



Reducing drift in differential tracking

Ali Rahimi ^{*}, Louis-Philippe Morency, Trevor Darrell

Massachusetts Institute of Technology, Computer Science and AI Lab, Cambridge, MA 02139, USA

Received 7 June 2005; accepted 8 December 2006

Available online 18 January 2007

Abstract

We present methods for turning pair-wise registration algorithms into drift-free trackers. Such registration algorithms are abundant, but the simplest techniques for building trackers on top of them exhibit either limited tracking range or drift. Our algorithms maintain the poses associated with a number of key frames, building a view-based appearance model that is used for tracking and refined during tracking. The first method we propose is batch oriented and is ideal for offline tracking. The second is suited for recovering egomotion in large environments where the trajectory of the camera rarely intersects itself, and in other situations where many views are necessary to capture the appearance of the scene. The third method is suitable for situations where a few views are sufficient to capture the appearance of the scene, such as object-tracking. We demonstrate the techniques on egomotion and head-tracking examples and show that they can track for an indefinite amount of time without accumulating drift.

© 2007 Elsevier Inc. All rights reserved.

Keywords: Tracking; Drift; View-based appearance models

1. Introduction

Robust algorithms for estimating the change in the pose of an object from one image to the next are abundant [34,10,22,3,2,11]. These estimators are an appealing component for building trackers because code for them is readily available, and they tend to be very robust and accurate. Furthermore, they abstract away details such as the image formation model, the computation of feature correspondences, and the optimization procedure for registering the images. Instead, we are left with the simpler problem of combining their reported pose change estimates into a pose for each frame. Unfortunately, the obvious approaches for doing this suffer from several problems. One such common approach is to accumulate the pose changes between successive pairs of frames to obtain the pose of the object in the current frame. Because each pose change is noisy, the error in the accumulated pose grows over time, and results in drift. An appealing alternative is to compute the pose

change between each incoming image and the first image in the image sequence. But because pairwise pose change estimation algorithms have limited range, this approach cannot work for arbitrarily large motions. To overcome the shortcomings of these approaches, we provide new methods that utilize pose-change estimators to build long-range trackers that exhibit limited drift.

Rather than computing pose changes between temporally adjacent frames, or between the current frame and the first frame, we propose computing pose changes between the current frame and a number of frames that previously appeared in the image sequence (we called these frames *key frames*). These pose changes are then merged to obtain a pose for the current frame. The pose of the key frames, are also subsequently updated. By updating the pose of the key frames (or of the entire trajectory) our tracker effectively maintains an image-based appearance model of the object to track. This appearance model can later be used to initialize the tracker on subsequent tracking sessions, or can be stitched into a 3D model of the object [17].

We propose three related methods for performing these updates. The first is a batch method, and serves as a good workhorse technique. The second method is an online

^{*} Corresponding author.

E-mail address: ali@mit.edu (A. Rahimi).

method well suited to egomotion problems, where capturing the appearance of the scene can require an indefinite number of key frames. This method retains all frames seen so far as candidate key frames, and updates the pose associated with them in sub-linear time. We demonstrate this technique by estimating the pose of a camera rig in a very large environment, and rely on an optical flow-based 2D tracker. The third method is an online method suitable for maintaining the appearance of relatively small objects, such as heads, that fit within the field of view of the camera, and can thus be captured with just a few key frames. This method recovers the pose of the objects and adjusts the pose of key frames using a Kalman update. Because its complexity does not grow over time, this method can track human heads without drift for an indefinite time. To demonstrate this, we rely on the 3D pose change estimator described in [27].

2. Related work

This article is a culmination of our work on building drift-free trackers [33,28,32]. Our batch algorithm is closely related to the problem of globally consistent registration, such as mosaicking of planar scenes [36,34] and stitching of laser range scans [39,31,6,37,4,21,9]. Our batch method is most similar to the method of Lu and Milios [21], which computes pose changes between pairs of laser range scans, and merges these pose differences into consistent poses by finding poses that are consistent with the measured pose changes. Other global registration methods take into account additional details about the scans, and so do not use pairwise registration as a subroutine: Stoddart and Hilton [37] attach virtual springs between corresponding points among all pairs of scans and relax the system to convergence. Chen and Medioni [4] rely on a version of the Iterated Closest Point (ICP) algorithm to iteratively compute correspondences and transformations between the scans. Sawhney et al. [34] perform global registration on the overlapping regions of the scans, which are assumed to be of planar structures.

The idea of anchoring tracking against landmarks to reduce drift was exploited in the rigid-body tracking work of Chiuso, Jin, Favaro, and Soatto [5,15]. Their tracker eliminates the drift incurred from fixing the coordinate system against a poorly estimated feature by anchoring the coordinate system against features whose pose is well-known when they reappear in the scene. Rather than anchoring against 3D patches, our methods anchor against images using a pose change estimator. In [40], the tracker is anchored against static 3D features that are provided to the tracker before hand.

The literature on simultaneous localization and mapping (SLAM) and structure from motion (SFM) and adaptive rigid body tracking [24,19,1,18,13,5,15,7] aims to update an appearance model of the environment (the map) or of an object while simultaneously tracking the pose of the object or of the camera. This body of work tends to differ from ours in that the appearance model is represented as a set of geometric features, such as 3D lines

and corners, or as an occupancy grid, rather than as a collection of pose-annotated frames. This representation is easier to operate on because it delegates explicit reasoning about geometry to the pose change estimation subroutine. A notable exception is [14] and related methods that capture appearance with a subspace model.

In the SLAM literature, both [38] and [9] represent the map as pose-annotated key frames, as we do. As loops in the trajectory are detected, a propagation step corrects backward poses in the loop. The updates are online, but increase in complexity linearly in the length of the loop in the case of [38] and cubically in the case of [9]. These algorithms are similar in purpose to the algorithm presented in Section 5.

Updating appearance models becomes more expensive as more features are introduced, so McLauchlan [24] proposed an efficient online algorithm based on the Kalman filter update that allows certain features to become fixed. Our algorithm of Section 6 is similar.

3. Tracking model

As each image becomes available, we compute its pose change with respect to several past key frames using an off-the-shelf pose change estimator, and combine these pose change measurements to update the estimate of the pose of the object throughout the sequence. To do this, we first describe a generative model for pose change estimators, and then describe algorithms for approximating the maximum *a posteriori* (MAP) estimate of the poses.

Let $y_{s,t}$ denote the measured pose change between frames t and s , and let $Y^T = \{y_{s,t}\}_{(s,t) \in \mathcal{O}}$ denote the set of measured pose changes up to time T , stacked vertically into a column vector. Let x_t denote the true (unknown) state of the object at time t . The state maintains information about the pose and the dynamics of the object. Let $X^T = \{x_t\}_{t=1 \dots T}$ be the state trajectory of the object from time $t=1$ to time $t=T$, stacked vertically into a column vector. Finally, let $x_{\mathcal{M}}$ denote the appearance model, which simply consists of the poses of a set of key frames.

We define a prior $p(X^T)$ on the trajectory of poses, and a likelihood $p(Y^T|X^T)$ on the trajectory, and show how to update $p(x_t, x_{\mathcal{M}}|Y^T)$, the posterior distribution over trajectories and the appearance model, as new pose changes become available. This distribution captures our estimate and uncertainty in the pose of the object and the appearance model given all measurements so far.

3.1. Trajectory prior $p(X^T)$

We will assume that the object's pose follows *a priori* Markovian dynamics:

$$p(X^T) = p(x_1) \prod_{t=2}^T p(x_t|x_{t-1}). \quad (1)$$

The transition density $p(x_t|x_{t-1})$ is a distribution over the state of the object at time t given only its state at time x_{t-1} .

We will assume linear Gaussian dynamics and let the state encode the pose of the object and the derivative of the pose.

For example, define the state $x_t = [u_t \ v_t \ \dot{u}_t \ \dot{v}_t]^\top$ to be the 2D location and 2D velocity to be tracked. A sensible state-transition model is:

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t|\mathbf{A}x_{t-1}, \mathbf{\Lambda}_x), \quad (2)$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3)$$

where $\mathcal{N}(x|\mu, \mathbf{\Lambda})$ denotes a multivariate Gaussian with mean μ and covariance $\mathbf{\Lambda}$. At each time step, this model updates the pose by adding the velocity and some noise to the previous pose, and updates the velocity by adding some random noise to it. Such a model can describe 3D motion by including rotation and angular velocity in the state. The recursive algorithms presented later can operate with nonlinear transition densities of the form $p(x_t|x_{t-1}) = \mathcal{N}(x_t|\mathbf{f}(x_{t-1}), \mathbf{\Lambda}_x)$ by linearizing \mathbf{f} at the current estimate of x_t at each iteration.

Eq. (2) can be re-written as

$$x_t = \mathbf{A}x_{t-1} + \omega_t, \quad (4)$$

where ω_t is a zero-mean Gaussian random variable with covariance $\mathbf{\Lambda}_x$.

3.2. Observation model $p(Y^T|X^T)$

For certain types of transformations, the ideal pose change $y_{s,t}^*$ is additively related to x_s and x_t , so that $x_t = y_{s,t}^* + x_s$. This holds, for example, if the poses represent translations. For other transformations, the relationship is multiplicative, so that $\hat{x}_t = \hat{y}_{s,t}^* \hat{x}_s$, where the $\hat{\cdot}$ operator reshapes a vector into a square matrix of the appropriate size (the inverse operation, $\text{vec}(\cdot)$, stacks the elements of a matrix into a column vector). To discuss both types of transformations in one framework, we use the notation $x_t = y_{s,t}^* \oplus x_s$, and write $y_{s,t}^* = x_t \ominus x_s$ to describe the inverse operation. To stay consistent with both multiplication and addition, these operators are associative but not commutative.

Observed pose changes are noisy versions of $x_t \ominus x_s$. The distribution $p(y_{s,t}|x_s, x_t)$ represents the uncertainty in pose-change estimation between two frames at known poses x_s and x_t . Each pose change $y_{s,t}$ is assumed to be mutually independent of other pose changes conditioned on x_s and x_t ¹:

¹ This independence assumption is a simplification: in general pose change measurements that share a frame are dependent, even when conditioned on the true poses. This is because the random variables $y_{s,t}|x_s, x_t$ and $y_{s,u}|x_s, x_u$ are obtained by registering images, so that $y_{s,t}|x_s, x_t = f(I_s, I_t)$ and $y_{s,u}|x_s, x_u = f(I_s, I_u)$. These images are in turn stochastic functions of the poses. Since $y_{s,t}|x_s, x_t$ and $y_{s,u}|x_s, x_u$ are built from a common random variable I_s , they are dependent on each other. The independence assumption is exact when I_s is a deterministic function of x_s , but in general, this independence assumption results in adequate performance and greatly simplifies the model.

