Visual Odometry and Map Correlation

Anat Levin* and Richard Szeliski
Microsoft Research

Abstract

In this paper, we study how estimates of ego-motion based on feature tracking (visual odometry) can be improved using a rough (low accuracy) map of where the observer has been. We call the process of aligning the visual ego-motion with the map locations as map correlation. *Since absolute estimates of camera position are unreliable, we use stable local information such as change in orientation to perform the alignment. We also detect when the observer's path has crossed back on itself, which helps improve both the visual odometry estimates and the alignment between the video and map sequences. The final alignment is computed using a graphical model whose MAP estimate is inferred using loopy belief propagation. Results are presented on a number of indoor and outdoor sequences.*

1 Introduction

Tracking visual features observed from a moving camera has long been used to estimate camera pose and location (ego-motion) [7, 15]. In the robotics community, this process is often called *visual odometry* [9]. Under favorable conditions, estimates of ego-motion based on such tracks can be quite accurate over the short run, but suffer from accumulated errors over longer distances, in addition to suffering from the well-known pose and scale ambiguities [9, 16].

A complementary source of information, such as ground control points, absolute distance measurements, or GPS is needed to compensate for these inaccuracies. However, such information may not be readily available, i.e., it presupposes working with surveying equipment or having good lines of sight to GPS satellites. In many situations, being able to rely solely on visual data is preferable, leading to a simpler, less expensive system, and enabling the interpretation of existing video footage.

In this paper, we examine how a rough hand-drawn map indicating where the observer has traveled can be integrated with feature-based ego-motion estimates to yield a more accurate estimate of absolute location. Our main application is the creation of real-world video-based tours of interesting architectural and outdoor locations [20]. In such situations, low-resolution maps are often readily available, and pre-planning a route by drawing on the map is usually done in any case so that a good camera trajectory can be determined ahead of time.

The main problem is how to accurately align vision-based ego-motion estimates with 2D map locations. If the ego-motion estimates themselves were reasonably accurate to begin with, e.g., up to an unknown similarity or affine transform from the ground truth, this process would be straightforward, and could be solved using global moment matching and time-warping (e.g., dynamic programming).

Unfortunately, while instantaneous ego-motion estimates of rotation and direction of motion can be quite accurate, accumulated errors eventually lead to large-scale global distortions in the estimated path [9, 16] (Figure 4c). This requires us to develop a technique that can exploit more *local* structure in the visual ego-motion and hand-drawn map data.

Tours through real-world environments often cross back over themselves in several places [15, 16, 20]. This additional information can be used both to improve the quality of the ego-motion estimates, and to further constrain the possible matches between the visual odometry and map data. We therefore develop an efficient technique to automatically detect when the path crosses back onto itself based on the visual data alone and later use this to augment the map correlation stage.

1.1 Previous work

In the field of mobile robotics, the problem of navigating a novel environment while building up a representation of both its structure and the observer's motion is often called *simultaneous localization and mapping* (SLAM) [3, 16]. In most of these applications, the robot builds a map of the environment as it moves around, although in some applications, the map is given beforehand. *Localization* refers to the process of finding out where you are once you have built an annotated map, perhaps using recognizable landmarks. The *correspondence problem* is determining when you have seen the same object (landmark) and/or come back to the same location.

Some of these systems use omnidirectional sensors from which panoramic images can be constructed [14, 15]. Such sensors are particularly good for exploring and visualizing environments, because not only do they capture a rich set of images that can be used for image-based rendering

