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 realtime 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 views of the environment that are collected as the robot moves. Connections among the views are formed by consistent geometric matching of their features. The key problem we solve is how to efficiently find and match a new view to the set of views already collected. Our approach uses a vocabulary tree to propose candidate views, and a new compact feature descriptor that makes view matching very fast – essentially, the robot continually re-recognizes where it is. We present experiments showing the utility of the approach on video data, including map building in large environments, map building without localization, and re-localization when lost.**

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, which is called a *skeleton*. Figure 1 gives an example of a skeleton constructed in a small indoor environment. Relations between the views are calculated by matching common features; the overall map is generated by nonlinear optimization of the system [1, 17, 33]. For efficient operation, the critical question is how to match a newly-acquired view to a large database of existing views. The matching system should be robust to changes in viewpoint, lighting, moving objects, and other distractors, and it must be fast enough to run online during image acquisition.

In this paper we present a system that solves the view-matching problem effectively, and can run in small, almost constant time over large view databases. The matching system is feature-based: hundreds of features from the current view are matched against features in a candidate skeleton view. Two views are considered to be matched when a sufficient number of their matched features pass a strict geometric consistency check used by structure-from-motion (SfM) analysis [14]. To limit the number of skeleton views that must be considered, we employ a vocabulary tree [26] to suggest candidate views for matching. The vocabulary tree response is not perfect, and there will be many false positives in the candidate views, which must all undergo feature matching to the current view; this is the biggest computational bottleneck for online processing. The main contributions of this paper are

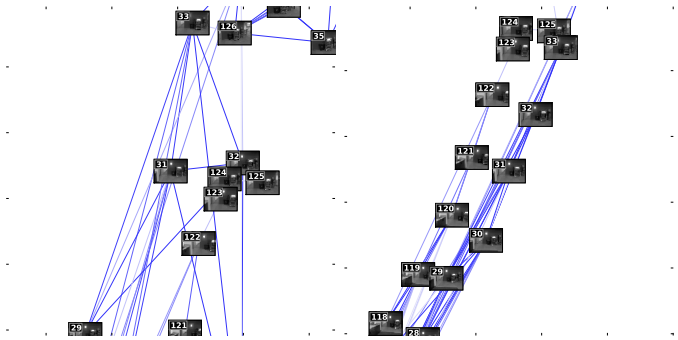

Fig. 1: Map reconstructed from view matching in an indoor environment, with no sequence information. On the left, the graph of view links, where each link encodes the relative position of its two views. On the right, the graph after optimization. Our system is able to integrate a new image against a large map database, with no a priori information about its position, in under 100 ms.

- The development and deployment of a new feature descriptor, based on random tree signatures [5], that is robust to view variation, yet extremely fast to compute and match. As an example, matching 512 features to 512 features takes about 6 ms, less than GPU-enhanced algorithms for other robust descriptors such as SURF [2].
- The integration of a visual vocabulary tree into a complete solution for online place recognition. We call this ability *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).
- A rigorous analysis of the false positive rejection ability of two-view geometry.
- The construction of a realtime system for robust, accurate visual map making over large and small spaces.

In the experiments section, we highlight some of the advantages of a view-based system. The view matching technique,

because of geometric consistency, is robust to object motion, nearly blank walls, and self-similar textures. View maps also scale well: maps with hundreds of views can be constructed and used in real time. Loop closure over large distances is possible; here we show a map with a 400 m trajectory. The same view matching method automatically relocalizes the camera within the existing view graph, recovering from occlusion, motion blur, etc. Finally, view matching with large numbers of points is inherently accurate, showing sub-centimeter precision over a desktop workspace.

Our solution uses stereo cameras for input images. The development of the feature descriptors and 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. Related Work

Visual map-making, or VSLAM, has received a lot of recent attention, starting with Davison's online monoSLAM [9, 10], and now including many variations [29, 32]. These systems all consider sets of 3D points as landmarks, and attempt to maintain a consistent EKF over them. The main limitation here is the filter size, which is only tractable in small (room-size) environments. An exception is [29], which uses a submap technique, although realtime performance has not yet been demonstrated.