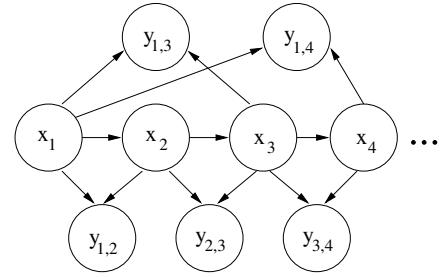


Fig. 1. Independence structure between trajectory and measured pose changes. Each hidden node x_t is the pose associated with a frame. The observed nodes $y_{s,t}$ are measured pose changes as recovered by a registration algorithm.

$$p(Y^T|X^T) = \prod_{(s,t) \in \mathcal{O}} p(y_{s,t}|x_s, x_t). \quad (5)$$

Fig. 1 depicts the independence relationship between poses and pose-change measurements.

To simplify computing the posterior distribution over X^T , we approximate the distribution $p(y_{s,t}|x_s, x_t)$ with a Gaussian, as explained in the appendix:

$$p(y_{s,t}|x_s, x_t) \approx \mathcal{N}(y_{s,t}|y^*, \mathbf{\Lambda}_{s,t}), \quad (6)$$

$$y^* = x_t \ominus x_s, \quad (7)$$

$$\mathbf{\Lambda}_{s,t} = \hat{\sigma}^2 \left[\sum_{p \in \mathcal{P}} \dot{u}(p, y^*)^\top \nabla I_t(p)^\top \nabla I_t(p) \dot{u}(p, y^*) \right]^{-1}, \quad (8)$$

$$\hat{\sigma}^2 = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} [I_s(i + u(i, y^*)) - I_t(i)]^2, \quad (9)$$

Under this approximation, the mean of the distribution is the true pose change. The covariance reflects the average sensitivity of the components of the registration model at each pixel, and the strength of the image features at that pixel. It is scaled by the reconstruction error $\hat{\sigma}^2$ after warping according to the recovered pose change. Equivalently, this can be re-written as

$$y_{s,t} = x_t \ominus x_s + \omega_{s,t}, \quad (10)$$

where $\omega_{s,t}$ is a zero-mean Gaussian with covariance $\mathbf{\Lambda}_{s,t}$.

4. Method 1: batch

The *a posteriori* most probable trajectory, given measurements Y^T , is defined as

$$X^* = \underset{X^T}{\operatorname{argmax}} p(X^T|Y^T) = \underset{X^T}{\operatorname{argmax}} p(Y^T|X^T)p(X^T). \quad (11)$$

X^* is the trajectory that is most consistent with the observations Y^T according to the generative model of the previous section.

When the likelihood $p(Y^T|X^T)$ is linear in X^T (for example, when \ominus is a simple difference) and $p(X^T)$ is Gaussian, X^* can be found by solving a sparse linear least-squares problem [33,21]. When $p(Y^T|X^T)$ is a Gaussian whose mean depends nonlinearly on X^T (when \ominus involves matrix inversion), X^* can be found by solving a sparse nonlinear least-squares problem.

4.1. Linear-Gaussian case

When $p(x_t|x_{t-1})$ is Gaussian and linear in x_{t-1} (for example, see Eq. (2)), X^T is a zero-mean Gaussian random variable with

$$p(X^T) \propto \exp(-(X^T)^\top \Lambda_X^{-1} X^T), \quad (12)$$

with the inverse covariance Λ_X^{-1} block tri-diagonal.

If the relationship between observation $y_{s,t}$ and x_s and x_t is additive, Eq. (7) becomes:

$$y_{s,t}^* = \mathbf{C}x_t - \mathbf{C}x_s = \mathbf{C}_{s,t}X^T, \quad (13)$$

$$\mathbf{C}_{s,t} = [0 \ \cdots \ -\mathbf{C} \ \cdots \ 0 \ \cdots \ \mathbf{C} \ \cdots], \quad (14)$$

The matrix \mathbf{C} extracts the pose components of the state x_t , so that, for example, if $x_t = [u_t \ v_t \ \dot{u}_t \ \dot{v}_t]^\top$, then $\mathbf{C}x_t = [u_t \ v_t]^\top$. The matrix $\mathbf{C}_{s,t}$ has $-\mathbf{C}$ at the location corresponding to s , \mathbf{C} at the location corresponding to t , and $\mathbf{0}$ elsewhere.

Stacking up the $\mathbf{C}_{s,t}$ into \mathcal{C} and $\Lambda_{s,t}$ into $\Lambda_{Y|X}$ gives the likelihood

$$p(Y^T|X^T) = \mathcal{N}(Y^T|\mathcal{C}X^T, \Lambda_{Y|X}). \quad (15)$$

This linear-Gaussian likelihood, in conjunction with the Gaussian prior (12) gives a Gaussian posterior $p(X^T|Y^T)$ over X^T . The covariance $\Lambda_{X|Y}$ and the mean $m_{X|Y}$ of this posterior can be obtained by solving these two equations [30]:

$$\Lambda_{X|Y}^{-1} = \Lambda_X^{-1} + \mathcal{C}^\top \Lambda_{Y|X}^{-1} \mathcal{C}, \quad (16)$$

$$\Lambda_{X|Y}^{-1} m_{X|Y} = \mathcal{C}^\top \Lambda_{Y|X}^{-1} Y^T. \quad (17)$$

When there are T trajectory steps and $|\mathcal{O}|$ observations, this computation can take $O(T^3)$ if performed naively. Using the conjugate gradient method, Eq. (17) can be solved in $O(T(T + |\mathcal{O}|))$: T iterations, each requiring a matrix multiplication that takes time proportional to the number of nonzero elements in $\Lambda_{X|Y}^{-1}$.

4.2. Nonlinear case

When Eq. (7) relates $y_{s,t}$ nonlinearly with x_t and x_s , we can linearize the relationship and apply the batch method of the previous section. The batch solution provides a new point about which to linearize the relationship, and the process is repeated.

In the case of affine transformations, for example, a first order Taylor series approximation of Eq. (7) about \hat{x}_t and \hat{x}_s gives [26]:

$$x_t \ominus x_s = \hat{x}_t \hat{x}_s^{-1} \approx \text{vec}(\hat{x}_t \hat{x}_s^{-1}) + d(\hat{x}_t \hat{x}_s^{-1}), \quad (18)$$

$$d(\hat{x}_t \hat{x}_s^{-1}) = (\mathbf{d}\hat{x}_t) \hat{x}_s^{-1} + \hat{x}_t d(\hat{x}_s^{-1}) \\ = (\mathbf{d}\hat{x}_t) \hat{x}_s^{-1} - \hat{x}_t \hat{x}_s^{-1} (\mathbf{d}\hat{x}_s) \hat{x}_s^{-1}, \quad (19)$$

$$\hat{\mathbf{d}}\hat{x}_s = \hat{x}_s - \hat{\hat{x}}_s, \quad (20)$$

$$\hat{\mathbf{d}}\hat{x}_t = \hat{x}_t - \hat{\hat{x}}_t. \quad (21)$$

Substituting terms, and applying the identity $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^\top \otimes \mathbf{A})\text{vec}(\mathbf{B})$, where \otimes is the Kronecker product [26], yields

$$y_{s,t}^* - \text{vec}(\hat{x}_t \hat{x}_s^{-1}) = (\hat{x}_s^{-\top} \otimes \mathbf{I})x_t - (\hat{x}_s^{-\top} \otimes \hat{x}_t \hat{x}_s^{-1})x_s, \quad (22)$$

where \mathbf{I} is the identity matrix. This relationship is now in the form of Eq. (13) and the method of the previous section can be applied.

Once the optimal x_s and x_t are found, the system is linearized again as per Eq. (22) and the operation is repeated.

4.3. Selecting pairs of frames

Applying the pose change estimator is the main computational bottleneck of the batch method. In practice, it is costly to compute the pose change between every pair of frame (s, t) , so \mathcal{O} must be chosen with care. In addition, registration algorithms work most reliably if the motion between the two views is small. For example, when tracking a head, the 6-degrees-of-freedom registration algorithm we use in our experiments returns a reliable pose estimate if the head has undergone a rotation of at most 10 degrees along any axis. To find frame pairs (s, t) that yield high-quality registration (as quantified by the trace of $\Lambda_{s,t}$) without actually performing registration on every pair of images, we search for pairs of images that are similar according to their sum of squared differences (SSD). If the SSD between frames s and t is within some threshold, (s, t) is inserted into \mathcal{O} and $y_{s,t}$ is computed using the registration algorithm.

5. Method 2: online updates

In online tracking, pose changes must be incorporated into the estimate of the trajectory X^T as soon as a new frame becomes available. We show an efficient way of approximately updating X^T with pose change measurements as they become available. Rather than re-running the batch algorithm of the previous section each time, we propose simplifying the *a posteriori* correlation between the pose estimates to allow measurements to be incorporated without solving the full system of Eqs. (16) and (17).

A new measurement is independent of past measurements conditioned on X^T , so the maximum *a posteriori* X^T can be updated recursively as follows:

$$\begin{aligned} \max_{X^T} p(X^T|Y^T) &= \max_{X^T} p(X^T|Y^{T-1}, y_{s,t}) \\ &= \max_{X^T} p(y_{s,t}|X^T)p(X^T|Y^{T-1}). \end{aligned} \quad (23)$$

This update uses $p(X^T|Y^{T-1})$ as a prior, and $p(y_{s,t}|X^T)$ as a likelihood. When $p(y_{s,t}|X^T)$ is linear and Gaussian (see (13)), and $p(X^T|Y^{T-1})$ is Gaussian with mean m_X and covariance Λ_X , the posterior over X^T can again be obtained in closed form by solving $\Lambda_{X|Y}$ and $m_{X|Y}$ [30]:

$$\Lambda_{X|Y}^{-1} = \Lambda_X^{-1} + \mathbf{C}_{s,t}^\top (\Lambda_{s,t})^{-1} \mathbf{C}_{s,t}, \quad (24)$$

$$\Lambda_{X|Y}^{-1} m_{X|Y} = \Lambda_X^{-1} m_X + \mathbf{C}_{s,t}^\top (\Lambda_{s,t})^{-1} y_{s,t}. \quad (25)$$

Solving for $m_{X|Y}$ using conjugate gradient takes $O(T^2 + T|\emptyset|)$. That is, incorporating a single $y_{s,t}$ takes as much work as incorporating all of Y^T all at once! Our solution is to approximate Λ_X by a tri-diagonal matrix before applying these updates. This makes $\Lambda_{X|Y}^{-1}$ invertible in time linear in T using forward–backward substitution, allowing us to incorporate new measurements in time $O(T)$.