*Current address: The Hebrew University of Jerusalem, Israel

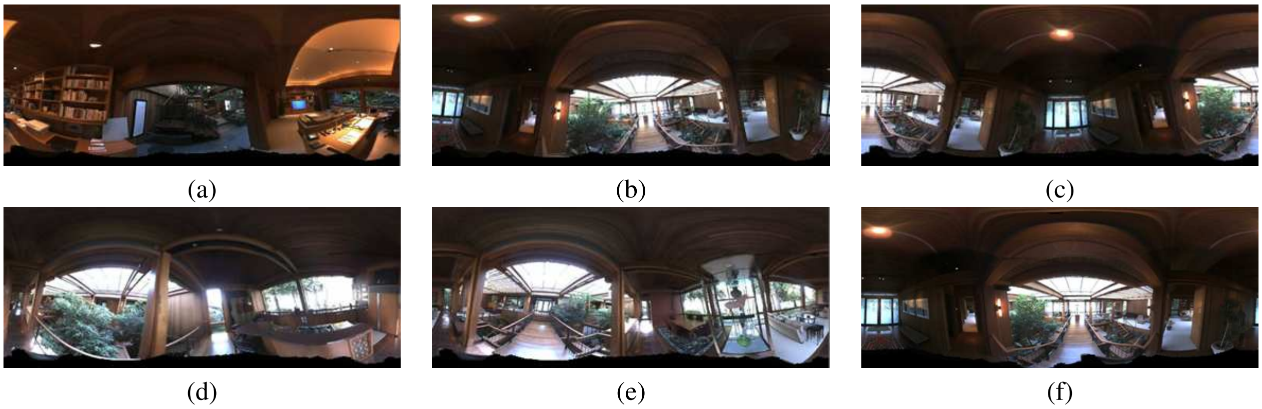


Figure 1. Spherical images from indoor house sequence. Frames (a) and (b) are correctly rejected using color histograms, while frames (b) and (c), while at different orientations, are correctly matched. Frames (d) and (e) on the other hand, are incorrectly matched (false positive) using color histograms. Frame (f) shows the results of aligning frame (c) to frame (b) using moment matching (flagged as a successful match). See the electronic version of this paper for larger images.

[15], they also provide the highest-possible accuracy in ego-motion estimation for a fixed number of pixels [10].

Taylor [15] reconstructs a sparse 3D model of the environment and robot locations from omnidirectional imagery, which can then be used for image-based rendering. Strelow and Singh [14] combine omnidirectional images with inertial sensors. (See also [9] for the use of inertial sensors with more conventional stereo.)

Some researchers have also used panoramic images directly for localization, e.g., using color histograms to recognize when a previously seen location is re-visited [2, 19]. Other researchers have looked at feature-based [12] and texture-based [17] methods to perform localization (sometimes called "place and object recognition"). In our work, we use a novel combination of these techniques, together with a novel moment-based matching scheme, to solve the localization problem with high computational efficiency and accuracy.

Overall, our approach to ego-motion estimation differs from previous work since we assume a *rough* hand-drawn map of the camera's trajectory, rather than having an accurate map from which features can be abstracted and matched against visual data [11, 16]. To our knowledge, this problem has not been previously explored.

1.2 Overview

Before describing the detailed components of our system, we first outline its overall structure. The inputs to our system are a stream of omnidirectional images together with a hand-drawn map of the path traversed, which is assumed to be topologically correct (i.e., the paths and branches are traversed in the same order as drawn). Our video comes from a Point Grey Ladybug$^{\mathrm{TM}}$ camera (//www.ptgrey.com/products/ladybug/), which produces six streams of 1024×768 video at 15 fps.

Our first task is to detect which frames of video were taken from the same (or nearby) locations. We first convert the six individual frames of video into a single flattened spherical image (Figure 1). We then use the efficient hierarchical matching techniques described in Section 2 to detect nearby frames.

Next, we track features from frame to frame, and chain pairwise matches together to form longer tracks. We also match frames that are identified as overlaps (nearby frames) in the first stage. We then use an essential matrix technique to extract inter-camera rotations between all nearby frames.

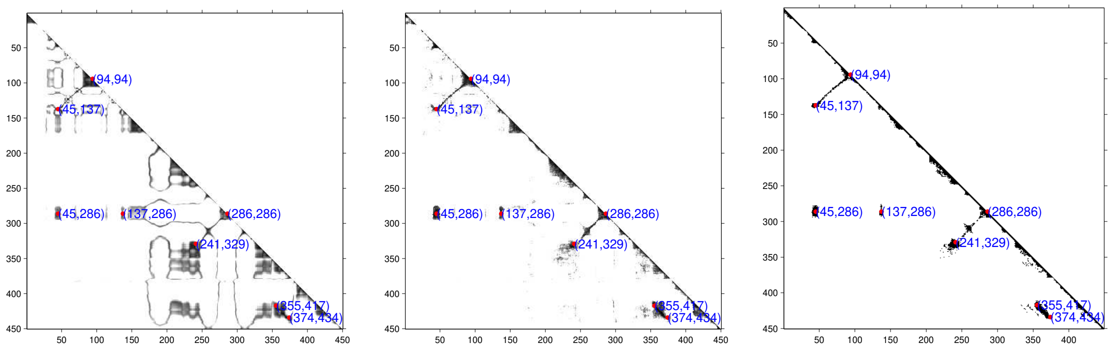
We are now in a position to match the visual odometry and map data using local orientation estimates (Section 3). We simultaneously optimize the similarity in change in orientation, the smoothness in the temporal correspondence assignment, and the consistency at crossings and overlaps. To solve this difficult non-local optimization problem, we use loopy belief propagation since it efficiently exploits the sparse graph structure of our constraints.

Once we have established a temporal correspondence between the ego-motion and map data, we can update the camera positions returned by initial ego-motion estimates to better reflect the corresponding map locations (Section 4). We then run a bundle adjustment algorithm to estimate the final camera positions.

We show experimental results on both an indoor and outdoor sequences, and close with a discussion of our results and some ideas for future work.

