The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Towards Lifelong Visual Maps

Kurt Konolige and James Bowman
Willow Garage
Menlo Park, CA
konolige,jamesb@willowgarage.com

Abstract— **The typical SLAM mapping system assumes a static environment and constructs a map that is then used without regard for ongoing changes. Most SLAM systems, such as FastSLAM, also require a single connected run to create a map. In this paper we present a system of visual mapping, using only input from a stereo camera, that continually updates an optimized metric map in large indoor spaces with movable objects: people, furniture, partitions, etc. The system can be stopped and restarted at arbitrary disconnected points, is robust to occlusion and localization failures, and efficiently maintains alternative views of a dynamic environment. It operates completely online at a 30 Hz frame rate.**

I. INTRODUCTION

A mobile robot existing in a space over a period of time has to deal with a changing environment. Some of these changes are ephemeral and can be filtered, such as moving people. Others are more permanent: objects like furniture are moved, posters change, doors open and close, and more infrequently, walls are torn down or built. The typical SLAM mapping system assumes a static environment and constructs a map, usually in a single continuous run. This map is then enshrined as the ground truth, and used without regard for ongoing changes, with the hope that a robust localization filter will be sufficient for navigation and other tasks.

In this paper we propose to end these limitations by focusing on the idea of a *lifelong map*. A lifelong map system must deal with at least three phenomena:

1) **Incremental mapping.** The system should be able to add new sections on to its map at any point, that is, it should be continuously localize *and* mapping. It should be able to wake up anywhere, even outside the current map, and connect itself to the map when it is encountered. It should continually check for loop closures and optimize them. It should work online.
2) **Dynamic environment.** When the world changes, the system should repair its map to reflect the changes. The system should maintain a balance between remembering past environments (to deal with short-term occlusions) and efficient map storage.
3) **Localization and odometry failure.** Typically a robot will fail to localize if its sensors are blocked or degraded in some way. The system should recover from these errors by relocalizing in the map when it gets the chance.

These principles have been explored independently in many research papers (see Section II on related work). But they have not yet been articulated as a coherent set of rules for a practical robotic system to exist in a dynamic environment. No current mapping and localization system adheres to all of them, and it is not obvious that combining current techniques would lead to a consistent realtime system.

Lifelong mapping as a concept is independent of the sensor suite. But just as laser sensors helped to solve a static SLAM problem that was difficult for sonars, so new techniques in visual place recognition (PR) can help with the difficult parts of lifelong mapping: loop closure and robust relocalization. Visual sensors have much more data, and are better at distinguishing scenes from a single snapshot – 2D laser scans are more ambiguous, making loop closure and relocalization a harder task.

The high information content in individual views also allows visual maps to more easily encode conflicting information that arises from dynamic environments. For example, two snapshots of a doorway, one with it open and one closed, could exist in a visual map. These views can be matched independently to a current view, and the best one chosen. In contrast, laser maps use a set of scans in a local area to construct a map, making them harder to dynamically update or represent alternative scenes.

In this paper we build on our recent work in online visual mapping using PR [19], and extend it to include incremental map stitching and repair, relocalization, and view deletion, which is critical to maintaining the efficiency of the map. The paper presents two main contributions: first, an overall system for performing lifelong mapping that satisfies the above criteria; and second, a view deletion model that maximizes the ability of the map to recognize situations that have occurred in the past and are likely to occur again. This model is based on clustering similar views, keeping exemplars of past clusters while allowing new information to be added. Experiments demonstrate that the lifelong map system is able to cope with map construction over many separate runs at different times, recovers from occlusion and other localization errors, and efficiently incorporates changes in an online manner.

II. Related Work

Work on mapping dynamic environments with laser-based systems has concentrated on two areas: ephemeral objects such as moving people that are distractors to an otherwise static map, and longer-term but less frequent changes to geometry such as displaced furniture. The former are dealt with

978-1-4244-3804-4/09/$25.00 ©2009 IEEE

using probabilistic filters to identify range measurements not consistent with a static model [11], [15], or by tracking moving objects [20], [29]. With visual maps, the problem of ephemeral objects is largely bypassed with geometric consistency of view matching, which rejects independently-moving objects [1].

