techniques, e.g. Locality Sensitive Hashing (LSH) [14] or Semantic Hashing [22]. Recently, Muja and Lowe [17] presented an algorithm for matching binary descriptors based on a hierarchical decomposition of the search space which performs better in comparison with the hashing approaches. In our approach, we modify this structure to be used as a binary vocabulary within a loop closure detection algorithm.

In a previous approach [13], we developed a topological mapping and localization approach based on an efficient index of invariant features and a map refinement strategy. In this work, we address the replacement of the loop closure detector by an online BoW scheme using binary features.

3 Online Binary Visual Dictionary

In order to obtain loop closure candidates using an online BoW approach, we need efficient structures for searching descriptors. Recently, Muja and Lowe [17] presented an algorithm for matching binary descriptors which performs a hierarchical decomposition of the search space by successively clustering the input data set and constructing a tree. Initially, all the points in the data set are clustered using a k-medoids algorithm with K centers selected randomly. This process is repeated recursively until the number of leaf nodes in each cluster is below some threshold. The search is performed starting from the root until reaching a leaf node and, then, the points contained within this leaf are linearly searched in order to find the closest candidates. They also proposed building multiple trees and using them in parallel during the search to improve the speed of the process. The performance of this index is directly related to the input parameters: the branching factor, the maximum leaf size and the number of search trees. According to the results presented in [17], their approach requires less storage space, scales better compared to LSH and seems to be very effective for matching binary descriptors.

In our loop closure detection approach, we use a modified version of the Muja and Lowe's approach as an incremental visual dictionary. It is combined with an inverted index, which contains, for each word in the dictionary, a list of images where it was found. Since our approach relies on an incremental visual dictionary based on binary features, an updating policy for combining binary descriptors is needed. Averaging each component of the vector is an option for real-valued descriptors, but it can not be considered for the binary case. We propose to use a bitwise AND operation. Formally, being B a binary descriptor:

$$B^{t}_{w_i} = B^{t-1}_{w_i} \wedge B_q \, , \tag{1}$$

where $B^{t-1}_{w_i}$ is the binary descriptor of the word w_i stored in the dictionary at time instant $t-1$, Bq is the query descriptor and $B^{t}_{w_i}$ is the merged descriptor stored for the word w_i at time t. This policy is inspired by the observation that each component of a binary descriptor is usually set to 1 or 0 according to the result of a comparison between a pair of image pixel intensities, e.g. BRIEF, ORB and FREAK. If the *i*-th bit is the same in both descriptors, it means that the result of this comparison between the pixel intensities was the same in both images. Otherwise, we experimentally prioritize the use of the zero value by means of the AND operation.

The index is initially built using the descriptors of the first image. When a new image needs to be added to the index, their descriptors are searched in the index. Given a query binary descriptor, we search for the two nearest neighbours traversing the tree from the root to the leafs and selecting at each level the node that minimizes the Hamming distance. Using these two neighbours, we apply the ratio test [16] in order to determine if both descriptors represent the same visual feature. If positive, the query descriptor and the visual word are merged using (1) and replaced in the dictionary. Otherwise, the query descriptor is considered a new feature and is added to the index as a new visual word. In both cases, the inverted index is updated accordingly, adding a reference to the current image in the list corresponding to the modified or added feature. The updating process of the visual dictionary is summarized in Algorithm 1.

Algorithm 1 Update Visual Dictionary

```
Require:
  $F_t$: Descriptors from image $I_t$.
  $\rho$: Ratio between nearest neighbours.
  for all $d$ in $F_t$ do
    [$n1$, $n2$] = nearestNeighbours($d$, 2)
    if dist($d$, $n1$) $<$ dist($d$, $n2$) * $\rho$ then
      $B = d \wedge n1$
      replaceDescriptor($n1$, $B$)
      addToInvertedFile($I_t$, $n1$)
    else
      addDescriptor($d$)
      addToInvertedFile($I_t$, $d$)
    end if
  end for
```

4 Bayesian 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 previously seen images, allowing us to deal with noisy measurements and helping us to discard false recognitions. The Bayesian framework is also used for ensuring temporal coherency between consecutive predictions, integrating past estimations over time. It can also be used for fusing sensory information from different sources, such as cameras, lasers or IMUs, provided that an observation model is available for each one. In our case, we use a monocular camera as input. The Bayes filter, which is described below, is based on our previous approach [13].