In a similar vein, the recent Parallel Tracking and Mapping (PTAM) system [18, 19] 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 tracking phase, leading to accurate and robust performance under many conditions. Still, it is limited to small environments 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).

The skeleton system deployed here comes directly from the work in [1, 21]. Other robotics work that employs similar ideas about constructing view-based constraints is in [33, 34]. 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. To solve the skeleton optimization problem, we use the technique of Grisetti et al. [12], which is an efficient implementation of stochastic gradient descent (SGD). Other relaxation methods for nonlinear constraint systems include [11, 27].

For fast lookup of similar places, we rely on the hierarchical vocabulary trees proposed by Nistér and Stewénius [26]; other methods include approximate nearest neighbor [30] and various methods for improving the response or efficiency of the tree [8, 15, 16]. In particular, Cummins and Newman [8] show how to use visual features for navigation and loop closure over very large trajectories. Our method differs from theirs in using a strong geometric check to do recognition on single views, rather than extended sequences. Callmer et al. [4] 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.

There is an interesting convergence between our work and recent photo stitching in the vision community [31]. They employ a similar skeletonization technique to limit the extent of bundle adjustment calculations, but run in batch mode, with no attempt at realtime behavior. Klopschitz et al. [20] use a vocabulary tree to identify possible matches in video stream, and then followed by a dynamic programming technique to verify a sequence of view matches. They are similar to our work in emphasizing online operation.

The ability to match keypoints across frames seen from potentially very different viewpoints is a key ingredient of establishing relationships between these frames. This requires keypoint descriptors that, such as SIFT [24] and GLOH [25], are robust to viewpoint changes. Faster SIFT-like descriptors such as SURF [2] achieve 3 to 7-fold speed-ups by exploiting the properties of integral images. However, it has recently been shown that even shorter run-times can be obtained without loss in discriminative power by reformulating the matching problem as a classification problem [23, 28]. This approach is not suitable for real-time SLAM applications, since it requires online training of new keypoints [35].

In recent work [5], we observed that if the classifier is trained offline on a randomly-chosen set of keypoints, all other keypoints can be characterized in terms of the response they induce in the classifier, their *signatures*. In this paper, we build on this technique by developing a more compact version of signatures that is extremely efficient and hence suitable for online view matching.

III. FrameSLAM Background

The view map system, which derives from FrameSLAM [1, 21], is most simply explained as a set of nonlinear constraints among camera views, represented as nodes and edges (see Figure 5 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.

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

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

where $\ominus$ is the inverse motion composition operator – in other words, c_j's position in c_i's frame. The covariance expresses

the strength of the constraint, and arises from the geometric matching step that generates the constraint, explained below.

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 [1, 13]. For realtime operation, it is more convenient to run an incremental relaxation step, and the recent work of Grisetti et al. [12] on SGD provides an efficient method of this kind, called Toro, which we use for the experiments.

A. Geometric View Matching

Constraints arise from geometric matching 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.
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

$$J^\top J \delta x = -J^\top \Delta z, \tag{2}$$

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 derives from $J^\top J$, which approximates the curvature at the solution point. As a practical matter, Toro accepts only diagonal covariances, so instead of using $J^\top J$, we scale a simple diagonal covariance based on the inlier response.

In cases where there are too few inliers, 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. Re-detection and Re-recognition

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 [22], because points are re-detected and matched over large areas of the image for each frame. Here we use the FAST detector and SAD matching of small patches around each keypoint as the matching step. 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 5 for sample frames). The PTAM methods of [18], which employ hundreds of points per frame, can also have good performance, with pyramid techniques to determine large motions. However, they are prone to fail when there is significant object motion, since they do not explore the space of geometrically consistent data associations

The focus of this paper is on fast, effective PR. The next section discusses an effective candidate view proposal method, and the geometric consistency check that eliminate false positives. The end result is an extremely fast and reliable PR method that takes on the order of 100 ms to find, match and orient multiple corresponding views over large view datasets. This method relies on no prior information about the current camera view relative to other views, and it does not need to maintain complicated covariance relations among views. It greatly simplifies the task of constructing the skeleton system, and allows it to operate over large spaces.

