$(y_1 DM)_j = 1$ if $j = (m+1)^2$, and $(y_1 DM)_j = 0$ otherwise. Let $(y_2)_j = 1$ if $j = m+1$, and $(y_2)_j = 0$ otherwise. The statement follows from Motzkin's Theorem since $y_1 DM - y_2 N = 0$. ■

Proof of Theorem 3.1: Let $V_{\min} = \emptyset$, $V = V_{\max}$, and $|V| = n = m+1$. Let δ^* be the vector of waiting intervals and arrival times defined in Trajectory 1. Note that the specification of the robots initial positions implies the specification of the arrival times $\mathcal{A}_{\min}$. Suppose that δ^* is not an optimal solution to the optimization problem A4, and let δ be a minimizer of A4. Then, $\|b + DM\delta\|_\infty < \|b + DM\delta^*\|_\infty$. It can be verified that the entries of $b + DM\delta^*$ indexed by $V_{\max}$ are all equal to each other. In addition, $b + DM\delta \succeq 0$, $b + DM\delta^* \succeq 0$. Hence

$$DM(\delta - \delta^*) \prec 0. \tag{A5}$$

Notice that, since $\delta_1^i(k)^* = 0$ and $\delta_1^i(k) \geq 0$ due to the constraint in A4, it follows that $\delta_1^i(k) - \delta_1^i(k)^* \geq 0$. Because of Lemma 6.2, the inequalities A5 with the constraint $\delta_1^i(k) - \delta_1^i(k)^* \geq 0$ are infeasible. Due to convexity, we conclude that δ^* is a global minimizer of A4 with $V_{\min} = \emptyset$.

Let $V_{\min} \neq \emptyset$, and observe that, because of Lemma 2.2, the weighted refresh time for the set of viewpoints $V_{\max} \cup V_{\min}$ cannot be smaller than the weighted refresh time for the set of viewpoints $V_{\max}$. Then, the vector δ^* that is defined in Trajectory 1 is a global minimizer of the optimization problem A4.

We now characterize the performance of Trajectory 1. Notice that $x_i(t) = x_{i+1}(t + \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1})$. Hence, the viewpoint v_α is not visited for an interval of length $\frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1} - \delta_\alpha$, where δ_α is the waiting interval at the viewpoint v_α. We have

$$\frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1} - \delta_\alpha = \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1} - \frac{\mathrm{RT}_{\mathrm{T}}^*(\phi_\alpha - \phi_1)}{\phi_\alpha \phi_1} = \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_\alpha}.$$

From (1), we have $\mathrm{RT}(X) = \max_\alpha \phi_\alpha \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_\alpha} = \mathrm{RT}_{\mathrm{T}}^*$. Note that

$$\frac{\Delta_1^m(1)}{\phi_1} = L + \mathcal{A}_{\min}(1) + \sum_{\alpha=1}^{n} \delta_\alpha^1(1) - \mathcal{A}_{\min}(m) - \delta_1^m(1)$$

where $\Delta_1^m(1) = \mathrm{RT}_{\mathrm{T}}^*$, $\delta_1^m(1) = 0$, and

$$\mathcal{A}_{\min}(1) - \mathcal{A}_{\min}(m) = -(m-1)\mathrm{RT}_{\mathrm{T}}^*/\phi_1$$

$$\sum_{\alpha=1}^{n} \delta_\alpha^1(1) = m\mathrm{RT}_{\mathrm{T}}^*/\phi_1 - \sum_{\phi \in \Phi_{\max}} \mathrm{RT}_{\mathrm{T}}^* \phi^{-1}.$$

Hence, $\mathrm{RT}_{\mathrm{T}}^* = \frac{L}{\sum_{\phi \in \Phi_{\max}} \phi^{-1}}$. Finally, for the Equal-Time-Spacing trajectory, it holds $\mathrm{RT}_1 = \mathrm{RT}_2 = \mathrm{RT}_{\mathrm{T}}^* = \mathrm{RT}(X)$. ■

REFERENCES

[1] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Trans*., vol. 46, no. 1, pp. 3–13, 2007.

[2] D. B. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Trans. Robot*., vol. 24, no. 6, pp. 1394–1404, Dec. 2008.

[3] S. Susca, S. Martínez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Trans. Control Syst. Technol*., vol. 16, no. 2, pp. 288–296, Mar. 2008.

[4] Y. Elmaliach, A. Shiloni, and G. A. Kaminka, "A realistic model of frequency-based multi-robot polyline patrolling," in *Proc. Int. Conf. Auton. Agents*, Estoril, Portugal, May 2008, pp. 63–70.

[5] I. I. Hussein and D. M. Stipanoviè, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Trans. Control Syst. Technol*., vol. 15, no. 4, pp. 642–657, Jul. 2007.

[6] C. G. Cassandras, X. C. Ding, and X. Lin. (2011, Aug.) An optimal control approach for the persistent monitoring problem [Online]. Available at: http://arxiv.org/pdf/1108.3221

[7] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *Proc. IEEE/WIC/ACM Int. Conf. Intell. Agent Technol*., Beijing, China, Sep. 2004, pp. 302–308.

[8] D. B. Kingston, R. S. Holt, R. W. Beard, T. W. McLain, and D. W. Casbeer, "Decentralized perimeter surveillance using a team of UAVs," presented at the AIAA Conf. Guid., Navigat. Control, San Francisco, CA, Aug. 2005.

[9] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis and approximation algorithms," *IEEE Trans. Robot*., 2012, DOI: 10.1109/TRO.2011.2179580.

[10] G. Cannata and A. Sgorbissa, "A minimalist algorithm for multirobot continuous coverage," *IEEE Trans. Robot*., vol. 27, no. 2, pp. 297–312, Apr. 2011.

[11] S. L. Smith and D. Rus, "Multi-robot monitoring in dynamic environments with guaranteed currency of observations," in *Proc. IEEE Conf. Decis. Control*, Atlanta, GA, Dec. 2010, pp. 514–521.

[12] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Trans. Robot*., vol. 28, no. 2, pp. 410–426, 2012, DOI: 10.1109/TRO.2011.2174493.

[13] D. Peleg, *Distributed Computing. A Locality-Sensitive Approac* (ser. Monographs on Discrete Mathematics and Applications). Philadelphia, PA: SIAM, 2000.

[14] A. Davoodi, P. Fazli, P. Pasquier, and A. K. Mackworth, "On multi-robot area coverage," in *Proc. Japan Conf. Comput. Geometry Graphs*, Kanazawa, Japan, Nov. 2009, pp. 75–76.

[15] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Network* (ser. Applied Mathematics Series). Princeton, NJ: Princeton Univ. Press, 2009.

[16] B. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for multi-robot and distributed sensor systems," in *Proc. Int. Conf. Adv. Robot*., Coimbra, Portugal, Jun. 2003, pp. 317–323.

[17] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artif. Intell*., vol. 128, nos. 1–2, pp. 99–141, 2001.

[18] J. W. Durham and F. Bullo, "Smooth nearness-diagram navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., Nice, France, Sep. 2008, pp. 690–695.

[19] O. L. Mangasarian, *Nonlinear Programming*. Philadelphia, PA: SIAM, 1994.

Bags of Binary Words for Fast Place Recognition in Image Sequences

Dorian Gálvez-López and Juan D. Tardós

***Abstract*—We propose a novel method for visual place recognition using bag of words obtained from accelerated segment test (FAST)+BRIEF features. For the first time, we build a vocabulary tree that discretizes a binary descriptor space and use the tree to speed up correspondences for geometrical verification. We present competitive results with no false positives in very different datasets, using exactly the same vocabulary and settings. The whole technique, including feature extraction, requires 22 ms/frame in a sequence with 26 300 images that is one order of magnitude faster than previous approaches.**

***Index Terms*—Bag of binary words, computer vision, place recognition, simultaneous localization and mapping (SLAM).**

Manuscript received September 23, 2011; revised February 13, 2012; accepted April 18, 2012. Date of publication May 18, 2012; date of current version September 28, 2012. This paper was recommended for publication by Associate Editor C. Stachniss and Editor D. Fox upon evaluation of the reviewers' comments. This work was supported by the European Union under Project RoboEarth FP7-ICT-248942, the Dirección General de Investigación of Spain under Project DPI2009-13710 and Project DPI2009-07130, and the Ministerio de Educación Scholarship FPU-AP2008-02272.