Given the current image I_t at time t, we denote z_t as the set of binary descriptors extracted from this image. These are the observations in our filter. We also denote L^t_i 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 previous image I_c whose index satisfies:

$$c = \underset{i=0,\ldots,t-p}{\arg\max} \left\{ P\left(L^t_i | z_{0:t}\right)\right\} , \tag{2}$$

where $P\left(L^t_i | z_{0:t}\right)$ is the full posterior probability at time t given all previous observations up to time t. As in [2], 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 would 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:

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

and then, using conditional probability properties, we can isolate our final goal to obtain:

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

where $P\left(z_t | z_{0:t-1}\right)$ can be seen as a normalizing factor since its computation does not depend on L^t_i. Under this premise and the Markov assumption, the posterior is defined as:

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

where η represents the normalizing factor, $P\left(z_t | L^t_i\right)$ is the observation likelihood and $P\left(L^t_i | z_{0:t-1}\right)$ is the prior, computed 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 Angeli [2] and Cummins [5], we do not model explicitly the probability of no loop closure in the posterior. If the probability of loop closure of I_t with I_c ($P\left(L_c^t|z_{0:t}\right)$) is not high enough, we discard L_c^t as a possible loop candidate.

4.1 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 × (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.

4.2 Observation Model

Once the prediction step is performed, the current observation needs to be included in the filter. We have to compute the most likely images given the current frame I_t and its keypoint descriptors z_t, but we want to avoid comparing I_t with each previous image, since this is not tractable. To this end, we use the visual online dictionary described in section 3 in combination with the inverted index.

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 a 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 $t-p$. We search each descriptor in z_t in the visual dictionary in order to find the closest word. Each time a word is found, the inverted index enables us to obtain a list of past image where this word was found. We then add a statistic about the word to the correspondent score for each retrieved image. This statistic is the Term Frequency-Inverse Document Frequency (tf-idf) weighting factor, which reflects how important a word is to the query image in a collection of the received images up to the current time t. Given the set of images $I_{0:t-p}$, which are the images inside the filter and the dictionary at time t, the tf-idf value $\rho_{w_j}^i$ computed given the word w_j and the image I_i is defined as:

$$\rho_{w_j}^i = \mathrm{tf}(w_j, I_i) \times \mathrm{idf}(w_j, I_{0:t-p}) , \tag{7}$$

which is the product of the term *tf*, the frequency of the word in the image, and the term *idf*, the inverse frequency of the images containing this word. The term *tf* is defined as:

$$\mathrm{tf}(w_j, I_i) = \frac{n_{w_j}^i}{N_i} , \tag{8}$$

being $n_{w_j}^i$ the number of occurrences of the word w_j in the image I_i, and N_i the total number of features found in the image I_i. The term *idf* is defined as:

$$\mathrm{idf}(w_j, I_{0:t-p}) = \log \frac{\#I_{0:t-p}}{n_{w_j}} = \log \frac{t-p}{n_{w_j}} , \tag{9}$$

where $\#I_{0:t-p}$ is the cardinal of set $I_{0:t-p}$, and n_{w_j} is the total number of images in $I_{0:t-p}$ containing the word w_j. This value is accumulated onto the corresponding score according to:

$$s\left(z_t, z_i\right) = s\left(z_t, z_i\right) + \rho_{w_j}^i , \tag{10}$$

being i the index of the image extracted from the inverted index. 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 [2]):

$$P\left(z_t|L_i^t\right) = \begin{cases} \frac{s(z_t,z_i)-2s_\sigma}{s_\mu} & \text{if } s\left(z_t, z_i\right) \geq s_\mu + 2\, s_\sigma \\ 1 & \text{otherwise} \end{cases} , \tag{11}$$

being, respectively, s_μ and s_σ the mean and the standard deviation of the set of scores. Notice that, by means of (11), given the current observation z_t, only the most likely locations update their posterior. After incorporating the observation into our filter, the full posterior is normalized in order to obtain a probability density function.

