2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Fast Relocalisation and Loop Closing in Keyframe-Based SLAM

Raúl Mur-Artal and Juan D. Tardós

Abstract— **In this paper we present for the first time a relocalisation method for keyframe-based SLAM that can deal with severe viewpoint change, at frame-rate, in maps containing thousands of keyframes. As this method relies on local features, it permits the interoperability between cameras, allowing a camera to relocalise in a map built by a different camera. We also perform loop closing (detection + correction), at keyframe-rate, in loops containing hundreds of keyframes. For both relocalisation and loop closing, we propose a bag of words place recognizer with ORB features, which is able to recognize places spending less than 39 ms, including feature extraction, in databases containing 10K images (without geometrical verification). We evaluate the performance of this recognizer in four different datasets, achieving high recall and no false matches, and getting better results than the state-of-art in place recognition, being one order of magnitude faster.**

I. Introduction

Visual SLAM (Simultaneous Localisation and Mapping with a camera as the only sensor) has been strongly developed in the last decade. Nowadays the most extended techniques are based on keyframes. Keyframe-based SLAM performs map reconstruction over selected frames, called keyframes, using batch optimization techniques, as mapping is not tied to frame-rate, providing very precise results. The most representative keyframe-based system is probably Parallel Tracking and Mapping (PTAM) [1]. In this work we focus on relocalisation and loop closing, two open problems that are essential for real SLAM applications.

Relocalisation consists in locating the robot on an existing map when its pose is unknown. Relocalisation is indispensable if we want to reuse a previous map, otherwise the map becomes useless apart from localising the camera during a certain operation. In addition relocalisation is essential for recovering after a tracking failure, very common in visual SLAM due to occlusions (a person moving in front of the camera), untextured areas or fast camera movements.

Loop closing is the task of detecting when a robot is revisiting a previously mapped area, in order to correct the error accumulated in the robot trajectory during exploration. Tracking data association is able to close small loops. In the case of large loops, the prediction of the camera location accumulates so much error, that data association fails to establish correspondences between the current image and the map. The result is that the system behaves as if the image were showing a new place and thus duplicating previously mapped areas.

At the core of loop closing and relocalisation there is a place recognition system, which recognizes revisited places.

The contributions of our work are the following:

- A place recognition system based on bag of words with ORB features, which recognizes revisited places from large viewpoint difference, taking less than 39 ms (without geometrical verification) to recognize a place in databases of 10K images. The system is validated in four different datasets, achieving high recall and no false positives. These results are compared with the state-of-art in place recognition, getting better results and being one order of magnitude faster.
- A relocalisation method that allows to relocalise a camera from very different viewpoints, at frame-rate in maps with thousands of keyframes. We measure experimentally the viewpoint change that our relocalisation is able to handle. The system is robust to changes in scale between around 0.5 and 2.5, any in-plane rotation and to viewpoint angle differences in excess of 50 degrees. As the relocalisation is based on a feature-based place recognizer it permits the interoperability between cameras. We perform an experiment where a camera is able to relocalise in a map built by a different camera.
- A loop closing procedure for keyframe-based SLAM that can detect and correct loops containing hundred of keyframes, before the system needs to insert a new keyframe to the map. The procedure explicitly fuse duplicated points and correct the camera localisation, so that tracking can continue smoothly after loop closure.

II. Related work

A. Place recognition

The survey of Williams et al. [2] compared several approaches for place recognition and concluded that techniques based on appearance, image to image matching, seemed to scale better in large environments than map to map or image to map methods. Within appearance based methods, bags of words techniques [3] are to the fore because of their high efficiency. A great example is FAB-MAP [4] but, as it relies on SURF [5], it spends 400 ms for feature extraction. The work of Gálvez-Lopez and Tardós [6], used for the first time bag of binary words obtained from BRIEF descriptors [7] along with the very efficient FAST feature detector [8], reducing in more than one order of magnitude the time

This work was supported by Dirección General de Investigación of Spain under Project DPI2012-32168, and Gobierno de Aragón Scholarship B121/13.

The authors are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, María de Luna 1, 50018 Zaragoza, Spain. {raulmur, tardos}@unizar.es

978-1-4799-3684-7/14/$31.00 ©2014 IEEE

needed for feature extraction. Nevertheless the use of BRIEF, neither rotation nor scale invariant, limited the system to in-plane trajectories and loop events with similar point of view. Here we extend that work using ORB [9], rotation invariant, scale aware features.

B. Loop closing and relocalisation

Williams et al. [10] perform relocalisation and loop closing on a SLAM system, based on a filter approach. They use randomized list of binary test classifiers to find correspondences between image features and previously trained map features. Classifiers are trained under many different synthetic warped patches and further feature observations (feature harvesting), yielding a relocalisation system robust to viewpoint changes. The main scalability limitation is the memory as it needs 1.25 Mb per map point class.

Eade and Drummond [11] unify relocalisation and loop closing in a graph SLAM of local filtered submaps. They use a bag of word approach building incrementally a vocabulary of 16-D SIFT descriptors that continually searches for revisited places. Once a node candidate is found they use a local landmark appearance model to find correspondences between features to compute a similarity transformation. As a new graph component is created anytime track is lost, relocalisation consist in adding a link between different graph components and loop closing within the same graph component.

