View-based Maps

Kurt Konolige, James Bowman, JD Chen, Patrick Mihelich Willow Garage
Menlo Park, CA 94025
Email: konolige@willowgarage.com

Michael Calonder, Vincent Lepetit, Pascal Fua EPFL
Lausanne, Switzerland
Email: michael.calonder@epfl.ch

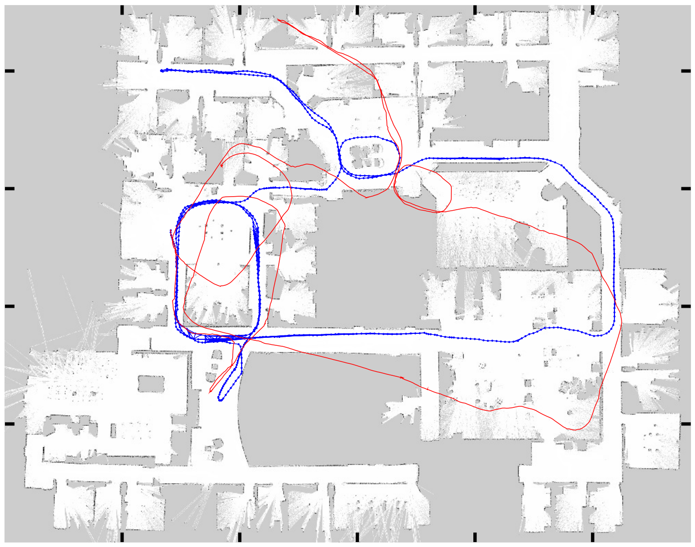
***Abstract*— Robotic systems that can create and use visual maps in real-time have obvious advantages in many applications, from automatic driving to mobile manipulation in the home. In this paper we describe a mapping system based on retaining stereo views of the environment that are collected as the robot moves. Connections among the views are formed by consistent geometric matching of their features. Out-of-sequence matching is the key problem: how to find connections from the current view to other corresponding views in the map. Our approach uses a vocabulary tree to propose candidate views, and a strong geometric filter to eliminate false positives – essentially, the robot continually re-recognizes where it is. We present experiments showing the utility of the approach on video data, including incremental map building in large indoor and outdoor environments, map building without localization, and re-localization when lost.**[1]

I. Introduction

Fast, precise, robust visual mapping is a desirable goal for many robotic systems, from transportation to in-home navigation and manipulation. Vision systems, with their large and detailed data streams, should be ideal for recovering 3D structure and guiding tasks such as manipulation of everyday objects, navigating in cluttered environments, and tracking and reacting to people. But the large amount of data, and its associated perspective geometry, also create challenging problems in organizing the data in an efficient and useful manner.

One useful idea for maintaining the spatial structure of visual data is to organize it into a set of representative views, along with spatial constraints among the views, called a *skeleton*. Figure 1 gives an example of a skeleton constructed in an indoor environment. Typically views are matched in sequence as the camera is moved around, so the skeleton mimics the camera trajectory (red trajectory). In loop closure, the camera enters an area already visited, and can re-connect with older views. The overall map is generated by nonlinear optimization of the system [2, 25, 47]. View-based maps have the advantage of *scalability*: using incremental techniques, new views can be added and the skeleton optimized online. Skeletons are similar to the pose graphs familiar from laser SLAM work [48], although the latter typical have just 2D implementations. Also, skeletons retain the images that are captured at each node, for future matching via place recognition.

[1]This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.



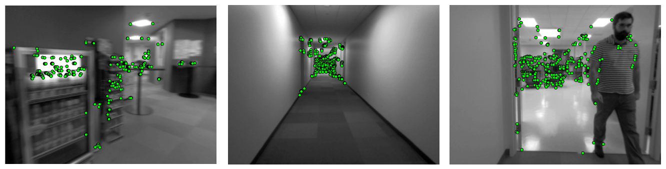


Fig. 1. Top: Skeleton map constructed online from just stereo images, registered against a laser map for reference. Red is visual odometry, blue is corrected by loop closure from visual place recognition. Tick marks at sides of map are 10m intervals. Bottom shows typical views, with blurring, clutter, people, and blank walls.

One problem is how to efficiently perform loop closure. Previous approaches used exhaustive search of the current view against all skeleton views that could possibly be in the area, given the relative uncertainty of views.

This approach does not scale well to larger skeletons, and involves constant calculation of relative covariance. Instead, to limit the number of views that must be considered for loop closure, we employ a vocabulary tree [39] to suggest candidate views, a type of *place recognition* (PR). The vocabulary tree allows us to efficiently filter thousands of skeleton views to find possible matches, as well as add new views online. We call this online PR *re-recognition*: the robot recognizes its position relative to the stored view map on every cycle, without any a priori knowledge of its position (unlike localization, which requires a position hypothesis).

The addition of vocabulary tree PR to view-based maps is a happy alignment of technologies that expands the utility of visual mapping in two important ways.

1) **Incremental mapping.** The system can add new sections on to its map at any point, that is, it can continuously localize *and* map. It is able to wake up anywhere, even outside the current map, and connect itself to the map when it is encountered. It can continually check for loop closures and optimize them. It works online.
2) **Localization and odometry failure.** Typically a robot will fail to localize if its sensors are blocked or degraded in some way. The system can recover from these errors by relocalizing in the map when it gets the chance.

Just as laser sensors helped to solve a static SLAM problem that was difficult for sonars, so new techniques in visual PR and online recognition eliminate the ambiguous nature of 2D laser scan matching, and enable online PR. Visual sensors have much more data, and are better at distinguishing scenes from a single snapshot.

This paper brings together several technologies from both the vision and laser SLAM fields, and integrates them into a robust, real-time system for visual mapping. The main contributions are

- The construction of a real-time system for robust and accurate visual map making over large and small spaces. This system exhibits the key properties of incremental anytime mapping, and recovery from localization failures.
- The use of views (images), view matching, and geometric relations between views as a uniform approach to short-term tracking and longer-term metric mapping and loop closure.
- The integration of a visual vocabulary tree into a complete solution for online place recognition.
- An analysis of the false positive rejection ability of two-view geometry.
- Extensive experiments with real data, showing the scalability of the technique.

In the Experiments section, we highlight some applications that show the scalability and flexibility of view-based maps. For example, even without sequence information, it is often possible to quickly reconstruct a skeleton map from a set of views (Figure 13 and Section V-F). Loop closure over large distances is possible: we show indoor maps with 800m trajectories (Figure 1), and outdoor rough-terrain maps with 5km trajectories. On a smaller scale, view matching with large numbers of points is inherently accurate, showing a few centimeter accuracy over a desktop workspace. Additional capabilities include automatic recovery from localization failures (e.g., occlusion and motion blur) and incremental construction of maps.

Our solution uses stereo cameras for input images. The development of place recognition is also valid for monocular cameras, with the exception that the geometric check is slightly stronger for stereo. However, the skeleton system so far has been developed just for the full 6DOF pose information generated by stereo matching, and although it should be possible to weaken this assumption, we have not yet done so.

II. VSLAM and View Maps

The view map system (Figure 2), which derives from FrameSLAM [2, 29], is most simply explained as a set of nonlinear constraints among camera views, represented as nodes and edges (see Figure 6 for a sample graph). Constraints are input to the graph from two processes, visual odometry (VO) and place recognition (PR). Both rely on geometric matching of views to find relative pose relationships; they differ only in their search method. VO continuously matches the current frame of the video stream against the last keyframe, until a given distance has transpired or the match becomes too weak. This produces a stream of keyframes at a spaced distance, which become the backbone of the constraint graph, or *skeleton*. PR functions opportunistically, trying to find any other views that match the current keyframe. This is much more difficult, especially in systems with large loops. Finally, an optimization process finds the best placement of the nodes in the skeleton.