2 Overlap and crossing detection

Given our input video, our first task is to detect which (non-contiguous) video frames were taken from nearby locations. This information is later used to improve the quality of the visual odometry estimates (by matching frames



(a) (b) (c)

Figure 2. Distance maps for the house (upper level) sequence: (a) after matching color histograms; (b) further pruning using moments; (c) final matches using epipolar constraints.

when the observer re-visits some previously seen location), and also to improve the quality of the temporal alignment (map correlation).

We begin by combining our six input images into a single spherical image using a previously computed calibration of the camera rig. A few examples from an indoor home sequence are shown in Figure 1. Our comparison technique needs to be rotation invariant since the viewer can return to a previously seen location facing a completely different direction. One approach would be to use spherical harmonics [8] or a simpler FFT-based technique to perform an alignment of each pair of images. The computational complexity of such an approach is $O(n^2 p \log p)$ where n is the number of images and p is the number of pixels. We could also extract characteristic features and match these to establish correspondences [12]. Naïve implementations are $O(n^2 f^2)$, where f is the number of features per image, although tree-based techniques can potentially reduce this to $O(n \log n)$ [1].

In selecting our approach, we need to keep in mind that a typical sequence in traversing an interesting interior or exterior space may have tens of thousands of frames. To process the sequence quickly, we try to reject a large number of obviously bad matches by comparing global color histograms. Next, we match global color moments to obtain a rough rotation estimate, and finally, we match features to confirm the correspondences. Each of these stages is described below in more detail.

2.0.0.1 Matching histograms. We begin by computing a color histogram for each image. Each histogram contains 6 bins in each of the RGB channels, for a total of 216 bins. We build a cumulative histogram table in which the (R, G, B) bin is the sum of all smaller bins:

$$c(R,G,B) = \sum_{r \le R, g \le G, b \le B} h(r,g,b), \tag{1}$$

where $h(r, g, b)$ is the percent of image pixels in the (r, g, b) bin. We use the distance between the cumulative color histograms as a first similarity score [22]. Using a conservative matching threshold, we can prune away over 95% of the bad matches, while retaining all of the good matches (Figures 2–3(a)).

2.0.0.2 Matching moments. For each pair of frames whose histograms distance is below a threshold, we compute a 3D rotation by matching simple moments. This relies on the first order moments of a spherical image being invariant under 3D rotation, i.e.,

$$\int_S f(I(R\hat{\mathbf{u}}))\hat{\mathbf{u}} d\hat{\mathbf{u}} = R^T \int_S f(I(\hat{\mathbf{u}}))\hat{\mathbf{u}} d\hat{\mathbf{u}}, \tag{2}$$

where $I(\hat{\mathbf{u}})$ is the RGB value of the sphere in direction $\hat{\mathbf{u}}$, and $f(I(\hat{\mathbf{u}}))$ is a simple function of the local color. In our current implementation, the features we use are simple binary values, checking if the color falls within a certain histogram bin. Again, 6 bins in each channel are used for a total of 216 features (spherical binary images). For each frame in the sequence, we evaluate the 216 vectors

$$\mathbf{m}_{ki} = \int_S f_i(I_k(\hat{\mathbf{u}}))\hat{\mathbf{u}} d\hat{\mathbf{u}}. \tag{3}$$

For each pair (k, l) passing the threshold of stage 1, we compute a 3D rotation between the moment representation using the Procrustes algorithm [5]. We then keep those pairs in which median error (computed using RANSAC) in the moments after rotation is low (Figures 2–3(b)).

2.0.0.3 Matching features. For the final stage, we extract Harris corners [6] and then estimate the epipolar geometry (Essential matrix) between the remaining candidate images using a robust RANSAC technique [7]. Pairs without enough consistently matching features are then discarded.

The feature matching stage is also used to produce highly accurate inter-frame rotation estimates, which are used in

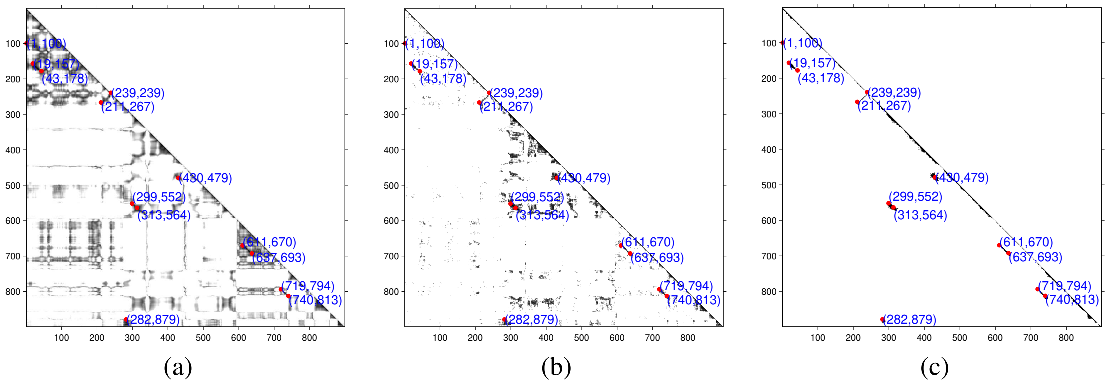


Figure 3. Distance maps for the botanical garden sequence: (a) after matching color histograms; (b) further pruning using moments; (c) final matches using epipolar constraints.

the next phase of our map correlation algorithm to perform temporal alignment of the video and map sequences.