Strasdat et al. [12] and Lim et al. [13] perform loop closing on keyframe-based SLAM systems that maintain a covisibility graph. They use bag of words approaches with a tree vocabulary of SURF descriptors to detect loops, adding then new links to the graph that are handled by their optimizers.

Pirker et al. [14] perform relocalisation and loop closing using FAB-MAP in a monocular SLAM system thought for long-term operation. The very recent work of Tan et al. [15], also on monocular SLAM in dynamic environments, perform global localization using SIFT, with GPU acceleration, and arranging map points in a KD-Tree. Also on long-term operation, Johannsson et al. [16] use a bag of words approach with SURF or BRIEF to detect loops in their reduced pose graph.

The improved PTAM version presented in [17] perform relocalisation comparing the current frame and previous keyframes by sum-square-difference of small resolution blurred images. It is limited to relocalise from very similar viewpoints, although this is not a problem for their augmented reality target application.

Our system is similar to [11], [12], [13] and [16] in that we use a bag of words approach to detect loops, but while they do not offer results of their recognizer performance, we validate the high efficiency and reliability of our place recognizer in several datasets. This results also shown a better performance than FAB-MAP, which is used in [14], being one order of magnitude faster. Our relocalisation system is achieved through the same place recognizer, allowing for high viewpoint changes in contrast to [17] ,and allowing the interoperability between cameras. Instead of using a GPU implementation of SIFT and KD-Trees as in [15], we achieve real time performance by using ORB and a bag of binary words approach.

III. Appearance-based Place Recognition

To perform relocalisation and loop closing, we need first a place recognition method. We want a recognizer as general as possible, in the sense that it permits to recognize a place from very difference viewpoints, and also being fast. We build on the very efficient DBoW2 recognizer [18] with the binary rotation invariant, scale aware ORB features [9]. For completeness we next review how DBoW2 works.

A. Review of DBoW2 [6]

In order to detect if an image corresponds to a revisited place, bag of words techniques summarize the content of an image by the visual words it contains. These visual words correspond to a discretization of the descriptor space, known as the visual vocabulary. DBoW2 creates a vocabulary structured as a tree, in an offline step over a big set of descriptors, extracted from a training image dataset.

Processing a new image consist on extracting keypoints and their descriptors, which are assigned to a visual word traversing the vocabulary tree. As descriptors are binary, distances are computed by the Hamming distance. The result is a bag of words vector, containing the term frequency-inverse document frequency (tf-idf) score for each word present in the image. This score is higher as more frequent is a word in the image and less it was in the training dataset. This bag of words vector is then compared against the bag of words vectors of the images in the database using an invert index, which stores for each word, in which images it has appeared.

The similarity between two bag of word vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ is the L_1-score:

$$s(\mathbf{v_1}, \mathbf{v_2}) = 1 - \frac{1}{2}\left| \frac{\mathbf{v_1}}{|\mathbf{v_1}|} - \frac{\mathbf{v_2}}{|\mathbf{v_2}|} \right| \quad (1)$$

This score is normalized with the score one would expect to get for an image showing the same place. This is approximated with the previous image processed, assuming we are processing an image sequence with a certain scene overlap.

$$\eta(\mathbf{v_i}, \mathbf{v_j}) = \frac{s(\mathbf{v_i}, \mathbf{v_j})}{s(\mathbf{v_i}, \mathbf{v_{i-1}})} \quad (2)$$

Images in the database close in time may have similar scores. DBoW2 takes advantage of it, grouping images close in time and computing only one score for the group, which is the sum of the individual scores. Individual scores have to be higher than a threshold α to be considered. Once the database is searched, the group with the highest score is selected and the image with the maximum individual score is considered as a loop candidate for the query image.

A loop candidate in order no to be rejected has to be consistent with k previous queries. It means that the groups with the highest scores for the last k images must form an overlapping sequence. This temporal consistency check improves the robustness of the system as not accepting a loop until there is enough evidence.

Finally a loop candidate is accepted if it passes a geometrical check. DBoW2 was originally applied to raw video sequences (no 3D information) and computes a fundamental matrix with RANSAC. The search for initial correspondences is performed exhaustively but only between those features that belong to the same node at a level l of the vocabulary tree. The database uses a direct index that stores for each feature in an image the node at level l it belongs.

B. Our approach: Place recognition with ORB

We build on DBoW2 with ORB features, which are rotation invariant and can deal with changes in scale, so that our place recognizer can recognize places from very different viewpoints. We rely on the ORB implementation in the OpenCV library. The default parameters of this implementation are to extract 500 keypoints, at 8 different scales with a scale factor of 1.2. As no non-maximum suppression is applied and keypoints are extracted at several scales for the same image position, keypoints are poorly distributed over the image. We found that applying a non-maximum suppression effectively improved the keypoint distribution but highly reducing the descriptor matching performance. Instead, we extract 1000 keypoints, which improves the keypoint distribution and the recall of our recognizer, at the expense of increasing slightly ORB extraction times.