IV. Matching Views

In this section we describe our approach to achieving efficient view matching over thousands of frames. We start with a new keypoint descriptor that is fast both to compute and to match. Next we develop a filtering technique for matching a new image against a dataset of reference images ($1 \times N$ matching), using a vocabulary tree to suggest candidate views from large datasets. Finally, we develop statistics to verify the rejection capability of the geometric consistency check. In all cases, we use FAST keypoint detectors because they are, well, fast.

A. Compact Randomized Tree Signatures

In Section II we introduced a keypoint descriptor that can be computed fast enough to be useful to demanding real-time problems such as SLAM [5]. The descriptor relies on the fact that if we train a Randomized Tree (RT) classifier to recognize a number of keypoints extracted from an image database, all other keypoints can be characterized in terms of their response to these classification trees. Remarkably, a fairly limited number of base keypoints—500 in our experiments—is sufficient. However, a limitation of this approach is that storing a pre-trained Randomized Tree takes a considerable amount of memory. Here we show that the signatures can be compacted into much denser and smaller vectors, as depicted by Figure 2, resulting in both a large decrease in storage requirement and substantially faster matching.

In [5], signatures are computed as follows. A set of B base keypoints are extracted from a representative image and the RT classifier is trained to recognize them under changes in scale, perspective, and lighting [23]. It consists of a set of N binary RTs T_i, where the binary test at a node is a simple comparison of two random points in a patch $\mathbf{p}$ around the keypoint. At each leaf of a tree T_i, there is a vector of responses for all base keypoints, computed from the training set. Let $\mathbf{t}_i(\mathbf{p})$ be the vector found by dropping the patch $\mathbf{p}$ through the tree T_i to a leaf node. The total response vector of $\mathbf{p}$ is taken to be

$$\mathbf{r}(\mathbf{p}) = \sum_{i=1}^{N} \mathbf{t}_i(\mathbf{p}) \ . \tag{3}$$

The response can be normalized to generate a probability of the patch $\mathbf{p}$ belonging to any member of the base set. Note that for $\mathbf{p}$ belonging to some keypoint that is similar to a base keypoint b, we expect $\mathbf{r}(\mathbf{p})$ to have high values at b's position in the vector.

For any new keypoint k not in the base set, the response $\mathbf{r}(\mathbf{p})$ will have high values at locations corresponding to base keypoints that are similar to k, and low values elsewhere. Thresholding the components of $\mathbf{r}(\mathbf{p})$ therefore results in a sparse vector that we take to be our signature. In practice, we obtain good results using $N = 50$ binary randomized trees of depth 10 and $B = 500$ base points. [5] compared the matching performance of sparse RTs with that of SIFT and found these comparable.

While sparse signatures are fairly efficient to generate and match, it is possible to make them even more so. First, the expensive operation in signature creation is the summation of the $\mathbf{t}_i$, which requires $50 * 500 = 25,000$ floating-point operations per signature. Second, the matching of sparse vectors is slower than desired, because it involves conditional tests. To address these issues, we compress the $\mathbf{t}_i$ responses into smaller vectors, and produce a dense signature. There are several ways to perform the compression, and the tradeoffs involved will be reported in an upcoming paper [6]. Here, we use a simple PCA scheme to extract a dense 176-element vector $\mathbf{t}'_i$ that replaces the 500-element $\mathbf{t}_i$ on each leaf node. As a further reduction, we found that each *element* of both $\mathbf{t}'_i$ and the corresponding signature $\mathbf{r}'(\mathbf{p}) = \sum \mathbf{t}'_i(\mathbf{p})$ could be represented by a single byte, rather than a floating-point number, and that signatures could be compared more quickly using sum of absolute differences. In tests on standard viewpoint matching sets, performance of the dense signatures degrades by about 10 percent relative to the original sparse ones.

As expected, compression to dense, small vectors greatly improves the timing of both descriptor creation and matching. In Table I, dense RTs are compared to the original sparse RTs, and also to the most efficient robust descriptor, U-SURF [2]. SURF matching is done on the smaller length 64 vectors; match times are from [7], creation times are from [2], and do not include keypoint detection. Overall, dense RTs are many times faster, and for matching even beat GPU implementations of U-SURF. Approximate Nearest Neighbor techniques [3] can be used to speed up the process, at some decrease in matching performance; but the overhead in applying them is not worthwhile given the matching speed.