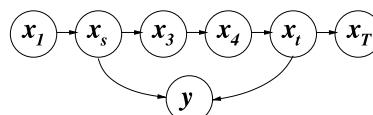
In this section, we present a method for simplifying $\Lambda_{X|Y}^{-1}$ to block tri-diagonal form after each measurement is inserted. Since $\Lambda_{X|Y}^{-1}$ serves as Λ_X for incorporating the next set of measurements, this ensures that all updates are in time $O(T)$ in the worst case, and in constant time in practice. For a set of Gaussian variables, a block tri-diagonal covariance matrix describes a Markovian independence structure. Intuitively, a Markov structure between the poses of chronologically adjacent frames expresses the prior knowledge that frames that are observed near the same time should have similar poses. We demonstrate in Section 7.1 that a simpler structure such as a diagonal matrix, which does not allow for such propagation, does not result in smooth trajectories. With minor adaptation, the method presented here can simplify $\Lambda_{X|Y}^{-1}$ to any tree structure, but the tri-diagonal or Markov structure is appealing because it is the simplest structure that allows the influence of measurements to propagate throughout the entire trajectory X^T , allowing loops to be closed and resulting in smooth trajectories.

The approximation strategy follows the pattern of assumed density filtering (ADF) [25]: at each iteration, $p(X^T|Y^{T-1})$ is approximated with a simpler distribution $q(X^T)$ that factors as a Markov chain. To incorporate a new measurement $y_{s,t}$, we apply Bayes rule with $q(X^T)$ as a prior and $p(y_{s,t}|x_s, x_t)$ as likelihood:

$$p(X^T|Y^T) \propto p(y_{s,t}|x_s, x_t) q(X^T). \quad (26)$$

To incorporate the next measurement, $p(X^T|Y^T)$ is again approximated with a simpler distribution q , and the process is repeated. Fig. 2 summarizes this process.

In the linear-Gaussian case, it is possible to compute in $O(T)$ a Gaussian approximation $q(X^T)$ with minimum Kullback-Leibler divergence to $p(X^T|Y^{T-1})$. One way to do this is



(a) Measurement appears.

to invoke tools for covariance extension (see, for example, [16,20]). The remainder of this section describes our variant of these techniques, which is formulated as a two-pass message-passing algorithm on a loop. This approach is particularly well suited for this problem because messages need not be propagated beyond a node if the influence of the message on the node is small. This allows early termination of the algorithm, allowing it to take time sub-linear in T .

5.1. Updating the trajectory

We would like to approximate an arbitrary distribution that factors according to $p(X) = \prod_t p_t(x_t|\text{Pa}[x_t])$ with one that factors according to the Markov property $q(X) = \prod_t q_t(x_t|x_{t-1})$. Here, $\text{Pa}[x_t]$ are the parents of node x_t in the graph prescribed by the factorization of $p(X)$. In Appendix B, we show that the distribution q that is closest to p according to the Kullback-Leibler divergence is

$$q_t(x_t|x_{t-1}) = p(x_t|x_{t-1}). \quad (27)$$

So the best conditional q_t is built up from the conditional marginals of p . Computing each q_t in a graph is generally an expensive operation, but in a graph with a single loop, it can be done in time proportional to the number of nodes in the graph.

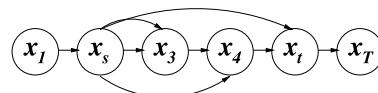
Instead of first computing $p(X^T|Y^T)$ and then simplifying it onto a suitable $q(X^T)$, we compute q directly from $p(X^T|Y^{T-1})$. We have just shown that this q factors into conditional distributions $q_t(x_t|x_{t-1}) = p(x_t|x_{t-1}, y_{s,t}, Y^{T-1})$. These factors are computed differently depending on their location in the graph. In what follows, we omit the dependence on Y^{T-1} for brevity, and assume that all distributions are conditioned on Y^{T-1} .

5.1.1. Finding $p(x_\tau|x_{\tau-1}, y_{s,t})$ for $s < \tau < t$

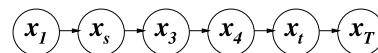
Because for every $s < \tau < t$, $p(x_\tau|x_{\tau-1}, x_t) = p(x_\tau|x_{\tau-1}, x_t, y_{s,t})$ (see Fig. 2), we have

$$p(y_{s,t}, x_{\tau-1}, x_\tau, x_t) = p(y_{s,t}, x_{\tau-1}, x_\tau) p(x_\tau|x_{\tau-1}, x_t), \quad (28)$$

We can find $p(x_\tau|x_{\tau-1}, y)$ by marginalizing out x_t from $p(y_{s,t}, x_{\tau-1}, x_\tau, x_t)$ and normalizing. We can also find $p(x_\tau|y)$ by marginalizing out both x_t and $x_{\tau-1}$ and normalizing. Finally, we can compute $p(y, x_\tau, x_t)$ for the next τ in the iteration.



(b) Hidden variables become correlated a posteriori.



(c) The correlation structure is approximated to allow new measurements to be incorporated easily.

Fig. 2. Using Assumed Density Filtering to maintain a Markov approximation to the correlation induced by measurements on the trajectory.

There are two missing pieces for computing $p(y_{s,t}, x_{\tau-1}, x_s, x_t)$: The first is $p(y_{s,t}|x_s, x_t) = p(y_{s,t}|x_s, x_t)p(x_s, x_t)$ for starting the recursion. $p(y|x_s, x_t)$ is the given measurement model, and $p(x_s, x_t)$ can be obtained by marginalizing over $p(X^T)$. 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. The recursion for calculating it is:

$$p(x_\tau|x_{\tau-1}, x_t) \propto p(x_t|x_\tau)p(x_\tau|x_{\tau-1}), \quad (29)$$

$$p(x_t|x_\tau) = \int dx_{\tau+1} p(x_t|x_{\tau+1})p(x_{\tau+1}|x_\tau). \quad (30)$$

The second equation describes a recursion that starts from t and goes down to s . It computes the influence of node τ on node t . Eq. (29) is coupled to this recursion and uses its output. Because this recursion runs in the opposite direction of the recursion described by (28), $p(x_\tau|x_{\tau-1}, x_t)$ has to be computed in a separate pass.

The forward and backward recursions visit every element of the chain once. To speed up computation, we stop a recursion in either direction when it does not modify the value of the current node significantly. This is an acceptable stopping criterion because a small change in a node guarantees that the iterations will not modify subsequent nodes significantly.

5.1.2. Finding $p(x_\tau|x_{\tau-1}, y_{s,t})$ for $1 \leq \tau \leq s$

Starting from $\tau = s - 1$, compute

$$p(y_{s,t}|x_\tau) = \int dx_{\tau+1} p(y_{s,t}|x_{\tau+1})p(x_{\tau+1}|x_\tau), \quad (31)$$

$$p(x_\tau|y_{s,t}) \propto p(y_{s,t}|x_\tau)p(x_\tau), \quad (32)$$

$$p(x_\tau|x_{\tau-1}, y_{s,t}) \propto p(y_{s,t}|x_\tau)p(x_\tau|x_{\tau-1}). \quad (33)$$

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

5.1.3. Finding $p(x_\tau|x_{\tau-1}, y_{s,t})$ for $t \leq \tau \leq T$

Starting from $\tau = t$, compute

$$p(x_\tau|y_{s,t}) = \int dx_{\tau-1} p(x_\tau|x_{\tau-1}, y_{s,t})p(x_{\tau-1}|y_{s,t}), \quad (34)$$

$$p(x_\tau|x_{\tau-1}, y_{s,t}) = p(x_\tau|x_{\tau-1}). \quad (35)$$

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

To handle nonlinear observations, one can again linearize the observation equation, as in Section 4.2. If so desired, after running the operation described in this section, the procedure may be repeated to yield a better answer. In this way, the procedure described here serves as an approximate Newton step.

5.2. Selecting frame pairs

At each time step t , the online algorithm must determine the set of frames $s < t$ to register against the frame at time t . Picking frames whose appearance is similar to the current

frame works well if there is a one-to-one mapping between appearance and pose. But in some situations, for example objects with repetitive texture such as floor tiles or a calibration cube with identical sides, different poses yield the same appearance. To disambiguate between these situations, key frames that sufficiently resemble the current frame in appearance are registered only if their pose is likely to be within tracking range of each other.

To assess the probability that the pose x_s of a potential base frame is within tracking range of the pose x_t of the current frame, we compute the probability that $|x_s - x_t|$ is above some tracking range Δx . If this probability is above a threshold, then the two frames are deemed likely to be close in pose and $y_{s,t}$ is measured. This probability can be estimated by evaluating

$$\int_{|x_s - x_t| \leq \Delta x} p(x_s, x_t | Y^{T-1}, y_{t-1,t}) dx_s dx_t, \quad (36)$$

which is the probability that x_s and x_t are within tracking range given all measurements prior to measuring the pose change between s and t . The mean and covariance of $p(x_s, x_t | Y^{T-1}, y_{t-1,t})$ can be read from the parameters of $p(X^T | Y^{T-1}, y_{t-1,t})$. Eq. (36) can be quickly approximated by Monte Carlo by drawing samples (x_s, x_t) from $p(x_s, x_t | Y^{T-1}, y_{t-1,t})$ and reporting the ratio of samples where $|x_s - x_t| \leq \Delta x$ to the total number of samples drawn.

6. Method 3: sparse key frames

For some tracking applications, only a few key frames are needed to capture the appearance of the scene. In these cases, instead of updating the entire trajectory history, we update the pose of the current frame and that of a few key frames only. The pose of these frames is maintained as a state vector using a Kalman filter. The prediction step of the filter can remove old entries or insert new entries in the state vector as needed. The update step of the filter incorporates new measurements. The resulting filter has a computational complexity that grows cubically in the size of the state vector, so in practice we limit the number of key frames to about 50.