The authors are with the Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, 50018 Zaragoza, Spain (e-mail: dorian@unizar.es; tardos@unizar.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2012.2197158

I. Introduction

One of the most significant requirements for long-term visual *simultaneous localization and mapping* (SLAM) is robust place recognition. After an exploratory period, when areas nonobserved for long are reobserved, standard matching algorithms fail. When they are robustly detected, loop closures provide correct data association to obtain consistent maps. The same methods used for loop detection can be used for robot relocation after track lost, due, for example, to sudden motions, severe occlusions, or motion blur. In [1], we concluded that, for small environments, map-to-image methods achieve nice performance, but for large environments, image-to-image (or appearance-based) methods, such as fast appearance-based mapping (FAB-MAP) [2] scale better. The basic technique consists in building a database from the images collected online by the robot so that the most similar one can be retrieved when a new image is acquired. If they are similar enough, a loop closure is detected.

In recent years, many algorithms that exploit this idea have appeared [2]–[6], basing the image matching on comparing them as numerical vectors in the bag-of-words space [7]. Bags of words result in very effective and quick image matchers [8], but they are not a perfect solution for closing loops, due mainly to perceptual aliasing [6]. For this reason, a verification step is performed later by checking the matching images to be geometrically consistent, requiring feature correspondences. The bottleneck of the loop closure algorithms is usually the extraction of features, which is around ten times more expensive in computation cycles than the rest of the steps. This may cause SLAM algorithms to run in two decoupled threads: one to perform the main SLAM functionality, and the other just to detect loop closures, as in [5].

In this paper, we present a novel algorithm to detect loops and establish point correspondences between images in real time, with a conventional central processing unit and a single camera. Our approach is based on bag of words and geometrical check, with several important novelties that make it much faster than current approaches. The main speed improvement comes from the use of a slightly modified version of the BRIEF descriptor [9] with features from accelerated segment test (FAST) keypoints [10], as explained in Section III. The BRIEF descriptor is a binary vector where each bit is the result of an intensity comparison between a given pair of pixels around the keypoint. Although BRIEF descriptors are hardly invariant to scale and rotation, our experiments show that they are very robust for loop closing with planar camera motions, which is the usual case in mobile robotics, offering a good compromise between distinctiveness and computation time.

We introduce a bag of words that discretizes a binary space, and augment it with a direct index, in addition to the usual inverse index, as explained in Section IV. To the best of our knowledge, this is the first time a binary vocabulary is used for loop detection. The inverse index is used for fast retrieval of images potentially similar to a given one. We show a novel use of the direct index to efficiently obtain point correspondences between images, speeding up the geometrical check during the loop verification.

The complete loop-detection algorithm is detailed in Section V. Similar to our previous work [5], [6], to decide that a loop has been closed, we verify the temporal consistency of the image matches obtained. One of the novelties in this paper is a technique to prevent images collected in the same place from competing among them when the database is queried. We achieve this by grouping together those images that depict the same place during the matching.

Section VI contains the experimental evaluation of our study, including a detailed analysis of the relative merits of the different parts in our algorithm. We present comparisons between the effectiveness of BRIEF and two versions of speeded up robust features (SURFs) [11], the descriptors most used for loop closing. We also analyze the performance of the temporal and geometrical consistency tests for loop verification. We, finally, present the results achieved by our technique after evaluating it in five public datasets with 0.7–4-km-long trajectories. We demonstrate that we can run the whole loop-detection procedure, including the feature extraction, in 52 ms with 26 300 images (22 ms on average), outperforming previous techniques by more than one order of magnitude.

A preliminary version of this study was presented in [12]. In this paper, we enhance the direct index technique and extend the experimental evaluation of our approach. We also report results in new datasets and make a comparison with the state-of-the-art FAB-MAP 2.0 algorithm [13].

II. Related Work

Place recognition based on appearance has obtained great attention in the robotics community because of the excellent results achieved [4], [5], [13], [14]. An example of this is the FAB-MAP system [13], which detects loops with an omnidirectional camera, obtaining a recall of 48.4% and 3.1%, with no false positives, in trajectories 70 and 1000 km in length. FAB-MAP represents images with a bag of words, and uses a Chow–Liu tree to learn offline the words' covisibility probability. FAB-MAP has become the gold standard regarding loop detection, but its robustness decreases when the images depict very similar structures for a long time, which can be the case when using frontal cameras [5]. In the work of Angeli *et al*. [4], two visual vocabularies (for appearance and color) are created online in an incremental fashion. The two bag-of-words representations are used together as input of a Bayesian filter that estimates the matching probability between two images, taking into account the matching probability of previous cases. In contrast with these probabilistic approaches, we rely on a temporal consistency check to consider previous matches and enhance the reliability of the detections. This technique has proven successful in our previous works [5], [6]. Our work also differs from the earlier ones in that we use a bag of binary words for the first time, as well as propose a technique to prevent images collected close in time and depicting the same place from competing between them during the matching so that we can work at a higher frequency.

To verify loop-closing candidates, a geometrical check is usually performed. We apply an epipolar constraint to the best matching candidate as done in [4], but we take advantage of a direct index to calculate correspondence points faster. Konolige *et al*. [3] use visual odometry with a stereo camera to create in real time a view map of the environment, detecting loop closures with a bag-of-words approach as well. Their geometrical check consists in computing a spatial transformation between the matching images. However, they do not consider consistency with previous matches, and this leads them to apply the geometrical check to several loop-closing candidates.

In most loop-closing works [4]–[6], [14], the features used are scale-invariant feature transform (SIFT) [15] or SURF [11]. They are popular because they are invariant to lighting, scale and rotation changes, and show a good behavior in view of slight perspective changes. However, these features usually require between 100 and 700 ms to be computed, as reported by the aforementioned publications. Apart from graphics processing unit implementations [16], there are other similar features that try to reduce this computation time by, for example, approximating the SIFT descriptor [17] or reducing the dimensionality [18]. The work of Konolige *et al*. [3] offers a qualitative change, since it uses compact randomized tree signatures [19]. This approach calculates the similarity between an image patch and other patches previously trained

in an offline stage. The descriptor vector of the patch is computed by concatenating these similarity values, and its dimensionality is finally reduced with random orthoprojections. This yields a very fast descriptor that is suitable for real-time applications [19]. Our study bears a resemblance with [3] in that we also reduce the execution time by using efficient features. BRIEF descriptors, along with other recent descriptors such as the binary robust invariant scalable keypoints (BRISK) [20] or the oriented FAST with uncorrelated BRIEF features (ORB) [21], are binary and require very little time to be computed. As an advantage, their information is very compact so that they occupy less memory and are faster to compare. This allows a much faster conversion into the bag-of-words space.

III. Binary Features

Extracting local features (keypoints and their descriptor vectors) is usually very expensive in terms of computation time when comparing images. This is often the bottleneck when these kinds of techniques are applied to real-time scenarios. To overcome this problem, in this paper, we use FAST keypoints [10] and the state-of-the-art BRIEF descriptors [9]. FAST keypoints are corner-like points detected by comparing the gray intensity of some pixels in a Bresenham circle of radius 3. Since only a few pixels are checked, these points are very fast to obtain, proving successful for real-time applications.