	Descriptor Creation (512 kpts)	N×N Matching (512×512 kpts)
Sparse RTs (CPU)	31.3 ms	27.7 ms
Dense 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 I: Timings for descriptor creation and matching.

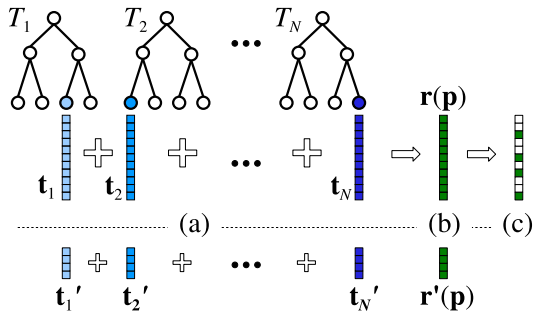

Fig. 2: Illustration of the signature creation process for an arbitrary, new keypoint k. For simplicity we show trees of depth 3; the typical value our implementation uses is 10. (a) The patch $\mathbf{p}_i$ around k is dropped through all trees T_i, $1 \leq i \leq N$, yielding the vectors $\mathbf{t}_i$. (b) All $\mathbf{t}_i$ are summed up yielding $\mathbf{r}(\mathbf{p})$, cf Equation 3. (c) In the upper row of vectors, the $\mathbf{r}(\mathbf{p})$ are thresholded yielding a sparse signal. This step does not occur in the lower row of vectors as they represent some compressed representation $\mathbf{t}'_i$ of the corresponding $\mathbf{t}_i$ that simply need to be summed.

B. Place Recognition

The $1 \times N$ image matching problem has received recent attention in the vision community [16, 26, 30]. We have implemented a place recognition scheme based on the vocabulary trees of Nistér and Stewénius [26] which has good performance for both inserting and retrieving images based on the compact RT descriptors.

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. In our case, we use about 1M training keypoints from 500 images, with $k = 10$, and create a tree of depth 5, resulting in 100K visual words. Nistér and Stewénius have shown that performance improves with the number of words, up to very large (>1M) vocabularies.

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 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 [26, 30].

Various extensions to the bag-of-words approach exemplified by the vocabulary tree are possible. Cummins and Newman [8] use pairwise feature statistics to address the perceptual aliasing problem, especially notable in man-made environments containing repeated structure. Jegou et al. [15] incorporate Hamming embedding and weak geometric consistency constraints into the inverted file to improve performance. We do not use such techniques in this work, relying instead on the strength of the geometric consistency check. Finally, Jegou et al. [16] 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. We simply note that for our scale of application (in the thousands of images), the number of reference images we must score is effectively a small constant.

To test the effectiveness of the vocabulary tree as a prefilter, we constructed a test set of some 180 keyframes over a 20m trajectory, and determined ground truth matches by performing geometric matching across all 180×180 possibilities. 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 3, top) 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 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 3, bottom).

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 around 50% of the correct matches to appear in the top 15 results.

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. 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

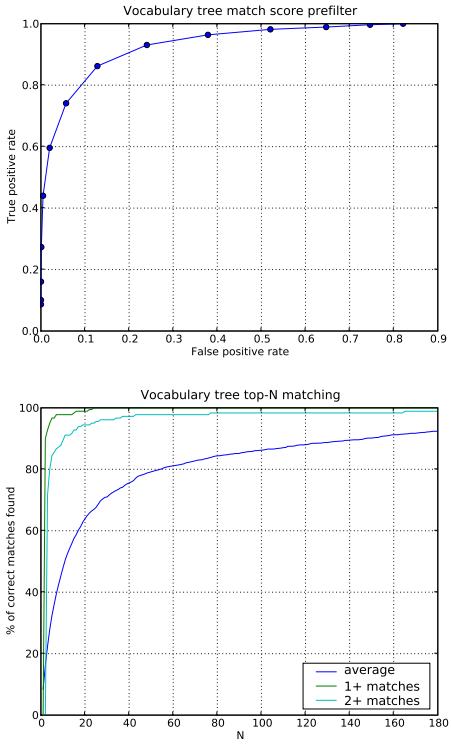

Fig. 3: Top: ROC curve for the vocabulary tree prefilter on the test dataset. Bottom: "Average" curve shows percentage of the correct matches among the top N results from the vocabulary tree (blue); other curves are the percentage of views with at least 1 or 2 matches in the top N.