4.3 Selection of a Loop Closure Candidate

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 sum the probabilities along a predefined neighbourhood. This neighbourhood is the same as defined in section 4.1, i.e. 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. Matchings that do not fulfill the epipolar constraint are discarded by means of

RANSAC. If the number of surviving matchings is above a threshold T_{ep}, the loop closure hypothesis is accepted; otherwise, it is definitely rejected.

Finally, we define another threshold T_{hyp} 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 full loop closure detection approach is outlined in Alg. 2.

Algorithm 2 Visual Loop Closure Detection

```
Require:
  $I_t$: Current image.
  $B$: Discrete Bayes filter.
  $D$: Binary visual dictionary.
  $p$: Number of recent images to be discarded.
  $\rho$: Ratio between nearest neighbours.

  /* Variables */
  $F_i$: Descriptors from image $I_i$.
  $M_t$: Matches between the descriptors of image $I_t$ and the words in dictionary.
  $c$: Candidate image index for closing a loop.
  $P_c$: Probability of candidate image c.
  $n_{hyp}$: Number of hypotheses in the Bayes filter.

  $t = t + 1$
  $I_t$ = getImage()
  storeImage($I_t$)
  $n$ = $t$ - $p$
  if $n < 0$ then
    continue
  end if
  $F_n$ = describeImage($I_n$)
  updateVisualDictionary($D$, $F_n$, $\rho$)
  addHypothesisToBayesFilter($B$, $I_n$)
  $F_t$ = describeImage($I_t$)
  $M_t$ = searchVisualDictionary($D$, $F_t$)
  BayesFilterPredict($B$)
  BayesFilterUpdate($B$, $M_t$)
  $[c, P_c]$ = getMostLikelyLocationFromBayesFilter($B$)
  if $P_c > T_{loop}$ and $n_{hyp} > T_{hyp}$ then
    inliers = epipolarGeometry($F_t$, $F_c$)
    if number_of(inliers) $> T_{ep}$ then
      /* Loop Detected */
      registerLoop(t, c)
    else
      /* Hypothesis Rejected */
      doNothing()
    end if
  else
    /* No Loop Detected */
    doNothing()
  end if
```

Parameter	City Centre	New College
Branching factor	10	
Search Trees	4	
Leaf node size	100	
p	20	
Inliers	12	60
Features	650	900
T_{hyp}	20	

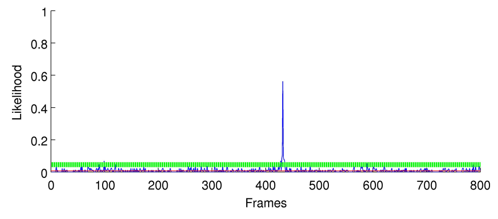
Table 1. Parameters used in our experiments for each dataset.

5 Experimental Results

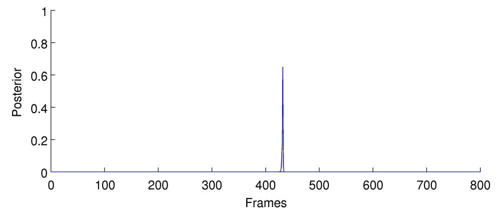
We evaluate our approach using two outdoor urban data sets, published for the validation of FAB-MAP [5]. The City Center and the New College datasets are composed, respectively, of 1237 and 1073 pairs of images of size 640×480 taken by the left and right cameras mounted on a robot while it travels through the environment. Since our approach has been developed to be used with monocular cameras, we merge left and right frames resulting into images of size 1280×480. The first data set was recorded to validate the ability of a system for matching images in the presence of scene changes, while the second one was recorded because of its high perceptual aliasing conditions. All experiments were performed on a desktop computer fitted with an Intel Core i3 at 2.27Ghz processor and 4GB of RAM memory. In order to obtain global performance measures, each dataset is provided with a ground truth, which indicates, for each image in the sequence, which images can be considered to close a loop and with which image. The assessment against this ground truth has been performed 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, the two following metrics were computed:

- *Precision*. Ratio between real loop closures and total amount of loop closures detected $\left(\frac{TP}{TP+FP}\right)$.
- *Recall*. Ratio between real loop closures and total amount of loop closures existing in the sequence $\left(\frac{TP}{TP+FN}\right)$.

