Bayesian Network for Online Global Pose Estimation

A. Rahimi   T. Darrell

MIT AI Lab
Cambridge, MA 02139

Abstract

We cast the location estimation problem in vision-based robotic navigation in a Bayesian framework. We derive an efficient online algorithm for updating the trajectory of a robot as new frames of data become available. For each new frame, the algorithm computes the pose of the robot relative to past frames and combines these relative pose changes to obtain a robust estimate of its trajectory. The complexity of this algorithm grows linearly with the number of frames so far processed. Because it is effectively tracking against an appearance-based map, our algorithm provides consistent results in circular environments, where the robot returns to places already visited.

1 Introduction

Many robot navigation tasks require accurate sensing of the robot's location. This is a particularly difficult problem when the environment is uninstrumented or has never been explored before. This paper addresses the problem of computing an accurate estimate of the robot's position given a sequence of scans from an optical sensor such as a monocular camera, a stereo camera, or a laser range scanner. No prior knowledge or instrumentation of the environment is required. Our algorithm computes a globally consistent trajectory, so that when the target returns to an already visited position, its estimated pose is consistent with the pose estimate produced on the earlier visit. To compute the trajectory, we build a non-causal filter which continually updates the past poses as new scans become available, maintaining an optimal trajectory at each time step. Our algorithm does not need to build a 3D model of the world, since it tracks with respect to past frames, and updates the trajectory in time proportional to the number of frames seen so far.

We revisit the problem of global registration in the Bayesian framework and cast the true trajectory to be estimated as hidden variables, where noise-corrupted estimates of the difference between these are available as observations. We presume an algorithm which, given two scans of the environment, can recover the pose change parameters between them and provide a certainty over its result. There exists an abundance of these trackers, ranging from planar [23, 26] and 3D [25, 30, 4] motion models using monocular camera to 3D motion models using range imagery [8, 13].

Our algorithm incorporates each measurement by updating the pose of every frame so far encountered. To insure that subsequent updates can happen in linear time, the resulting correlation structure of the poses is approximated with a simpler Markov chain. This scheme can be thought of as an instance of Assumed Density Filtering (ADF) [17, 1].

2 Related Work

Global pose estimation has been applied to a variety of tasks, ranging from aerial photography [19], mosaicking of planar scenes[26, 23], stitching of laser range scans [6, 21, 3, 27, 2], robot navigation[15, 5, 29], head tracking [22, 11, 9] and camera-based 3D model acquisition [31]. Global pose estimation has typically involved building a mosaic, a 3D model, or a feature-based map of the environment. Alternatively, instead of maintaining an explicit geometric model, poses can be associated with the scans themselves, providing a kind of image-based model of the environment. Depending on the representation, it may be necessary to construct a topological map of how scans are spatially related to each other before registering them together. Updates and refinements to the model and the trajectory can be performed either in batch, once all scans have been made, or online, as scans become available. The following subsections compare the existing literature along these three axes.

2.1 Representation of map

A common approach to global registration is to register each incoming frame against a map, and to subsequently update the map with the new frame. An early version of Sawhney and Kumar's system

[24] registered each incoming frame against a mosaic and then pasted the warped image unto the mosaic. Work in 3D model acquisition [6] has used a similar strategy for building 3D models by registering structured light range scans against an accumulated model.

These methods work well when there are few scans. But since scans are committal, future frames are not able to resolve inconsistencies in the past. For example, when the trajectory closes on itself, these methods cannot explicitly take advantage of this information to adjust the trajectory, and so the quality of the map deteriorates over time.

A common way to address this issue is to represent the map as a feature vector. For example, McLauchlan [15] represents the map as a vector of parameters which describe geometric features in the environment, such as 3D lines and corners. This state vector is updated recursively for each new scan using a Kalman Filter. Jebara [9] uses a similar representation for features on the face of a human subject. Over time, the location of these features are refined to describe the subject's facial geometry.