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 4 shows the theoretic probabilities for the planar stereo case, based on Equation 5. The graph peaks sharply at 2 inliers (out of 250 matches), showing the rejection ability of the geometric check. Actual values were computed for the indoor dataset, using 200 images with Harris keypoints from each of two disjoint sets. The actual values are less peaked and average just under 3 inliers – the real world has structure that violates

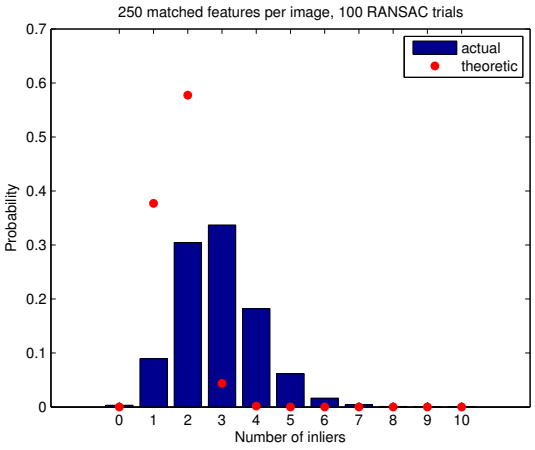

Fig. 4: The probability of getting x inliers from a random unrelated view match, based on 250 keypoint matches per image and 100 RANSAC steps, with a keypoint match probability of 0.00059.

the random match assumption. The key part is the tail: there are no actual matches with greater than 10 inliers.

V. Experiments

As explained in Section III, 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 video data captured at 30 Hz from a stereo camera at 640x480, with a 9 cm baseline and a 90 degree FOV. Rectification is done on the stereo head; VO consumes 11/33 ms per video frame, leaving 22/33 ms for view integration, 2/3 of the available time. As in PTAM [18], 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.

The goal of the system is to integrate as many views as possible, while giving priority to VO in processing the video stream. The view integration cycle takes the latest keyframe produced by VO, runs the vocabulary tree prefilter to determine likely match candidates, performs geometric consistency checking against the candidates, and then runs Toro to optimize the skeleton. With the exception of Toro, all of these components take constant time (the vocabulary tree prefilter is essentially constant up to very large numbers of views). Since Toro can run incrementally, we limit the amount of time it takes by stopping iterations when the error delta is small, or the number of iterations exceeds a threshold. In cases where the error is growing, we then limit the addition of new keyframes to the skeleton graph, until the error comes down.

Skeleton graph density is controlled by view integration. When it has finished matching and optimizing its current skeleton node, it checks if the most recent keyframe is far enough in angle or distance (typically 10 degrees or 0.5 m) from the previous keyframe. 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. Typically the graph contains about 1/2 of the keyframes produced by VO. 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. We have not explored these strategies yet.

A. Large Office Loop

The first experiment is a large office loop of about 400m in length. The trajectory was done by joysticking a robot at around 1m/sec. Figure 5 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 12K images in the trajectory, with 1540 keyframes, 628 graph nodes, and 1275 edges. Most of the edges are added from neighboring nodes along the same temporal path, but a good portion come from loop closures and parallel trajectories (Figure 5, bottom right).

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, which is a good measure of the accuracy of the technique: for the view-based system, RMS error was 22 cm; for open-loop VO, it was 50 cm.

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 [9, 10, 29, 32].

The time spent in view integration is broken down by category in Figure 6. Averages for for adding to and searching the vocabulary tree are 25 ms, and for the geometry check, 65 ms. Toro does almost no work at the beginning of the trajectory, then grows to over 1000 ms by the end. The big jump comes when the large loop is closed, which creates a long optimization loop in Toro. At this point, optimization starts to limit the number of new keyframes coming in to the graph, and the distance between nodes stretches to about 1m. On other parts of the trajectory, view integration can run at full speed.