It is interesting to note that current methods in visual SLAM divide in the same way as in laser-based SLAM, namely, those that keep track of landmarks using an EKF filter (monoSLAM [13, 14] and variations [41, 46]), and those that, like ours, maintain a constraint graph of views, similar to the original Lu and Milios method [34] current GraphSLAM systems [48], and delayed information filter systems [35, 18]. The main limitation

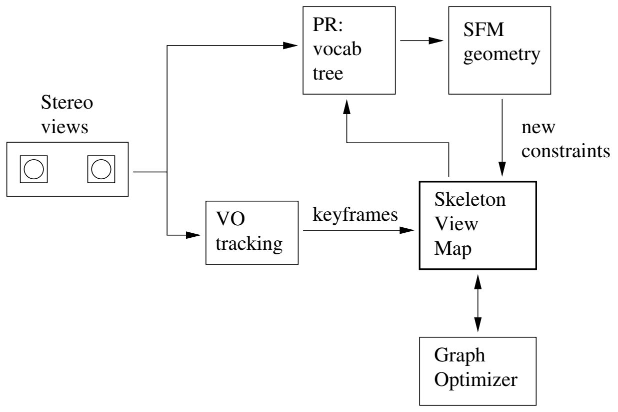


Fig. 2. System overview.

of the landmark method is the filter size, which is only tractable in small (room-size) environments. An exception is [41], which uses a submap technique, although real-time performance has not yet been demonstrated. Landmark systems also tend to be less accurate, because they typically track only a few tens of landmarks per frame. In contrast, our visual odometry technique tracks 300 points per frame, and we construct maps containing several thousand views (and thus hundreds of thousands of points).

In a similar vein, the recent Parallel Tracking and Mapping (PTAM) system [26, 27] also uses 3D landmarks, but employs standard SfM bundle adjustment to build a map from many views. Many more points can be handled in the decoupled tracking phase, leading to accurate and robust performance under many conditions. Still, it is limited to small environments (around 150 keyframes) by the number of points and by bundle adjustment. It is also subject to tracking failures on self-similar textures (e.g., bushes), object motion, and scene changes (e.g., removal of an object). In contrast, view-based maps use consistent view geometry to robustly estimate poses even in the presence of distractors.

The skeleton system deployed here comes directly from the frameSLAM work in [2, 29]. Several other systems employ constraint graphs as the basic map structure. Fraundorfer et al. [19] have a monocular system that represents only direction information between views, and produce only a topological map. Eade and Drummond [16] employ a hybrid approach, using EKF landmarks within a local area called a *node*, then connecting the nodes via similarity relations. An interesting point of their graph optimization is the use of cycles to constrain relative scale between nodes. Other robotics work that employs similar ideas about constructing view-based constraints is in [47, 49]. These systems also keep a constraint network of relative pose information between frames, based on stereo visual odometry, and solve it using nonlinear least square methods. The main difference with our system is that frameSLAM represents the relationships as nonlinear constraints, which are more accurate over angular deformations, and can reduce the size of the skeleton graph to deal with larger areas as required. Recently Sibley et al. [43] have developed a local-consistency optimizer that is similar to constraint networks, but parameterizes the system using relative rather than global variables. They show constant-time updates, even in the presence of loop closures, to create a locally-consistent system.

An interesting variation on pose constraints are the delayed information filter systems [35, 18]. Here an information filter maintains estimates of the relationships between poses (the "delayed" part), updating them in a manner similar to EKF. Since only poses and not features are represented, larger systems can be computed in reasonable time. One drawback is that these systems are linear, and do not relinearize on loop closure; an exception is the Smoothing and Mapping (SAM) technique of [15, 24]. As with other pose-based systems, they develop approximate techniques for generating relative covariances, to limit search for loop closures. Our work bypasses this requirement by performing continuous place recognition over the entire image dataset.

A. *Related Place Recognition Work*

Visual place recognition is an image classification problem; new views are classified against a set of previously seen views. For use in VSLAM, the classifier must support efficient online learning of new reference views. Image matching techniques based on bag-of-words matching are ideally suited to this purpose. For fast lookup of similar places, we rely on the hierarchical vocabulary trees proposed by Nistér and Stewénius [39], which has the advantage of fast online learning of new places. Other methods include alternative approximate nearest neighbor algorithms [44, 37] and various refinements for improving the response or efficiency of the tree [11, 12, 22, 23].

Cummins and Newman [11, 12] show how to use visual features for navigation and loop closure over very large trajectories. Their most recent system, FAB-MAP 2.0 [12], has much in common with our PR method: an inverted index for fast retrieval of relevant frames, and a geometric check for false positive rejection. There are two main differences. FAB-MAP uses a Bayesian filter to determine the probability that a newly-seen frame is indeed a new place or matches a previously-seen one. This filter is implemented very efficiently, and helps to deal with perceptual aliasing in man-made environments with repeated structures. Surviving candidates undergo a further "soft" geometric check for approximately correct structure in matching features. In contrast, we dispense with any Bayesian analysis, and instead depend on a strong structure-from-motion geometric check to eliminate false positives. We think that the strong geometric check is important for matching in domains with high ambiguity, such as the open terrain of Figure 9, which would be rejected by Bayesian analysis because of abundant similar textures. However, we have not done a detailed comparison, which remains as future work. In terms of computation, the efficient Bayesian analysis should allow FAB-MAP 2.0 to scale better than our techniques, since it eliminates large numbers of candidates with little effort.

Jegou et al. [22] have a system similar to FAB-MAP, in which they incorporate Hamming embedding of the features and weak geometric consistency constraints into the inverted file to improve performance. Jegou et al. [23] also note that even using inverted files, query time is linear in the number of reference images; they propose a two-level inverted file scheme to improve the complexity. Our experiments do show linearly increasing query/update time, but with a very small constant factor (Figure 7). For our scale of application (in the thousands of images), the query time of the vocabulary tree is nearly constant, and such sophistication is unnecessary.

In application to graph-based VSLAM, Callmer et al. [6] propose a loop closure procedure that uses a vocabulary tree in a manner similar to ours, along with a weak geometric check to weed out some false positives. Eade and Drummond [17] have extended their node approach with a PR method based on bag of words, in which they learn the words online. They give few statistics on the performance of PR, so it isn't possible to compare directly – they have the advantage of learning based on the observed features, but have far fewer words (3000 vs. 100,000 in our case). They have independently introduced some of the same applications of PR as given here: recovery from localization error and stitching together trajectories when common views are found. Finally, Williams et al. [50] also recover from localization errors in a landmark-based VSLAM framework, by training a classifier to recognize landmarks online; so far their system has been limited to 80 landmarks, mostly because of EKF processing.

There is an interesting convergence between our work and recent photo stitching in the vision community [45]. They employ a similar skeletonization technique to limit the extent of bundle adjustment calculations, but run in batch mode, with no attempt at real-time behavior. Klopschitz et al. [28] use a vocabulary tree to identify possible matches in video stream. To limit false positives, they look at a sequence of subsequent matches to verify the initial match. They are similar to our work in emphasizing online operation.