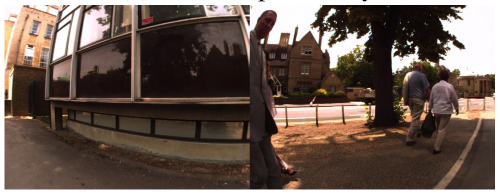
We process each sequence using our algorithm configured with the parameters indicated in Table 1. Initially, the visual dictionary is empty and grows as the sequence progresses, following the policy explained in this paper. In this work we extract FAST [20] features for each image, due to its speed in corner detection, and then, each feature is described by means of the BRIEF [4] binary descriptor. Note that our approach is descriptor-independent and another binary descriptor could be used instead of BRIEF.



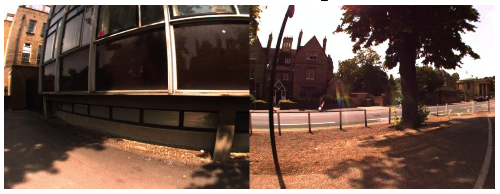
(a) Likelihood



(b) Location probability.



(c) Current image



(d) Loop closure image.

Figure 1. Example of loop closure detection. Red and green lines show respectively s_μ and $s_\mu + 2\,s_\sigma$ values.

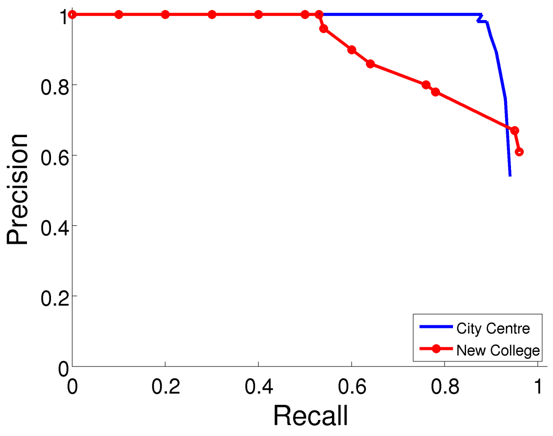


Figure 2. Precision-recall curves for each dataset. These curves were generated modifying the threshold for loop acceptance (T_{loop}).

Dataset	#Imgs	TP	TN	FP	FN	Pr	Re
City Center	1237	497	676	0	64	100	88
New College	1073	220	656	0	193	100	53

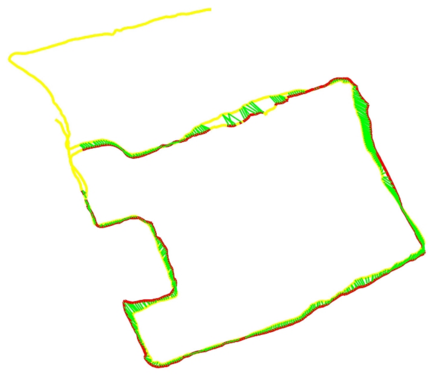
Table 2. Results for the two data sets. Precision (Pr) and Recall (Re) columns are expressed as percentages. See text for details

An example of a loop closure detection is given in Fig. 1. Fig. 1 (a) shows the likelihood computed given the image 971, which is consistent with the posterior shown in Fig 1 (b). Both plots present a high peak around the image 430. Fig. 1 (c) and Fig. 1 (d) are respectively the current robot view and the retrieved location. As can be seen, our approach is able to detect the loop closure situation despite there are changes in the scene.