At time t , the filter maintains a Gaussian distribution $p(x_t, x_{\mathcal{M}} | Y^T)$ over the pose of the current frame and the poses $x_{\mathcal{M}} = \{x_{M_1}, x_{M_2}, \dots\}$ of the key frames. Define the variable \mathcal{X} as the vertical concatenation x_t and $x_{\mathcal{M}}$:

$$\mathcal{X} = [x_t, x_{M_1}, x_{M_2}, \dots]. \quad (37)$$

To represent $p(x_t, x_{\mathcal{M}} | Y^T)$, the tracker need only maintain the mean $m_{\mathcal{X}}$ and covariance $\Lambda_{\mathcal{X}}$ of $p(\mathcal{X} | Y^T)$. To incorporate a new measurement $y_{s,t}$, the tracker applies a Kalman update to $p(\mathcal{X} | Y^{T-1})$ to obtain $p(\mathcal{X} | Y^{T-1}, y_{s,t})$. After all pairwise measurements involving frame t have been performed, the prediction step turns x_t into a key frame, and may decide to evict one of the older key frames. Then the tracker must prepare to incorporate pose change estimates involving a new frame at time $t + 1$. We examine each step separately.

6.1. Incorporating a new measurement

In the linear case, as before, incorporating $y_{s,t}$ with the prior $p(\mathcal{X}|Y^T)$ gives a Gaussian $p(\mathcal{X}|Y^T, y_{s,t})$ whose mean and covariance can be obtained by solving for $m_{\mathcal{X}|Y}^{new}$ and $\Lambda_{\mathcal{X}|Y}^{new}$ in

$$[\Lambda_{\mathcal{X}}^{new}]^{-1} = [\Lambda_{\mathcal{X}}]^{-1} + \mathbf{C}_{s,t}^\top \Lambda_{s,t}^{-1} \mathbf{C}_{s,t}, \quad (38)$$

$$[\Lambda_{\mathcal{X}}^{new}]^{-1} m_{\mathcal{X}}^{new} = [\Lambda_{\mathcal{X}}]^{-1} m_{\mathcal{X}} + \mathbf{C}_{s,t}^\top \Lambda_{s,t}^{-1} y_{s,t}. \quad (39)$$

The main computational burden in introducing a new measurement is solving for $m_{\mathcal{X}}^{new}$ in Eq. (39). $[\Lambda_{\mathcal{X}}^{new}]^{-1}$ will be dense after a few measurements, so solving the linear system takes cubic time in the number of key frames. We can bound the complexity of this update by capping the number of key frames. When pose change measurements are nonlinearly related to poses, the relationship can be linearized prior to applying Eqs. (38) and (39) using Eq. (22).

6.2. Preparing for a new frame

After incorporating all pose changes relating to frame t , the frame at time t becomes a key frame, and the mean and covariance of \mathcal{X} are augmented to make room for the pose x_{t+1} , to represent the distribution $p(x_{t+1}, x_t, \mathcal{X}_M | Y^T)$. If we have no *a priori* information about the pose x_{t+1} , we simply set the corresponding entry in the inverse covariance to 0. This translates to infinite marginal variance on x_{t+1} :

$$m_{\mathcal{X}}^{aug} = [0, m_{\mathcal{X}}^{new}], \quad (40)$$

$$[\Lambda_{\mathcal{X}}^{aug}]^{-1} = \begin{bmatrix} 0 & 0 \\ 0 & [\Lambda_{\mathcal{X}}^{new}]^{-1} \end{bmatrix}. \quad (41)$$

Since the augmented blocks of $\Lambda_{\mathcal{X}}$ are 0, we have assumed that *a priori*, x_{t+1} is independent of x_t . If there is side knowledge about dynamics, it can be incorporated in this step. $m_{\mathcal{X}}^{aug}$ and $\Lambda_{\mathcal{X}}^{aug}$ are used as $m_{\mathcal{X}}$ and $\Lambda_{\mathcal{X}}$ when incorporating the next measurement.

6.3. Evicting old key frames

When a key frame is no longer needed, it is marginalized out of the distribution $p(\mathcal{X}|Y^T)$. This is accomplished by removing the corresponding elements of $m_{\mathcal{X}}$ and the corresponding rows and columns from $\Lambda_{\mathcal{X}}$.

In our tracker, a frame $t - 1$ is always used as a base frame for frame t . Often, this frame is eliminated from the key frame set after estimating its pose change with frame t . In addition, older key frames are sometimes dropped to make room for key frames corresponding to poses that are more commonly visited.

6.4. Picking good key frames

To populate the key frames, we seek frames with accurate pose estimates that capture representative views of the object. A key frame should be available whenever the

object returns near a previously visited pose. To identify poses that the object is likely to revisit, the pose-space is tessellated into adjacent regions, each region maintaining a key frame. A key frame is assigned to a region if it can be ascertained that the pose of the key frame falls within the region with high probability.

After estimating the pose of frame t , the tracker determines whether frame $t - 1$ should become a key frame. The probability that x_{t-1} belongs to a region centered at x_r is:

$$\Pr[x_{t-1} \in B(x_r)] = \int_{x \in B(x_r)} \mathcal{N}(x|E[x_{t-1}], \Lambda_{t-1}) dx, \quad (42)$$

where $B(x)$ is the region centered around a location x , and $E[x_{t-1}]$ and Λ_{t-1} can be read from $m_{\mathcal{X}}^{new}$ and $\Lambda_{\mathcal{X}}^{new}$.

If frame x_{t-1} belongs to a region with higher probability than any other frame so far, it is deemed the best representative for that region, and it is assigned to that region. If the pose does not belong to any region with sufficiently high probability, or all regions already maintain key frames with higher probability, the frame is discarded.

These criteria exhibit several desirable properties: (1) Frames are assigned to regions near their estimated pose. (2) Frames with low certainty in their pose do not become key frames, because the integral of a Gaussian under a fixed volume decreases with the variance of the Gaussian. (3) key frames are replaced when better key frames are found for a given region.

7. Experiments

The online algorithm of Section 6 is well-suited for scenes or objects whose appearance can be captured with a few key frames. We apply it to a face tracking application that uses stereo cameras. The algorithm of Section 5 retains the pose of all frames seen so far, and is well suited for situations where the appearance of the scene or object cannot be captured well with a few frames. We demonstrate it on an egomotion experiment in a large environment whose appearance requires many views to capture. Since the batch algorithm of Section 4 can also potentially use all frames as key frames, we compare it against the online algorithm of Section 5 to evaluate the quality of the Markov chain simplification the latter performs.

7.1. Egomotion in a room

To demonstrate the online algorithm of Section 5, we manually maneuvered a monocular camera inside a large environment and attempted to recover the location of the camera. The camera faced upward and observed its motion relative to the ceiling. The excursions were about three minutes long, producing about 6000 frames of data for each experiment. The trajectory was marked on the floor before the experiment so we could revisit specific locations (see the schematics of Figs. 3 and 5). This was done to

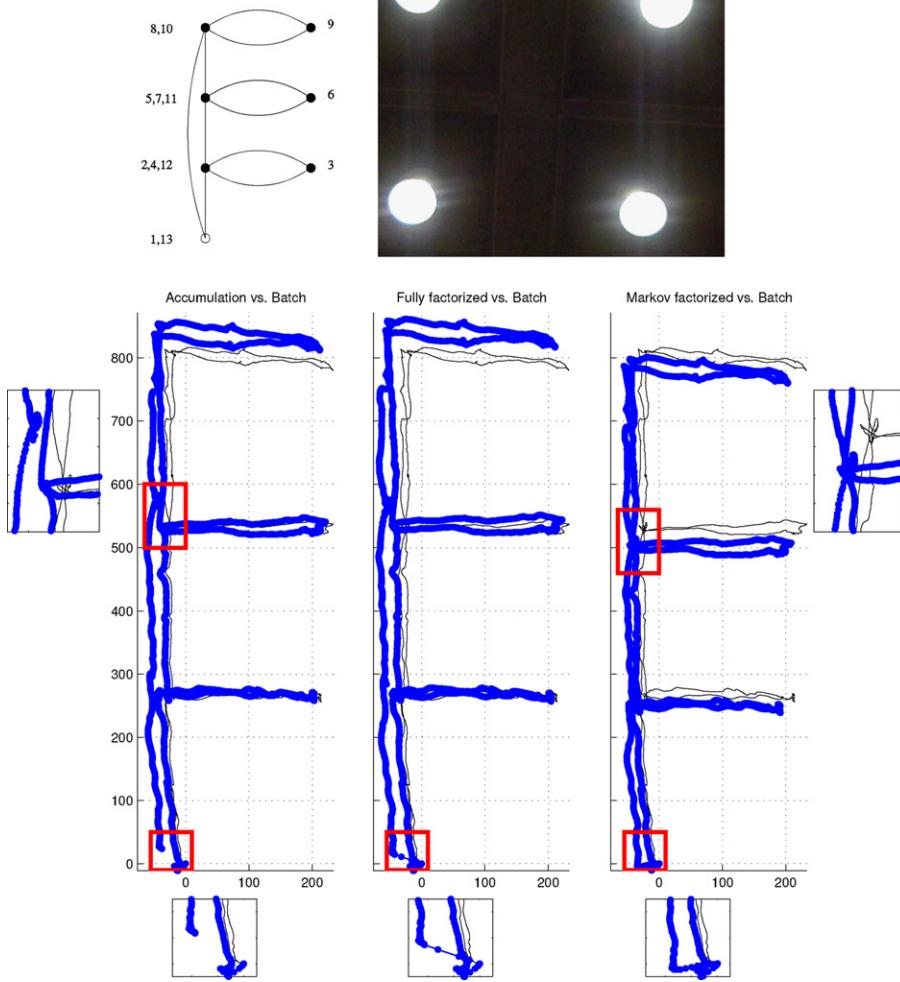


Fig. 3. Recovered trajectories from egomotion experiments. Axes are labeled in centimeters. The camera smoothly followed the trajectory shown in the schematic on the upper left, visiting the waypoints in the order specified by the numbers, starting with waypoint 1. The image in the upper right shows a typical snapshot of the ceiling. The bottom left panel compares the trajectory recovered by accumulating pose changes between successive frames (thick dotted path) to the batch solution of Section 4 (solid path). Loops are not closed well (bottom and left insets). The middle panel compares the trajectory recovered by fully factorizing the covariance after incorporating each measurement (dotted path) and the batch solution (solid path). Loops are closed abruptly, resulting in large jumps in the trajectory (bottom inset). The right panel compares the algorithm of Section 5 to the batch solution. Loops are closed, and the trajectory is smooth. The average distance between pose estimates between the batch solution and each online algorithm was 27 cm for naive accumulation, 26 cm for the fully factorized solution, and 23 cm for the Markov factorization solution.

make the evaluation of the results simpler. The trajectory estimation worked at frame rate for the duration of both trajectories, although it was processed offline to simplify data acquisition.