B. *Implemented Systems*

There are several recent systems that incorporate similar ideas to the ones in this paper; there appears to be a convergence to using new techniques in place recognition to formulate robust, integrated visual mapping. The system of Eade and Drummond [17] has already been mentioned. Newman et al. [38] presented an impressive integrated system with laser and stereo vision that is similar to the one described here: visual odometry for front-end tracking, and FAB-MAP for online loop closure. They also note, as we do, that maps can be stiched together over non-contiguous runs, although they use batch-mode processing to do so, rather than the online method presented here. An interesting variation of this system uses the locally-consistent optimizer mentioned above, rather than a pose graph [36].

III. FrameSLAM Background

The view map system, which derives from our work on FrameSLAM [2, 29, 31], is most simply explained as a set of nonlinear constraints among camera views, represented as nodes and edges (see Figures 11 and 13 for sample graphs). Constraints are input to the graph from two processes, visual odometry (VO) and place recognition (PR). Both rely on geometric matching of stereo views to find relative pose relationships. The poses are in full 3D, that is, 6 degrees of freedom, although for simplicity planar projections are shown in the figures of this paper.

VO and PR differ only in their search method and features. VO uses FAST features [42] and SAD (Sum of Absolute Differences) correlation, continuously matching the current frame of the video stream against the last keyframe, until a given distance has transpired or the match becomes too weak. This produces a stream of keyframes at a spaced distance, which become the backbone of the constraint graph, or *skeleton*. PR functions opportunistically, trying to find any other views that match the current keyframe, using random tree signatures [7] for viewpoint independence. This is much more difficult, especially in systems with large loops. Finally, an optimization process finds the best placement of the nodes in the skeleton.

For two views c_i and c_j with a known relative pose, the constraint between them is

$$\Delta z_{ij} = c_i \ominus c_j, \text{ with covariance } \Lambda^{-1} \qquad (1)$$

where $\ominus$ is the inverse motion composition operator – in other words, c_j's position in c_i's frame; we abbreviate the constraint as c_{ij}. The covariance expresses the strength of the constraint, and arises from the geometric matching step that generates the constraint. In our case, we match two stereo frames using a RANSAC process with 3 random points to generate a relative pose hypothesis. The hypothesis with the most inliers is refined in a final nonlinear estimation, which also yields a covariance estimate. In cases where there are too few inliers, the match is rejected; the threshold varies for VO (usually 30) and PR (usually 80).

Given a constraint graph, the optimal position of the nodes is a nonlinear optimization problem of minimizing $\sum_{ij} \Delta z_{ij}^\top \Lambda \Delta z_{ij}$; a standard solution is to use preconditioned conjugate gradient [2, 21]. For real-time operation, it is more convenient to run an incremental relaxation step, and the recent work of Grisetti et al. [20] on stochastic gradient descent provides an efficient method of this kind, called Toro, which we use for the experiments. Toro has an incremental mode that allows amortizing the cost of optimization over many view insertions.

Because Toro accepts only a connected graph, we have used the concept of a *weak link* to connect a disjoint sequence to the main graph. A weak link has a very high covariance so as not to interfere with the rest of the graph, and is deleted as soon as a normal connection is made via place recognition.

A. Geometric Consistency Check and Pose Estimation

Constraints arise from the perspective view geometry between two stereo camera views. The process can be summarized by the following steps:

1) Match features in the left image of one view with features in the left image of the other view ($N \times N$ matching).
2) (RANSAC steps) From the set of matches, pick three candidates, and generate a relative motion hypothesis between the views. Stereo information is essential here for giving the 3D coordinates of the points.
3) Project the 3D points from one view onto the other based on the motion hypothesis, and count the number of inliers.
4) Repeat 2 and 3, keeping the hypothesis with the best number of inliers.
5) Polish the result by doing nonlinear estimation of the relative pose from all the inliers.

The last step iteratively solves a linear equation of the form

$$J^\top J \delta x = -J^\top \Delta z, \qquad (2)$$

where Δz is the error in the projected points, δx is a change in the relative pose of the cameras, and J is the Jacobian of z with respect to x. The inverse covariance (precision) derives from $J^\top J$, which approximates the curvature at the solution point. As a practical matter, Toro accepts only diagonal precisions, so instead of using $J^\top J$, we scale a simple diagonal precision based on the inlier response, as follows

$$\mathrm{diag}(1,1,1,100,100,100)\ln(n-r+1), \qquad (3)$$

where n is the number of inliers, and r is the rejection threshold. Here the angles are given higher weight than translation, in accordance with precision matrices generated by $J^\top J$.

In cases where there are too few inliers ($n < r$), the match is rejected; this issue is explored in detail in Section IV-C. The important result is that geometric matching provides an almost foolproof method for rejecting bad view matches.

B. Visual Odometry and Re-detection

Our overriding concern is to make the whole system robust. In outdoor rough terrain, geometric view matching for VO has proven to be extremely stable even under very large image motion [30], because points are re-detected and matched over large areas of the image for each frame. For this paper's experiments, we use a recently-developed scale-space detector called STAR[2] outdoors, and the FAST detector indoors.

There is no motion assumption to drive keypoint match prediction, although for computational efficiency we limit the match range of each keypoint in the left image to a corresponding area of size 128x64 pixels in the right image – this allows for very large jumps between images. Keypoints are matched using SAD correlation of a 16x16 patch. Robust geometric matching then determines the best pose estimate. Keyframes are switched when the match inlier count goes below 100, or the camera has moved 0.3m or 10 degrees.

In a 400 m circuit of our labs, with almost blank walls, moving people, and blurred images on fast turns, there was not a single VO frame match failure (see Figure 6 for sample frames). The PTAM methods of [26], which employ hundreds of points per frame, can also have good performance, with pyramid techniques to determine large motions. For localization failures, they

[2]STAR is a multiscale center-surround detector very similar to the CenSurE detector [1]. The main difference is that it uses combinations of squares rather than octagons to approximate circles. It is not yet published; the code is available in the ROS repository at www.ros.org.

have added a relocalization module [50], which finds matching features in the map using a randomized tree search, which is a type of on-demand place recognition.

C. System-Level Algorithm

The robot stores a skeleton map M that represents its current global map. Every time the robot wakes up, it runs the algorithm of Table I for visual mapping. In general form, the algorithm is very simple. Waking up, the robot is lost, and inserts a weak link to keep the whole map connected (see Section III). Then it processes stereo frames at 30 Hz, using VO to connect each frame to the last. If there is a failure in VO, it proceeds as with wakeup, putting in a weak link. Otherwise, it tests for a keyframe addition, which happens if the match score falls below a threshold, or the robot has moved a certain amount (usually 0.3m or 10 degrees).

The VO module provides a constant stream of keyframes to be integrated into the skeleton graph. To control the size of the graph for large environments, only a subset of the keyframes need to be kept in the graph. As each keyframe is generated by VO, it is kept in a small sequential buffer until enough distance has accumulated to integrate it into the skeleton. At this point, all the views in the buffer are reduced to a single constraint between the first and last views in the buffer. The reduction process is detailed in [2]; for a linear sequence of constraints, it amounts to compounding the pose differences $\Delta z_{01} \oplus \Delta z_{12} \oplus \cdots \oplus \Delta z_{n,n-1}$. As an example, in the 5km outdoor runs, a typical distance between skeleton views is 5m.