Dealing with longer-term changes is a difficult problem, and relatively little work has been done. Burgard et al. [5] learn distinct configurations of laser-generated local maps – for example, with a door open and closed. They use fuzzy k-means to cluster similar configurations of local occupancy grids. They extend particle-filter localization to include the configuration that is most consistent with the current laser scan. Our approach is similar in spirit, but clustering is done by a view matching measure in a small neighborhood, and is on a much finer scale. We also concentrate on efficiency in keeping as few views as possible to represent different scenes.

Another approach with similarities to ours is the long-term mapping work of Biber and Duckett [4]. They sample local laser maps at different time scales, and merge the samples in each set to create maps with short and long-term elements. The best map for localization is chosen by its consistency with current readings. They have shown improved localization using their updated maps over time scales of weeks.

Both these methods rely on good localization to maintain their dynamic maps. They create an initial static map, and cannot tolerate localization failures: there is no way to incorporate large new sections into the map. In contrast, our method explicitly deals with localization failure, incremental map additions, and large map changes.

A related and robust area of research is topological visual mapping, usually using omnidirectional cameras ([26], [27], [12] among many). As with laser mapping, there have been few attempts to deal with changing environments. Dayoub and Duckett [9] develop a system that gradually moves stable appearance features to a long-term memory, which adapts over time to a changing environment. They choose to adapt the set of features from many views, which is a visual analog to the Biber and Ducket time-scale approach; they report increased localization performance over a static map. Andreasson et al. [3] also provide a robust method for global place recognition in scenes subject to change over long periods of time, but without modifying the initial views. Both these methods assume the map positions are known, while we continuously build and optimize a metric map of views.

For place recognition, we rely on the hierarchical vocabulary trees proposed by Nistér and Stewénius [21]; other methods include approximate nearest neighbor [23] and various methods for improving the response or efficiency of the tree [8], [16], [17]. Callmer et al. [6], Eade and Drummond [10], Williams et al. [28], and Fraundorfer et al. [12] all use vocabulary tree methods to perform fast place recognition and close loops or recover from localization failures.

III. Background: View-Based Maps

A. FrameSLAM and Skeleton Graphs

The view map system, which derives from our work on FrameSLAM [2], [18], [19], is most simply explained as a set of nonlinear constraints among camera views, represented as nodes and edges (see Figures 6 and 7 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 [22] and SAD 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], [14]. For realtime operation, it is more convenient to run an incremental relaxation step, and the recent work of Grisetti et al. [13] 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.

B. Deleting Views

View deletion is the process of removing a view from the skeleton graph, while preserving connections in the graph. We show this process here for a simple chain. Let c_0, c_1 and c_2 be three views, with constraints c_{01} and c_{12}. We can construct a constraint c_{02} that represents the relative pose and covariance between c_0 and c_2. The construction is:

$$\Delta z_{02} = \Delta z_{01} \oplus \Delta z_{12}, \tag{2}$$
$$\Gamma_{02}^{-1} = J_1 \Gamma_{01}^{-1} J_1^T + J_2 \Gamma_{12}^{-1} J_2^T. \tag{3}$$

J_1 and J_2 are the jacobians of the transformation z_{02} with respect to c_1 and c_2, respectively. The pose difference is constructed by compounding the two intermediate pose differences, and the covariances Γ are rotated and summed appropriately via the Jacobians (see [24]).

Under the assumption of independence and linearity, this formula is exact, and the node c_1 can be deleted if it is only desired to retain the relation between c_0 and c_2; otherwise it is approximately correct when c_0 and c_2 are separated by Δz_{02}. We use view deletion extensively in Section V to get rid of unnecessary views and keep the graph small.

C. Place Recognition

The place recognition problem (matching one image to a database of images) has received recent attention in the vision community [17], [21], [23]. We have implemented a place recognition scheme based on the vocabulary trees of Nistér and Stewénius [21] which has good performance for both inserting and retrieving images. Features are described with a compact version of random-tree signatures [7], [19], which are efficient to compute and match, and have good performance under view change.

For a given reference image, the vocabulary tree generates an ordered set of matches. We pick the top N candidates (where N is ~ 15) and subject them to a geometric consistency check. In previous work [19], we found experimentally and theoretically that a match count of 30 features or greater produces no false positives.[1] We also found, in test cases, that the vocabulary tree produced at least one good match in the top 15 candidates for 97% of matchable reference views. For any given reference view, we expect almost 60% of the correct matches to appear in the top 15 results. Figure 1 shows the recognition rate for an actual map (Figure 6), 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.