We create the visual vocabulary in an offline step with the dataset Bovisa 2008-09-01 [19]. This dataset is a sequence with outdoors and indoors areas, yielding a vocabulary that will provide good results in both scenarios. We extract ORB features in 10K images from the dataset and build a vocabulary of 6 levels and 10 clusters per level, getting one million words. Such a big vocabulary is suggested in [6] to be efficient for recognition in large image databases.

Our geometrical verification will depend on the scenario, in which we apply the recognizer. When processing raw video sequences (recognizer evaluation), there is no 3D information and we compute a fundamental matrix between the matched images. In the case of relocalisation we have 3D information in the keyframe candidate, but not in the current frame, so we solve a Perspective-n-Point (PnP) [20] problem. In loop closing we have 3D information in both the current and the matched keyframe, then we compute a 3D to 3D similarity transformation [21]. In all the three cases we use a RANSAC scheme. The initial correspondences between ORB features are computed by exhaustive descriptor comparison between those features that belong to the same node at level 4 in the vocabulary tree (counting from leaves up), and applying a nearest neighbor ratio of 0.6. We also propose the use of an orientation consistency test to filter out wrong correspondences, improving the robustness and speeding up the geometrical verification. For each initial match, we compute the rotation increment between the ORB keypoints in both images, voting for rotations in a discretization of $2\pi/60$, which we found enough in our experiments to discard most of the outliers. Only those correspondences of the three most voted rotations will be accepted. We consider more than one rotation because objects at different depths make the rotation increments not to be uniform along the image.

The settings of our place recognizer are shown in Table I.

C. Experimental evaluation

In order to evaluate the performance of our place recognizer, we run the system in four image sequences, NewCollege [22], Bicocca25b [19], Malaga6L [23] and CityCentre [24]. We measure the recall and precision of the loops detected, processing the sequences at the same frequency as in [6] and with the same geometrical check (fundamental matrix with RANSAC) in order to make comparisons. In the geometrical verification we include our rotation consistency check. Experiments were performed in an Intel Core i5 @ 2.30 GHz computer. Table II shows our results and the results provided in [6] for DBoW2 with BRIEF and FAB-MAP 2.0 (only for two of the datasets).

As shown in Table II, our place recognizer achieves high recall in all datasets with no false positives (100 % precision). It makes better than BRIEF in all sequences, with the exception of Bicocca25b, an indoor sequence where loops have a very similar viewpoint. On the other hand NewCollege, Malaga6L and CityCentre are outdoors sequences where there exist bigger viewpoint differences at loop events, and therefore ORB, rotation invariant and scale aware, performs better. Our approach gets also higher recall than FAB-MAP 2.0 in Malaga6L and CityCentre. Fig. 1 shows the loops detected by our recognizer in each dataset.

D. Execution time

In order to complete the performance analysis of our proposed place recognition system, we ran the system in a 10K image sequence from NewCollege and measured the execution time for ORB extraction and loop detection. We

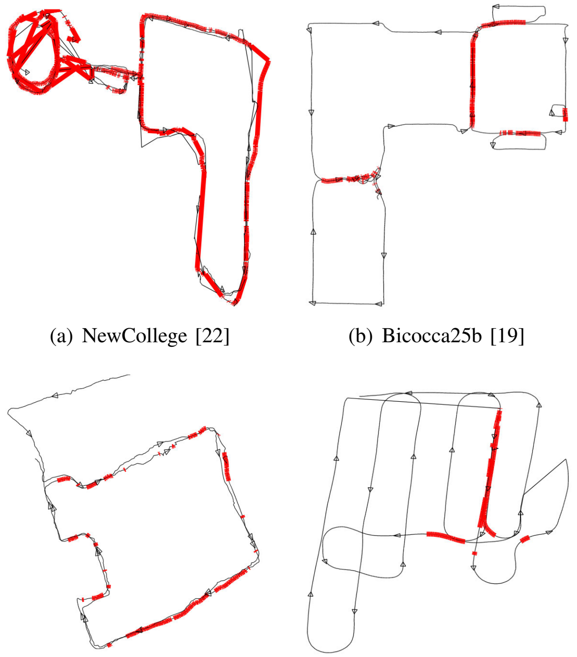
TABLE I
RECOGNIZER PARAMETERS

DBoW2 parameters	ORB parameters
Temporal consistency (k): 3 Score threshold (α): 0.3 Direct index level (l): 4 Nearest. neighbor ratio: 0.6	Number of features: 1000 Scale levels: 8 Scale factor: 1.2

Visual vocabulary
Vocabulary tree levels: 6 Vocabulary clusters/level: 10

TABLE II
COMPARISON WITH THE RESULTS PROVIDED IN [6].