One can imagine many other schemes for skeleton construction that try to balance the density of the graph, but this simple one worked quite well. In the case of lingering in the same area for long periods of time, it would be necessary to stop adding new views to the graph, which otherwise would grow without limit. The frameSLAM graph reduction supports online node deletion, and we are starting to explore strategies for controlling the density of views in an area.

If a skeleton view is added, it checks all views in the graph for matches, and adds any links it finds, removing the now-unnecessary weak link. Finally, the graph is incrementally optimized for a few iterations of Toro. The optimization can be amortized over time, allowing online operation for fairly large graphs, up to several thousand views (see the timings in Figure 7).

```
Anytime Mapping
    Input: skeleton view map $M$
    Output: updated map $M$
  1) On wakeup, initialize the current keyframe $K_c$ and
     insert a weak link between $K_c$ and the last encountered
     map keyframe.
  2) Get new stereo frame $S$
  3) Perform VO to get the relative pose $K_c \leftrightarrow S$
  4) VO failure?
       a) Add weak link from $S$ to $K_c$
       b) If previous $S$ was a VO failure, delete it
       c) Continue at step (2)
  5) Switch keyframes?
       a) $K_c \Leftarrow S$
       b) Add skeleton node?
            i) $M \Leftarrow M \cup \{S\}$
            ii) Place recognition for $S$?
                 A) Add PR links to $M$
                 B) Remove any weak links
                 C) Incrementally optimize $M$
  6) If not shut down, continue from step (2)
```

TABLE I
SYSTEM-LEVEL SKELETON GRAPH CONSTRUCTION.

IV. Matching Views

In this section we describe our approach to achieving efficient view matching against thousands of frames. We develop a filtering technique for matching a new image against a dataset of reference images (PR), using a vocabulary tree to suggest candidate views from large datasets. From a small set of the top candidates, we apply the geometric consistency check, using Randomized Tree signatures [7] as an efficient viewpoint-invariant descriptor for keypoint matching. Finally, we develop statistics to verify the rejection capability of this check.

A. Compact Randomized Tree Signatures

Our approach to matching keypoints relies on statistical learning techniques to compute a probabilistic model of the patches surrounding them. Since the set of possible appearances of patches around an image feature, seen under changing perspective and lighting conditions, can be treated as a class, a classifier based on Randomized Trees [3] can be trained to recognize them independently of pose. This is done using a database of patches that is obtained by warping keypoints of a reference image by randomly chosen homographies. The resulting algorithm has very fast run-time performance but requires a computationally intensive training phase that precludes online learning of new feature points [32, 40].

In recent work, we overcame this limitation based on the following observation: If we train a Randomized Tree classifier [32] to recognize a number of *base* keypoints extracted from an image database, all other keypoints can be characterized in terms of their response to these classification trees, which we will refer to as their *signature*. Because the signature can be computed very fast, the learning becomes quasi-instantaneous and therefore practical for online applications. We attribute this desirable behavior to the fact that, assuming the initial set of keypoints is rich enough, the new keypoints will be similar to some of those initial points and the signature will summarize these similarities. In other words, we replace a hand-crafted descriptor such as SIFT [33] by one that has been empirically learned from training data to be very discriminative. Remarkably, this can be done using a fairly limited number—500 in our experiments—of base keypoints [8].

Furthermore, our signatures are long sparse vectors that can be compacted into small dense ones by multiplying them by random ortho projection matrices. This results in a substantial speed increase over the fastest competing descriptors such as SURF [4], at essentially the same recognition rates [9]. As shown in Table I, we are about 32 times faster than SURF when running on the same CPU. Furthermore, our CPU implementation is even slightly faster than a GPU implementation of SURF.

	Descriptor Creation (512 keypts)	N×N Matching (512×512 keypts)
Compact RTs (CPU)	**7.9 ms**	**6.3 ms**
U-SURF64 (CPU)	150 ms	120 ms 73 ms (ANN)
U-SURF64 (GPU)		6.8 ms

TABLE II

TIMINGS FOR DESCRIPTOR CREATION AND MATCHING.

B. A Prefilter for Place Recognition

We have implemented a place recognition scheme based on the vocabulary trees of Nistér and Stewénius [39] which has good performance for both inserting and retrieving images based on the compact RT descriptors. We call this step a *prefilter* because it just suggests candidates that could match the current view, which must then be subject to the geometric consistency check for confirmation and pose estimation. VO and PR both use the geometric check, but PR has the harder task of finding matches against all views in the skeleton, while VO only has to match against the reference keyframe. The prefilter is a bag-of-words technique that works with monocular views (the left image of the stereo pairs).

The vocabulary tree is a hierarchical structure that simultaneously defines both the visual words and a search procedure for finding the closest word to any given keypoint. The tree is constructed offline by hierarchical k-means clustering on a large training set of keypoint descriptors. The set of training descriptors is clustered into k centers. Each center then becomes a new branch of the tree, and the subset of training descriptors closest to it are clustered again. The process repeats until the desired number of levels is reached.

The discriminative ability of the vocabulary tree increases with the number of words, at a cost of greater quantization error [5] and increased memory requirements. Nistér and Stewénius have shown that performance improves with the number of words, up to very large (>1M) vocabularies. In our experiments, we use about 1M training keypoints from 500 images in the Holidays dataset [22], with $k = 10$, and create a tree of depth 5, resulting in 100k visual words. The Holidays dataset consists of mostly outdoor images, so the vocabulary tree is trained on data visually dissimilar to the indoor environments of most of our experiments.

The vocabulary tree is populated with the reference images by dropping each of their keypoint descriptors to a leaf and recording the image in a list, or *inverted file*, at the leaf. To query the tree, the keypoint descriptors of the query image are similarly dropped to leaf nodes, and potentially similar reference images retrieved from the union of the inverted files. In either case, the vocabulary tree describes the image as a vector of word frequencies determined by the paths taken by the descriptors through the tree. Each reference image is scored for relevance to the query image by computing the L1 distance between their frequency vectors. The score is entropy-weighted to discount very common words using the Term Frequency Inverse Document Frequency (TF-IDF) approach described in [39, 44].

To evaluate the vocabulary tree as a prefilter, we constructed a small test set of some 180 keyframes over a 20m trajectory, and determined ground truth matches by performing geometric matching across all 180×180 possibilities. In this dataset, each keyframe averages 11.8 ground truth matches. We inserted these keyframes, along with another 553 non-matching distractor keyframes, into the vocabulary tree. Querying the vocabulary tree with each of the 180 test keyframes in turn, we obtained their similarity scores against all the reference images. The sensitivity of the vocabulary tree matching is shown by the ROC curve (Figure 4, left) obtained by varying a threshold on the similarity score.

Since we can only afford to put a limited number of candidates through the geometric consistency check, the critical performance criterion is whether the correct matches appear among the most likely candidates. Varying N, we counted the percentage of the ground

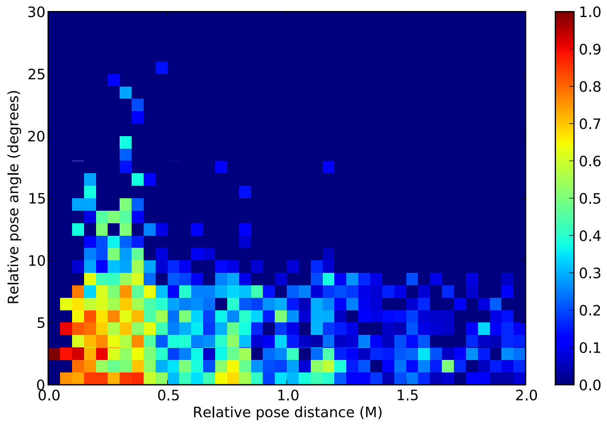


Fig. 3. Recognition rate. The plot shows the proportion of recognized poses for varying pose angle and pose distance. The poses are taken from the final map in Figure 11.