IV. Lifelong Visual Mapping

In the introduction, we presented the three principles of lifelong mapping, which can be summarized as:

- Continual, incremental mapping and loop closing.
- Map repair to reflect a changing environment.
- Recovery from localization failure.

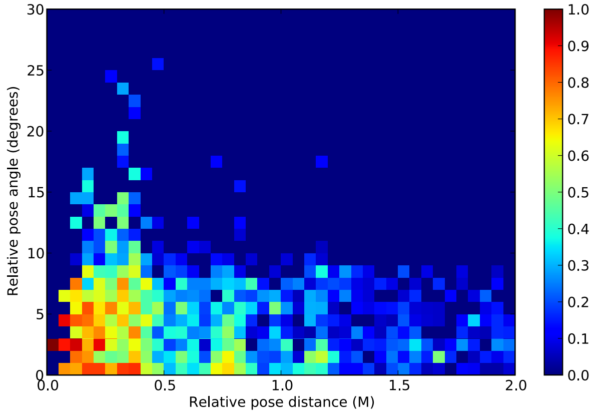

Fig. 1. Recognition rate. The plot shows the proportion of recognized poses for varying pose angle and pose distance. The poses are taken from the final map in Figure 6.

Here we show how a lifelong map can be created and maintained using the techniques of the view map skeleton. The most interesting and difficult part is deciding when to delete views from the map.

A. System-Level Algorithm

We assume that the robot stores a skeleton map M that represents its current global map. Every time the robot wakes up, it runs the following algorithm for visual mapping.

```
Lifelong 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) Check for view deletion using $K_c$
                D) Incrementally optimize $M$
  6) If not shut down, continue from step (2)
```

This algorithm uses techniques from Section III to maintain the global view map; the one addition is view deletion in line 5C, which is explained further below. In general form,

[1] 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. Filters based on positional information could be used in these cases.

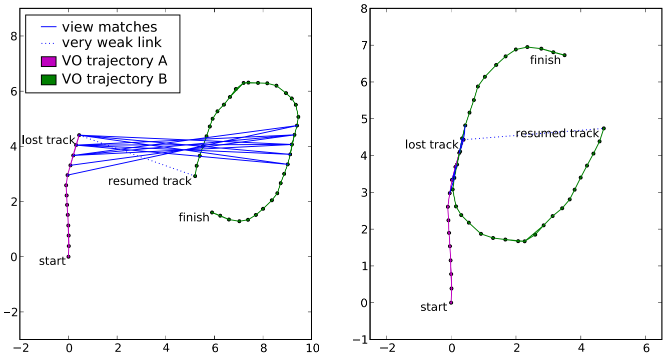

Fig. 2. Sequence stitching. The robot traverses the first sequence (red), then is transported 5m and restarted (green). After continuing a short time, a correct view match inserts the new trajectory into the map.

the algorithm is very simple. Waking up, the robot is lost, and inserts a weak link to keep the map connected. Then it processes stereo frames at 30 Hz, using VO to connect each frame to the last. If there is a failure, 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). On keyframe addition, a further check is made on whether to add a skeleton view, which is the same test as for keypoint switching, with a further constraint of at least 15 frames (0.5 seconds) between skeleton views. 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. The view deletion algorithm is run on K_c (see Section V) to thin out crowded skeleton views. Finally, the graph is incrementally optimized, and the process repeats until the robot shuts down.

B. Map Stitching

Loop closure, recovery from VO failure, and global localization on wakeup are handled by a uniform place-recognition mechanism. To illustrate, consider two small sequences that overlap somewhere along their length. These can represent any of the three scenarios above. In Figure 2, the first sequence in red ends in a VO failure or robot shutdown. The second sequence (green) starts some 5m away, without any *a priori* knowledge of its relation to the first. The weak link (dotted line) connects the last view of the red sequence with the first view of the green sequence, to maintain a connected graph. After traveling along the green sequence, PR makes matches to the first sequence. Optimization then brings the sequences into correct alignment, and the weak link can be deleted.