2.0.0.4 Matching results. The resulting cascaded pruning scheme is extremely fast. Processing the set of all pairs using the first two stages in a 1000 frame sequence takes just a few minutes using our unoptimized Matlab implementation.

Figure 1 shows some matching results for the first two stages, as described in the figure caption. Figures 2–3 show the distance maps computed using this process, with matching pairs marked with numerals. For the house, the matches after the first two stages are already good enough. For the botanical garden sequence, there are still frames taken at different geometric location with similar statistics. To eliminate such pairs, we require the final feature matching stage. Figures 2–3(c) show the final matches, which we can think of a binary matrix $M(i, j)$. Note that in interpreting these distance maps, clusters (segments) of matching points parallel to the main diagonal indicate a path that was retraversed in the same direction, while segments perpendicular to the diagonal indicate a reverse traversal (e.g., when exiting a “dead-end”).

Figure 4 shows the matches plotted on both the hand-drawn map (using a hand-labeled correspondence between video frames and map positions for the purposes of visualization) and a rough ego-motion reconstructed path (whose construction is described below). Again, we can see that the overlaps and crossings are all correctly identified.

2.0.0.5 Orientation estimation. Once we have established feature correspondences between spatially adjacent video frames, we estimate pairwise inter-frame rotations and translation directions using the Essential matrix technique [7]. These could be used to initialize a global bundle adjustment [18] on all of the structure and motion parameters, but this would result in a huge system that would converge slowly and would still be susceptible to low-frequency errors [9, 13]. Alternatively, we could keep only the inter-pair rotation estimates, and use these to optimize for the global orientation of each camera. This is a simple non-linear least-squares problem in the rotation estimates, which could be optimized using Levenberg-Marquardt.

Instead, we have found that the global orientation estimation stage can be bypassed entirely, and that simple (but globally inaccurate) estimates of orientation based on chaining together rotation estimates between successive frames are adequate for the map correlation phase. A similar chaining of displacement estimates assuming uniform observer velocity (which is clearly wrong when turning around at the end of a corridor or path) is used to estimate the initial ego-motion (location) estimates shown in Figure 4(b).

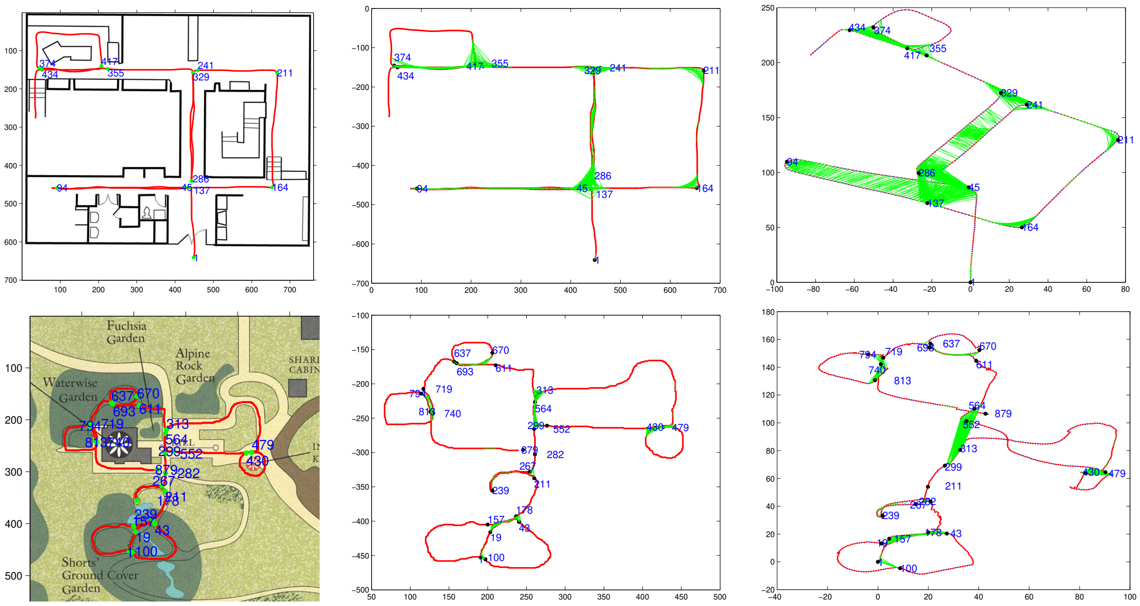
3 Map correlation

At this point, we are ready to match up the visual data (from which we have extracted local estimates of orientation and global estimates of path overlaps and crossings) with the hand-drawn map. This might at first glance seem to be a simple case of time warping, which could be solved using dynamic programming. However, as we will shortly see, this would not allow us to exploit all of the constraints available for solving this problem.