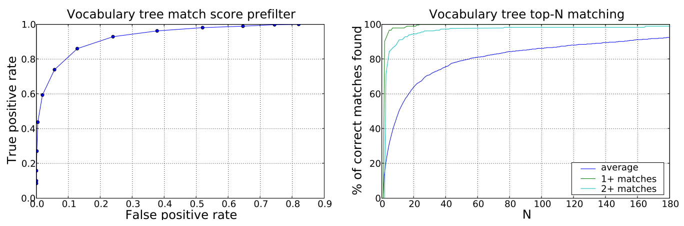


Fig. 4. Left: ROC curve for the vocabulary tree prefilter on the test dataset. Right: "Average" curve shows percentage of the correct matches among the top N results from the vocabulary tree (blue); other curves are the percentage of views with at least 1 or 2 matches in the top N.

truth matches appearing in the top-N results from the vocabulary tree. For robustness, we want to be very likely to successfully relocalize from the current keyframe, so we also count the percentage of test keyframes with at least one or at least two ground truth matches in the top-N results (Figure 4, right).

In our experiments, we take as match candidates the top $N = 15$ responses from place recognition. We expect to find at least one good match for 97% of the keyframes and two good matches for 90% of the keyframes. For any given keyframe, we expect almost 60% of the correct matches to appear in the top 15 results. Figure 3 shows the recognition rate for a full map (Figure 11), as a function of distance and angle to a view. Within a 0.5m radius, the place recognition algorithm gives very high recall when the angle is 10 degrees or less.

C. Geometric Consistency Check

We can predict the ability of the geometric consistency check (Section III-A) to reject false matches by making a few assumptions about the statistics of matched points, and estimating the probability that two unrelated views I_0 and I_1 will share at least M matches, given a relative pose estimate. Based on perspective geometry, any point match will be an inlier if the projection in I_1 lies on the epipolar line of the point in I_0. In our case, with 640×480 images, an inlier radius of 3 pixels, the probability of being an inlier is:

$$A_{track}/A_{image} = (6 * 640)/(640 * 480) = .0125 \quad (4)$$

This is for monocular images; for stereo images, the two image disparity checks (assuming disparity search of 128 pixels) yield a further factor of (6/128)*(6/128). In the more common case with dominant planes, one of the image disparity checks can be ignored, and the factor is just (6/128). If the matches are random and independent (i.e., no common objects between images), then counting arguments can be applied. The distribution of inliers over N trials with probability p of being an inlier is $B_{p,N}$, the binomial distribution. We take the maximum inliers over K RANSAC trials, so the probability of having less than x inliers is $(1 - B_{p,N}(x))^K$. The probability of exactly x inliers over all trials is

$$(1 - B_{p,N}(x))^K - (1 - B_{p,N}(x - 1))^K \quad (5)$$

Figure 5 shows the probabilities for the planar stereo case, based on Equation 5. The graph peaks sharply at 2 inliers (out of 250 matches), showing the theoretic rejection ability of the geometric check. However, the real world has structure, and some keypoints form clusters: these factors violate the independent match assumption. Figure 5 compares actual rejections from the three datasets in the Experiments section, with two different types of keypoints, FAST and STAR. These show longer tails, especially FAST, which has very strong clustering at corners. Note that repetitive structure, which causes false positives for bag-of-words matching, as noted in [11], is rejected by the geometric check – for example, the windows in Figure 8. Even with the long tail, probabilities are very low for larger numbers of inliers, and the rejection filter can be set appropriately.

Of course, there is always the possibility of visual aliasing, e.g., the same large poster in two locations could produce false positives, although we haven't yet found such a case in many hundreds of thousands of matches. In such cases, a good technique would be filters based on positional information.

V. Experiments

As explained in Section II, the view-based system consists of a robust VO detector that estimates incremental poses of a stereo video stream, and a view integrator that finds and adds non-sequential links to the skeleton graph, and optimizes the graph. We carried out a series of tests on stereo data from three different environments, listed below.

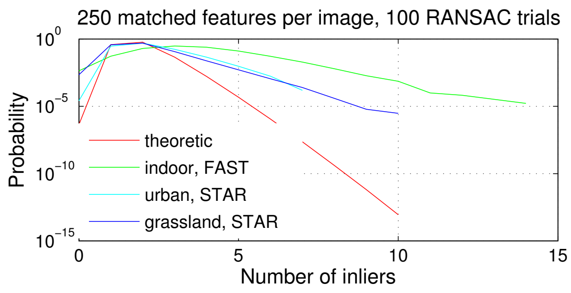


Fig. 5. The probability of getting x inliers from a random unrelated view match. Theoretic probability (see text) compared to three different datasets. Note log scale for probabilities.

Type	length	image res	image rate	stereo base	skeleton views
Office1	0.8 km	640x480	30 Hz	9 cm	4.2k
Office2	0.9 km	640x480	30 Hz	9 cm	5.1k
Urban	0.4 km	768x568	25 Hz	100 cm	0.5k
Terrain	10 km	512x384	15 Hz	50 cm	14.6k

Rectification is not counted in timings; for the indoor sequence it is done in the stereo hardware. VO consumes 11 ms per video frame, leaving 22 ms for view integration, 2/3 of the available time at the fastest frame rate. As in PTAM [26], view integration can be run in parallel with VO, so on a dual-core machine view matching and optimization could consume a whole processor. Given its efficiency, we publish results here for a single processor only. In all experiments, we restrict the number of features per image to $\sim$300, and use 100 RANSAC iterations for geometric matching.

A. *Large Office Loop*

The first experiment is a large office loop of about 800m in length. The trajectory was done by joysticking a robot at around 1m/sec. Figure 1 shows some images: there is substantial blurring during fast turns, sections with almost blank walls, cluttered repetitive texture, and moving people. There are a total of 24k images in the trajectory, with 10k keyframes, 4235 skeleton views, and 21830 edges (Figure 1 shows the first 400m). Most of the edges are added from neighboring nodes along the same trajectory, but a good portion come from loop closures and parallel trajectories (Figure 6).

View matching has clearly captured the major structural aspects of the trajectory, relative to open-loop VO. It closed the large loop from the beginning of the trajectory to the end, as well as two smaller loops in between. We also measured the planarity of the trajectory: for the view-based system, RMS error was 22 cm; for open-loop VO, it was 50 cm.

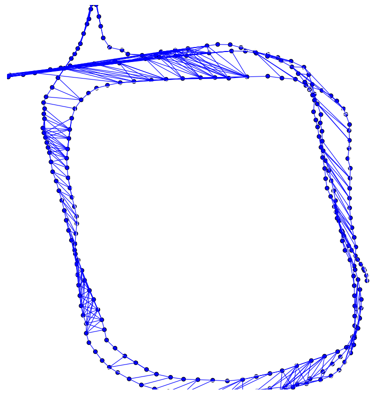


Fig. 6. A closeup from the office dataset showing the matched views on a small loop. The optimizer has been turned off to show the links more clearly.