For each FAST keypoint, we draw a square patch around them and compute a BRIEF descriptor. The BRIEF descriptor of an image patch is a binary vector where each bit is the result of an intensity comparison between two of the pixels of the patch. The patches are previously smoothed with a Gaussian kernel to reduce noise. Given before the size of the patch S_b, the pairs of pixels to test are randomly selected in an offline stage. In addition to S_b, we must set the parameter L_b: The number of tests to perform (i.e., the length of the descriptor). For a point $\mathbf{p}$ in an image, its BRIEF descriptor vector $\mathbf{B}(\mathbf{p})$ is given by

$$B^i(\mathbf{p}) = \begin{cases} 1, & \text{if } I(\mathbf{p}+\mathbf{a}_i) < I(\mathbf{p}+\mathbf{b}_i) \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in [1,\, L_b] \qquad (1)$$

where $B^i(\mathbf{p})$ is the ith bit of the descriptor, $I(\cdot)$ is the intensity of the pixel in the smoothed image, and $\mathbf{a}_i$ and $\mathbf{b}_i$ are the 2-D offset of the ith test points with respect to the center of the patch, with value in $[-\frac{S_b}{2} \ldots \frac{S_b}{2}] \times [-\frac{S_b}{2} \ldots \frac{S_b}{2}]$, randomly selected in advance. Note that this descriptor does not need training; it just needs an offline stage to select random points that hardly takes time. The original BRIEF descriptor proposed by Calonder *et al*. [9] selects each coordinate of the test points $\mathbf{a}_i$ and $\mathbf{b}_i$ according to a normal distribution $\mathcal{N}(0, \frac{1}{25}S_b^2)$. However, we found that using close test pairs yielded better results [12]. We select each coordinate j of these pairs by sampling the distributions $a_i^j \sim \mathcal{N}(0, \frac{1}{25}S_b^2)$ and $b_i^j \sim \mathcal{N}(a_i^j, \frac{4}{625}S_b^2)$. Note that this approach was also proposed by Calonder *et al*. [9] but not used in their final experiments. For the descriptor and patch sizes, we chose $L_b = 256$ and $S_b = 48$, because they resulted in a good compromise between distinctiveness and computation time [12].

The main advantage of BRIEF descriptors is that they are very fast to compute (Calonder *et al*. [9] reported 17.3 μs per keypoint when $L_b = 256$ bits) and to compare. Since one of these descriptors is just a vector of bits, measuring the distance between two vectors can be done by counting the amount of different bits between them (Hamming distance), which is implemented with an *xor* operation. This is more suitable in this case than calculating the Euclidean distance, as is usually done with SIFT or SURF descriptors, which are composed of floating point values.

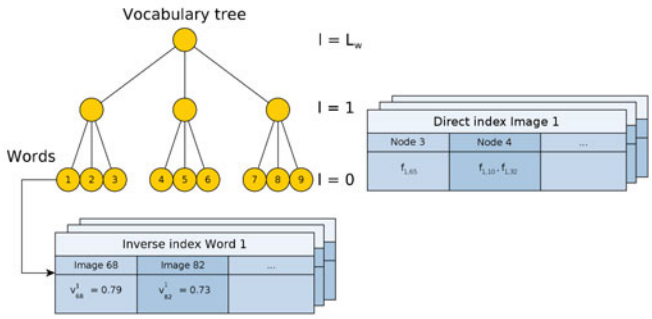

Fig. 1. Example of vocabulary tree and direct and inverse indexes that compose the image database. The vocabulary words are the leaf nodes of the tree. The inverse index stores the weight of the words in the images in which they appear. The direct index stores the features of the images and their associated nodes at a certain level of the vocabulary tree.

IV. Image Database

In order to detect revisited places, we use an image database composed of a hierarchical bag of words [7], [8] and direct and inverse indexes, as shown in Fig. 1.

The bag of words is a technique that uses a visual vocabulary to convert an image into a sparse numerical vector, allowing us to manage big sets of images. The visual vocabulary is created offline by discretizing the descriptor space into W visual words. Unlike with other features like SIFT or SURF, we discretize a binary descriptor space, creating a more compact vocabulary. In the case of the hierarchical bag of words, the vocabulary is structured as a tree. To build it, we extract a rich set of features from some training images, independently of those processed online later. The descriptors extracted are first discretized into k_w binary clusters by performing k-medians clustering with the k-means++ seeding [22]. The medians that result in a nonbinary value are truncated to 0. These clusters form the first level of nodes in the vocabulary tree. Subsequent levels are created by repeating this operation with the descriptors associated with each node, up to L_w times. We, finally, obtain a tree with W leaves, which are the words of the vocabulary. Each word is given a weight according to its relevance in the training corpus, decreasing the weight of those words which are very frequent and, thus, less discriminative. For this, we use the term frequency–inverse document frequency (*tf–idf*), as proposed by Sivic and Zisserman [7]. Then, to convert an image I_t, taken at time t, into a bag-of-words vector $\mathbf{v}_t \in \mathbb{R}^W$, the binary descriptors of its features traverse the tree from the root to the leaves, by selecting at each level the intermediate nodes that minimize the Hamming distance.

To measure the similarity between two bag-of-words vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, we calculate a L_1-score $s(\mathbf{v}_1, \mathbf{v}_2)$, whose value lies in $[0, 1]$:

$$s(\mathbf{v}_1, \mathbf{v}_2) = 1 - \frac{1}{2}\left|\frac{\mathbf{v}_1}{|\mathbf{v}_1|} - \frac{\mathbf{v}_2}{|\mathbf{v}_2|}\right|. \qquad (2)$$

Along with the bag of words, an inverse index is maintained. This structure stores for each word w_i in the vocabulary a list of images I_t where it is present. This is very useful when querying the database, since it allows us to perform comparisons only against those images that have some word in common with the query image. We augment the inverse index to store pairs $< I_t, v_t^i >$ to quickly access the weight of the word in the image. The inverse index is updated when a new image I_t is added to the database and accessed when the database is searched for some image.

These two structures (the bag of words and the inverse index) are often the only ones used in the bag-of-words approach for searching images. However, as a novelty in this general approach, we also make use of a direct index to conveniently store the features of each image. We separate the nodes of the vocabulary according to their level l in the tree, starting at leaves, with level $l = 0$, and finishing in the root, $l = L_w$. For each image I_t, we store, in the direct index, the nodes at level l that are ancestors of the words present in I_t, as well as the list of local features f_{tj} associated with each node. We take advantage of the direct index and the bag-of-words tree to use them as a means to approximate nearest neighbors in the BRIEF descriptor space. The direct index allows us to speed up the geometrical verification by computing correspondences only between those features that belong to the same words or to words with common ancestors at level l. The direct index is updated when a new image is added to the database, and accessed when a candidate matching is obtained and geometrical check is necessary.

V. Loop-Detection Algorithm

To detect loop closures, we use a method based on our previous work [5], [6] that follows the four stages detailed next.

A. Database Query

We use the image database to store and retrieve images similar to any given one. When the last image I_t is acquired, it is converted into the bag-of-words vector $\mathbf{v}_t$. The database is searched for $\mathbf{v}_t$, resulting in a list of matching candidates $\langle \mathbf{v}_t, \mathbf{v}_{t_1} \rangle$, $\langle \mathbf{v}_t, \mathbf{v}_{t_2} \rangle$, . . ., associated with their scores $s(\mathbf{v}_t, \mathbf{v}_{t_j})$. The range these scores varies is very dependent on the query image and the distribution of words it contains. We then normalize these scores with the best score we expect to obtain in this sequence for the vector $\mathbf{v}_t$, obtaining the normalized similarity score η [6]

$$\eta(\mathbf{v}_t, \mathbf{v}_{t_j}) = \frac{s(\mathbf{v}_t, \mathbf{v}_{t_j})}{s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})}. \tag{3}$$

Here, we approximate the expected score of $\mathbf{v}_t$ with $s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$, where $\mathbf{v}_{t-\Delta t}$ is the bag-of-words vector of the previous image. Those cases where $s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$ is small (e.g. when the robot is turning) can erroneously cause high scores. Thus, we skip the images that do not reach a minimum $s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$ or a required number of features. This minimum score trades off the number of images that can be used to detect loops with the correctness of the resulting score η. We use a small value to prevent valid images from being discarded. We then reject those matches whose $\eta(\mathbf{v}_t, \mathbf{v}_{t_j})$ does not achieve a minimum threshold, which is denoted α.