In these experiments, the pose parameters were (x, y) locations on the floor. All experiments assume the motion dynamics detailed in Section 3.1. For each new frame, pose changes were computed with respect to at most three base frames. The selection of base frames was based on a comparison of the appearances of the current frame and all past frames. The pose-change estimator was a Lucas-Kanade optical flow tracker [23]. To compute pose displacements, we computed a robust average of the flow vectors using an iterative outlier rejection scheme. We used the number of inlier flow vectors as a crude estimate of the precision of $p(y_{s,t}|x_s, x_t)$.

Figs. 3 and 5 compare the algorithm of Section 5 against three others: a naive differential tracker that accumulates

the pose change between adjacent frames, the batch approach of Section 4, and a variant of the online method of Section 4 that projects the covariance matrix to a fully factorized form after incorporating each pose change. Fig. 4 plots the distance between the location recovered by the batch method and each of the three algorithms. Although our recovered trajectories do not coincide exactly with the batch solutions, ours are smooth and consistent.

In contrast, more naive methods of reconstructing trajectories do not exhibit these two desiderata. Estimating the motion of each frame with respect to only the previous base frame yields an unsatisfactory trajectory. Furthermore, loops can not be closed correctly (for example, the robot is not found to return to the origin).

Projecting the correlation structure to a fully factored form is a very simple way of taking into account multiple base frames. This corresponds to using a diagonal matrix to represent the correlation between the poses (instead of

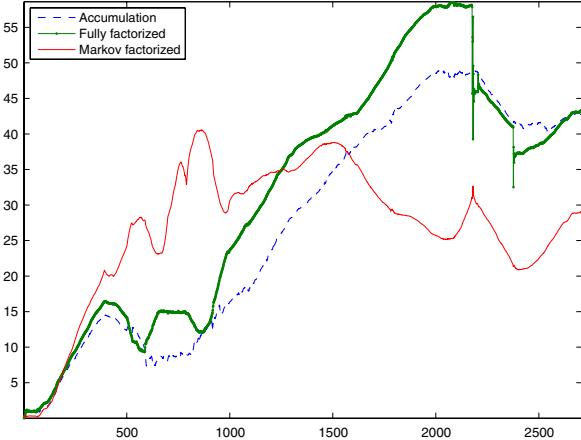


Fig. 4. The distance between the batch algorithm and the three other algorithms. While the Markov factorization method exhibits some shrinkage early in the sequence, it overtakes the other two algorithms later on.

the tri-diagonal inverse covariance matrix our algorithm uses). This method also fails to meet our requirements: the resulting trajectory is not smooth, and loops are not closed well.

The trajectories were each about 2000 frames long. The computational complexity of the online algorithm of 5 is linear in the number of frames in the worst case, but thanks to the stopping criterion of the algorithm, the first and last frames both about 5 ms to incorporate (MATLAB code on a 1.2 Ghz PIII), while incorporating a loop closure took about 20 ms.

By taking into account a minimum amount of correlation between frame poses, loops have been closed correctly and the trajectory is correctly found to be smooth.

7.2. Key-frame-based 6-DOF stereo tracking

In this experiment, we qualitatively compare three approaches for head-pose tracking: differential tracking, tracking using the first and previous frames as key frames, and the algorithm of Section 6. All three approaches use a 6-DOF (degree-of-freedom) registration algorithm (described in [27] and in the following subsection) to track and create an appearance model of a head undergoing large movements in the near field (approximately one meter from the camera) for several minutes. The camera system is a Videre Design stereo camera pair [41] that delivers depth and intensity maps in real time. In the second experiment, we present a quantitative analysis of our view-based tracking approach by comparing it with the *Inertia Cube*² inertial sensor.

7.2.1. 6-DOF registration algorithm

At each time step, the camera system provides an intensity image I_t and a depth image Z_t . Given frames (I_t, Z_t) and (I_s, Z_s) , the registration algorithm estimates a 6-DOF (3 rotation using the twist parameterization [29] and 3

translation parameters) pose change $y_{s,t}$ between these frames. It first identifies the object of interest by assuming that it is the front-most object in the scene, as determined by the range images Z_s and Z_t . For both frames, the foreground pixels are grouped into a connected component, and the background is masked out. The registration parameters are computed in several steps: First the centers of mass of the regions of interest are aligned in 3D translation. This provides a good starting point for aligning the images using 2D cross-correlation in the image plane. The output of this alignment provides a good initialization point for a finer-grained registration algorithm based on iterative closest point (ICP) and the brightness constancy constraint equation (BCCE).

The ICP algorithm iteratively computes correspondences between points in the depth images and finds the 6-DOF transformation parameters that minimize the distance between these pixels. By using depth values obtained from the range images, the BCCE can also be used to recover 3D pose-change estimates [10]. We have found that the BCCE provides superior performance in estimating rotations, whereas ICP provides more accurate translation estimates.

The registration technique of [27] minimizes the sum of the objective functions of ICP and BCCE iteratively, taking advantage of the strengths of both algorithms. At each step of the minimization, correspondences are computed for building the ICP cost function. Then the ICP and BCCE cost functions are linearized, and the locally optimal solution is found using a robust least-squares solver [12]. This process usually converges within 3–4 iterations. For more details, see [27].

7.2.2. Head-pose tracking

We tested our view-based approach with sequences recorded at 5 Hz. The tracking was performed using the method of Section 6. The pose space used for acquiring the view-based model was evenly tessellated in rotation only. On a 1.7 GHz Pentium 4, our C++ implementation of the tracking framework, including frame grabbing, 3D-view registration, and pose updates, runs at 7 Hz.

Tracking requires no manual intervention: the tracker searches for a frontal face in the scene using a face detector [42]. Because the face is frontal, we use this first frame to establish the origin. The system begins with no key frames, so accurate tracking in the early stages requires the user to move slowly. As more key frames are acquired, the system becomes robust to very fast movements. During steady state, the head tracker maintains about 50 key frames.

Fig. 6 shows tracking results from a video sequence in which the subject underwent rotations of about 110° and translations of about 80 cm, including translation along the Z-axis. While this sequence was 2 min long, in practice, tracking can continue indefinitely under these conditions. We have run experiments where the tracker was stable for 30 min, limited only by the patience of the subject. The tracker is tolerant of most hazards, including lighting variations and occlusions, as long as the segmentation

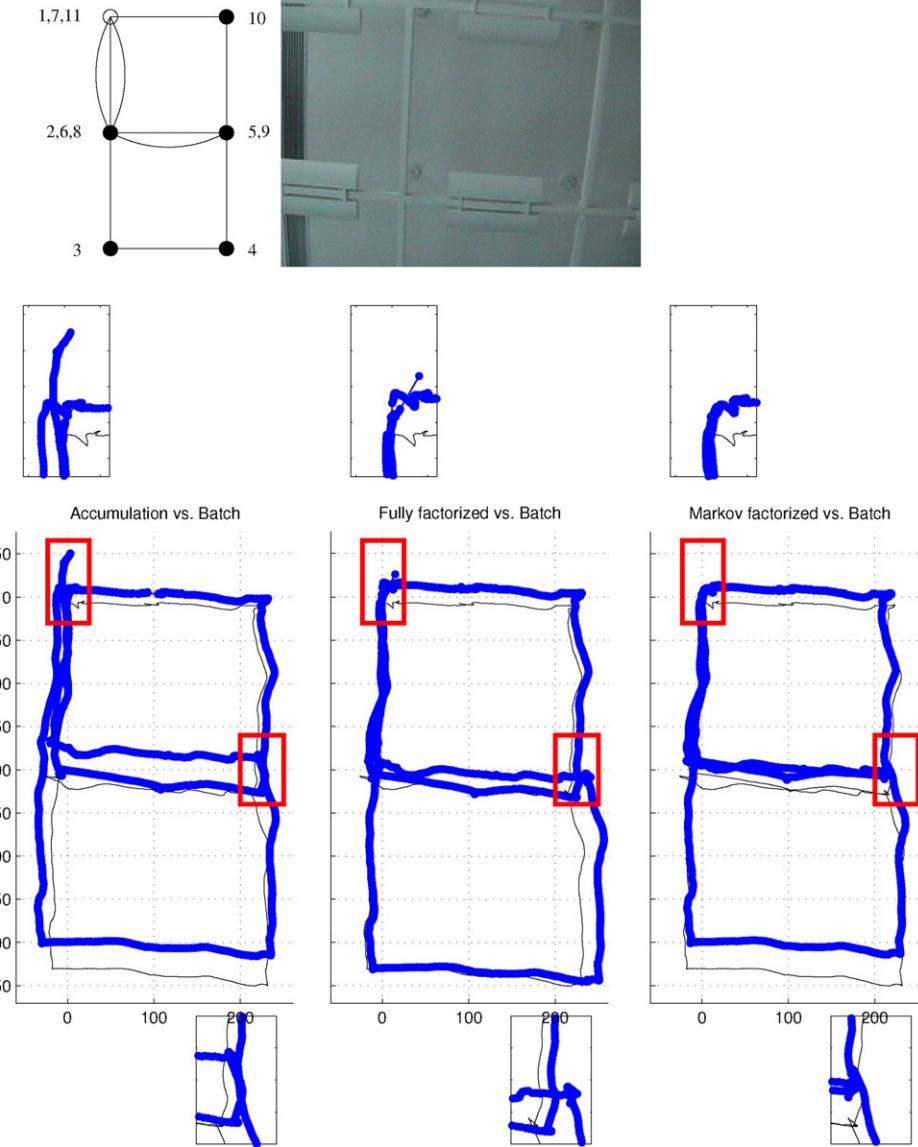


Fig. 5. See caption of Fig. 3 for the setup. Naive accumulation does not close loops well, full factorization results in abrupt jumps in the trajectory, and the Markov factorization results in smooth trajectories that close loops well. The average distance between pose estimates between the batch solution and each online algorithm was 26 cm for naive accumulation, 18 cm for the fully factorized solution, and 12 cm for the Markov factorization solution.

algorithm provides good foreground pixels. In the current implementation, the segmentation algorithm simply returns pixels near the camera, so intervening objects can interfere with tracking. The left column of Fig. 6 shows that a differential tracker based on our pose-change estimation algorithm drifts after a short while on this sequence. When tracking with only the first and previous frames as key frames (center column), the pose estimate is accurate when the subject is near-frontal but drifts when moving outside this region. The view-based approach (right column) gives accurate poses during the entire the sequence for both large and small movements. Usually, the view-based tracker used 2 or 3 base frames (including the previous frame) to estimate each pose.