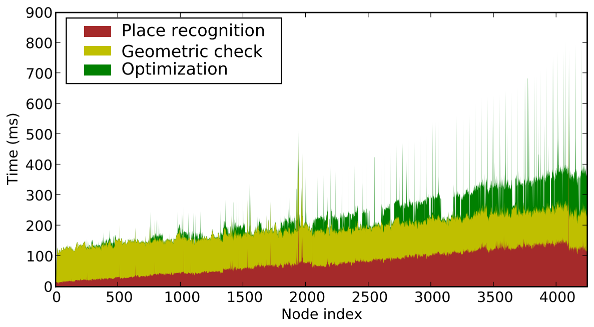


Fig. 7. Timing for view integration per view during the office loop trajectory.

Note that the vocabulary tree prefilter makes no distinction between reference views that are temporally near or far from the current view: all reference views are treated as places to be recognized. By exploiting the power of geometric consistency, there is no need to compute complex covariance gating information for data association, as is typically done for EKF-based systems [13, 14, 41, 46].

The time spent in view integration is broken down by category in Figure 7. The vocab tree prefilter grows linearly, to about 100 ms at the end; the geometry check is constant at 65 ms. Toro does almost no work at the beginning of the trajectory, then grows to average 120 ms at the end, with maximum time of 500 ms. VO can run at frame rates, while simultaneously adding and optimizing skeleton frames at 2 Hz.

B. *Versailles Rond*

We ran viewmap on an outdoor urban sequence from a car in Versailles, a trajectory of about 400m (Figure 8). The skeleton map contained 140 views, and PR found 12 matches after looping around, even when the car moved into an adjacent lane. The Versailles images have a lot of

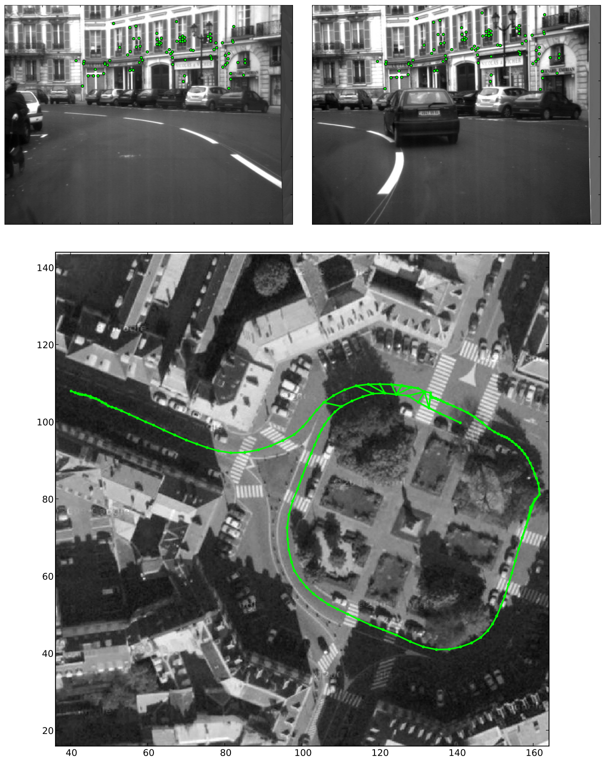


Fig. 8. Versailles Rond sequence of 700 video frames taken from a moving vehicle, 1m baseline, narrow FOV. (Dataset courtesy of Andrew Comport [10]) Top: matched loop closure frames. Bottom: top-down view of trajectory superimposed on satellite image.

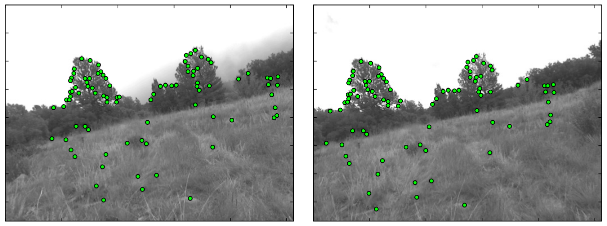


Fig. 9. Matched loop closure frames from the rough-terrain dataset. The match was made between two separate autonomous 5km runs, several hours apart: note the low cloud in the left image.

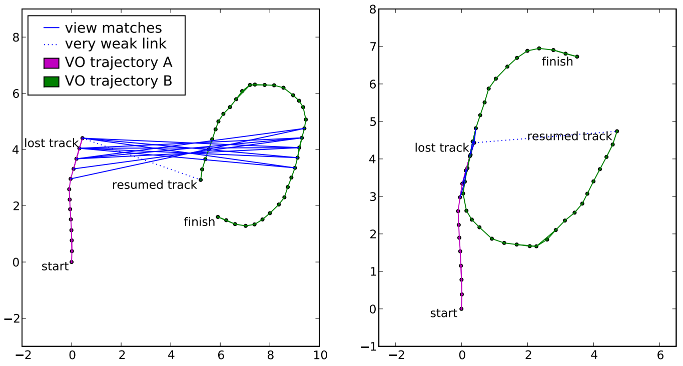


Fig. 10. Relocalization (a.k.a the Kidnapped Robot Problem). There is a cut in the VO process at the last frame in the left trajectory, and the robot is transported 5m. After continuing a short time, a correct view match inserts the new trajectory into the map.

self-similarity in the windows, but the geometric check rejects false positives. This sequence easily runs online.

C. Rough-Terrain Loops

Large off-road trajectories present the hardest challenge for VSLAM. Grass, trees and other natural terrain have self-similar texture and few distinguishing landmarks. The dataset we used was taken by a very aggressive offroad autonomous vehicle, with typical motion of 0.5 m between frames, and sometimes abrupt roll, pitch, and vertical movement. VO fails on about 2% of the frames, mostly because of complete occlusion of one camera; we fill in with IMU data. There are two 5 km trajectories of 30k frames that overlap occasionally. To test the system, we set the skeleton view distance to only 1m. The resultant graph has 14649 nodes and 69545 edges, of which 189 are cross-links between the two trajectories. The trajectories are largely corrected via the crosslinks – the error at the end of the loop changes from over 100m with raw VO to less than 10m. Note that there are no loop closures within each trajectory, only between them. Figure 9 shows such a match. The PR system has the sensitivity to detect close possibilities, and the geometric check eliminates false positives – in Section IV-C we tested 400k random non-matching image pairs from this dataset, and found none with over 10 inliers (Figure 5).

D. Relocalization

Under many conditions, VO can lose its connection to the previous keyframe. If this condition persists (say the camera is covered for a time), then it may move an arbitrary distance before it resumes. The scenario is sometimes referred to as the "kidnapped robot" problem. View-based maps solve this problem with no additional machinery. To illustrate, we took the small loop sequence from the TrajectorySynth experiment, and cut out enough frames to give a 5m jump in the actual position of the robot. Then we started the VO process again, using a very weak link to the previous node so that we could continue using the same skeleton graph. After a few keyframes, the view integration process finds the correct match, and the new trajectory is inserted in the correct place in the growing map (Figure 10). This example clearly indicates the power of constant re-recognition.