B. Match Grouping

To prevent images that are close in time to compete among them when the database is queried, we group them into *islands* and treat them as only one match. We use the notation T_i to represent the interval composed of timestamps $t_{n_i}, \ldots, t_{m_i}$, and V_{T_i} for an island that groups together the matches with entries $\mathbf{v}_{t_{n_i}}, \ldots, \mathbf{v}_{t_{m_i}}$. Therefore, several matches $\langle \mathbf{v}_t, \mathbf{v}_{t_{n_i}} \rangle, \ldots, \langle \mathbf{v}_t, \mathbf{v}_{t_{m_i}} \rangle$ are converted into a single match $\langle \mathbf{v}_t, V_{T_i} \rangle$ if the gaps between consecutive timestamps in $t_{n_i}, \ldots, t_{m_i}$ are small. The islands are also ranked according to a score H

$$H(\mathbf{v}_t, V_{T_i}) = \sum_{j=n_i}^{m_i} \eta(\mathbf{v}_t, \mathbf{v}_{t_j}). \tag{4}$$

The island with the highest score is selected as matching group and continues to the temporal consistency step. Besides avoiding clashes between consecutive images, the islands can help establish correct matches. If I_t and $I_{t'}$ represent a real loop closure, I_t is very likely to be similar to $I_{t'\pm\Delta t}$, $I_{t'\pm 2\Delta t}, \ldots$, producing long islands. Since we define H as the sum of scores η, the H score favors matches with long islands as well.

C. Temporal Consistency

After obtaining the best matching island $V_{T'}$, we check it for temporal consistency with previous queries. In this paper, we extend the temporal constraint applied in [5] and [6] to support islands. The match $\langle \mathbf{v}_t, V_{T'} \rangle$ must be consistent with k previous matches $\langle \mathbf{v}_{t-\Delta t}, V_{T_1} \rangle, \ldots, \langle \mathbf{v}_{t-k\Delta t}, V_{T_k} \rangle$, such that the intervals T_j and T_{j+1} are close to overlap. If an island passes the temporal constraint, we keep only the match $\langle \mathbf{v}_t, \mathbf{v}_{t'} \rangle$, for the $t' \in T'$ that maximizes the score η, and consider it a loop-closing candidate, which finally has to be accepted by the geometrical verification stage.

D. Efficient Geometrical Consistency

We apply a geometrical check between any pair of images of a loop-closing candidate. This check consists in finding with random sample consensus (RANSAC) a fundamental matrix between I_t and $I_{t'}$ supported by at least 12 correspondences. To compute these correspondences, we must compare the local features of the query image with those of the matched one. There are several approaches to perform this comparison. The easiest and slowest one is the exhaustive search, which consists in measuring the distance of each feature of I_t to the features of $I_{t'}$ in the descriptor space, to select correspondences later according to the *nearest neighbor distance ratio* [15] policy. This is a $\Theta(n^2)$ operation in the number of features per image. A second technique consists in calculating approximate nearest neighbors by arranging the descriptor vectors in k-dimensional (k-d) trees [27].

Following the latter idea, we take advantage of our bag-of-words vocabulary and reuse it to approximate nearest neighbors. For this reason, when adding an image to the database, we store a list of pairs of nodes and features in the direct index. To obtain correspondences between I_t and $I_{t'}$, we look up $I_{t'}$ in the direct index and perform the comparison only between those features that are associated with the same nodes at level l in the vocabulary tree. This condition speeds up the correspondence computation. The parameter l is fixed beforehand and entails a tradeoff between the number of correspondences obtained between I_t and $I_{t'}$ and the time consumed for this purpose. When $l = 0$, only features belonging to the same word are compared (as we presented in [12]) so that the highest speedup is achieved, but fewer correspondences can be obtained. This makes the recall of the complete loop-detection process decrease due to some correct loops being rejected because of the lack of correspondence points. On the other hand, when $l = L_w$, the recall is not affected but the execution time is not improved either.

We only require the fundamental matrix for verification, but note that after calculating it, we could provide the data association between the images matched to any SLAM algorithm that would run beneath, with no extra cost.

VI. Experimental Evaluation

We evaluate the different aspects of our proposal in the following sections. In Section VI-A, we introduce the methodology we followed to evaluate our algorithm. Next, we compare the reliability of BRIEF and SURF in our system in Section VI-B. In Section VI-C, we analyze the effect of the temporal consistency of our algorithm, and the efficiency of our geometrical verification based on the direct

TABLE I
DATASETS

Dataset	Camera	Description	Total length (m)	Revisited length (m)	Avg. Speed ($m \cdot s^{-1}$)	Image size (px × px)
New College [23]	Frontal	Outdoors, dynamic	2260	1570	1.5	512×384
Bicocca 2009-02-25b [24]	Frontal	Indoors, static	760	113	0.5	640×480
Ford Campus 2 [25]	Frontal	Urban, slightly dynamic	4004	280	6.9	600×1600
Malaga 2009 Parking 6L [26]	Frontal	Outdoors, slightly dynamic	1192	162	2.8	1024×768
City Centre [2]	Lateral	Urban, dynamic	2025	801	-	640×480

index is checked in Section VI-D. Finally, the execution time and the performance of our complete system are evaluated in Sections VI-E and F.

A. Methodology

The aspects to evaluate loop-detection results are usually assumed to be of general knowledge. However, little detail is given in the literature. Here, we explain the methodology we followed to evaluate our system.

1) Datasets: We tested our system in five publicly available datasets (see Table I). These present independent indoor and outdoor environments, and were collected at different speed by several platforms, with in-plane camera motion. CityCentre is a collection of images gathered at low frequency, with little overlap. The others provide images at high frequency (8–20 Hz).

2) Ground Truth: To measure the correctness of our results, we compare them with a ground-truth reference. Most of the datasets used here do not provide direct information about loop closures; therefore, we manually created a list of the actual loop closures. This list is composed of time intervals, where each entry in the list encodes a query interval associated with a matching interval.

3) Correctness Measure: We measure the correctness of the loop-detection results with the *precision* and *recall* metrics. The precision is defined as the ratio between the number of correct detections and all the detections fired and the recall as the ratio between the correct detections and all the loop events in the ground truth. A match fired by the loop detector is a pair of query and matching timestamps. To check if it is a true positive, the ground truth is searched for an interval that contains these timestamps. The number of loop events in the ground truth is computed as the length of all the query intervals in the ground truth multiplied by the frequency at which the images of the dataset are processed. When a query timestamp is associated with more than one matching timestamps in the ground truth because of multiple traversals, only one of them is considered to compute the amount of loop events.

4) Selection of System Parameters: It is a common practice to tune system parameters according to the evaluation data, but we think that using different data to choose the configuration of our algorithm and to evaluate it demonstrates the robustness of our approach. We then separate the datasets shown in Table I into two groups. We use three of them that present heterogeneous environments with many difficulties (NewCollege, Bicocca25b, and Ford2) as *training* datasets to find the best set of parameters of our algorithm. The other two datasets (CityCentre and Malaga6L) are used as *evaluation* data to validate our final configuration. In these cases, we only use our algorithm as a black box with a predefined configuration.

5) Settings: Our algorithm is used with the same settings throughout all the experiments. The same vocabulary tree was used to process all the datasets. This was built with $k_w = 10$ branches and $L_w = 6$ depth levels, yielding one million words, and trained with nine million features acquired from 10 000 images of an independent dataset (*Bovisa 2008-09-01* [24]). We used a threshold of 10 units in the response function of FAST, and 500 in the Hessian response of SURF. For each processed image, we kept only the 300 features with highest response.