The alternative is to represent the map as a collection of scans with associated pose parameters. For example, Stoddart and Hilton [27] first find corresponding points between pairs of scans, attach virtual springs between all corresponding points in the entire scan set, and relax the system to convergence. Chen and Medioni [2] propose a version of the Iterated Closest Point (ICP) algorithm which iteratively computes correspondences between scan pairs, and brings all scans into registration together using these correspondences. The process is iterated until convergence. Sawhney and Kumar [23] define a cost function for minimizing the appearance difference between the overlapping areas of images, which are assumed to be of planar structures. This cost function is reduced as a function of the pose of the planar patches. Lu and Milios [12] first compute pose differences between pairs of scans, and merge these pose differences into consistent poses by solving a maximum likelihood problem. [22] use a similar approach.

2.2 Topology determination

Gutmann and Konolige [7] and Sawhney and Kumar [23] both represent the environment map as scan sets. In order to determine which scans to use as base frames for each incoming scan, [23] use a coarse pass over the data which computes rough estimates of 2D location. Proximity in that parameters space is used to identify suitable matches. [7] builds a 2D top view map of the environment solely for use with topology determination.

We pursue a hybrid approach. Since the pose of all past frames are assumed to be accurate, the pose of an incoming frame is computed with respect to a given base frame only if they are similar in both pose and appearance (measured as the L2 norm of the pixel differences).

2.3 Online vs. Batch

The cost functions used in global pose estimation typically have the form

$$\epsilon(x) = \sum_{(i,j)\in P} d_{ij}(x).$$

The function d_{ij} is a measure of registration error between frames i and j. This optimization is costly and can involves large matrix multiplications or inversions of Jacobians and Hessians. This makes it difficult to build an online global pose estimation system, since the introduction of a new frame couples many poses and increases the size of the Jacobians of the optimization. McLauchlan [15] and Thrun [29] address the issue of efficient online updates of the map as each new scan is introduced.

McLauchlan proposes an online recursive algorithm for updating the map, which is represented as a feature vector. Since adding features increases the complexity of the updates, a method is provided for forgetting old features. Thus, to curb the complexity growth of the updates, certain features are simply not updated. Hence this algorithm does not perform well in large trajectory loops.

Thrun suggests another way of updating the map online. As loops in the trajectory are detected, a propagation step corrects backward poses in the loop. The updates are online, but increase linearly in complexity with the length of the loop.

Our update algorithm is derived as an approximation to the true posterior trajectory estimator for the robot. We obtain an algorithm which takes linear time with respect to the number of frames seen so far to perform its updates. Unlike [15], our algorithm does not forget the past. Instead, we use an approximation which simplifies the correlation structure of the poses and take advantage of an efficient inference algorithm for computing posterior pose estimates.

Much of the global pose estimation literature assumes an algorithm for recovering the pose change between two scans of the sensor. We assume such an algorithm is available and that in addition, it can report its uncertainty. The next section casts pose tracking in a Bayesian framework. Section 4 builds up a probabilistic model for globally consistent pose estimation using a pose change algorithm. Section 5 shows how to use this model to track the pose of an object in real-time. Finally, we present results on a 2D trajectory estimation problem.

3 Generative Model

This section defines a probabilistic generative model where poses are hidden variables and pose changes are measured by a pose change estimator. Subsequent sections explain how to use this model to estimate the true poses given pose change estimates.

Let $X = \{x_t\}_{t=1..T}$ be the trajectory of the robot up to time T, with each x_t its pose at time t. These poses can represent any parametrization of pose, for example as 3D rotations and translation, 2D translations, or even non-rigid deformations such as affine. Call Y the set of measured pose changes up to time T: $Y = \{y_s^t | s < t < T$ and pose between (s, t) has been measured$\}$.

The problem of estimating the current pose involves finding the posterior distribution $p(x_t|Y)$. Global pose consistency, on the other hand, involves adjusting the entire trajectory as each new pose change measurement becomes available, by computing the joint posterior poses $p(X|Y)$. To compute this posterior trajectory, we need a prior model $p(X)$ for the pose trajectory and an observation model $p(Y|X)$ to describe the output of the pose change estimator in terms of the true poses it measures.

We model the pose dynamics absent any measurements as a Markov chain:

$$p(X^T) = \prod_{t=1}^{T} p(x_t|x_{t-1}).$$

We model the pose change estimator as observing scans I_t and I_s from the environment and estimating the pose change between them. We assume that these measurements are independent of each other even when they share a base image, and model the pose change estimator as measuring the true pose change $d(x_s, x_t)$ directly:

$$p(y_s^t|x_s, x_t) = \mathcal{N}(y_s^t|d(x_s, x_t), \Lambda_{y|xx})$$

where $\Lambda_{y|xx|}$ is the uncertainty in the measurement and must be provided by the pose change estimator.