Fig. 7 shows the key frames acquired during a tracking session. Fig. 8 demonstrates how loop closures reduce drift and cause key frames to undergo adjustments.

To quantitatively analyze our algorithm, we compared our results to an *Inertia Cube*² sensor from InterSense. *Inertia Cube*² is an inertial 3-DOF orientation tracking system. The sensor was mounted on the inside structure of a construction hat. By sensing gravity and the earth's magnetic field, *Inertia Cube*² estimates for the X- and Z-axes (where Z points outside the camera and Y points up) are mostly drift-free, but the Y-axis estimate can suffer from drift. InterSense reports an absolute pose accuracy of 3° RMS when the sensor is moving.

We recorded 4 sequences, using the *Inertia Cube*² sensor. The sequences were recorded at 6 Hz and their average length was 801 frames (~133 s). During recording, subjects underwent rotations of about 125° and translations of about 90 cm, including translation along the Z-axis. Fig. 9 shows the pose estimates of our adaptive view-based tracker for one of the sequences. Fig. 10 compares the

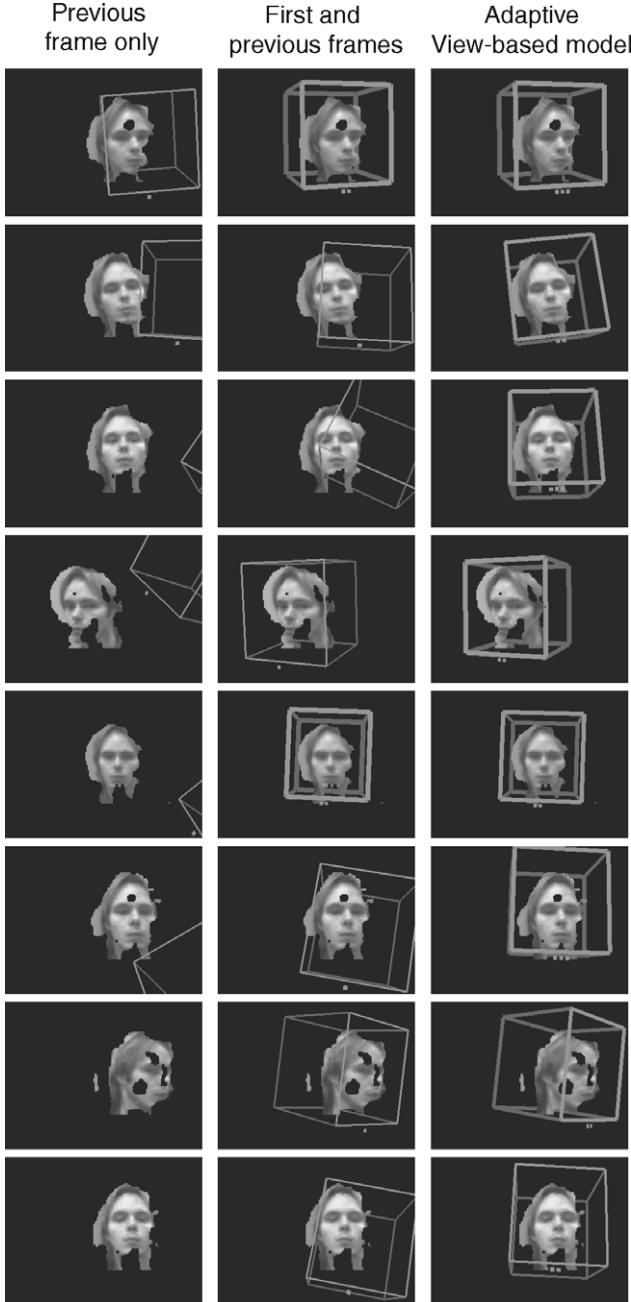


Fig. 6. Comparison of face tracking results using a 6-DOF registration algorithm. Rows represent results at 31.4, 52.2, 65, 72.6, 80, 88.4, 113, and 127 s into the video. Each image shows the foreground pixels and a rendered cube representing the pose of the head. The thickness of the lines defining the box around each face is inversely proportional to the uncertainty in the pose estimate (the determinant of the covariance of x_t , read from Λ_x). The number of indicator squares below the box indicate the number of base frames used during tracking. Background pixels and pixels where no range data was available are shown in black. Differential tracking (left column) drifts after a few seconds. Tracking with only the first and the previous frame is inaccurate for long excursions away from the first frame, but tracking does not drift (center column). Adaptively adding new key frames provides the best accuracy and suffers no drift.

tracking results of this sequence with those of the inertial sensor. Since the inertial sensor can only recover rotations, we only compare these parameters. The average root mean

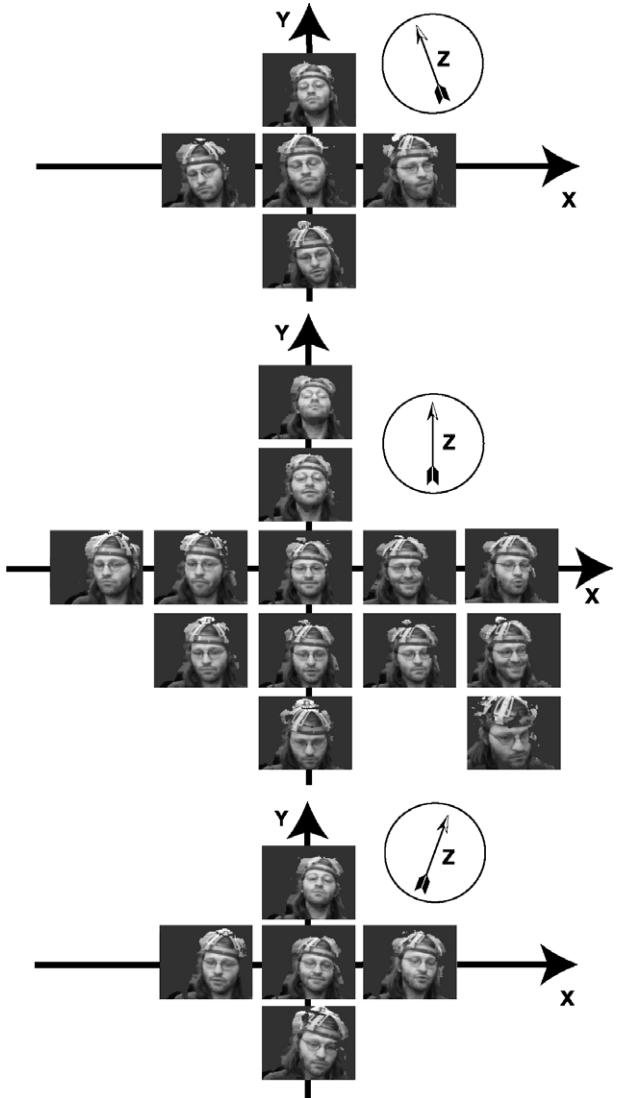


Fig. 7. The key frames acquired during one sequence, organized according to their pose. The key frames are spread evenly in the space of rotations to provide additional base frames for as many frames as possible.

squared distance to the output of the inertial sensor is 2.8° , which means our results are accurate to within the resolution of the *Inertia Cube*² sensor.

8. Conclusion

We have shown how to turn pose-change estimators into accurate drift-free trackers. Our algorithms use pose-change estimators to track poses while simultaneously updating a view-based appearance model. This makes it relatively easy to build trackers, since pose change estimators are available off-the-shelf. We derived a Gaussian approximation to represent the uncertainty in the output of parametric pose-change estimators. Three tracking recipes were presented, each suitable for different situations. The batch method is exact and uses all frames as key frames. The online method of Section 5 is appropriate for situations where many key frames are needed, but it grows linearly in the number of

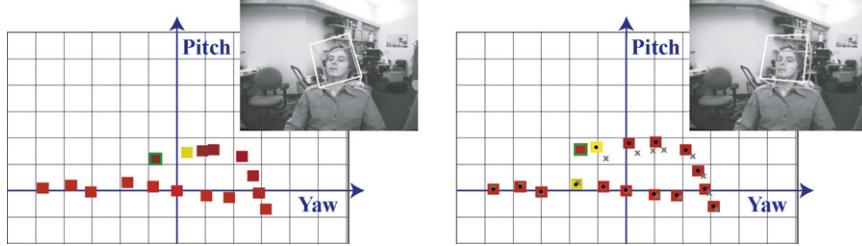


Fig. 8. Because key frames are correlated, closures can refine the pose of many key frames. The left panel shows two degrees of rotation of 15 key frames (solid squares) and of the current estimate of the pose of the head (outlined square). The light square is the key frame used for registration. It can be seen from the misalignment of the overlaid cube that the estimate pose of the head is slightly wrong. When the next frame becomes available, the current frame is compared against two key frames. Because one of these key frames was acquired much earlier, this closes a loop (right panel). The estimate of the pose of the head is corrected, and the key frames on the upper leg are adjusted as a result of the closure (the crosses depict the position of these key frames in the left panel).