3.1 Problem formulation

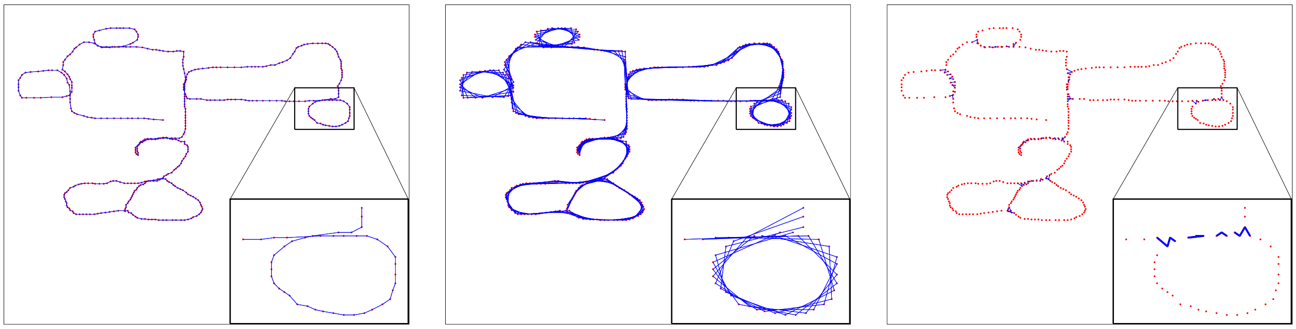
As we mentioned before, while the vision-based ego-motion (visual odometry) estimates capture the local structure of the path, global errors tend to accumulate over time. Therefore a global matching of map-based observer orientations and visual odometry orientation estimates will not work. Instead, we need to match the frames and map points based on the *local* orientation differences. In our work, we only use rotations around the vertical axis, since these are the only ones that can be measured from the map.

To find a matching between map points $i \in \{1, \ldots, m\}$



(a) (b) (c)

Figure 4. Matches overlaid on a map: the first line presents a house sequence and the second a Botanical garden (outdoor) sequence. (a) original hand-drawn map; (b) matching pairs plotted as green lines on the map; (c) matching pairs plotted on the initial ego-motion reconstruction. The frame numbers were selected by hand and the correspondences were manually established to gauge the quality of the overlap results. Figure 6 shows the automatically computed correspondences. (See the electronic version of this paper for larger figures.)



(a) (b) (c)

Figure 5. Graphical model: (a) edges in group E_1; (b) edges in group E_2; (c) edges in group E_3.

and video frames $k \in \{1, \dots, n\}$, we define a matching function $f : 1, ..., m \rightarrow 1, ..., n$, where m is the number of map points (after some discretization) and n is the number of frames in the sequence.

Let $\alpha(i, j)$ be the change in orientation between map point i and map point j. (We can estimate the local orientation by smoothing the hand-drawn map slightly and computing tangent directions.) Similarly, let $\hat{\alpha}(f_i, f_j)$ be the change in orientation from frame f_i to frame f_j, as computed from the visual odometry.

A good match should satisfy the following three criteria:

1. The matching function f should be monotonic and smooth.
2. Orientation differences $|\alpha(i, i + w) - \alpha(f_i, f_{i+w})|$ would be small, where w is the temporal window size, which should be high enough to overcome local noise, but low enough to avoid global drift. In our current experiments, we use a value of $w = 6$.
3. Whenever the map indicates close proximity between two points (i, j), the visual data should indicate that frames (f_i, f_j) were taken from nearby locations.

3.2 The graphical model

How do we encode these constraints into an optimization framework and find the desired correspondence? The answer is to associate the map with an undirected graphical model. The nodes in the model are the sampled map points. Each node in the graph i is assigned a value $f_i \in \{1, \dots, n\}$ representing the matching frame in the video sequence.

To express the above three criteria, the graph is built out of three edges types:

1. To encode monotonicity and smoothness, nodes i and $i+1$ are connected by an edge in E_1.

2. To encode orientation consistency, nodes i and $i+w$ are connected by an edge in E_2.

3. To encode overlaps (proximity), each pair of points (i,j) within a close geometric location on the map is connected by an edge in E_3.