By way of example, suppose the pose is parametrized as the affine appearance deformation with respect to the last frame. Let x_t be the affine transformation required to bring the image at time t into registration with the frame at time s. The pose change estimator recovers a pose change y_s^t according to $p(y_s^t|x_s, x_t)$, a distribution centered around $d(x_s, x_t) = I - x_s x_t^{-1}$. The model is:

$$\begin{aligned} p(y_s^t|x_s, x_t) &= \mathcal{N}(\text{vec}\left[y_s^t\right] | I - \text{vec}\left[x_s x_t^{-1}\right], \Lambda_{y|xx}) \\ p(x_{t+1}|x_t) &= \mathcal{N}(\text{vec}\left[x_{t+1}\right] | \text{vec}\left[x_t\right], \Lambda_{x|x}), \end{aligned}$$

where the vec operator stacks up the elements of a matrix into a column vector [18].

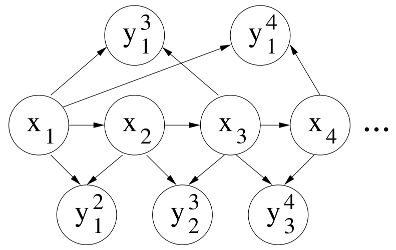

Figure 1: *Independence diagram for multiple-base frame pose estimation.*

4 Inference with Multiple Base Frames

Given a pose change estimator, the simplest way to recover the trajectory of a robot is to compute pose changes between temporally adjacent frames, and to accumulate these changes into a pose estimate. This formulation suggests a Kalman filter where the state vector is a compounding of the pose of the current frame and the past frame, and the observation is the difference between these, corrupted by noise. However, this approach results in drift, as witnessed by the growing variance in the estimated pose (see conditions of convergence in [10]).

To overcome this drift, we compute pose changes between non-adjacent frames in addition to temporally adjacent frames. As each new frame appears, its pose is computed with respect to several strategically chosen back frames. These pose change measurements are used to calculate the pose of the incoming frames and to update the pose of all past frames. Thus we effectively maintain a kind of image-based map of the world, which is updated for each new frame. Tracking with respect to past frames can then be thought of as tracking with respect to an adjustable map of the environment [23, 22, 12].

Figure 1 shows the independence diagram for this scheme, where pose changes are allowed to span over a wider set of poses and can provide redundant information. As each new pose change measurement y_s^t is computed (with $s < t$), we wish to compute the posterior trajectory $p(x_1..x_t|y_1^2..y_s^t)$.

This update can be accomplished recursively by considering the estimate of X from old observations Y^{old} and conditioning on the latest measurement y_s^t:

$$p(X|Y^{new}) \propto_X p(X|Y^{old})p(y_s^t|X, Y^{old}). \qquad (1)$$

Since a new pose measurement is independent of past measurements when the true poses are known (see the independence diagram in figure 1) [16, 20],

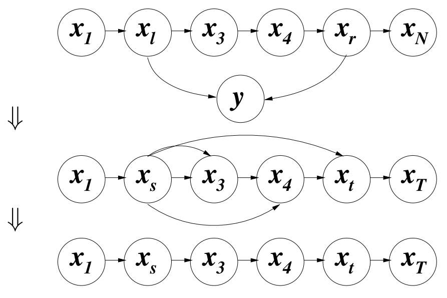

Figure 2: *The measurement (top) correlates the hidden variables (middle), whose correlation is then simplified (bottom), and is ready to accept a new measurement.*

$p(y_s^t|X, Y^{old}) = p(y_s^t|X)$. So the update (1) is Bayes' rule with the trajectory so far estimated as the prior, and the measurement model for the new measurement as the likelihood.

This recursion was used by [12]. This update requires time proportional to $O(T^2)$ since that $p(X|Y^{old})$ is a T-dimensional Gaussian with arbitrarily complex correlation structure (as shown by figure 1. See for example the Kalman update equations [10]).

Some methods have been proposed for curbing this cost. For example, [15] lowers the dimensionality of the problem by fixing some of the poses and dropping them from the state vector. Unfortunately, this method doesn't apply to situations with very large loops. When the structure of figure 1 suggests a sparse structure, suitable sparse methods can be used [28, 15]. But these algorithms still run slower than $O(T)$.