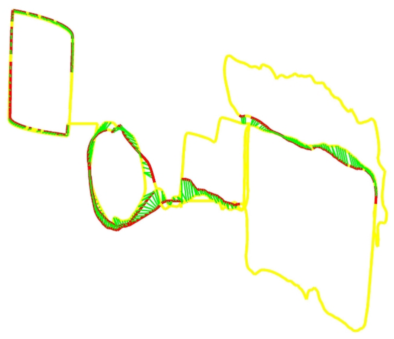
The precision-recall curves for each dataset are shown in Fig 2. These curves were plotted modifying the threshold for loop acceptance (T_{loop}). For an easier understanding of the algorithm performance, the best results for a 100% of precision are shown in Table 2. As can be seen, no false positives 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 both datasets. Our approach outperforms FAB-MAP in all datasets [5], obtaining a higher recall for a 100% of precision.

As can be seen, a high rate of correct detections were obtained from all experiments. False negatives are due to, on the one hand, the sensitivity of the filter. In effect, 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 increase the posterior for this hypothesis. This introduces a delay in the loop closure detection, which derives in false negatives. This sensitivity can be tuned by modifying the transition model of the filter, although a higher sensitivity can introduce loop detection errors, i.e. false positives. On the other hand, false negatives can also be due to camera rotations. When the camera is turning around a corner, it is difficult to find and match features in the images, 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}.

Figure 3 shows navigation results obtained using our approach for the two data sets. These figures are plotted following the same colours as in the FAB-MAP's original paper [5]. Each image of the sequence is labelled with a yellow dot. When a loop closure is detected, images representing this loop are labelled in red and linked with a green line. As can be seen, we detect more loop closures than FAB-MAP for the City Center data set, demonstrat-



(a) City Center



(b) New College

Figure 3. Appearance-based loop closure results for the City Center (left) and New College (right) datasets. GPS positions of the images are plotted with a yellow dot. Wherever an image closes a loop with another one, both are marked with a red dot and joined with a green line.

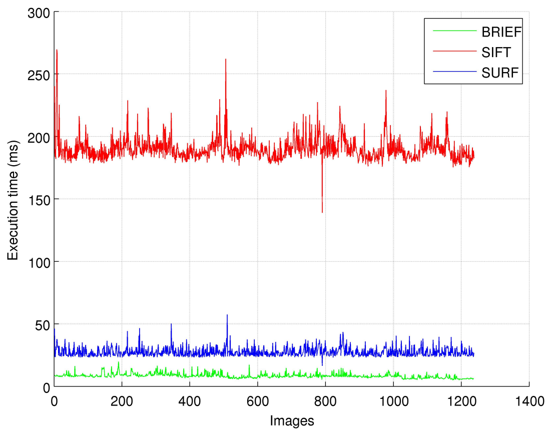


Figure 4. Description times for describing 650 features obtained with FAST.

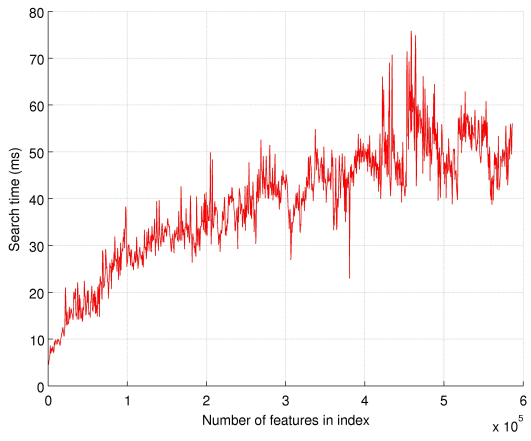


Figure 5. Time for searching 650 descriptors regarding the number of features stored in the index.

ing that our approach can deal with scene changes. For the case of the New College data set, we obtain a similar path, showing that, at least, we can obtain similar results for environments under high perceptual aliasing conditions.

We evaluate the performance of our approach in terms of computational time. Fig. 4 shows the times needed to describe an image using SIFT, SURF and BRIEF for a set of 650 corners extracted using the FAST detector. As shown in the figure, BRIEF outperforms the real-valued descriptors in computational terms. This implies a benefit in the approach, saving time in the description phase.