The three edges types for the botanical garden sequence are shown in Figure 5.

To find the optimal map correlation, we search for the assignment $\{f_1, ..., f_m\}$ that maximizes the probability

$$P(f) = \prod_{(i,j)\in E_1} \Psi^1_{i,j}(f) \prod_{(i,j)\in E_2} \Psi^2_{i,j}(f) \prod_{(i,j)\in E_3} \Psi^3_{i,j}(f). \quad (4)$$

For each of the three edges types, the pairwise potential $\Psi^p_{i,j}(f)$ is set to express the corresponding matching criterion. The first potential is

$$\Psi^1_{i,i+1} = \exp(-|f_i - f_{i+1}|^2)[f_i \leq f_{i+1}], \quad (5)$$

where the predicate $[f_i \leq f_{i+1}]$ constrains the match to be monotonic, and maximizing $exp(-||f_i - f_{i+1}||^2)$ is equivalent to minimizing the squared difference between f_i, f_{i+1}, which results in a smooth match. The second potential is

$$\Psi^2_{i,i+w} = \exp(-|\alpha(i, i+w) - \hat{\alpha}(f_i, f_{i+w})|). \quad (6)$$

Maximizing this term minimizes the local difference between the ego-motion (visual odometry) and map-based orientation estimates. The last potential is

$$\Psi^3_{i,j} = M(i,j), \quad (7)$$

i.e., $\Psi^3_{i,j}$ equals 1 iff the frames (f_i, f_j) were identified as being in nearby positions by the overlap detection stage.

3.3 Loopy belief propagation

Since the edges in groups E_2 and E_3 introduce loops in the model, dynamic programming cannot be used to find an optimal assignment. Instead, we use loopy belief propagation [21] to estimate the MAP (*maximum a posteriori*) assignment. This is similar in spirit to the work of Coughlan and Ferreira [4], who use loopy belief propagation to recognize letters.

Applying the loopy belief propagation message passing scheme requires some care, since the above graphical model contains a large number of loops, and the belief propagation process is not guaranteed to converge to the global optimum (or to converge at all). To improve convergence, we use an asynchronous message passing scheme and select the updating order in the following way.

Notice that the edges in group E_1 are a loop-free chain. The edges in the second group are also a union of w plain chains of the form $j, j+w, j+2w, \dots$. Recall that one iteration of an asynchronous message passing on a chain (or any tree structured graph) converges to the global optimum with respect to the potentials on the edges of this chain alone.

We therefore propagate the messages in the following order: (1) the chain in group E_1; (2) the loops in group E_3; (3) the first chain of group E_2; (4) the chain in group E_1; (5) the loops in group E_3; (6) the second chain of group E_2, *etc.*

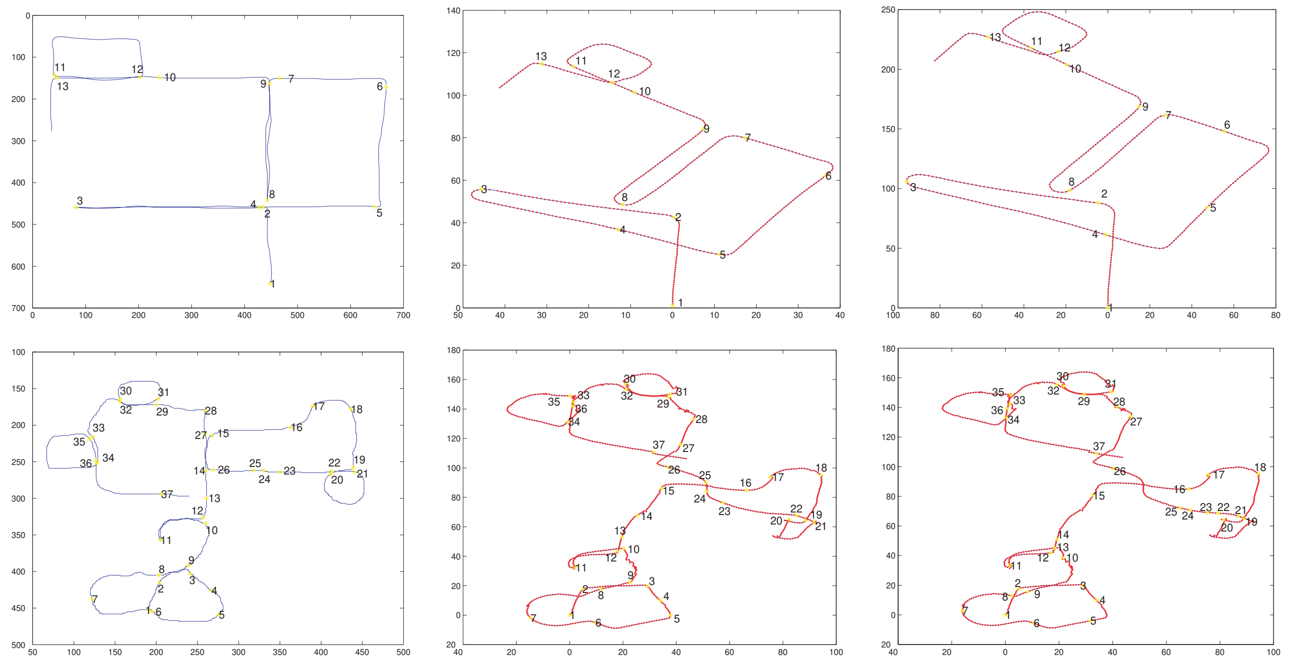
Note that if we could have sampled the map in such a way that $w = 1$ were already a noise free window, the second group would contain only one chain and could have been merged with the first group. Message passing in this case would be simple. However, we noticed that while sampling the map too densely indeed converged easily, the solution was not "nailed down" to the desired accuracy. The above process aims to nail a few chains in the map simultaneously, which results in significantly more accurate matching.