5 An Approximate Update Scheme

In this paper, we reduce this cost to $O(T)$ by resorting to Assumed Density Filtering (ADF) [17]: we approximate the estimate of X given old data, $p(X|Y^{old})$, with a simpler distribution $q(X|Y^{old})$. To incorporate a new measurement y_s^t, we apply the update

$$p(X|Y^{new}) \overset{Bayes}{\propto} p(y_s^t|x_s, x_t)q(X|Y^{old}). \tag{2}$$

This new $p(X|Y^{new})$ has a more complicated independence structure than $q(X|Y^{old})$, so incorporating subsequent measurements would require more work and the resulting posterior would be even hairier. So we approximate it again with a $q(X|Y^{new})$ that has a simpler independence structure. Specifically, we force q to always obey Markovian independence. Subsequent measurements can again be incorporated easily using this new q. Figure 2 summarizes this process.

To see that the distribution $p(X|Y^{new})$ has the independence structure of figure 2(middle), call X_{left} the nodes to the left of x_s, X_{loop} the nodes between x_s and x_t, and X_{right} the nodes to the right of x_t. Write the posterior as:

$$p(X|y_s^t, Y^{old}) = \frac{1}{p(y_s^t)} p(X_{left})p(X_{loop}, y_s^t|x_s)p(X_{right}|x_t),$$

where we've dropped the condition on old data for clarity. $p(X_{left}|y_s^t, Y^{old})$ has the structure of a Markov chain with an observation on the tail, and so is a Markov chain. $p(X_{right}|y_s^t, Y^{old})$ is a Markov chain with an observation on the head, so it is also a Markov chain. Because the variables in the loop are conditioned on x_s and y, they also form a Markov chain (x_s is the cutset of the loop [20]). However, the observations appear on the head and on the tail, with y_s^t appearing on the tail with distribution $p(y_s^t|x_t)$ parametrized by x_s. So unless we condition on x_s, it is not a Markov chain.

The following section discusses how to find a Markovian q so as to minimize the KL divergence between p and q. Section 5.2 shows how to incorporate a pairwise measurement on the resulting Markov chain using equation (2).

5.1 Simplifying the independence structure

We would like to approximate an arbitrary distribution which factors according to $p(X) = \prod_t p_t(x_t|\text{Pa}[x_t])$ using one which factors into $q(X) = \prod_t q_t(x_t|\text{Qa}[x_t])$. Here, $\text{Pa}[x_t]$ are the parents of node x_t in the graph prescribed by $p(X)$, and $\text{Qa}[x_t]$ are the parents of node x_t as prescribed by $q(X)$. Specifically, we want q to have Markov structure, so $\text{Qa}[x_t] = x_{t-1}$.

The objective is to minimize:

$$\begin{aligned} q^* &= \arg\min_q KL\left(\prod p_t \middle\| \prod q_t\right) \qquad (3) \\ &= \int_x p(X) \ln \frac{p(X)}{\prod_i q_t(x_t|\text{Qa}[x_t])}. \end{aligned}$$

After splitting the objective into two sums, this is equivalent to maximizing

$$\begin{aligned} &\int_X p(X) \sum_t \ln q_t(x_t|\text{Qa}[x_t]) \\ &= \sum_t \int_X p(X) \ln q_t(x_t|\text{Qa}[x_t]). \end{aligned}$$

Since each q_t can be optimized independently, we can maximize the terms one by one. Each term can be further simplified:

$$\int_X p(X)\ln q_t(X_t|\mathrm{Qa}[x_t]) =$$
$$\int_{\mathrm{Qa}[x_t]} p(\mathrm{Qa}[x_t]) \int_{x_t} p(x_t|\mathrm{Qa}[x_t]) \ln q_t(x_t|\mathrm{Qa}[x_t]).$$

Again, we can optimize this by optimizing the inner integral for all $\mathrm{Qa}[x_t]$, so

$$\begin{aligned} q_t^* &= \arg\max_{q_t} \int_{x_t} p(x_t|\mathrm{Qa}[x_t]) \ln q_t(x_t|\mathrm{Qa}[x_t]) \\ &= p(x_t|\mathrm{Qa}[x_t]) \end{aligned}$$

This says that the best conditional q_t is built up from the corresponding p_t by marginalizing out the conditions that were removed in the graph. In general, this is not easy, but as we show below, it is very useful in performing ADF updates.