We also evaluate our visual dictionary for searching words. Fig. 5 presents times for searching 650 features per image according to the number of visual words stored in the dictionary. As can be seen, when the index arrives at approximately 60K features, the total search time is only about 50 ms, which shows that our approach can deal with a high number of features despite it is an online approach. The searching times initially grow fast, but for larger amounts of features in the index, the growth is far more contained, as shown in the figure.

6 Conclusions

In this paper we have introduced a probabilistic loop closure detection algorithm based on a discrete Bayes filter, which uses an online BoW approach to search for loop candidates. Unlike other approaches that make use of visual dictionaries generated offline, in this work we propose an online visual dictionary which is based on a hierarchical decomposition of the search space by means of a tree. This index uses binary descriptors, which improves the speed of the searching process. We extent an existing binary indexing algorithm to be used as an online visual dictionary, which in combination with an inverted index, enables us to obtain loop closure candidates in an efficient way. We validate our approach using two outdoor datasets and compare our results with FAB-MAP, one of the state-of-the-art loop closure detection approaches. We demonstrate that our solution can be used for loop closure detection, improving the detection speed and showing very promising results.

Referring to future work, we intend to explore: (a) the inclusion of some information about the spatial arrangement of the visual words to improve the recognition performance; (b) the use of other binary descriptors, e.g. ORB [21] or BRISK [15]; and (c) the execution of the Bayes filter in a Graphics Processing Unit (GPU) to further speed up the loop closure detection.

Acknowledgments

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

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK : Fast Retina Keypoint. In *International Conference on Computer Vision and Pattern Recognition*, pages 510–517, 2012.

[2] 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*, 24(5):1027–1037, 2008.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417, 2006.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF : Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792, 2010.

[5] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *International Journal of Robotics Research*, 27(6):647–665, 2008.

[6] M. Cummins and P. Newman. Accelerating FAB-MAP With Concentration Inequalities. *IEEE Transactions on Robotics*, 26(6):1042–1050, 2010.

[7] M. Cummins and P. Newman. Appearance-Only SLAM at Large Scale with FAB-MAP 2.0. *International Journal of Robotics Research*, 30(9):1100–1123, 2011.

[8] H. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 2:99–110, 2006.

[9] F. Ferreira, G. Veruggio, M. Caccia, E. Zereik, and G. Bruzzone. A Real-Time Mosaicking Algorithm using Binary Features for ROVs. In *Mediterranean Conference on Control and Automation*, pages 1267–1273, 2013.

[10] D. Filliat. A Visual Bag of Words Method for Interactive Qualitative Localization and Mapping. In *IEEE International Conference on Robotics and Automation*, pages 3921–3926, 2007.

[11] F. Fraundorfer, C. Engels, and D. Nister. Topological Mapping, Localization and Navigation Using Image Collections. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, 2007.

[12] D. Galvez-Lopez and J. Tardos. Real-Time Loop Detection with Bags of Binary Words. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 51–58, 2011.

[13] E. Garcia-Fidalgo and A. Ortiz. Vision-Based Topological Mapping and Localization by means of Local Invariant Features and Map Refinement. *Robotica (in press)*, 2014.

[14] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *International Conference on Very Large Data Bases*, pages 518–529, 1999.

[15] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *International Conference on Computer Vision*, pages 2548–2555, 2011.

[16] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[17] M. Muja and D. G. Lowe. Fast Matching of Binary Features. In *Conference on Computer and Robot Vision*, pages 404–410, 2012.

[18] T. Nicosevici and R. Garcia. On-line Visual Vocabularies for Robot Navigation and Mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 205–212, 2009.

[19] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.

[20] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision*, number 1 in Lecture Notes in Computer Science, pages 430–443, 2006.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *International Conference on Computer Vision*, volume 95, pages 2564–2571, 2011.

[22] R. Salakhutdinov and G. Hinton. Semantic Hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July 2009.

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