C. Computation and Storage

The Lifelong Mapping algorithm can be run online using a single processor core. The time spent in view integration is broken down by category in Figure 3. VO takes 11 ms average per frame; there are a maximum of two skeletons views added per second, leaving at least 330 ms to process

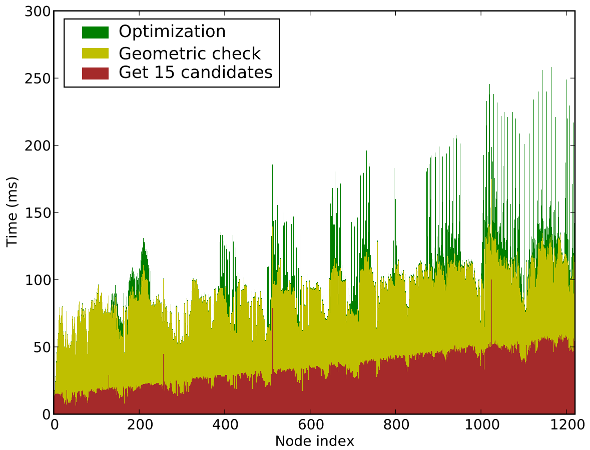

Fig. 3. Timing for view integration per view during integration of the last sequence of Figure 6.

each one. Averages for for adding to and searching the vocabulary tree are 25 ms, and for the geometry check, 65 ms. Optimization by Toro in incremental mode uses less than 10 ms per view.

Storage for each skeleton view consumes 60KB (average 300 features at 200 bytes for the descriptor); we can easily accommodate 50K views in memory (3GB). With a 50m x 50m building, assume that the robot's trajectories are spread over, say, 33% of the building's area; then the maximum density of views in the map is 50 per square meter, more than adequate for good recognition. [2]

V. Forgetting Views

A robot running continuously in a closed environment will eventually accumulate enough images in its graph to stress its computational abilities. Pruning the graph is essential to long-term viability. The question is how to prune it in a "reasonable" way, that is, what criteria guide the deletion of views? If the environment were static, a reasonable choice is reconstruction quality, which was successfully developed in the Photo Tourism project [25]. With a dynamic map, we want to maximize the chance that a given view in the map will be used in recognizing the pose of the robot via a new view.

A. Visual Environment Types

Consider a stereo camera at a fixed position capturing images at intervals throughout a period of time. Figure 4 shows the matching evolution for different persistence classes. In a completely static scene, a view will continue to match all view that come after it. In a static scene with some small changes (e.g., dishes or chairs being moved), there will be a slow degradation of the matching score over time, stabilizing to the score for features in the static areas. If there is a large, abrupt change (e.g., a large poster moved), then there is a large falloff in the matching score. There are

[2] We are experimenting with just saving the vocabulary word index for each feature as in [12], which would increase the limit to 500K views, and a density of 500 per square meter.

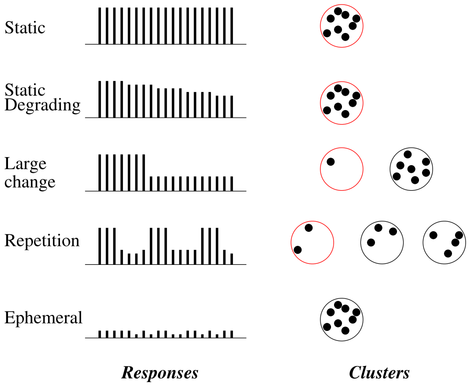

Fig. 4. View environments. On the left, a schematic of view responses over time. Every bar represents how well a view matches subsequent views at the same position for the given environment type. On the right, the clusters that are induced by the environment (see Section V-D). The red circled cluster is from the original view, which does not survive in the Ephemeral case.

changes that repeat, like a door opening and closing, that lead to occasional spikes in the response graph. Finally, a common occurrence is an ephemeral view, caused by almost complete occlusion by a dynamic object such as a person – this view matches nothing subsequently. An environment could have a mixture of these types, for example, occlusion can occur with any of them.

Our main idea is similar to the environment-learning strategy of [5], which attempts to learn distinct configurations of laser-generated local maps – for example, with a door open and closed. In our case, we want to learn clusters of views that represent a similar and persistent visual environment.

B. *View Clustering*

To cluster views based on their recognition environment, the obvious score is the inlier match percentage. Let v and v' be two views, m the minimum count of their two feature sets, and $\widetilde{m}$ the number of inliers in their match. The closeness of the two views is defined as

$$c(v, v') \equiv \frac{m}{\widetilde{m}} - 1. \qquad (4)$$

The closeness measure is zero if two views match exactly, and increases to infinity when there are no matching features. Note that closeness is not a distance measure, since it does not obey the triangle inequality (informally, two images may be close to a third image, but not close to each other).