B. 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.

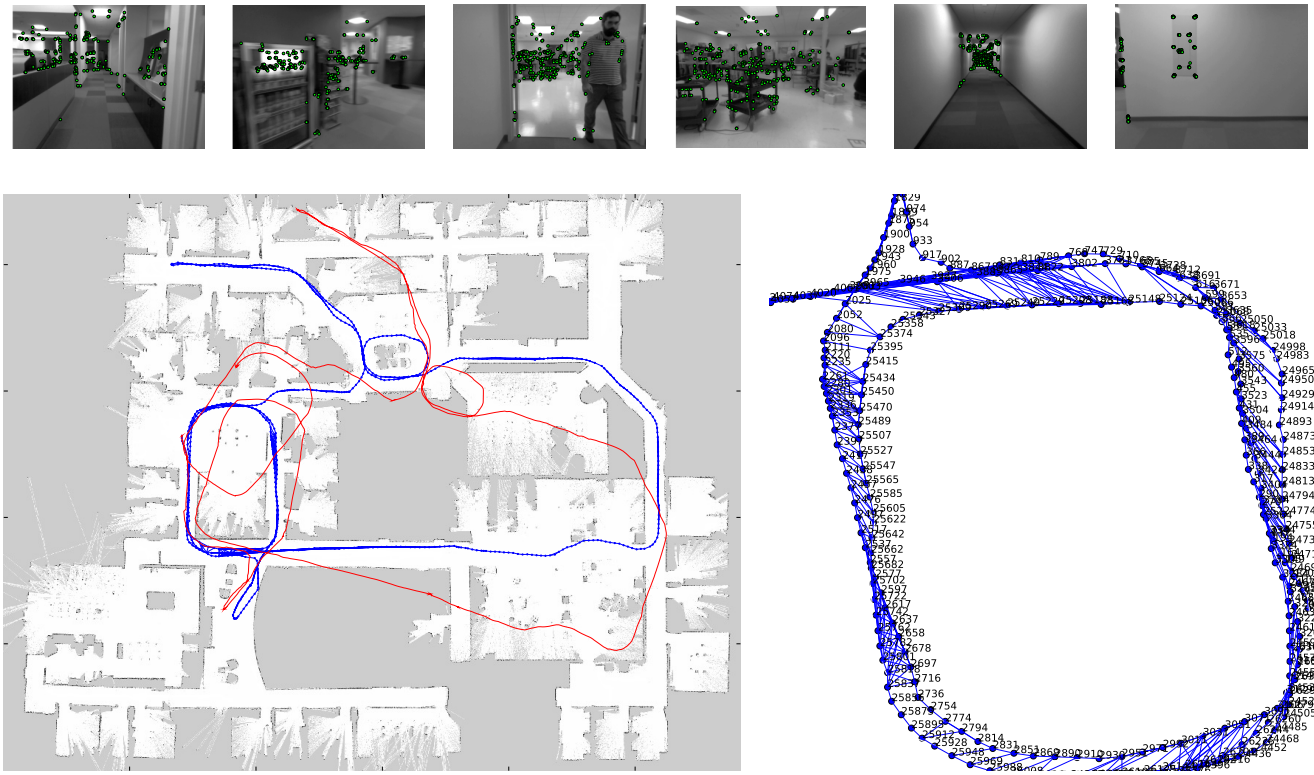

Fig. 5: Top: representative scenes from the large office loop, showing matched features in green. Note blurring, people, cluttered texture, nearly blank walls. Bottom: resultant skeleton graph (in blue) of 628 nodes and 1275 edges, overlaid on a laser map of the building. For comparison the VO trajectory without view match correction is shown in red. On the right is a closeup showing the matched views on a small loop. The optimizer has been turned off to show the links more clearly.

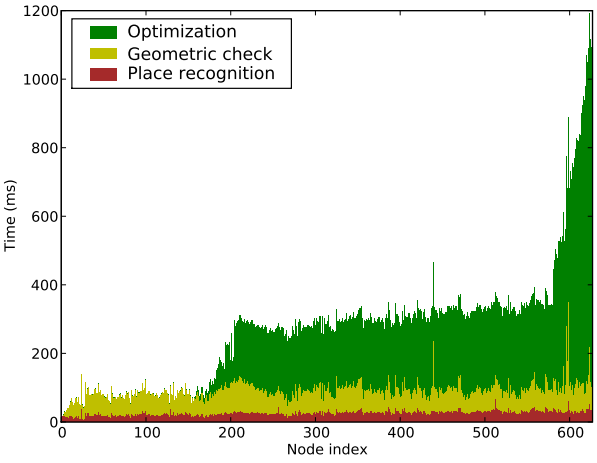

Fig. 6: Timing for view integration per view during the office loop trajectory. Toro dominates the latter part of the run.