	Our approach		DBoW2 [6]		FAB-MAP 2 [6]	
Datasets	Prec. (%)	Recall (%)	Prec. (%)	Recall (%)	Prec. (%)	Recall (%)
NewCollege	100	70.29	100	55.92	-	-
Bicocca25b	100	76.60	100	81.20	-	-
Malaga6L	100	81.51	100	74.75	100	68.52
CityCentre	100	43.03	100	30.61	100	38.77

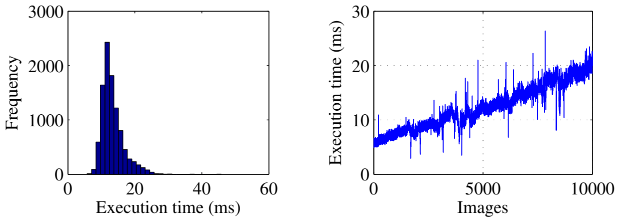

(a) NewCollege [22]

(b) Bicocca25b [19]

(c) CityCentre [24]

(d) Malaga6L [23]

Fig. 1. Loop detection results in each dataset. When the system is detecting a loop, the trajectory is shown in red.



(a) ORB extraction (histogram)

(b) Loop detection

Fig. 2. Execution times for 10K images from NewCollege. Loop detection times do not include geometrical check.

are not interested in the time spent by RANSAC to compute the fundamental matrix because we use another geometrical check for loop closing and relocalisation. Times are shown in Fig. 2. ORB extraction is performed in 13.3 ms on average, and loop detection in less than 25 ms for a database containing 10K images. These times are far from the 400 ms needed by SURF extraction [4].

IV. Loop Closing in Keyframe-based SLAM

In this section we present how we detect and correct loops inside PTAM. We implement our method in the mapping thread, as it is fast enough to close medium-size loops without much delaying the mapping tasks. Next we describe the different steps of our method.

A. Keyframe database

When a new keyframe arrives, we extract ORB features and associate a visual word to each descriptor, traversing the vocabulary tree. We then compose the bag of words vector and insert it in a keyframe database. In order to guarantee the recognizer performance, we extract ORB features, as done in section III, instead of using the features tracked by PTAM. Describing PTAM features with ORB yields very poor results due to the scale pyramid, with a scale factor of two, that PTAM uses for extracting its features. ORB requires much lower scale factors (we use 1.2) to provide a good matching performance. Once the new keyframe has been inserted in the database, it is added to a loop queue.

B. Search for a loop candidate

After the mapping thread has finished handling new keyframes and performing local bundle adjustment, it extracts the oldest keyframe in the loop queue, which ideally should contain only the last inserted keyframe. Then the place recognizer will search in the keyframe database for a loop candidate, which must be consistent with 3 previous keyframe matches. If a candidate is found, we check if the loop was already closed by PTAM, searching for map point measurements in common between the current and the loop candidate keyframe, in which case no additional computation is needed.

C. Geometrical verification: Similarity transformation

The first step to correct a loop is to compute the transformation from the current keyframe camera coordinate system to the loop candidate one. This transformation is the only way to know the drift accumulated during exploration, which in monocular SLAM can occur in seven degrees of freedom: three translations, three rotations and scale [25]. Therefore we compute a similarity transformation, Sim(3) [25], from current keyframe k to loop candidate l:

$$\mathbf{S}_{k,l} = \begin{bmatrix} s_{k,l}\mathbf{R}_{k,l} & \mathbf{t}_{k,l} \\ 0 & 1 \end{bmatrix} \qquad (3)$$

where $s \in \mathbb{R}^+$ is the scale factor, $\mathbf{R} \in \mathrm{SO}(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector. The computation of this transformation serves us as geometrical verification, if it is successful we proceed to correct the loop, otherwise we reject the loop candidate.

To compute the transformation, first we need to find a set of initial correspondences between ORB features in both the current and the loop candidate keyframe. We use the procedure explained at the end of section III-B.

ORB features used for recognition are different from the features used by PTAM for mapping. To recover the 3D coordinates of the ORB features, we interpolate the depth of each feature with its three nearest PTAM tracked features. We only interpolate the z coordinate (along the optical axis), and use this information along with the x and y coordinates in the image, to recover the 3D coordinates in the camera coordinate system. All that features that are further away than 10 pixels from a tracked feature are discarded, as interpolation might be poor. After this step we have a set of 3D to 3D point correspondences.

Now we try to find a similarity transformation supported by enough correspondences, using a RANSAC scheme. At each iteration, we select three correspondences and compute a similarity transformation using our own implementation of the closed-form solution of [26]. Then we count how many correspondences support this transformation, checking the reprojection error of the 3D points in both images. A correspondence is counted as inlier if the sum of the reprojection error in the current and the loop candidate images is less than 6 pixels. If RANSAC finds a transformation supported by the 40% of the initial correspondences in less than 70 iterations (0.99 probability of success), the geometrical verification is considered successful. We refine the transformation computing it again with all inliers, and recomputing it once again if more inliers were found.

D. Loop optimization

To effectively close the loop, the bundle adjustment in the backend will optimize the points and poses of the whole map. However due to the drift accumulated, the current state of the map is far from the solution and the optimizer, specially when using a robust cost function will not converge. We therefore compute an initial solution optimizing the pose graph formed from the current to the loop keyframe pose. We follow the approach of [25].

First we convert for all poses their SE(3) absolute transformation, $\mathbf{T}_{i,w}$, into a similarity Sim(3), $\mathbf{S}_{i,w}$, maintaining the rotation and translation and setting the scale to 1. We then compute the relative transformation $\Delta\mathbf{S}_{i,j}$, between one pose and the following, closing the loop with the similarity transformation computed between the current keyframe and the loop keyframe $\mathbf{S}_{k,l}$.