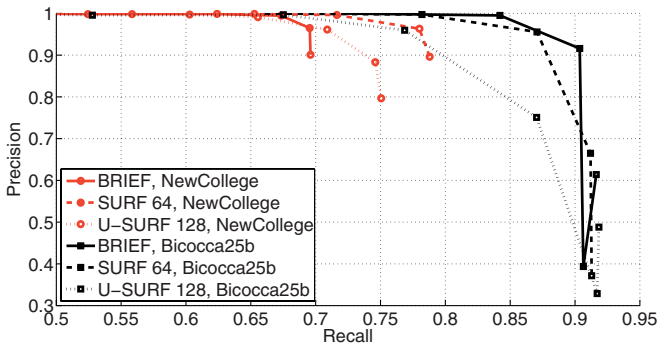

Fig. 2. Precision–recall curves achieved by BRIEF, SURF64, and U-SURF128 in the training datasets, without geometrical check.

B. Descriptor Effectiveness

A BRIEF descriptor encodes much less information than a SURF descriptor, since BRIEF is not scale or rotation invariant. In order to check if BRIEF is reliable enough to perform loop detection, we compared its effectiveness with that of SURF. We selected two versions of SURF features: 64-D descriptors with rotation invariance (SURF64) and 128-D descriptor without rotation invariance (U-SURF128). We selected these features because they are the usual choices to solve the loop-detection problem [5], [13].

We created vocabulary trees for SURF64 and U-SURF128 in the same way we built it for BRIEF and ran our system on Bicocca25b and NewCollege, processing the image sequences at $f = 2$ Hz. We deactivated the geometrical verification, fixed the required temporal consistency matches k to 3, and varied the value of the normalized similarity threshold α to obtain the precision–recall curves shown in Fig. 2. The first remark is that the curve of SURF64 dominates that of U-SURF128 on both datasets. We can also see that BRIEF offers a very competent performance compared with SURF. In Bicocca25b, BRIEF outperforms U-SURF128 and is slightly better than SURF64. In NewCollege, SURF64 achieves better results than BRIEF, but BRIEF still gives very good precision and recall rates.

To better illustrate the different abilities of BRIEF and SURF64 to find correspondences, we have selected some loop events from the previous experiments. In Fig. 3, the features that are associated with the same word of our vocabulary are connected with lines. These are the only matches taken into account to compute the normalized similarity score. In most cases, BRIEF obtains as many correct word correspondences as SURF64, in spite of the slight perspective changes, as shown in the first example (first row). In the second example, only BRIEF is able to close the loop, since SURF64 does not obtain enough word correspondences. These two examples show that BRIEF finds

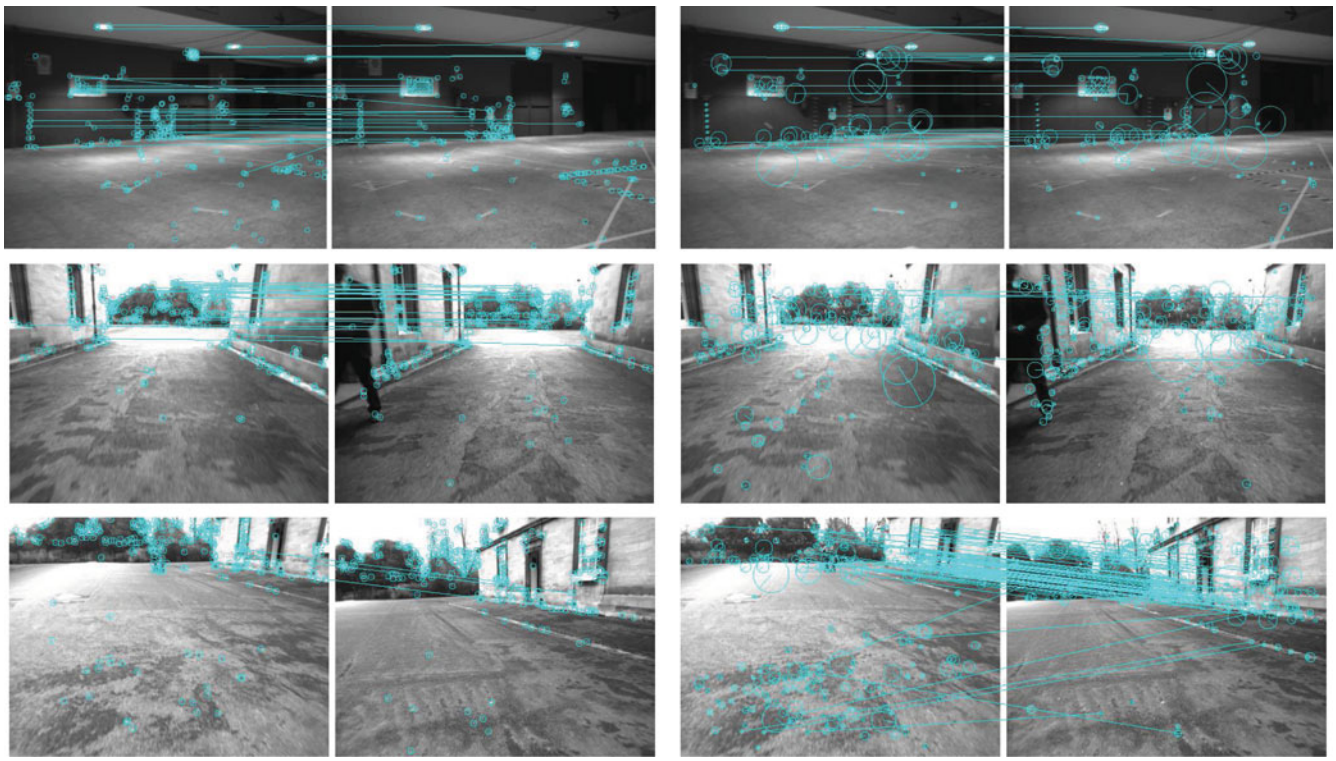

Fig. 3. Examples of words matched by using (pair on the left) BRIEF and (pair on the right) SURF64 descriptors.

correspondences in objects that are at a middle or large distance, such as the signs on the wall or the trees in the background. In general, distant objects are present in most of the imagery of our datasets. Since the scale of the keypoints extracted from distant objects hardly varies, BRIEF is suitable to match their patches. In cases where objects are close to the camera, SURF64 is more suitable because of its invariance to scale changes. However, we observed very few cases where this happened. In the third example in Fig. 3, the camera tilted, making the image appear rotated in some areas. This along with the scale change prevented BRIEF from obtaining word correspondences. In this case, SURF64 overcame these difficulties and detected the loop.

Our results show that FAST features with BRIEF descriptors are almost as reliable as SURF features for loop-detection problems with in-plane camera motion. As advantages, not only they are much faster to obtain (13 ms/image instead of 100–400 ms), but they also occupy less memory (32 MB instead of 256 MB to store a one million word vocabulary) and are faster to compare, speeding up the use of the hierarchical vocabulary.

C. Temporal Consistency

After selecting the features, we tested the number k of temporally consistent matches required to accept a loop closure candidate. For this, we ran our system in the training datasets with $f = 2$ Hz, for several values of k and α and with no geometrical constraint. We tested k for values between 0 (i.e., disabling the temporal consistency) and 4. We observed a big improvement between $k = 0$ and $k > 0$ for all the working frequencies. As k increases, a higher recall is attained with 100% precision, but this behavior does not hold for very high values of k, since only very long closures would be found. We chose $k = 3$ since it showed a good precision–recall balance in the three training datasets. We repeated this test in Bicocca25b for frequencies $f = 1$ and 3 Hz, as well to check how dependent parameter k is on the processing frequency. We show, in Fig. 4, the precision–recall curves obtained in Bicocca25b by varying the parameter α; for clarity, only $k = 0$ and 3 are shown. This shows that the temporal consistency is a valuable mechanism to avoid mismatches, as previously seen in [12]. We can also see that $k = 3$ behaves well even for different frequency values so that we can consider this parameter stable.