Figure 6(a-b) shows the final computed matches. For comparison Figure 6(c) shows the matches obtained using simple dynamic programming. For this comparison, the DP matched absolute orientation angles (after a single global shift) and included a smoothness term as well as a term to encourage map crossings to match visual crossings. As can be seen by inspecting the individual frame numbers, our algorithm successfully established high-accuracy matches between all of the video frames and the map data, while dynamic time warping produces a less accurate registration (particularly at bends in the path).

4 Pose update and re-estimation

Given the matching function (map correlation) between the video frames and the map points, we now wish to correct the global ego-motion using the map data. Note, however, that while the map accurately encodes the global structure of the world, it is often locally inaccurate, both because the map was manually drawn by a user, and because the actual tour the camera took in the world contains jumps and high frequencies that the map is unable to encode. Such high frequency motions can be accurately recovered using visual odometry.

To compute a global correction to the pose estimates, we divide the frames into a set of small segments. For each of these segments, we solve for a Euclidean transformation A_j minimizing the least squares distance between the camera positions given by the ego-motion and the map. Limit-



(a) (b) (c)

Figure 6. Loopy belief propagation matches: (a) input map. (b) ego-motion matches, using full loopy belief propagation. (c) ego-motion matches, using only dynamic time warping. The numbers show selected correspondences picked by hand to illustrate the quality of the overall correspondences.

ing the transformation to a simple parametric form enables global corrections of the ego-motion, but prevents the loss of the high frequencies. Furthermore, the set of transformations should connect neighboring segments in a smooth way. We therefore solve for the set of transformations $\{A_j\}$ simultaneously, requiring A_j and A_{j+1} to operate on the last few frames of the j segment and the first few frames of the $j+1$ segment in a similar way.

Figure 7 presents the results of the pose update step. Notice how the significant structures in the ego-motion are preserved. For example, in the right loop of the botanical garden sequence, the map was drawn inaccurately as a circle, while the camera carrier actually remembers walking in a square-like path. This square structure was captured by the visual ego-motion, and our transformation is robust enough to bring the ego-motion close to the map while preserving this square structure. Similarly, the S-shaped wiggle seen to the left of that loop actually reflect the true structure of the pathway, which makes a series of $90°$ turns while climbing a set of steps.

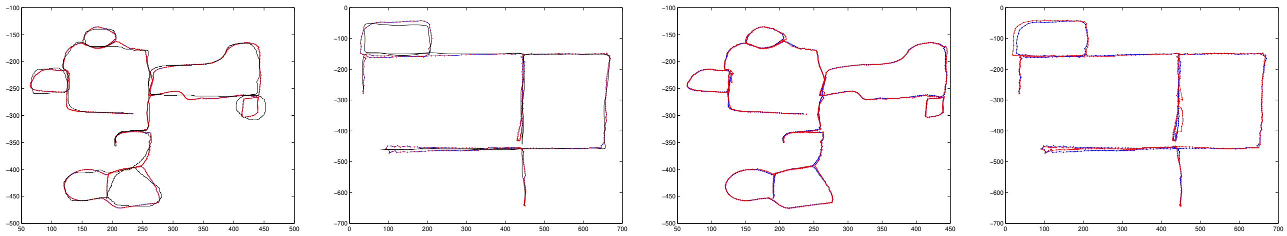
Once we have performed the initial adjustment of the rough ego-motion estimates, we can re-solve for the camera positions using a full feature-based bundle adjustment, using the map positions as weak priors on the camera positions [18]. The results of doing this are shown in Figure 7(c–d). As you can see, the final bundle adjustment does not change the results significantly. We need to perform further tests to determine whether this is because the global registration was indeed close enough, or whether the bundle adjustment problem is suffering from poor conditioning.