E. Incremental Construction

Continuous PR and relocalization allow us to incrementally construct maps using the Anytime Mapping

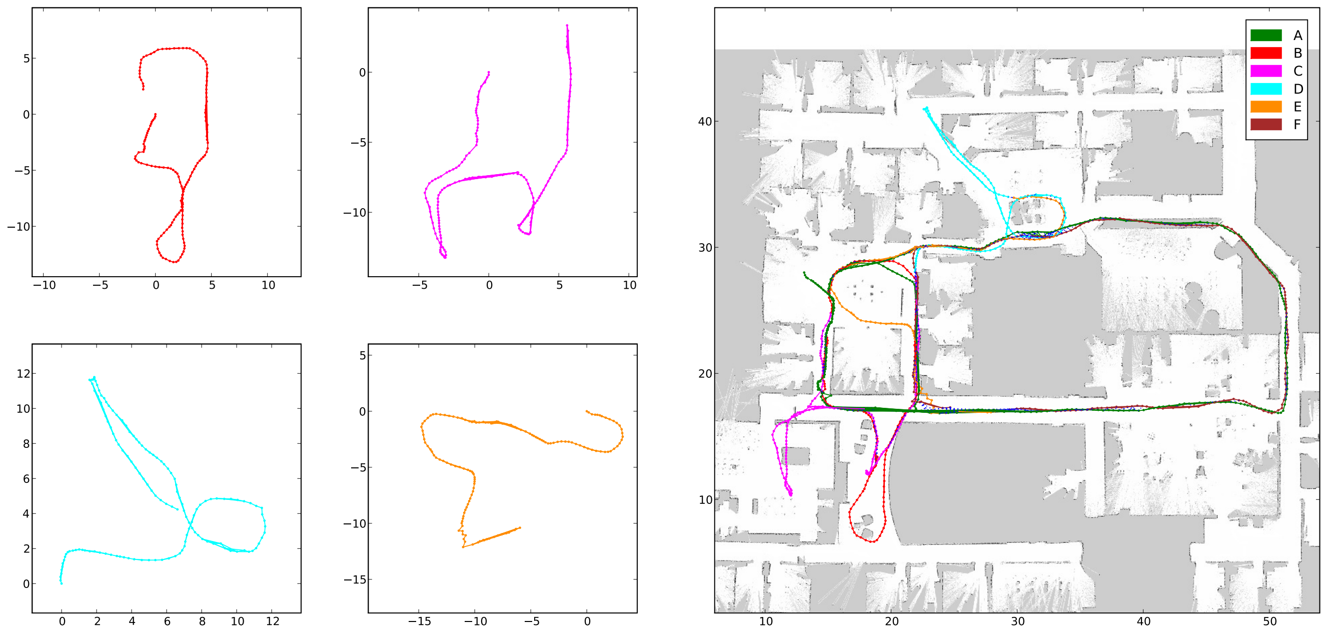


Fig. 11. Trajectories from robot runs through an indoor environment. Left: four typical trajectories shown without correction. Right: the complete map, using multiple trajectories. The map has 1228 views and 3826 connecting links. Distances are in meters.

algorithm of Table I. Over the course of two days, we collected a set of six sequences covering the same large office area as in Figure 1. The sequences were done without regard to forming a full loop or connecting to each other – see the four submaps on the left of Figure 11. There were no VO failures in the sequences, even with lighting changes, narrow corridors, and walls with little texture.

After capturing the sequences, we ran them through the Anytime Mapping algorithm, considering the start of each new sequence to be a "wake-up" event. Each new sequence started with a weak link to the map, and when a PR event took place, the sequence was attached in its proper place, as in the previous subsection. The full map stitching result (right side, Figure 11) shows that PR and optimization melded the maps into a consistent global whole. A detail of the map in Figure 12 illustrates the density of links between sequences, even after several days between sequences.

To show that the map can be constructed incrementally without regard to the ordering of the sequences, we redid the runs with a random ordering of the sequences, producing the same overall map with only minor variation. In some cases, several detached "islands" were grown, where the sequences in each island had no common views. When a sequence with views in both islands was added, they were merged into a common map.

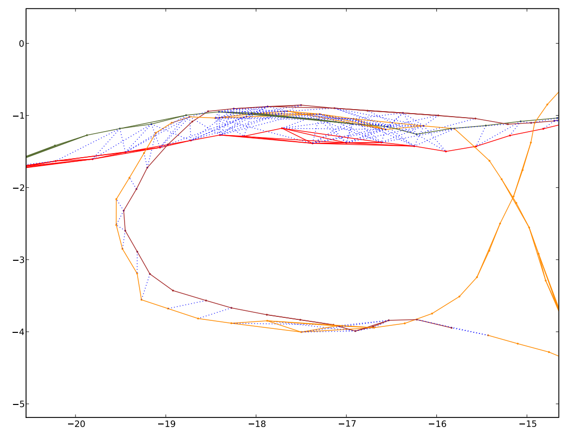


Fig. 12. Detail of a portion of the large map of Figure 11. The cross-links between the different sequences are shown in blue.

F. TrajectorySynth

To showcase the capability of view integration, we performed a reconstruction experiment without any temporal information provided by video sequencing or VO, relying just on view integration. We take a small portion of the office loop, extract 180 keyframes, and push them into the vocabulary tree. We then choose one keyframe as the seed, and use view integration to add all valid view matches to the view skeleton. The seed is marked as used, and one of the keyframes added to the skeleton

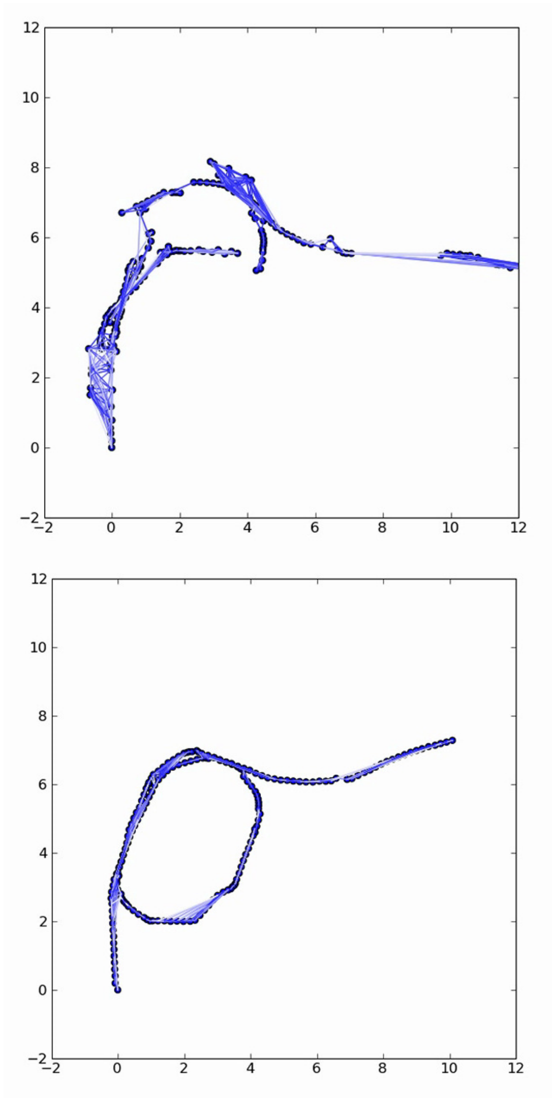


Fig. 13. Trajectory synthesis with no sequencing information: view constraints from PR at left; final optimized map at right.

is chosen as the next seed. The process repeats until all keyframes are marked as used.

The resultant graph is shown in Figure 13, left. The nodes are placed according to the first constraint found; some of these constraints are long-range and weak, and so the graph is distorted. Optimizing using Toro produces the consistent graph on the right. The time per keyframe is 150 ms, so that the whole trajectory is reconstructed in 37 seconds, about 2 times faster than real time. The connection to view stitching [45] is obvious, to the point where we both use the same term "skeleton" for a subset of the views. However, their method is a batch process that uses full bundle adjustment over a reduced set of views, whereas our approximate method retains just pairwise constraints between views.