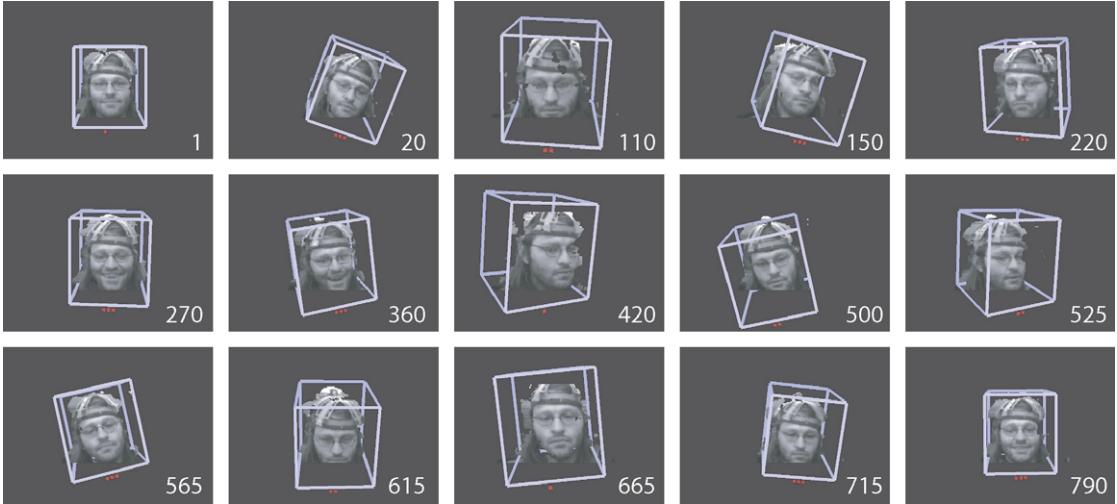


Fig. 9. To gather ground truth, the subject wore on his head an inertial sensor, whose output is compared with that of our adaptive view-based tracker in Fig. 10.

frames used. By contrast, the complexity of the online method of Section 6 does not grow over time. This algorithm is well suited for situations where the appearance of the scene can be captured with a small number of key frames. Our experiments showed that these trackers can run for a very long time without significant drift.

Appendix A. Uncertainty in parametric pose-change estimation

Some pose change estimators do not provide an uncertainty measure over their output. We present here a generic form for a distribution $p(y_{s,t}|x_s, x_t)$ that captures the uncertainty in the pose change estimate obtained from a large class of pose change estimators. We assume that $p(y_{s,t}|x_s, x_t)$ can be approximated by a Gaussian $\mathcal{N}(y_{s,t}|y^*, \Lambda_{s,t})$, where $y^* = x_t \ominus x_s$ is the true pose change, and proceed to derive a form for $\Lambda_{s,t}$. We assume that the pose change estimator minimizes a least-squares registration error using a known motion model².

Note that $p(y_{s,t}|x_s, x_t)$ is a function of the true poses, but that registration algorithms operate on images. When the appearance of an image is largely governed by its pose, so that the distributions $p(I|x)$ are peaked, we have $p(y_{s,t}|x_s, x_t) \approx p(y_{s,t}|I_s, I_t)$ as a function of $y_{s,t}$. We therefore proceed by finding a Gaussian approximation to the distribution $p(y_{s,t}|I_s, I_t)$.

We assume that the pose change estimators in question are parametric motion estimators that search for the mode of the following distribution as a function of $y_{s,t}$:

$$p(y_{s,t}|I_s, I_t) \propto \exp \left(-\frac{1}{2\sigma^2} \sum_{i \in \mathcal{P}} [I_s(i + u(i, y_{s,t})) - I_t(i)]^2 \right), \quad (\text{A.1})$$

where the summation is over the pixels \mathcal{P} of I_t , and σ^2 is an unknown variance parameter. The function u warps a pixel i by an amount dictated by its second argument. Finding the mode of this distribution amounts to finding the pose change that minimizes the residual between the two images under the family of warpings u .

To approximate $p(y_{s,t}|I_s, I_t)$, we substitute the maximum likelihood estimate of σ , and set the mean y^* of the approx-

² An alternative idea, suggested by one of the anonymous reviewers, is to learn $p(y_{s,t}|x_s, x_t)$ from labeled data.

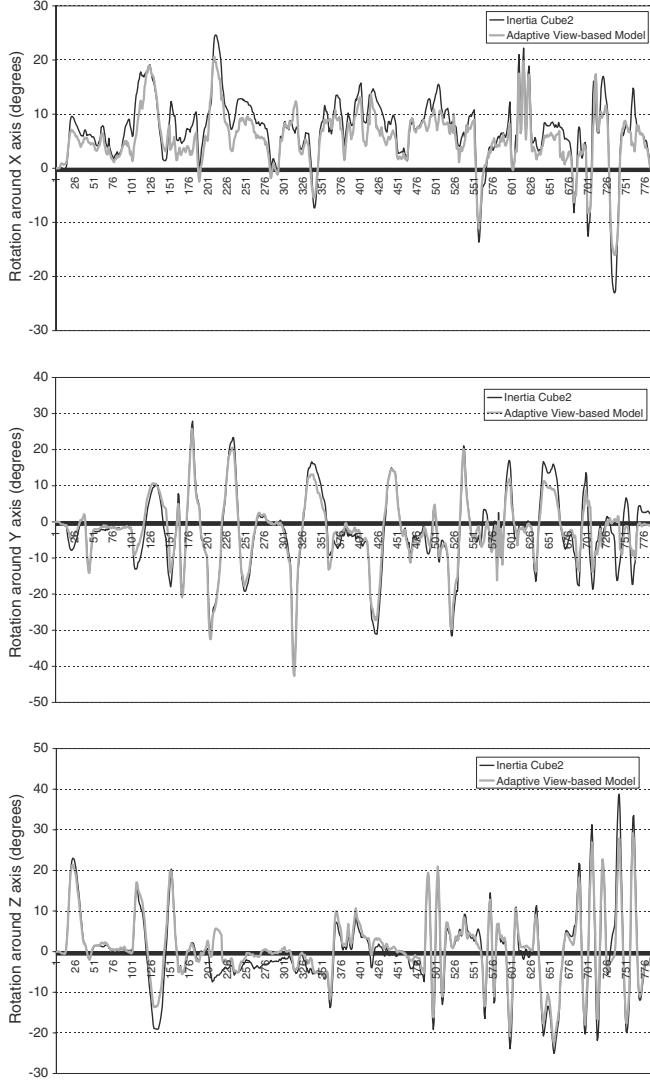


Fig. 10. Comparison of the head-pose estimation from our adaptive view-based approach with the measurements from the *Inertia Cube²* sensor. The *Inertia Cube²* only recovers rotations, so only the rotation axes are shown. The root mean squared difference between the output of the inertial sensor and our adaptive view-based tracker was about 2.8°, which is within the accuracy of the *Inertia Cube²* itself.

imating Gaussian to $x_t \ominus x_s$, which will be near a mode of $p(y_{s,t}|I_s, I_t)$ when there is not much imaging noise. The covariance $\Lambda_{s,t}$ will be set to the inverse curvature of $\log p(y_{s,t}|I_s, I_t)$ at $y_{s,t} = y^*$ following Laplace's approximation [8].

To justify this choice of covariance matrix, take the second order Taylor series expansion of $\log p(y_{s,t}|I_s, I_t)$ about y^* . The first order term vanishes because the first derivative is zero near the mode:

$$\log p(y_{s,t}|I_s, I_t) \approx \log p(y^*|I_s, I_t) + (y_{s,t} - y^*)^\top \mathbf{H}(y_{s,t} - y^*). \quad (\text{A.2})$$

Exponentiating both sides gives a Gaussian approximation to the posterior:

$$\begin{aligned} p(y_{s,t}|I_s, I_t) &\approx \kappa \exp((y_{s,t} - y^*)^\top \mathbf{H}(y_{s,t} - y^*)) \\ &= \mathcal{N}(y_{s,t}|y^*, -\frac{1}{2}\mathbf{H}^{-1}). \end{aligned} \quad (\text{A.3})$$

This approximation effectively fits a Gaussian near the mode of $p(y_{s,t}|I_s, I_t)$, matching its curvature there.

The Hessian \mathbf{H} of $\log p(y_{s,t}|I_s, I_t)$ at $y_{s,t} = y^*$ is

$$\begin{aligned} \mathbf{H} = -\frac{1}{\hat{\sigma}^2} \sum_{i \in \mathcal{P}} \dot{u}(i, y^*)^\top \nabla I_s(i + u(i, y^*))^\top \nabla I_s(i \\ + p(i, y^*)\dot{u}(i, y^*) + r(y^*)\nabla^2 I_s(i + u(i, y^*))), \end{aligned} \quad (\text{A.4})$$

where $r(y^*)$ is the residual of the images after warping I_s by y^* . At y^* , the residual is very small, and $I_s(i + u(i, y^*)) \approx I_t(i)$, so \mathbf{H} can be approximated by

$$\mathbf{H} \approx -\frac{1}{\hat{\sigma}^2} \sum_{i \in \mathcal{P}} \dot{u}(i, y^*)^\top \nabla I_t(i)^\top \nabla I_t(i) \dot{u}(i, y^*). \quad (\text{A.5})$$

Finally, the maximum likelihood estimate of σ can be found by differentiating $p(y_{s,t}|I_s, I_t)$ with respect to σ , and setting to zero:

$$\hat{\sigma}^2 = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} [I_s(i + u(i, y^*)) - I_t(i)]^2, \quad (\text{A.6})$$

where $|\mathcal{P}|$ denotes the number of pixels in \mathcal{P} . Putting everything together, we get the Gaussian approximation

$$p(y_{s,t}|x_s, x_t) \approx \mathcal{N}(y_{s,t}|y^*, \Lambda_{s,t}), \quad (\text{A.7})$$

$$y^* = x_t \ominus x_s, \quad (\text{A.8})$$

$$\Lambda_{s,t} = \hat{\sigma}^2 \left[\sum_{i \in \mathcal{P}} \dot{u}(i, y^*)^\top \nabla I_t(i)^\top \nabla I_t(i) \dot{u}(i, y^*) \right]^{-1}. \quad (\text{A.9})$$

Eq. (A.9) has an intuitive interpretation. The variance $\hat{\sigma}^2$ is the RMS reconstruction error after warping according to the recovered pose change. The summation measures the average sensitivity of each component of u , weighted by the strength of the features in the image. This is because $\nabla I_t(i)^\top \nabla I_t(i)$ represents the strength of a feature at location x (see [35]), and $\dot{u}(i)$ is the sensitivity to $y_{s,t}$ at various points in the image.

In the translational case, $u(i, y) = i + y$. So $\frac{\partial}{\partial y} u(i, y) = \mathbf{I}$. The covariance becomes

$$\Lambda_{\text{translation}} = \hat{\sigma}^2 \left[\sum_{i \in \mathcal{P}} \nabla I_t(i)^\top \nabla I_t(i) \right]^{-1}, \quad (\text{A.10})$$

which is just the reconstruction error weighted by a measure of how textured the image is.