The closeness measure defines the graph of the skeleton set, where an edge between two views exists if $c(v, v') < \tau$ for some match threshold τ. For a set S of views, a *cluster* of S is any maximal connected subset of S.

In deleting a view v from a cluster S, it is important to retain the connectedness of edges coming into a cluster. If there is an edge (v, a) from an external node a that has no other link to the cluster, then a new link is formed from a to a node v' of the cluster that is connected to v, using the technique of Section III-B. If the cluster is a singleton, then all pairs of nodes a, b linked to v are connected.

C. *Metric Neighborhood*

While running, the robot's views in the skeleton will seldom fall on exactly the same pose. Thus, to cluster views, we need to define a *view neighborhood* η_d, η_ϕ over which to delete redundant views. The size of the neighborhood is dependent on the scale of the application and the statistics of the match data. In large outdoor maps it may be sufficient to have a view every 10m or so; for close-up reconstruction of a desktop we may want views every 10cm. For the typical indoor environments of this paper, the data of Figure 1 suggest that $\eta_d = 0.5m$ is a reasonable neighborhood, and we use this in the experiments of Section VI.

The view angle η_ϕ is also important in defining the neighborhood, since matching views must have an overlapping FOV. Again, the data suggest a neighborhood value of $\eta_\phi = 10^o$.

The skeleton graph induces a global 3D metric on its nodes, but the metric is not necessarily globally accurate: as the graph distance between nodes increases, so does their relative uncertainty. The best way to see this is to note that a large loop could cause two nodes at the beginning and end of the loop to be near each other in the graph global pose, but not in ground truth, if the uncertainty of the links connecting the nodes is high. So, we search just the local graph around a view to find views that are within its neighborhood.

D. *An LRU Algorithm*

Within a metric neighborhood consisting of a set S of views, we want to do the following:

- limit the number of views to a maximum of κ;
- preserve diversity of views;
- preferentially remove older, unmatched views.

The main idea of the following view deletion algorithm is to use clusters of S to indicate redundant views, and try to preserve single exemplars from as large a number of clusters as possible, using a least-recently used (LRU) algorithm.

```
LRU View Deletion
Input: set of $n \leq \kappa$ views $v_i$ in a neighborhood, a new view $v$, a view limit $\kappa$, and a view match threshold $\tau$.
Output: updated set of views of size $\leq \kappa$.

  • Add $v$ to the graph
  • If $c(v, v_i) > \tau$ for all $v_i$ (no match), set the timestamp of $v$ to -1 (oldest timestamp)
  • While $n > \kappa$ do
      – If any cluster has more than one exemplar, delete the oldest exemplar among these clusters
      – Else delete the oldest exemplar
```

E. *Analysis*

This algorithm obviously preserves the sparsity of views in a neighborhood, keeping it at or below κ. To preserve exemplar diversity, the algorithm will keep adding views until κ is reached. Thereafter, it will add new (unmatched) views at the expense of thinning out all the clusters, until some cluster must be deleted. Then it chooses the oldest

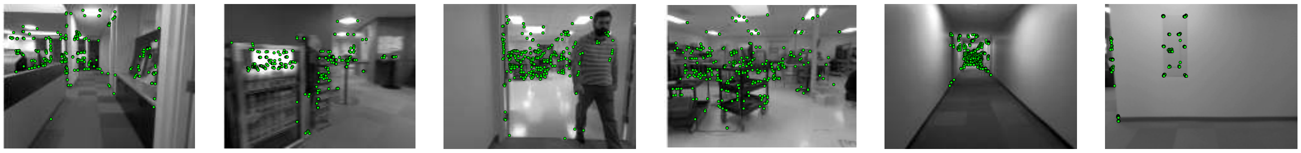

Fig. 5. Representative scenes from the large office loop, showing matched features in green. Note blurring, people, cluttered texture, nearly blank walls.