5.2 Computing posterior transitions on a graph with a single loop

This result suggests a simplification to the update of equation (2). Because the ultimate goal is to compute $q(X|Y^{new})$, not $p(X|Y^{new})$, we only need to compute the posterior transitions $p(x_t|x_{t-1}, Y^{new})$. Thus, we circumvent having to first find p then project it onto q. We propose computing these transitions in three steps, one for the transitions to the left of x_s, another for the loop, and the third for transitions to the right of x_r.

Finding $p(x_\tau|x_{\tau-1}, y)$ for $\tau = s..t$. For every $s < \tau < t$, notice that

$$p(y, x_{\tau-1}, x_t)p(x_\tau|x_{\tau-1}, x_t) = p(y, x_{\tau-1}, x_\tau, x_t), \tag{4}$$

because according to figure 2, $p(x_\tau|x_{\tau-1}, x_t) = p(x_\tau|x_{\tau-1}, x_t, y)$. If we could find this joint distribution for all τ, we could find $p(x_\tau|x_{\tau-1}, y)$ by maringalizing out x_t and normalizing. We could also find $p(x_\tau|y)$ by marginalizing out both x_t and $x_{\tau-1}$, then normalizing. Finally, we could compute $p(y, x_\tau, x_t)$ for the next τ in the iteration.

So there are two missing pieces: The first is $p(y, x_s, x_t)$ for starting the recursion. Computing this term is easy, because $p(y|x_s, x_t)$ is the given measurement model, and $p(x_s, x_t)$ can be obtained easily from the prior by successively applying the total probability theorem.

The second missing piece is $p(x_\tau|x_{\tau-1}, x_t)$. Note that this quantity does not depend on the measurements and could be computed offline if we wanted to. The recursion for calculating it is:

$$p(x_\tau|x_{\tau-1}, x_t) \overset{Bayes}{\propto} p(x_t|x_\tau)p(x_\tau|x_{\tau-1}) \tag{5}$$
$$p(x_t|x_\tau) = \int \mathbf{d}x_{i+1}\, p(x_t|x_{i+1})p(x_{\tau+1}|x_\tau) \tag{6}$$

The second equation describes a recursion which starts from t and goes down to s. It computes the influence of node τ on node t. Equation (5) is coupled to this equation and uses its output. It involves applying Bayes rule to compute a function of 3 variables. Because of the backward nature of (6), $p(x_\tau|x_{\tau-1}, x_t)$ has to be computed using a pass which runs in the opposite direction of the process of (4).

Finding $p(x_\tau|x_{\tau-1}, y)$ for $\tau = 1..s$. Starting from $\tau = s-1$, compute

$$\begin{aligned} p(y|x_\tau) &= \int \mathbf{d}x_{\tau+1}\, p(y|x_{\tau+1})p(x_{\tau+1}|x_\tau) \\ p(x_\tau|y) &\overset{Bayes}{\propto} p(y|x_\tau)p(x_\tau) \\ p(x_\tau|x_{\tau-1}, y) &\overset{Bayes}{\propto} p(y|x_\tau)p(x_\tau|x_{\tau-1}) \end{aligned}$$

The recursion first computes the influence of x_τ on the observation, then computes the marginal and the transition probability.

Finding $p(x_\tau|x_{\tau-1}, y)$ for $\tau = t..T$. Starting from $\tau = t$, compute

$$\begin{aligned} p(x_\tau|y) &= \int \mathbf{d}x_{\tau-1}\, p(x_\tau|x_{\tau-1}, y)p(x_{\tau-1}|y) \\ p(x_\tau|x_{\tau-1}, y) &= p(x_\tau|x_{\tau-1}) \end{aligned}$$

The second identity follows from the independence structure on the right side of observed nodes.

5.3 Behavior of the Algorithm

Instead of computing $p(X|Y^{new})$, we have justified that it is sufficient to compute its transitions to obtain a good approximation. Intuitively, as each measurement appears on the chain, the poses in the loop are "stretched" appropriately to match the observation. Poses to the left of the loop are similarly relaxed to fit with the origin on one end and the beginning of the loop on the other. Poses to the right of the loop are simply shifted according to the end of the loop.