5 Experiments

We have tested our algorithms on three different data sets. The first was an outdoor botanical garden sequence, while the other two were the upper and lower floors of a house (not shown, due to space limitations), which were processed separately because of the need to draw separate maps. The original botanical garden data consists of 10,000 frames, which we subsampled to every 10th frame before processing. Similarly, each section of the house contained about 5000 original frames, which we subsampled down to 500.

The results of our experiments are shown in Figures 1–7. In general, the algorithms performed well. The novel multi-stage localization algorithm quickly rejected unlikely matches and selected final matches that found all major overlaps without any false positives. The loopy belief propagation map correlation phase performed much better than a simple dynamic program and correctly matched up all of the major turn points and segments.

The one thing that we are currently missing is ground truth data for the actual motion. This could be obtained either using a robot with carefully calibrated odometry, or



(a) (b) (c) (d)

Figure 7. Initial local transformation of ego-motion estimation to map (a–b), and final results after bundle adjustment (c–d). The original paths (before updates) are shown in blue, while the updated paths are shown in red.

using photogrammetric techniques, e.g., by surveying visible landmarks (ground control points) and identifying them manually in the images.

6 Conclusions

In this paper, we have introduced and solved a novel variant on the visual odometry (camera localization) problem, where the system is provided with a rough hand-drawn map of the camera's trajectory. Our approach relies on a fast multi-stage correspondence algorithm to identify visually similar panoramic views, followed by a graphical model for finding the optimal temporal correspondence, which we optimize using loopy belief propagation.

In future work, we would like to investigate the use of additional information and constraints, such as vanishing points, the fact that the camera is on average not tilted and that floors are flat (indoors), and the use of absolute orientation sensors [9].

We also plan to investigate recomputing the map correlation in alternation with the final bundle adjustment to see if our results could be improved. This will also entail the development of more efficient algorithms for solving very large structure from motion problems [13].

References

[1] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *CVPR'97*, pp. 1000–1006, 1997.

[2] P. Blaer and P. Allen. Topological mobile robot localization using fast vision techniques. In *ICRA'02*, v. 1, pp. 1031–1036, 2002.

[3] M. Bosse *et al.* An *Atlas* framework for scalable mapping. In *ICRA'03*, v. 2, pp. 1899–1906, 2003.

[4] J. M. Coughlan and S. J. Ferreira. Finding deformable shapes using loopy belief propagation. In *ECCV 2002*, v. III, pp. 453–468, 2002.

[5] G. Golub and C. F. Van Loan. *Matrix Computation, third edition*. The John Hopkins University Press, 1996.

[6] C. Harris and M.J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pp. 147–152, 1988.

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, Cambridge, UK, Sept. 2000.

[8] A. Makadia and K. Daniilidis. Direct 3d-rotation estimation from spherical images via a generalized shift theorem. In *CVPR'2003*, v. II, pp. 217–224, 2003.

[9] C. F. Olson *et al.* Stereo ego-motion improvements for robust rover navigation. In *ICRA'01*, v. 2, pp. 1099–1104, 2001.

[10] R. Pless. Using many cameras as one. In *CVPR'2003*, v. II, pp. 587–593, 2003.

[11] D. P. Robertson and R. Cipolla. Building architectural models from many views using map constraints. In *ECCV 2002*, v. II, pp. 155–169, 2002.

[12] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *ECCV 2002*, v. I, pp. 414–431, 2002.

[13] D. Steedly and I. Essa. Propagation of innovative information in non-linear least-squares structure from motion. In *ICCV 2001*, v. 2, pp. 223–229, 2001.

[14] D. Strelow and S. Singh. Optimal motion estimation from visual and inertial measurements. In *WACV 2002*, pp. 314–319, 2002.

[15] C. J. Taylor. Videoplus: a method for capturing the structure and appearance of immersive environments. *IEEE Trans. Visual. Comp. Graphics*, 8(2):171–182, April-June 2002.

[16] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*, pp 1–36. Morgan Kaufmann, 2002.

[17] A. Torralba *et al.* Context-based vision system for place and object recognition. In *ICCV'03*, pp. 273–280, 2002.

[18] B. Triggs *et al.* Bundle adjustment — a modern synthesis. In *Intl. Workshop Vision Algorithms*, pp. 298–372, 1999.

[19] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *ICRA'00*, v. 2, pp. 1023–1029, 2000.

[20] M. Uyttendaele *et al.* High-quality image-based interactive exploration of real-world environments. *IEEE Computer Graphics and Applications*, 24(3), May-June 2004.

[21] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

[22] M. Werman, S. Peleg, and A. Rosenfeld. A distance metric for multidimensional histograms. *Computer Vision, Graphics and Image Processing*, 32(3):328–336, 1985.