G. *Accuracy of View-based Maps*

To verify the accuracy of the view-based map, we acquired a sequence of video frames that are individually tagged by "ground truth" 3D locations recorded by the IMPULSE Motion Capture System from PhaseSpace Inc. The trajectory is about 23 m in total length, consisting of 4 horizontal loops with diameters of roughly 1.5 m and elevations from 0 to 1m. There are total of 6k stereo images in the trajectory, with 224 graph nodes, and 360 edges. The RMS error of the nodes was 3.2 cm for the view-based system, which is comparable to the observed error for the mocap system. By contrast, open-loop VO had an error of 14 cm.

It is important to understand the nature of these accuracy results. For *any* VSLAM system that has no access to external reference, accuracy with respect to global ground truth will always degrade as the system moves further from the initial frame. For local areas, we expect accuracy to stay constant, as long as views keep getting re-recognized – hence the results in this small area. For larger regions such as the rough-terrain trajectories of Section V-C, any frame far from the initial one can have significant global error, because small changes in angles can propagate to large changes along a long trajectory. In assessing the accuracy of large outdoor trajectories in [2], we showed that accuracy within local areas stayed constant and was dependent on the density of the skeleton graph, and we refer the reader to those results for a detailed account.

VI. Conclusion

We have presented a complete system for online generation of view-based maps, with an emphasis on *anytime mapping*: incrementally constructing maps whenever new information presents itself. The use of re-recognition, where the robot's position is re-localized at each cycle with no prior information, leads to robust performance, including automatic relocalization and map stitching.

There are some issues that emerged in performing this research that bear further scrutiny. First, SGD optimization takes too long on very large graphs, since its convergence is sublinear. A better strategy is to use a few iterations of SGD, followed by Gauss-Seidel iterations to converge quickly. Second, we would like to investigate the monocular case, where full 6DOF constraints are not present in the skeleton graph. Finally, the use of the strong geometric check bears further investigation: how good is it for differentiating similar environments, e.g., almost-identical hotel rooms? It would also be interesting to compare the prefilter+geometric check to the FAB-MAP technology, both in terms of scalability and performance.

All software used in this paper is open-source, and revised versions are available in the ROS repository: see www.ros.org.

Acknowledgements

We would like to thank the anonymous referees for their comments and suggestions in improving the paper.

References

[1] M. Agrawal and K. Konolige. CenSurE: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, 2008.

[2] M. Agrawal and K. Konolige. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5), October 2008.

[3] Y. Amit and D. Geman. Shape Quantization and Recognition with Randomized Trees. *Neural Computation*, 9(7):1545–1588, 1997.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 10(3):346–359, 2008.

[5] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008.

[6] J. Callmer, K. Granström, J. Nieto, and F. Ramos. Tree of words for visual loop closure detection in urban SLAM. In *Proceedings of the 2008 Australasian Conference on Robotics and Automation*, page 8, 2008.

[7] M. Calonder, V. Lepetit, and P. Fua. Keypoint signatures for fast learning and recognition. In *European Conference on Computer Vision*, 2008.

[8] M. Calonder, V. Lepetit, and P. Fua. Keypoint signatures for fast learning and recognition. In *European Conference on Computer Vision*, Marseille, France, October 2008.

[9] M. Calonder, V. Lepetit, K. Konolige, J. Bowman, P. Mihelich, and P. Fua. Compact signatures for high-speed interest point description and matching. In *International Conference on Computer Vision*, Kyoto, Japan, September 2009.

[10] A. Comport, E. Malis, and P. Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In *International Conference on Robotics and Automation*, 2007.

[11] M. Cummins and P. M. Newman. Probabilistic appearance based navigation and loop closing. In *International Conference on Robotics and Automation*, 2007.

[12] M. Cummins and P. M. Newman. Highly scalable appearance-only SLAM – FAB-MAP 2.0. In *Robotics Science and Systems*, 2009.

[13] A. Davison. Real-time simultaneaous localisation and mapping with a single camera. In *International Conference on Computer Vision*, pages 1403–1410, 2003.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[15] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics Science and Systems*, 2005.

[16] E. Eade and T. Drummond. Monocular SLAM as a graph of coalesced observations. In *International Conference on Computer Vision*, 2007.

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

[18] R. M. Eustice, H. Singh, J. J. Leonard, and M. R. Walter. Visually mapping the RMS Titanic: conservative covariance estimates for SLAM information filters. *Intl. J. Robotics Reserach*, 25(12), 2006.

[19] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *International Conference on Intelligent Robots and Systems*, pages 3872–3877, 2007.

[20] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics Science and Systems*, 2007.

[21] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, California, November 1999.

[22] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, 2008.

[23] H. Jegou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. pages 1–8, 2007.

[24] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *International Conference on Robotics and Automation*, Rome, 2007.

[25] A. Kelly and R. Unnikrishnan. Efficient construction of globally consistent ladar maps using pose network topology and nonlinear programming. In *Proceedings 11th International Symposium of Robotics Research*, 2003.

[26] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[27] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision*, 2008.

[28] M. Klopschitz, C. Zach, A. Irschara, and D. Schmalstieg. Generalized detection and merging of loop closures for video sequences. In *3D Data Processing, Visualization, and Transmission*, 2008.

[29] K. Konolige and M. Agrawal. Frame-frame matching for realtime consistent visual mapping. In *International Conference on Robotics and Automation*, 2007.

[30] K. Konolige, M. Agrawal, and J. Solà. Large scale visual odometry for rough terrain. In *Proc. International Symposium on Research in Robotics (ISRR)*, November 2007.

[31] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Colander, V. Lepetit, and P. Fua. View-based maps. In *Robotics Science and Systems*, 2009.

[32] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, Sept. 2006.

[33] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004.

[34] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[35] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson. Efficient view-based SLAM using visual loop closures. *IEEE Transactions on Robotics*, 24(5):1002–1014, October 2008.

[36] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant time efficient stereo slam system. In *British Machine Vision Conference*, 2009.

[37] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, 2009.

[38] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, recognizing and describing urban spaces with vision and lasers. *Int. J. Rob. Res.*, 28(11-12):1406–1433, 2009.

[39] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[40] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast Keypoint Recognition using Random Ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. Accepted for Publication.

[41] L. Paz, J. Tardós, and J. Neira. Divide and conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5), October 2008.

[42] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, 2006.

[43] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics Science and Systems*, Seattle, USA, 2009.

[44] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *International Conference on Computer Vision*, page 1470, 2003.

[45] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal sets for efficient structure from motion. In *Proc. Computer Vision and Pattern Recognition*, 2008.

[46] J. Solà, M. Devy, A. Monin, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *International Conference on Robotics and Automation*, 2005.

[47] B. Steder, G. Grisetti, C. Stachniss, S. Grzonka, A. Rottmann, and W. Burgard. Learning maps in 3d using attitude and noisy vision sensors. In *International Conference on Intelligent Robots and Systems*, 2007.

[48] S. Thrun and M. Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *Int. J. Rob. Res.*, 25(5-6):403–429, 2006.

[49] R. Unnikrishnan and A. Kelly. A constrained optimization approach to globally consistent mapping. In *International Conference on Intelligent Robots and Systems*, 2002.

[50] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *International Conference on Computer Vision*, 2007.