6 Results

We manually navigated a camera rig along two trajectories. The camera faced upward and recorded the ceiling. The robot took about 3 minutes to trace each path, producing about 6000 frames of data for each experiment. The floor was marked so that the camera revisited specific locations on the floor throughout its trajectory (see figure 4). This was done to make the evaluation of the results simpler.

In these experiments, the pose parameters were (x, y) locations on the floor. The dynamics were taken to

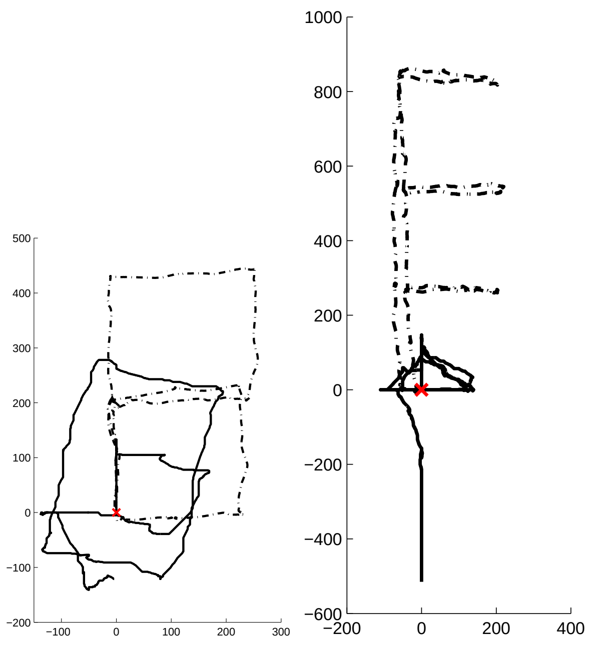

Figure 3: *Recovered pose. Naive accumulation (solid) and belief propagation (dashed) for two different experiments. The X marks the common starting position.*

be Gaussian Markov. For each new frame, at most three pose changes were computed. The selection of base frames was based on a measure of appearance between the current frame and all past frames. The pose change estimator was a Lucas-Kanade optical flow tracker [14]. To compute pose displacements, we computed a robust average of the flow vectors using an iterative outlier rejection scheme. We used the number inlier flow vectors as a crude estimate of the precision of $p(y_s^t|x_s, x_t)$.

The trajectory estimation worked at frame rate, although it was processed offline. Figure 3(left) compares the belief propagation algorithm against the accumulation of pose changes over time. Both algorithms used the same pose changes estimates and were run concurrently. Figure 4 depicts the trajectory the rig underwent. Figure 3(right) shows similar performance on a different path.

The naive accumulation of pose changes suffered from severe inaccuracies. In figure 3(right), the tracker recovered a spurious motion towards the left every time the camera moved downwards. This bias was corrected by the belief propagation algorithm because the upward trajectory had been recovered accurately. Note that all of the excursions away from the horizontal spine have similar lengths, as they should, and that the excursions are horizontally equidistant, as they should be. Also note that the trajectory correctly returns to its initial position. Since the rig did not retrace its steps exactly

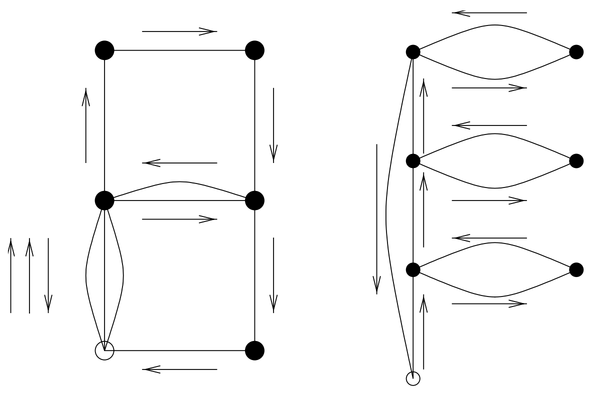

Figure 4: *Intended trajectories. Both path started at the bottom left. Circles indicated locations in the path where the camera was intentionally made to revisit the same place.*

on the return path to the home position, the return path does not meet the origins of the excursions on the recovered trajectory. The accumulation in figure 3(left) doesn't have a systematic flaw, other than underestimating the path length during the first leg, and suffering from lots of noise throughout the trajectory.