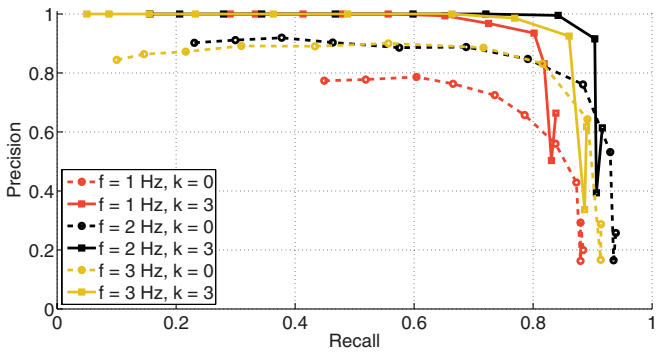

Fig. 4. Precision–recall curve in Bicocca25b with no geometrical check, for several values of similarity threshold α, number of temporally consistent matches k, and processing frequency f.

D. Geometrical Consistency

According to Fig. 4, we could select a restrictive value of α to obtain 100% precision, but this would require to tune this parameter for each dataset. Instead, we set a generic value and verify matches with a geometrical constraint consisting in finding a fundamental matrix between two images I_t and $I_{t'}$ of a loop-closing candidate. Computing the corresponding points between I_t and $I_{t'}$ is the most time-consuming step of this stage. We compared our proposal of using the direct index to

TABLE II
PERFORMANCE OF DIFFERENT APPROACHES TO OBTAIN CORRESPONDENCES IN NEWCOLLEGE

Technique	Recall (%)	Execution time (ms / query)		
		Median	Min	Max
DI_0	38.3	0.43	0.25	16.50
DI_1	48.5	0.70	0.44	17.14
DI_2	56.1	0.78	0.50	19.26
DI_3	57.0	0.80	0.48	19.34
Flann	53.6	14.09	13.79	25.07
Exhaustive	61.2	14.17	13.65	24.68

compute correspondences, coined DI_l, with the exhaustive search and a Flann-based approach [27]. The parameter l stands for the level in the vocabulary tree at which the ancestor nodes are checked. In the Flann approach, the Flann library [27] (as implemented in the OpenCV library) is used to build a set of k-d trees with the feature descriptors of I_t. This allows us to obtain, for descriptors of $I_{t'}$, the approximate nearest neighbors in I_t. After computing distances with any of these methods, the nearest neighbor distance ratio, with a threshold of 0.6 units, was applied. Although both the Flann and the vocabulary tree approaches are useful to approximate nearest neighbors, they are conceptually different here: Our vocabulary tree was created with training data, so that the neighbor search is based on independent data, whereas the k-d trees are tailored to each $I_{t'}$.

We ran each of the methods in the NewCollege dataset with $f = 2$ Hz, $k = 3$, and $\alpha = 0.3$. We selected this dataset because it presents the longest revisited trajectory and many perceptual aliasing cases. In Table II, we show the execution time of the geometrical check per query, along with the recall of the loop detector in each case. The precision was 100% in all the cases. The time includes the computation of correspondence points, the RANSAC loops, and the computation of the fundamental matrices. The highest execution time of all the methods was obtained when the maximum number of RANSAC iterations was reached. The exhaustive search achieves higher recall than the other methods, which are approximate, but exhibits the highest execution time as well. We see that the Flann method takes nearly as long as the exhaustive search method. The speedup obtained when computing the correspondences is not worth the cost of building a Flann structure per image. On the other hand, DI_0 presents not only the smallest execution time but the lowest recall level as well. As we noticed previously [12], selecting correspondences only from features belonging to the same word is very restrictive when the vocabulary is big (one million words). We finally chose the method DI_2 for our geometrical check since it showed a good balance between recall and execution time.

E. Execution Time

To measure the execution time, we ran our system in the NewCollege dataset with $k = 3$, $\alpha = 0.3$, and DI_2. By setting the working frequency to $f = 2$ Hz, a total of 5266 images were processed, yielding a system execution time of 16 ms/image on average and a peak of less than 38 ms. However, in order to test the scalability of the system, we set the frequency to $f = 10$ Hz and obtained 26 292 images. Even with $k = 3$, the system yielded no false positives. This shows that the behavior of the temporal consistency parameter k is stable even for high frequencies.

The execution time consumed per image in that case is shown in Fig. 5. This was measured on a Intel Core i7 @2.67 GHz machine. We also show, in Table III, the required time of each stage for this number of images. The *features*' time involves computing FAST keypoints and removing those with lower corner response when there are too many, as well as smoothing the image with a Gaussian kernel and computing

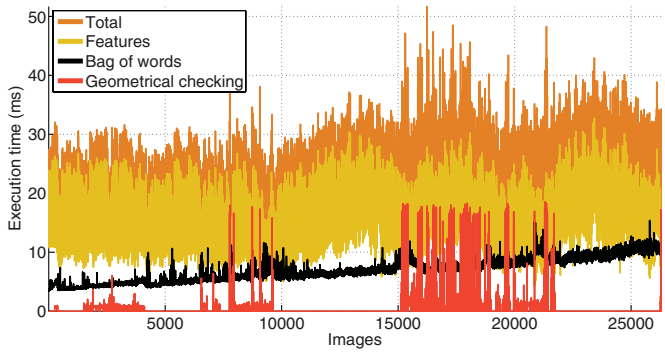

Fig. 5. Execution time in NewCollege with 26 292 images.

TABLE III
EXECUTION TIME IN NEWCOLLEGE WITH 26 292 IMAGES

		Execution time (ms / query)			
		Mean	Std	Min	Max
Features	FAST	11.67	4.15	1.74	30.16
	Smoothing	0.96	0.37	0.79	2.51
	BRIEF	1.72	0.49	1.51	4.62
Bag of words	Conversion	3.59	0.35	3.27	8.81
	Query	3.08	1.91	0.01	9.19
	Islands	0.12	0.04	0.08	0.97
	Insertion	0.11	0.02	0.06	0.28
Verification	Correspondences and RANSAC	1.60	2.64	0.61	18.55
Whole system		21.60	4.82	8.22	51.68

BRIEF descriptors. The *bag-of-words* time is split into four steps: The conversion of image features into a bag-of-words vector, the database query to retrieve similar images, the creation and matching of islands, and the insertion of the current image into the database (this also involves updating the direct and inverse indexes). The *verification* time includes both computing correspondences between the matching images, by means of the direct index, and the RANSAC loop to calculate fundamental matrices.

We see that all the steps are very fast, including extracting the features and the maintenance of the direct and inverse indexes. This allows us to obtain a system that runs in 22 ms/image, with a peak of less than 52 ms. The feature extraction stage presents the highest execution time, most of it due to the overhead produced when there are too many features and only the best 300 ones must be considered. Even so, we have achieved a reduction of more than one order of magnitude with respect to other features, such as SIFT or SURF, removing the bottleneck of these loop closure detection algorithms. In the bag-of-words stage, the required time of managing the islands and the indexes is negligible, and the conversion of image features into bag-of-words vectors takes as long as the database query. Its execution time depends on the number of features and the size of the vocabulary. We could reduce it by using a smaller vocabulary, since we are using a relatively big one (one million words instead of 10 000–40 000 [5], [14]). However, we found that a big vocabulary produces more sparse inverse indexes associated with words. Therefore, when querying, fewer database entries must be traversed to obtain the results. This reduces the execution time strikingly when querying, trading off, by far, the time required when converting a new image. We conclude that big vocabularies can improve the computation time when using large image collections. Furthermore, note that querying a database with more than 26 000 images takes 9 ms only, suggesting this step scales well with tens of thousands images. The geometrical verification exhibits a long execution time in the worst

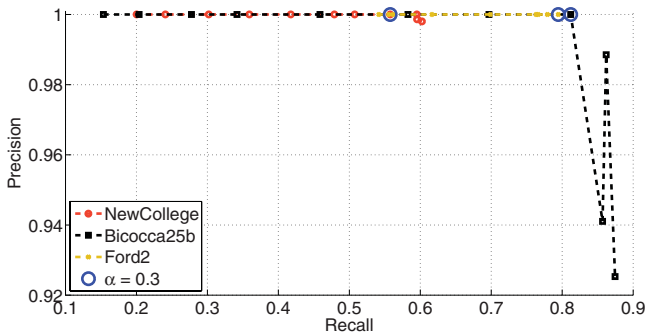

Fig. 6. Final precision–recall curves in the training datasets with $f = 2$ Hz, with the selected working point $\alpha = 0.3$.