We then want to minimize the residual error $\mathbf{r}_{i,j}$ between the pose $\mathbf{S}_{i,w}$ and $\mathbf{S}_{j,w}$ with respect to the constraint $\Delta\mathbf{S}_{i,j}$ in the tangent space $\mathfrak{sim}(3)$ and in a minimal representation:

$$\mathbf{r}_{i,j} = \left(\log_{\mathrm{Sim}(3)}(\Delta\mathbf{S}_{i,j} \cdot S_{j,w} \cdot S_{i,w}^{-1})\right)^{\vee}_{\mathfrak{sim}(3)} \tag{4}$$

where $\log_{\mathrm{Sim}(3)} : \mathrm{Sim}(3) \rightarrow \mathfrak{sim}(3)$ maps from the over-parametrized representation of the transformation to the tangent space and $(\cdot)^{\vee}_{\mathfrak{sim}(3)} : \mathfrak{sim}(3) \rightarrow \mathbb{R}^7$ is the *vee-operator* that maps from the tangent space to the minimal representation with the same elements as the degrees of freedom of the transformation [21].

Initially all the residuals are zero, except for the loop. We then optimize the poses to distribute this error along the graph. The cost function to minimize is defined as follows:

$$\chi^2 = \sum_{i,j} \mathbf{r}_{i,j}^{\top} \Lambda_{i,j} \mathbf{r}_{i,j} \tag{5}$$

where $\Lambda_{i,j}$ is the inverse covariance of the residual $\mathbf{r}_{i,j}$ and is set to the identity. We exclude the loop keyframe pose from the optimization to fix the seven degrees of freedom of the solution. We use the Levenberg-Marquadt method implemented in the graph optimizer g2o [27] and exploit the sparse structure of the problem by using the solver CHOLMOD based on Cholesky decomposition.

After poses have been optimized we need to correct the 3D points associated to them. For each point $\mathbf{x}_j$ we select a source keyframe $\mathbf{T}_{i,w}$ and map the point using the optimized $\mathbf{S}^{\mathrm{cor}}_{i,w}$ as follows:

$$\mathbf{x}_j^{\mathrm{cor}} = (\mathbf{S}^{\mathrm{cor}}_{i,w})^{-1} \cdot \mathbf{T}_{i,w} \cdot \mathbf{x}_j \tag{6}$$

The last step is to convert the corrected similarity transformations $\mathbf{S}^{\mathrm{cor}}_{i,w}$ back to 3D rigid body transformations $\mathbf{T}^{\mathrm{cor}}_{i,w}$. Map point scale has been corrected in (6), so we just get rid of the scale factor in the similarity and maintain the rotation as it is not affected by the scale. However translation was computed for the scale factor of the similarity and we need to re-scale it.

$$\mathbf{S}^{\mathrm{cor}}_{i,w} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \rightarrow \mathbf{T}^{\mathrm{cor}}_{i,w} = \begin{bmatrix} \mathbf{R} & \frac{1}{s}\mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{7}$$

The poses and points will be jointly optimized with this seed, next time PTAM performs global bundle adjustment.

E. Fuse points

When a loop is not closed by PTAM itself, map points are created in keyframes that show previously reconstructed areas, thus duplicating map points (more than one point corresponding to the same physical point in space). Therefore after correcting the pose graph we have to take care of the duplicated points at the loop closure.

We start from the current keyframe $\mathbf{T}_{k,w}$ and project on it the points seen in the loop keyframe $\mathbf{T}_{l,w}$, erasing all that points in the current keyframe that are in a distance less than 20 pixels to a reprojected point. Afterwards we do the same but projecting the points from $\mathbf{T}_{l-\delta,w}$ and $\mathbf{T}_{l+\delta,w}$ for δ from 1 until no points are erased. This process is also applied to $\mathbf{T}_{k-\delta',w}$ for δ' from 1 until no points are erased. Once we got rid of duplicated points we need to perform new data association in all that keyframes where points where erased. We use a large search radius because poses and points are still not jointly optimized. This data association is very important because it is the only way in which the loop closure is attached, when bundle adjustment is later performed by the backend of PTAM.

F. Tracking correction

The last step in loop closing is to inform the frontend about the loop closure and correct the camera location and velocity. First we compute the relative transformation from the current camera pose $\mathbf{T}_{c,w}$ to the non-corrected pose of the current keyframe $\mathbf{T}_{k,w}$:

$$\Delta\mathbf{T}_{c,k} = \mathbf{T}_{c,w} \cdot \mathbf{T}_{k,w}^{-1} \tag{8}$$

we then have to scale the translation of $\Delta\mathbf{T}_{c,k}$ by the scale factor $s_{k,l}$ from the similarity transformation computed in section IV-C:

$$\Delta\mathbf{T}_{c,k} = \begin{bmatrix} \mathbf{R}_{c,k} & \mathbf{t}_{c,k} \\ 0 & 1 \end{bmatrix} \rightarrow \Delta\mathbf{T}^{\mathrm{cor}}_{c,k} = \begin{bmatrix} \mathbf{R}_{c,k} & \frac{1}{s_{k,l}}\mathbf{t}_{c,k} \\ 0 & 1 \end{bmatrix} \tag{9}$$

We now apply $\Delta\mathbf{T}^{\mathrm{cor}}_{c,k}$ to the corrected current keyframe pose $\mathbf{T}^{\mathrm{cor}}_{k,w}$ to recover the camera pose:

$$\mathbf{T}^{\mathrm{cor}}_{c,w} = \Delta\mathbf{T}^{\mathrm{cor}}_{c,k} \cdot \mathbf{T}^{\mathrm{cor}}_{k,w} \tag{10}$$

Finally we need to correct the decay velocity motion model dividing the linear speed by $s_{k,l}$.