The belief propagation algorithm was helpful even on straightaways, before the trajectory ever crossed itself. Computing the pose of a frame while taking into account multiple back frames improved accuracy dramatically, as has been shown in [5].

7 Conclusions and Future Work

We have described an algorithm which computes globally consistent pose estimates using an efficient approximate belief propagation algorithm. The algorithm updates the trajectory for each frame and constructs a map of the world in the form of pose-attributed images. Its main strengths are that it maintains a simple world model and that it runs in linear time for each update. We expect to improve this running time by taking advantage of the fact that updates on a Markov chain have do not propagate beyond an unaffected node. So that in practice, it will be sufficient to run the left-right propagation updates on a small neighborhood.

We have observed that the algorithm not only returns consistent pose estimates near locations which have already been visited, but also improves accuracy for paths which do not close on themselves, by taking advantage of redundant pose change information.

We are currently working on producing results using a more sophisticated pose estimator based on [8], for computing globally consistent pose estimates in six degrees of freedom.

References

[1] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Uncertainty in Artificial Intelligence*, 1998.

[2] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. In *Porceedings of the IEEE Internation Conference on Robotics and Authomation*, pages 2724–2728, 1991.

[3] B. Curless. From range scans to 3d models. *Computer Graphics*, 33(4), november 1999.

[4] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Cambridge, Massachusetts, 1993.

[5] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV*, pages 311–326, 1998.

[6] G.Turk and M. Levoy. Zippered polygon meshes form range images. In *SIGGRAPH*, pages 311–318, 1994.

[7] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2000.

[8] M. Harville, A. Rahimi, T. Darrell, G.G. Gordon, and J. Woodfill. 3d pose tracking with linear depth and brightness constraints. In *ICCV99*, pages 206–213, 1999.

[9] T. Jebara and A. Pentland. Parametrized structure from motion for 3d adaptive feedback tracking of faces. In *CVPR*, 1997.

[10] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation.* Prentice Hall, 2000.

[11] M. LaCascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of textured-mapped 3d models. *PAMI*, 22(4):322–336, April 2000.

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

[13] Feng Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Robotics and Autonomous Systems*, 22(2):159–178, 1997.

[14] B. D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[15] Philip F. McLauchlan. A batch/recursive algorithm for 3d scene reconstruction. *Conf. Computer Vision and Pattern Recognition*, 2:738–743, 2000.

[16] T.P. Minka. Independence diagrams. Technical report, Media Lab, http://www.stat.cmu.edu/~minka/papers/diagrams.html, 1998.

[17] T.P. Minka. Expectation propagation for approximate bayesian inference. In *UAI*, 2001.

[18] T.P. Minka. Old and new matrix algebra useful for statistics. Technical report, Media Lab, http://www.media.mit.edu/~tpminka/papers/matrix.html, 2001.

[19] H. F. Moffitt and E. Mikhail. *Photogrammetry.* Harper and Row, 1980.

[20] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1997.

[21] K. Pulli. Multiview registration for large data sets. In *Int.Conf. on 3D Digital Imaging and Modeling*, pages 160–168, 1999.

[22] A. Rahimi, L-P. Morency, and T. Darrell. Reducing drift in parametric motion tracking. In *ICCV*, volume 1, pages 315–322, June 2001.

[23] Harpreet S. Sawhney, Steve Hsu, and Rakesh Kumar. Robust video mosaicing through topology inference and local to global alignment. In *Proc ECCV 2*, pages 103–119, 1998.

[24] H.S. Sawhney and R. Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(3):235–243, 1999.

[25] A. Shashua. Trilinearity in visual recognition by alignment. In *ECCV*, pages 479–484, 1994.

[26] H.-Y. Shum and R. Szeliski. Construction of panoramic mosaics with global and local alignment. In *IJCV*, pages 101–130, February 2000.

[27] A. Stoddart and A. Hilton. Registration of multiple point sets. In *IJCV*, pages B40–44, 1996.

[28] E. Sudderth. Embedded trees: Estimation of gaussian processes on graphs with cycles. Master's thesis, MIT, 2002.

[29] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[30] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[31] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.