TABLE IV
PARAMETERS

FAST threshold	10
BRIEF descriptor length (L_b)	256
BRIEF patch size (S_b)	48
Max. features per image	300
Vocabulary branch factor (k_w)	10
Vocabulary depth levels (L_w)	6
Min. score with previous image ($s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$)	0.005
Temporally consistent matches (k)	3
Normalized similarity score threshold (α)	0.3
Direct index level (l)	2
Min. matches after RANSAC	12

TABLE V
PRECISION AND RECALL OF OUR SYSTEM

Dataset	# Images	Precision (%)	Recall (%)
NewCollege	5266	100	55.92
Bicocca25b	4924	100	81.20
Ford2	1182	100	79.45
Malaga6L	869	100	74.75
CityCentre	2474	100	30.61

case, but as we saw in the previous section, this rarely occurs, whereas the 75% of the cases require less than 1.6 ms.

Our results show that we can reliably detect loops against databases with 26 000 images in 52 ms (22 ms on average). This represents an improvement of one order of magnitude with respect to the 300–700 ms required by algorithms based on SIFT or SURF [4]–[6], [13], [14]. For example, the state-of-the-art algorithm FAB-MAP 2.0 [13] needs 423 ms to extract SURF, 60 ms for conversion into bag of words, 10 ms to retrieve matching candidates against 25 000 images, and 120 ms (worst case) for RANSAC geometric verification. Our algorithm also outperforms the extremely efficient loop detector developed by Konolige *et al*. [3], based on compact randomized tree signatures. According to [3, Fig. 6], the method requires around 300 ms to perform the complete loop detection against a database with 4000 images.

F. Performance of the Final System

In previous sections, we showed the effect of the parameters of our system in the correctness of the results. For our algorithm, we chose the generic parameters $k = 3$, $\alpha = 0.3$, and the DI_2 method to compute correspondences, since they proved effective under several kinds of environments in the training datasets. A summary with the parameters of the algorithm and the vocabulary is shown in Table IV. In Fig. 6, we show the precision–recall curves obtained in these datasets with these parameters, processing the sequences at $f = 2$ Hz. In Table V, we show the figures of those curves with the final configuration. We achieved a high recall rate in the three datasets with no false positives.

TABLE VI
PRECISION AND RECALL OF FAB-MAP 2.0

Dataset	# Images	Min. p	Precision (%)	Recall (%)
Malaga6L	462	98%	100	68.52
CityCentre	2474	98%	100	38.77

In order to check the reliability of our algorithm with new datasets, we used Malaga6L and CityCentre as evaluation datasets. For these, we used our algorithm as a black box, with the default configuration given previously and the same vocabulary. For Malaga6L, we processed the sequence at $f = 2$ Hz, and for CityCentre, we used all the images, since these are already taken far apart. We also compared our algorithm with the state-of-the-art FAB-MAP 2.0 algorithm [13], configured by default as it is available in its authors' website.[1] Given a query image, FAB-MAP returns a vector with the probability p of being at the same place than some previous image. Only those matches with p higher than a threshold are accepted. This parameter must be set by the user. We chose $p \geq 98\%$ because it showed the highest recall for 100% precision in these datasets. Tables V and VI show the results in the evaluation datasets. For sake of fairness, we remark on how this comparison was performed: FAB-MAP 2.0 software does not apply any geometrical constraint to the returned matches by default; therefore, we applied a verification stage similar to ours, consisting of computing a fundamental matrix with the exhaustive search method. The input for FAB-MAP 2.0 must be a sequence of disjoint images. For Malaga6L, we fed it with images taken at frequency 1 Hz. We also tried 0.25 and 0.5 Hz, but 1 Hz yielded better results. For CityCentre, we used all the available images. Finally, FAB-MAP 2.0 provides a vocabulary of 11 000 words of 128 float values, built from outdoor disjoint images, whereas our vocabulary contains one million words of 256 bits, created from a sequence of images.

As shown in Table V, our algorithm with the parameters by default is able to achieve large recall with no false positives in both evaluation datasets. Our recall level is similar to that yielded by FAB-MAP 2.0, but with lower execution time. In the Malaga6L dataset, all the loops are correct in spite of the illumination difficulties and the depth of the views. The results in CityCentre differ between our method and FAB-MAP 2.0 because the change between loop closure images is bigger than that in other datasets. This hinders the labor of the DI_2 technique because features are usually more distinct and are separated in early levels in the vocabulary tree. Note that this highlights the little invariance of BRIEF, since others as SURF may be able to produce more similar features between the images. Anyhow, we see that our method is still able to find a large amount of loop events in this dataset. This test shows that our method can work fine out of the box in many environments and situations and that it is able to cope with sequences of images taken at low or high frequency, as long as they overlap. We can also remark that the same vocabulary sufficed to process all the datasets. This suggests that the source of the vocabulary is not so important when it is big enough.

We show, in Fig. 7, the detected loops in each dataset. No false detections were fired. The trajectory in NewCollege is based on partially corrected GPS data so that some paths are inaccurately depicted. Note that part of the vehicle where the camera is mounted is present in all the images of Ford2; we removed the features that lay on it. We see that detecting 55.92% of the loop events is enough to, for example, widely cover all the loop areas in a long trajectory as that of NewCollege.