In the case of an affine tracker, the partial of u is:

$$\frac{\partial}{\partial y} u(i, y) = \begin{bmatrix} i_1 & i_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & i_1 & i_2 & 1 \end{bmatrix}. \quad (\text{A.11})$$

If we set $\nabla I_t(i)^\top \nabla I_t(i) = \mathbf{I}$, effectively assigning the same texture to all points, the covariance becomes

$$\Lambda_{aff} = \hat{\sigma}^2 \sum_{i \in \mathcal{P}} \begin{bmatrix} i_1^2 & i_1 i_2 & i_1 & \\ i_1 i_2 & i_2^2 & i_2 & \mathbf{0} \\ i_1 & i_2 & 1 & \\ & i_1^2 & i_1 i_2 & i_1 \\ \mathbf{0} & i_1 i_2 & i_2^2 & i_2 \\ & i_1 & i_2 & 1 \end{bmatrix}^{-1}. \quad (\text{A.12})$$

Consistent with intuition, points away from the center of the coordinate system reduce the uncertainty in the multiplicative portion of the affine transformation more than the central points. In addition all points contribute equally to the translation parameters.

Appendix B. Minimum KL-divergence simplification of a factored distribution

We would like to approximate a distribution $p(X) = \prod_t p_t(x_t | \text{Pa}[x_t])$ with a distribution $q(X) = \prod_t q_t(x_t | \text{Qa}[x_t])$ whose factors $q_t(x_t | \text{Qa}[x_t])$ depend on a subset of the variables that appear in the corresponding factor p_t (ie, $\text{Qa}[x_t] \subset \text{Pa}[x_t]$). We want q to be as close as possible to p in the KL-divergence sense, where the KL-divergence between two distributions p and q is defined as $\text{KL}(p\|q) = \int_X p(X) \ln \frac{p(X)}{q(X)}$. It is well known that the closest such q is obtained by dropping the additional edges from the factors of p .

To see this, expand the KL divergence:

$$\text{KL}(p\|q) = \int_X p(X) \ln \frac{\prod_t p_t(x_t | \text{Pa}[x_t])}{\prod_t q_t(x_t | \text{Qa}[x_t])} \quad (\text{B.1})$$

$$= \int_X p(X) \sum_t \ln \frac{p_t(x_t | \text{Pa}[x_t])}{q_t(x_t | \text{Qa}[x_t])} \quad (\text{B.2})$$

$$= \sum_t \int_{x_t, \text{Pa}[x_t]} p(x_t, \text{Pa}[x_t]) \ln \frac{p_t(x_t | \text{Pa}[x_t])}{q_t(x_t | \text{Qa}[x_t])}. \quad (\text{B.3})$$

Since each term in the summation can be optimized over q_t separately, after dropping terms that do not depend on q and flipping signs, we get

$$q_t^*(x_t | \text{Qa}[x_t]) = \arg \max_{q_t} \int_{x_t, \text{Pa}[x_t]} p_t(x_t, \text{Pa}[x_t]) \ln q_t(x_t | \text{Qa}[x_t]) \quad (\text{B.4})$$

$$= \arg \max_{q_t} \int_{\text{Qa}[x_t]} p(\text{Qa}[x_t]) \int_{x_t} p_t(x_t | \text{Qa}[x_t]) \ln q_t(x_t | \text{Qa}[x_t]) \quad (\text{B.5})$$

$$= p_t(x_t | \text{Qa}[x_t]). \quad (\text{B.6})$$

The last statement follows because the inner integral in Eq. (B.5) is the KL divergence between $p_t(x_t | \text{Qa}[x_t])$ and $q_t(x_t | \text{Qa}[x_t])$.

References

- [1] N. Ayache, O. Faugeras, Maintaining representations of the environment of a mobile robot, *IEEE Transactions on Robotics and Automation* 5 (6) (1989) 804–819.
- [2] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, Hierarchical model-based motion estimation, in: European Conference on Computer Vision (ECCV), 1992, pp. 237–252.
- [3] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, *Pattern Analysis and Machine Intelligence (PAMI)* 14 (2) (1992) 239–256.
- [4] Y. Chen and G. Medioni, Object modelling by registration of multiple range images, in: International Conference on Robotics and Automation, 1991, pp. 2724–2728.
- [5] A. Chiuso, P. Favaro, H. Jin, S. Soatto, Structure from motion causally integrated over time, *Pattern Analysis and Machine Intelligence (PAMI)* 24 (4) (2002) 523–535.
- [6] B. Curless, From range scans to 3D models, *Computer Graphics* 33 (4) (1999) 38–41.
- [7] D. DeCarlo, D. Metaxas, Adjusting shape parameters using model-based optical flow residuals, *Pattern Analysis and Machine Intelligence (PAMI)* 24 (6) (2002) 814–823.
- [8] A. Gelman, J. Carlin, H. Stern, D. Rubin, *Bayesian Data Analysis*, Chapman and Hall/CRC, 1995.
- [9] J. Gutmann and K. Konolige, Incremental mapping of large cyclic environments, in: IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), 2000.
- [10] M. Harville, A. Rahimi, T. Darrell, G.G. Gordon, and J. Woodfill, 3D pose tracking with linear depth and brightness constraints, in: International Conference on Computer Vision (ICCV), 1999, pp. 206–213.
- [11] B.K.P. Horn, E.J. Weldon Jr., Direct methods for recovering motion, *International Journal of Computer Vision (IJCV)* 2 (1) (1988) 51–76.
- [12] P.J. Huber, *Robust statistics*, Addison-Wesley, New York, 1981.
- [13] T. Jebara and A. Pentland, Parametrized structure from motion for 3D adaptive feedback tracking of faces, in: Computer Vision and Pattern Recognition (CVPR), 1997, pp. 144–152.
- [14] A.D. Jepson, D.J. Fleet, and T. El-Maraghi, Robust on-line appearance models for visual tracking, in: Computer Vision and Pattern Recognition (CVPR), 2001, pp. 415–422.
- [15] H. Jin, P. Favaro, S. Soatto, A semi-direct approach to structure from motion, *The Visual Computer* 19 (2003) 1–183.
- [16] A. Kavčić, J.M.F. Moura, Matrices with banded inverses: inversion algorithms and factorization of gauss-markov processes, *IEEE Transactions on Information Theory* 46 (4) (2000) 1495–1509.
- [17] T. Darrell, L.-P. Morency, A. Rahimi, Fast 3D model acquisition from stereo images, in: 3D Data Processing, Visualization and Transmission (3DPVT), 2002.
- [18] M. LaCascia, S. Sclaroff, V. Athitsos, Fast, reliable head tracking under varying illumination: an approach based on registration of textured-mapped 3D models, *Pattern Analysis and Machine Intelligence (PAMI)* 22 (4) (2000) 322–336.
- [19] J.J. Leonard and P.M. Newman, Consistent, convergent, and constant-time SLAM, in: International Joint Conferences on Artificial Intelligence (IJCAI), 2003, pp. 1143–1150.
- [20] H. Lev-Ari, S.R. Parker, T. Kailath, Multidimensional maximum-entropy covariance extension, *IEEE Transactions on Information Theory* 35 (1989) 497–508.
- [21] F. Lu, E. Milios, Globally consistent range scan alignment for environment mapping, *Autonomous Robots* 4 (1997) 333–349.
- [22] F. Lu, E. Milios, Robot pose estimation in unknown environments by matching 2D range scans, *Robotics and Autonomous Systems* 22 (2) (1997) 159–178.
- [23] B.D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, in: International Joint Conference on Artificial Intelligence, 1981, pp. 674–679.
- [24] P.F. McLauchlan, A batch/recursive algorithm for 3D scene reconstruction, *Computer Vision and Pattern Recognition (CVPR)* 2 (2000) 738–743.
- [25] T.P. Minka, Expectation propagation for approximate bayesian inference, in: Uncertainty in Artificial Intelligence (UAI), 2001, pp. 362–369.

- [26] 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.
- [27] L.-P. Morency and T. Darrell, Stereo tracking using ICP and normal flow constraint, in: International Conference on Pattern Recognition (ICPR), 2002, pp. 367–372.
- [28] L.-P. Morency, A. Rahimi, and T. Darrell, Adaptive view-based appearance models, in: Computer Vision and Pattern Recognition (CVPR), 2003, pp. 803–812.
- [29] R.M. Murray, Z. Li, S.S. Sastry, A Mathematical Introduction to Robotic Manipulation, CRC Press, 1994.
- [30] A. Papoulis, Probability, random variables, and stochastic processes, third ed., McGraw-Hill, New York, 1991.
- [31] K. Pulli, Multiview registration for large data sets, in: Int. Conf. on 3D Digital Imaging and Modeling, 1999, pp. 160–168.
- [32] A. Rahimi and T. Darrell, Location estimation with a differential update network, in: Neural Information Processing Systems (NIPS), 2002, pp. 1049–1056.
- [33] A. Rahimi, L.-P. Morency, and T. Darrell, Reducing drift in parametric motion tracking, in: International Conference on Computer Vision (ICCV), vol. 1, 2001, pp. 315–322.
- [34] H.S. Sawhney, S. Hsu, and R. Kumar, Robust video mosaicing through topology inference and local to global alignment, in: European Conference on Computer Vision (ECCV), 1998, pp. 103–119.
- [35] J. Shi and C. Tomasi, Good features to track, in: Computer Vision and Pattern Recognition (CVPR), 1994, pp. 593–600.
- [36] H.-Y. Shum and R. Szeliski, Construction of panoramic mosaics with global and local alignment, in: International Journal of Computer Vision (IJCV), 2000, pp. 101–130.
- [37] A. Stoddart and A. Hilton, Registration of multiple point sets, in: International Conference on Pattern Recognition (ICPR), vol. 2, 1996, pp. 40–44.
- [38] S. Thrun, A probabilistic online mapping algorithm for teams of mobile robots, International Journal of Robotics Research 20 (5) (2001) 335–363.
- [39] G. Turk and M. Levoy, Zippered polygon meshes form range images, in: SIGGRAPH, 1994, pp. 311–318.
- [40] L. Vaccetti, V. Lepetit, and P. Fua, Fusing online and offline information for stable 3-D tracking in real-time, in: Computer Vision and Pattern Recognition (CVPR), 2003, pp. 241–249.
- [41] Videre Design, MEGA-D megapixel digital stereo head, <http://www.ai.sri.com/konolige/ssv>, 2000.
- [42] P. Viola, M. Jones, Robust real-time face detection, International Journal of Computer Vision (IJCV) 57 (2004) 137–154.