V. Relocalisation in Keyframe-based SLAM

Our relocalisation system is launched anytime the track is lost or as initialization to localise in a previous map. Next we describe the steps of our relocalisation procedure.

First we extract ORB features in the incoming frame from the camera, with the parameters presented in section III-B (1000 keypoints, 8 scale levels with a scale factor of 1.2). Then ORB descriptors are associated to a visual word and we compose the bag o words vector for the current frame. Our place recognizer will search then for a keyframe candidate in a database. We assume that a keyframe database exists, which could be the same created for loop closing or another created, for example in an offline step. Here we want to get a candidate as soon as possible and thus, we use neither a minimum threshold α nor a temporal consistency check. Although this reduce the robustness, further geometrical verification and the requirement for the system to be able to track the map after relocalisation, mitigate this loss of robustness.

If the place recognizer finds a keyframe candidate, we next compute a set of initial correspondences between ORB features in the current frame and the keyframe candidate, using the procedure explained at the end of section III-B.

Once we have a set of initial correspondences, we need to retrieve the 3D coordinates of the ORB features in the keyframe candidate. This is done by interpolation with the three nearest tracked features in the keyframe candidate. As done in loop closing (section IV-C), we do not interpolate those ORB features that are further away than 10 pixels from tracked features, as interpolation would be poor. After this step we have a set of 2D-3D point correspondences (from current frame to keyframe candidate).

At this point we try to find a camera pose that agrees with the 2D-3D correspondences. We use a RANSAC scheme, selecting at each iteration four correspondences an solving the Perspective-n-Point problem, which returns a camera pose. We use the C++ implementation of [20]. We count at each iteration the number of inliers, that is the correspondences that support the camera pose, checking the reprojection errors. If RANSAC is able to find a camera pose supported for more than the 40% of the initial correspondences (but at less 20), in less than 178 iterations (99% of success if there exist a solution), then it is considered as successful. The camera pose is refined, computing it with all inliers, and recomputing it once again if more inliers are found.

Finally, using the computed camera pose, the tracking thread must be able to track the camera in the map. It must be able to track the camera for more than 20 frames, to let the mapping thread insert new keyframes again. In case the camera pose were wrong the tracking thread would fail tracking the camera and the relocalisation process would be launched again.

VI. Experiments

We evaluate the performance of our PTAM version with the proposed relocalisation and loop closing modules, in several experiments. We perform the experiments in the same Intel Core i5 @ 2.30 GHz computer than in section III-C.

A. Closing a loop

In this experiment the sequence consist in a hand-held camera moving laterally inside a bedroom, facing the walls and completing a loop trajectory. The camera is a Unibrain Fire-i™ with 640×480 resolution and 30 FPS. Fig. 3 shows the image sequence of the first and second times the same place is seen. PTAM itself cannot close the loop, as shown in Fig. 4. When we run the system with our loop closing module activated, the loop is detected and closed. Fig. 5 shows the two keyframes matched by our loop detector. Fig. 6 shows the instant just after the loop is closed. The final reconstructions of the room performed by PTAM without and with the loop closing module activated are shown in Fig. 7. Original PTAM is not able to close the loop and duplicates the map yielding a poor reconstruction. Using our loop closing method the loop is closed at run-time, highly improving the reconstruction.

We measure the time in three executions and find that the system spent 98.2 ms on average, to detect and close the loop, which contained 55 keyframes. Table III shows the time spent at each step.

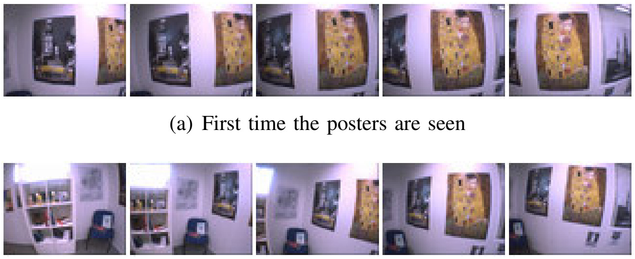
B. Relocalisation limits

The purpose of this experiment is to find the change in viewpoint that our relocalisation system is able to handle. Specifically we measure the change in scale, the in-plane rotation and the change in viewpoint angle. We start from the reconstruction of a wall and then the camera is occluded and moved to another place, so that the system try to relocalize. Fig. 8 shows the results of the experiments. Our system can handle scale changes between 0.36 and 2.93, any in-plane rotation, and the camera can relocate to a keyframe with an

TABLE III

Average loop closing execution times

ORB extraction	16.2 ms
Loop detection	9.7 ms
Geometrical verification	29 ms
Loop optimization	18.9 ms
Fuse points + Correct Tracking	24.38 ms
Total:	98.2 ms



(a) First time the posters are seen

(b) Second time the posters are seen

Fig. 3. Image sequences of the loop (only some image samples).