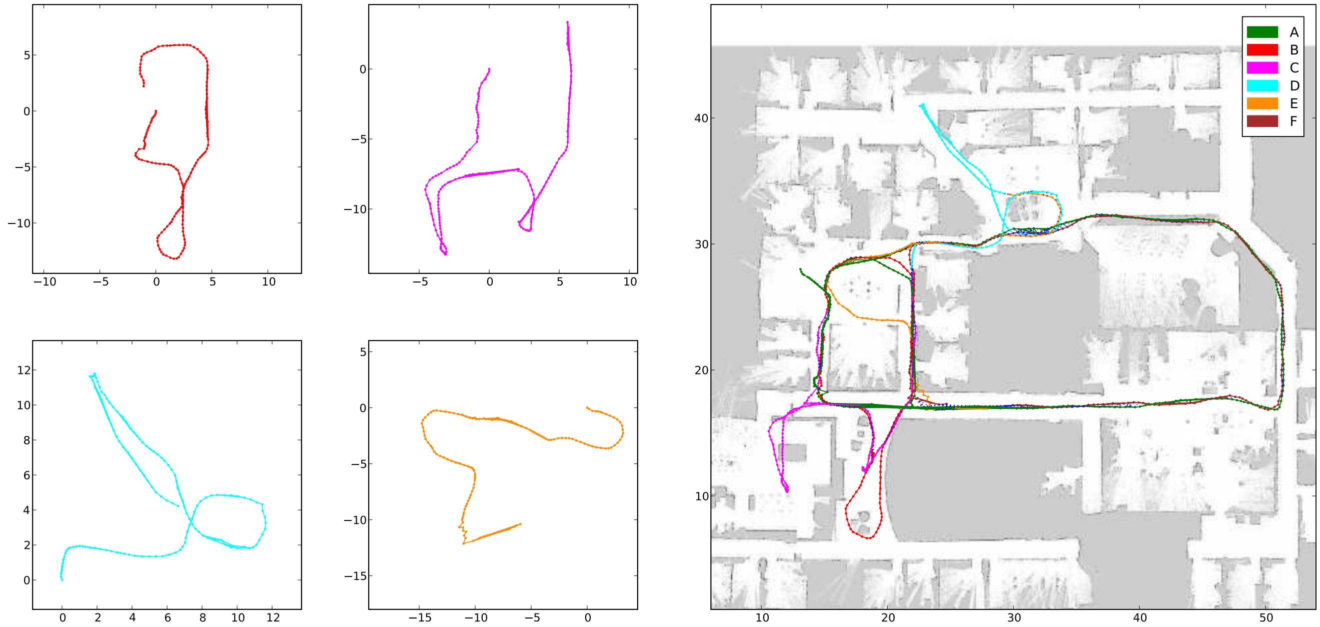

Fig. 6. Trajectories from robot runs through an indoor environment. Left: four typical trajectories shown without correction. Right: the complete lifelong map, using multiple trajectories. The map has 1228 views and 3826 connecting links. Distances are in meters.

cluster. Figure 4 shows examples of cluster evolution for the different environments. In the case of static environments, there is just a single cluster, populated by the most recent views. When there is a large change, a new cluster will grow to have all the exemplars but one – should the change be reversed as in the repetition scene, that cluster will again grow. Notice that the bias towards newer views reduces older cluster to single exemplars in the long term.

One of the interesting aspects of the view deletion algorithm is how it deals with new ephemeral views. Such a view v starts a new cluster with the oldest timestamp. The next view to be added will not match v; assuming the neighborhood is full, and all clusters are singletons, v will be deleted. Only if the cluster is confirmed with the next addition matching will it survive.

In the long term, the equilibrium configuration of any neighborhood is a set of singleton clusters, representing the most recent κ stable environments with the most recent exemplars for each environment. This is the most desired outcome for any algorithm.

Our clustering algorithm has similarities to the fuzzy-means clustering done by Burgard et al. [5]. For their 2D laser maps, a local occupancy grid is treated as a vector, and the vectors are clustered to find representative environments. The number of clusters is traded off against the divergence of each cluster using a Bayesian Information Criteria. In our case, we have a more direct means of comparing views, using the closeness measure $c(v, v')$. We can vary the threshold τ to obtain different bounds on the cluster variance. Our technique has the advantage of low storage requirements and an aging process to get rid of long-unused views. Finally, there is no need to choose a particular cluster for localization, as in [5], because a new view is compared against all views to find the best match.

Our algorithm differs from the sample-based approach of Biber and Duckett [4], which forms clusters as randomly-chosen samples at different time scales and synthesizes the common features into an exemplar. Instead, we keep single recent exemplars for each cluster, which have a better chance of matching new views.