[1] http://www.robots.ox.ac.uk/∼mobile

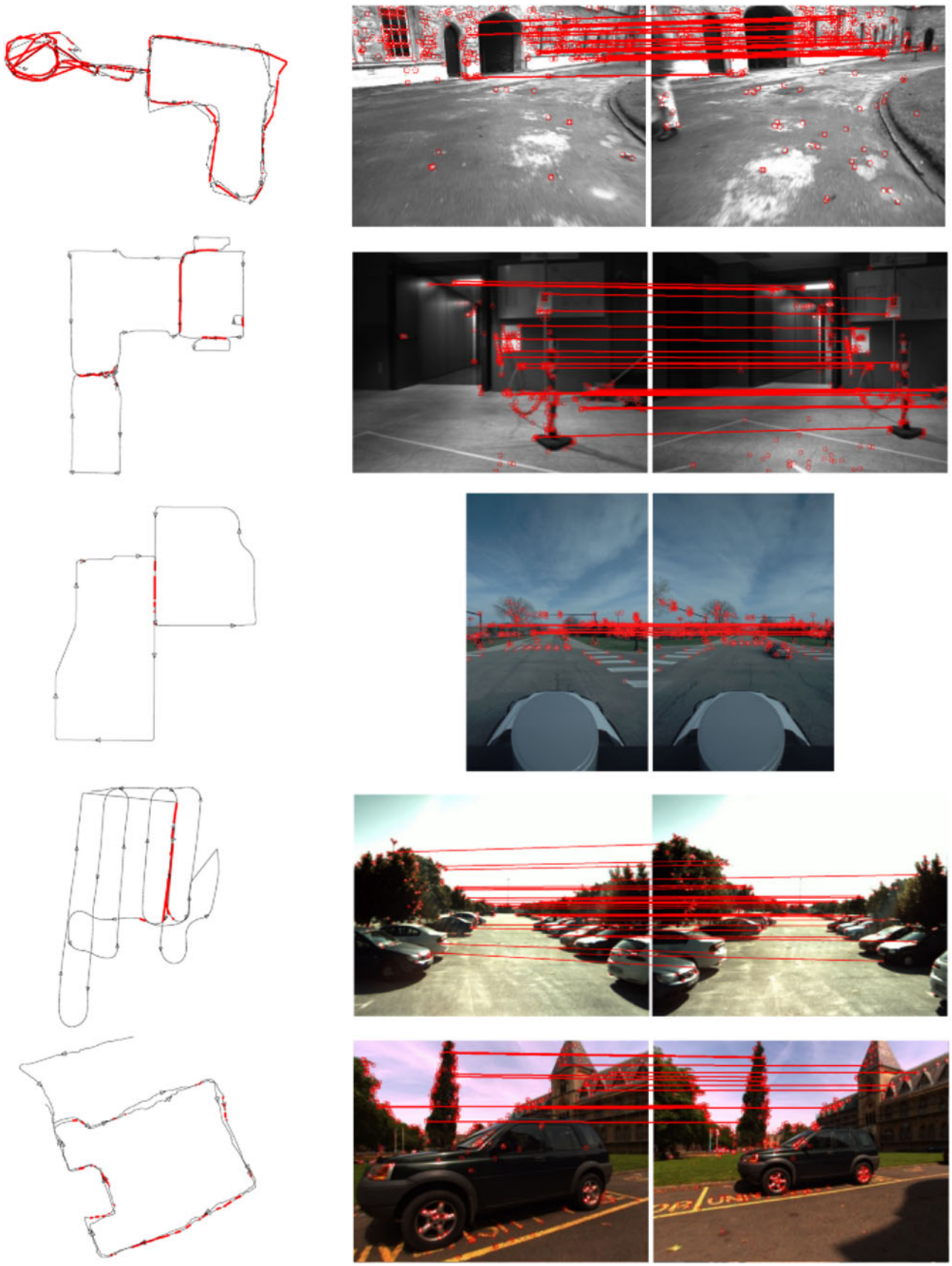

Fig. 7. Loops detected by our system in the five datasets (Top to bottom: NewCollege, Bicocca25b, Ford2, Malaga6L, and CityCentre), with some examples of correct loops detected in scenes with motion blur and slight scale and perspective change. On the right-hand side, lines depict final corresponding features. On the left-hand side, the trajectory of the robot is depicted with thin black lines in new places and with thick red lines in revisited areas. There are no false positives in any case.

On the right-hand side of Fig. 7, we show examples of correct loop detections in the training and evaluation datasets, with the final corresponding features. These examples make the limited scale invariance of BRIEF descriptors apparent. Most of the features matched are distant, as we noticed in Section VI-B. The scale change that BRIEF tolerates is shown in the correspondences that are close to the camera in NewCollege and Bicocca25b, as well as those on the cars in Malaga6L. However, BRIEF cannot handle such a large-scale change as that produced on the car in CityCentre, where all correspondences were obtained from distant features. On the other hand, whenever features are matched in background objects, a loop can be detected despite medium translations. This is visible in CityCentre and Ford2, where the vehicle moved along different lanes of the road.

VII. Conclusion

We have presented a novel technique to detect loops in monocular sequences. The main conclusion of our study is that binary features are very effective and extremely efficient in the bag-of-words approach. In particular, our results demonstrate that FAST+BRIEF features are as reliable as SURF (either with 64 dimensions or with 128 and without rotation invariance) to solve the loop-detection problem with in-plane camera motion, the usual case in mobile robots. The execution time and memory requirements are one order of magnitude smaller, requiring no special hardware.

The reliability and efficiency of our proposal have been shown on five very different public datasets depicting indoor, outdoor, static, and dynamic environments, with frontal or lateral cameras. Departing from most previous works, to avoid overtuning, we restricted ourselves to present all results using the same vocabulary obtained from an independent dataset and the same parameter configuration obtained from a set of training datasets without peeking on the evaluation datasets. Therefore, we can claim that our system offers robust and efficient performance in a wide range of real situations, with no additional tuning.

The main limitation of our technique is the use of features that lack rotation and scale invariance. It is enough for place recognition in indoor and urban robots but surely not for all-terrain or aerial vehicles, humanoid robots, wearable cameras, or object recognition. However, our demonstration of the effectiveness of the binary bag-of-words approach paves the road for the use of new and promising binary features such as ORB [21] or BRISK [20], which outperform the computation time of SIFT and SURF, maintaining rotation and scale invariance.

As a final contribution to the community, the implementation of our algorithm is publicly available online.[2]

References

[1] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robot. Auton. Syst.*, vol. 57, pp. 1188–1197, Dec. 2009.

[2] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, 2008.

[3] C. Cadena, D. Gálvez-López, J. D. Tardós, and J. Neira, "Robust place recognition with stereo sequences," *IEEE Trans. Robot.*, DOI 10.1109/TRO.2012.2189497, to be published.

[4] A. Angeli, D. Filliat, S. Doncieux, and J. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1027–1037, Oct. 2008.

[5] P. Piniés, L. M. Paz, D. Gálvez-López, and J. D. Tardós, "CI-graph SLAM for 3D reconstruction of large and complex environments using a multicamera system," *Int. J. Field Robot.*, vol. 27, no. 5, pp. 561–586, Sep./Oct. 2010.

[6] C. Cadena, D. Gálvez-López, J. D. Tardós, and J. Neira, "Robust place recognition with stereo sequences," *IEEE Trans. Robot.*, vol. 28, no. 4, 2012, to be published.

[7] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2003, vol. 2, pp. 1470–1477.

[8] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2006, vol. 2, pp. 2161–2168.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2010, vol. 6314, pp. 778–792.

[10] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis.*, May 2006, vol. 1, pp. 430–443.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[12] D. Gálvez-López and J. D. Tardós, "Real-time loop detection with bags of binary words," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 51–58.

[13] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *Int. J. Robot. Res.*, vol. 30, no. 9, pp. 1100–1123, Aug. 2011.

[14] R. Paul and P. Newman, "FAB-MAP 3D: Topological mapping with spatial and visual appearance," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2649–2656.

[15] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[16] S. Heymann, K. Maller, A. Smolic, B. Froehlich, and T. Wiegand, "SIFT implementation and optimization for general-purpose GPU," presented at the Int. Conf. Central Eur. Comput. Graph., Vis. Comput. Vis., Plzen, Czech Republic, Jan. 2007.

[17] M. Grabner, H. Grabner, and H. Bischof, "Fast approximated SIFT," in *Proc. Asian Conf. Comput. Vis.*, 2006, pp. 918–927.

[18] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2004, vol. 2, pp. 506–513.

[19] M. Calonder, V. Lepetit, P. Fua, K. Konolige, J. Bowman, and P. Mihelich, "Compact signatures for high-speed interest point description and matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2010, pp. 357–364.

[20] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2548–2555.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[22] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. ACM-SIAM Symp. Discr. Algorithms*, Jan. 2007, pp. 1027–1035.

[23] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *Int. J. Robot. Res.*, vol. 28, no. 5, pp. 595–599, May 2009.

[24] RAWSEEDS. (2007–2009). Robotics advancement through web-publishing of sensorial and elaborated extensive data sets (Project FP6-IST-045144). [Online]. Available: http://www.rawseeds.org/rs/datasets

[25] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *Int. J. Robot. Res.*, vol. 30, no. 13, pp. 1543–1552, Nov. 2011.

[26] J.-L. Blanco, F.-A. Moreno, and J. González, "A collection of outdoor robotic datasets with centimeter-accuracy ground truth," *Auton. Robots*, vol. 27, no. 4, pp. 327–351, Nov. 2009.

[27] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, Feb. 2009, pp. 331–340.

[2] http://webdiis.unizar.es/∼dorian