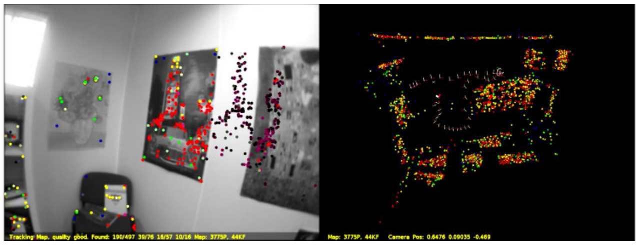


Fig. 4. The camera trajectory accumulates so much error, that when projecting the poster into the image, it appears displaced and no correspondences are found in the image, as shown on the left side of the figure. The loop is not closed by PTAM itself and the poster is duplicated, as shown at the upper left corner of the map reconstruction (on the right side of the figure).

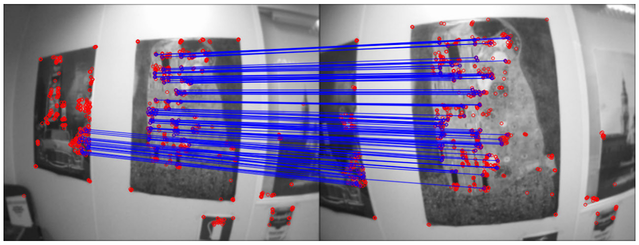


Fig. 5. Keyframes matched by our loop detector. Left keyframe is the current one and the right is the loop keyframe found. Blue lines are ORB correspondences that support the similarity transformation computed.

angle difference in the optical axis up to 59 degrees. This results are only indicative as they can vary with the scene properties (textures, object distribution, etc.).

C. Interoperability: changing the camera

In this experiment we demonstrate that our system, which rely on local features and PnP, allows the interoperability between cameras. We part from the map of the bedroom, built in the experiment described in section VI-A, with the Unibrain camera. Then we run a sequence in which a robot with a Kinect on board navigates inside the room. We deactivate the mapping thread and look if PTAM is able to perform localisation only using the previous map. Although the Kinect and Unibrain cameras have the same resolution, the field of view angle of the Unibrain is about 80° and Kinect 60°. Our relocalisation system was able to initially relocate the camera and PTAM tracked the camera as the robot moved. More relocalisations were performed when PTAM lost the track because the robot faced areas with not enough map features. Fig 9 shows some of the relocalisations performed and Table IV shows the average relocalisation times in the experiment, where the initial map contained 76 keyframes.

Although the sequences were recorded at different points in time, the scenario remained mostly unchanged and therefore relocalisation and PTAM tracking can be performed. There was one wrong relocalisation against a big poster that was not exactly at the same place, see the bottom-right image in Fig. 9. Recent research as [14], [15] have shown promising results on SLAM in dynamic environments.

The accompanying video shows the real-time executions of the loop closing and the interoperability experiments.

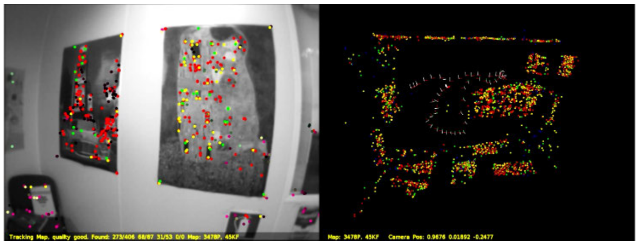
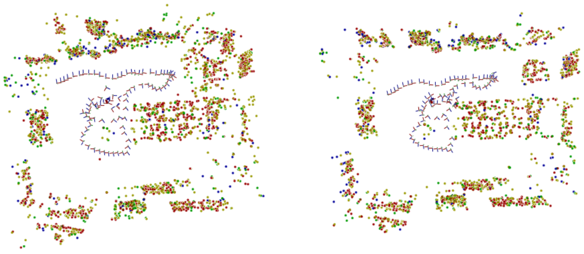


Fig. 6. Our system just after closing the loop. Now the tracked features on the posters are map points created at the first time those poster were seen. Duplicated map points at the upper left corner of the map have been fused.



(a) without loop closing

(b) with loop closing

Fig. 7. PTAM reconstruction for the bedroom sequence. The camera starts at the top-left corner and moves clockwise performing a loop trajectory. Without the loop closing procedure the loop is not closed and some parts of the map (upper left corner) are duplicated. PTAM with our loop closing module is able to close the loop as it runs.