The seed is marked as used, and one of the keyframes added to the skeleton is chosen as the next seed. The process repeats until all keyframes are marked as used.

The resultant graph is shown in Figure 1 (first page), 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 realtime. The connection to view stitching [31] 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.

C. 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 7). This example clearly indicates the power of constant re-recognition.

D. 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

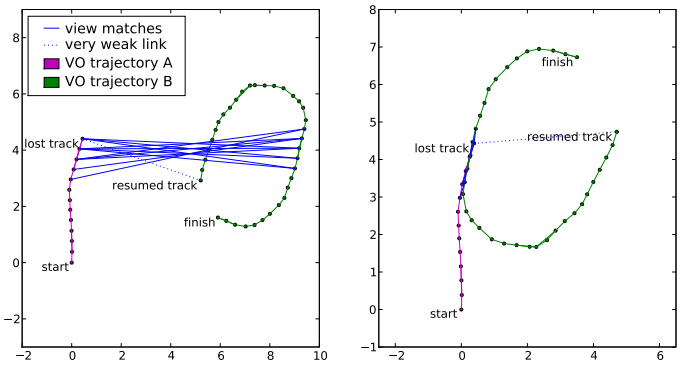

Fig. 7: Kidnapped robot problem. There is a cut in the VO process at the last frame in the left trajectory, and the robot is transported 5m. After continuing a short time, a correct view match inserts the new trajectory into the map.

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.

VI. Conclusion

We have presented a complete system for online generation of view-based maps. 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, the time taken by SGD optimization will not be acceptable for graphs with more than a few thousand edges, and better methods, perhaps hierarchical, should be found. Second, we would like to investigate the monocular case, where full 6DOF constraints are not present in the skeleton graph.

References

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

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *CVIU*, 110(3):346–359, 2008.

[3] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1000, 1997.

[4] 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.

[5] M. Calonder, V. Lepetit, and P. Fua. Keypoint signatures for fast learning and recognition. In *ECCV*, 2008.

[6] M. Calonder, V. Lepetit, K. Konolige, P. Mihelich, and P. Fua. High-speed keypoint description and matching using dense signatures. In *To be submitted*, 2009.

[7] A. Chariot and R. Keriven. GPU-boosted online image matching. In *ICPR*, 2008.

[8] M. Cummins and P. M. Newman. Probabilistic appearance based navigation and loop closing. In *ICRA*, 2007.

[9] A. Davison. Real-time simultaneaous localisation and mapping with a single camera. In *ICCV*, pages 1403–1410, 2003.

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE PAMI*, 29(6), 2007.

[11] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

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

[13] 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.

[14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[15] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[16] H. Jegou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[17] 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.

[18] 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.

[19] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *ECCV*, 2008.

[20] M. Klopschitz, C. Zach, A. Irschara, and D. Schmalstieg. Generalized detection and merging of loop closures for video sequences. In *3DPVT*, 2008.

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

[22] 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.

[23] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE PAMI*, 28(9):1465–1479, Sept. 2006.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[25] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *PAMI*, 27(10):1615–1630, 2004.

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

[27] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor estimates. In *In ICRA*, 2006.

[28] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *In Proc. IEEE Conference on Computing Vision and Pattern Recognition*, 2007.

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

[30] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *Computer Vision, IEEE International Conference on*, 2:1470, 2003.

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

[32] J. Solà, M. Devy, A. Monin, and T. Lemaire. Undelayed initialization in bearing only slam. In *ICRA*, 2005.

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

[34] R. Unnikrishnan and A. Kelly. A constrained optimization approach to globally consistent mapping. In *Proceedings International Conference on Robotics and Systems (IROS)*, 2002.
[35] B. Williams, G. Klein, and I. Reid. Real-time slam relocalisation. In *ICCV*, 2007.