VI. Experiments

We performed a series of experiments using a robot equipped with a stereo head from Videre Design. The FOV was approximately 90 degrees, with a baseline of 9 cm, and a resolution of 640x480. The experiments took place in the Willow Garage building, a space of approximately 50x x 50m, containing several large open areas with movable furniture, workstations, whiteboards, etc. All experiments

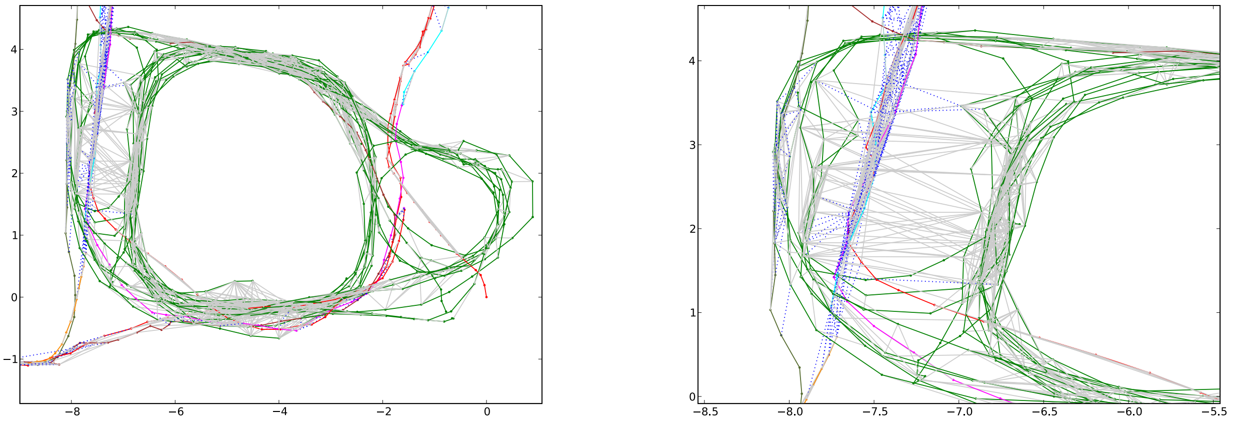

Fig. 7. Additional later sequence added to the map of Figure 6; closeup is on the left. The new VO path is in green, and the intra-path links are in gray. There are 650 views and 2,676 links in this sequence, bringing the total map to almost 2K views. All inter-path links are shown in blue. Note the relatively few links between the new path and the old ones, showing the environment has changed considerably. No view deletion is done in this figure.

were done during the day with no attempt to change the environment or discourage people from moving around the robot. Figure 5 shows some representative scenes from the runs.

A. Incremental Construction

Over the course of two days, we collected a set of six sequences covering a major portion of Willow Garage. 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 6. There were no VO failures in the sequences, even with lighting changes, narrow corridors, and walls with little texture. The map stitching result (right side, Figure 6) shows that PR and optimization melded the maps into a consistent global whole. A detail of the map in Figure 8 shows the density of links between sequences in a stable portion of the environment, 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.

For the whole map, there were a total of 29K stereo frames, resulting in a skeleton map with 1228 view nodes and 3826 links. The timings when adding the last sequence are given in Figure 3. Given that we run PR and skeleton integration only at 2 Hz, timings show that the system can run online.

B. Map Repair

After an interval of 4 days, we made an additional sequence of some 13K frames, covering just a small 10m x 5m portion of the map. During this sequence, the stereo pair was covered at random times, simulating 20 extended VO failures. The area was a high-traffic one with lots of movable furniture, so the new sequence had a significantly different visual environment from the original sequences. The object was to test the system's ability to repair a map, and do so under adverse conditions, namely VO failures.

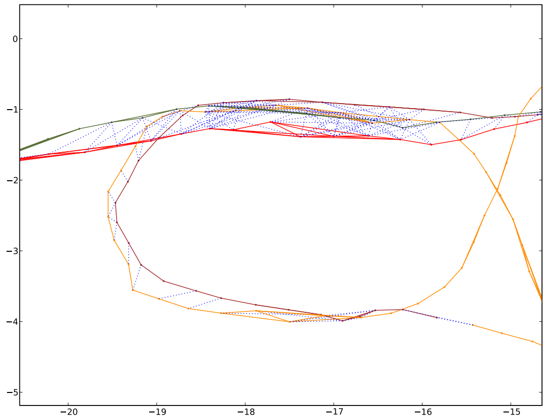

Fig. 8. Detail of a portion of the large map of Figure 6. The cross-links between the different sequences are shown in blue.