TABLE IV

AVERAGE RELOCALISATION EXECUTION TIMES

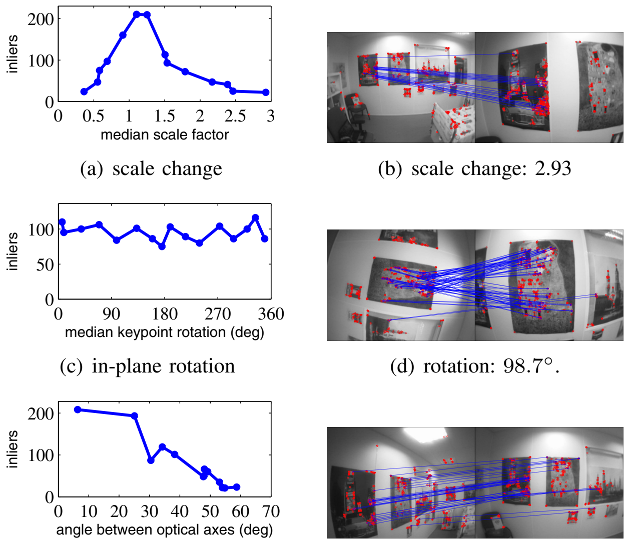
ORB extraction	14.4 ms
Keyframe candidate selection	6.4 ms
Camera pose computation	14.9 ms
Total:	35.7 ms

VII. CONCLUSIONS

Our place recognizer based on bag of words from ORB features has demonstrated to be a very efficient and robust recognizer for general camera trajectories. To the best of our knowledge there is no faster system that can recognize places from such large different viewpoints.

We proved that our relocalisation system can recover the camera pose from severe viewpoint change. The experiments showed, that it can relocalise in less than 36 ms in maps containing 76 keyframes. The system spent only 6.4 ms in finding a keyframe candidate, which is the only step that depends on the map size. The place recognizer scales well, taking less than 10 ms, to find an image candidate, on databases containing thousands of images (as shown in Fig. 2(b)). Thus we state that our relocalisation system can work at frame-rate ($\sim$ 25 Hz) in maps containing thousands of keyframes. In addition we have demonstrated, that our relocalisation system is able to relocate a camera in a map built, previously by another camera. So it allows for the interoperability between cameras.

Our loop closing method spent on average 98.2 ms to close a loop containing 55 keyframes, taking only 9.7 ms to detect



(a) scale change (b) scale change: 2.93

(c) in-plane rotation (d) rotation: 98.7°.

(e) viewpoint angle (f) angle difference: 53°.

Fig. 8. On the left: PnP inliers at each relocalisation. On the right: an example for each experiment. Blue lines are the inlier correspondences.

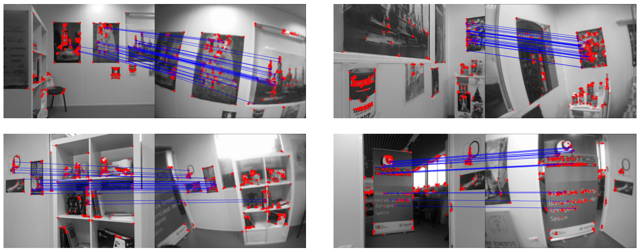


Fig. 9. Interoperability experiment. Matches between kinect frames and unibrain keyframes.

the loop and 18.9 ms to optimize the pose graph, which are the only times that depends on map size. The time interval between keyframes is around 670 ms. According to the high amount of free time, our system could easily close loops of some hundreds of keyframes, before a new keyframe arrives.

Further research could focus on integrating the proposed relocalisation and loop closing procedures in a large scale SLAM system. We also plan to investigate how to extend the system to work in the long-term in dynamic environments, as some recent promising works [14], [15].

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[2] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[3] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[4] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *IEEE European Conference on Computer Vision (ECCV)*, 2006.

[6] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *IEEE European Conference on Computer Vision (ECCV)*, 2010.

[8] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *IEEE European Conference on Computer Vision (ECCV)*, 2006.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[10] B. Williams, G. Klein, and I. Reid, "Automatic relocalization and loop closing for real-time monocular SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1699–1712, 2011.

[11] E. Eade and T. Drummond, "Unified loop closing and recovery for real time monocular SLAM." in *British Machine Vision Conference*, 2008.

[12] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[13] J. Lim, J. M. Frahm, and M. Pollefeys, "Online environment mapping," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[14] K. Pirker, M. Ruther, and H. Bischof, "CD SLAM-continuous localization and mapping in a dynamic world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[15] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[16] H. Johannsson, M. Kaess, M. Fallon, and J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[17] G. Klein and D. Murray, "Improving the agility of keyframe-based slam," in *IEEE European Conference on Computer Vision (ECCV)*, 2008.

[18] D. Gálvez-López and J. D. Tardós, "Software: DBoW2: Enhanced hierarchical bag-of-word library for C++," 2012, download date: 2013. [Online]. Available: http://webdiis.unizar.es/ dorian

[19] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardós, "RAWSEEDS: Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets," in *Intelligent Robots and Systems (IROS) Workshop on Benchmarks in Robotics Research*, 2006.

[20] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[21] H. Strasdat, "Local Accuracy and Global Consistency for Efficient Visual SLAM," Ph.D. dissertation, Imperial College, London, October 2012.

[22] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, 2009.

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

[24] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[25] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM." in *Robotics: Science and Systems (RSS)*, 2010.

[26] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[27] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.