In Figure 7, the new sequence is overlaid against the original ones. Despite numerous long VO failures, the new sequence is integrated correctly and put into the right relation with the original map. Because the environment was changed, there are relatively few links between the original sequences and the new one (see the detail on the right), while there are very dense connections among the older sequences. In the new environment, the old sequences are no longer as relevant for matching, and the map has been effectively repaired by the new sequence.

C. Deletion in a Small Area

We did not do view deletion in the first experiment because the density of views was not sufficient to warrant it. With the small area sequence just presented, there were enough views to make deletion worthwhile. We collected statistics on the map with different values for κ, the maximum number of views allowed in a neighborhood. These statistics are for the area occupied by the new sequence.

κ	∞	7	5	4	3	2
Views	643	232	162	134	104	78
Edges	2465	361	213	184	293	269
Views/nb	17.7	6.1	4.6	3.8	2.9	2.0
Clusters/nb	2.0	2.3	2.1	2.0	1.8	1.5

The View line shows the total number of new views in the map, which decrease significantly with κ. New edges also decline until $\kappa = 4$, and which point more edges are needed as clusters are deleted and their neighbors must be linked.

The most interesting line is clusters per neighborhood. In general, there are two clusters, one for each direction the robot traverses along a pathway – views from these directions have no overlapping FOV. Note that decreasing κ keeps the number of clusters approximately constant, while reducing the number of views substantially. It is the clusters that preserve view diversity.

VII. Conclusion

We believe this paper makes a significant step towards a mapping system that is able to function for long periods of time in a dynamic environment. One of the main contributions of the paper is to present the criteria for practical lifelong mapping, and show how such a system can be deployed. The key technique is the use of robust visual place recognition to close loops, stitch together sequences made at different times, repair maps that have changed, and recover from localization failures, all in real time. To operate in larger environments, it is also necessary to build a realtime, optimized map structure connecting views, a role filled by the skeleton map and Toro.

The role of view deletion in maintaining an efficient map has been highlighted. With the LRU deletion algorithm, we have shown that exemplars of different environments can be maintained in a fine-grained manner, while minimizing the storage required for views.

One of the main limitations of the paper is the lack of long-term data and results on how the visual environment changes, and how to maintain matches over long-term changes. Not all features are stable in time, and picking out those that are is a good idea. We have begun exploring the use of linear features as key matching elements, since indoors the intersection of walls, floors and ceilings are generally stable.

VIII. ACKNOWLEDGMENTS

We would like to thank Giorgio Grisetti and Cyrill Stachniss for providing Toro, and for their help in getting the incremental version to work. We also would like to thank Michael Calonder and Patrick Mihelich for their work on the place recognition system.

References

[1] M. Agrawal and K. Konolige. Rough terrain visual odometry. In *Proc. International Conference on Advanced Robotics (ICAR)*, August 2007.

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

[3] H. Andreasson, A. Treptow, and T. Duckett. Self-localization in non-stationary environments using omni-directional vision. *Robotics and Autonomous Systems*, 55(7):541–551, July 2007.

[4] P. Biber and T. Duckett. Dynamic maps for long-term operation of mobile service robots. In *RSS*, 2005.

[5] W. Burgard, C. Stachniss, and D. Haehnel. Mobile robot map learning from range data in dynamic environments. In *Autonomous Navigation in Dynamic Environments*, volume 35 of *Springer Tracts in Advanced Robotics*. Springer Verlag, 2007.

[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 *ECCV*, 2008.

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

[9] F. Dayoub and T. Duckett. An adaptive appearance-based map for long-term topological localization of mobile robots. In *IROS*, 2008.

[10] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular slam. In *BMVC*, 2008.

[11] D. Fox and W. Burgard. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[12] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *IROS*, pages 3872–3877, 2007.

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

[14] 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.

[15] D. Haehnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *ICRA*, pages 1557–1563, 2003.

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

[17] 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.

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

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

[20] M. Montemerlo and S. Thrun. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *ICRA*, pages 695–701, 2002.

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

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

[23] 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.

[24] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4), 1986.

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

[26] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological mapping. In *ICRA*, 2000.

[27] C. Valgren, A. Lilienthal, and T. Duckett. Incremental topological mapping using omnidirectional vision. In *IROS*, 2006.

[28] B. Williams, G. Klein, and I. Reid. Real-time slam relocalisation. In *ICCV*, 2007.

[29] D. Wolf and G. Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *ICRA*, 2004.