Reducing Drift in Parametric Motion Tracking

A. Rahimi L.-P. Morency T. Darrell

MIT AI Lab
Cambridge, MA 02139

Abstract

We develop a class of differential motion trackers that automatically stabilize when in £nite domains. Most differential trackers compute motion only relative to one previous frame, accumulating errors inde£nitely. We estimate pose changes between a set of past frames, and develop a probabilistic framework for integrating those estimates. We use an approximation to the posterior distribution of pose changes as an uncertainty model for parametric motion in order to help arbitrate the use of multiple base frames. We demonstrate this framework on a simple 2D translational tracker and a 3D, 6-degree of freedom tracker.

1. Introduction

Tracking the pose of an object requires that image transformation parameters be recovered for each frame of a video sequence. A common class of approaches for estimating these parameters involves accumulating motion parameters between pairs of temporally adjacent frames. These differential techniques suffer from accumulated drift which limits their effectiveness when dealing with long video sequences. The proposed method reduces this drift by anchoring each frame to many past frames. We then use a maximum likelihood formalism to fuse these pose change estimates to obtain poses which exhibits less error.

Various methodologies for avoiding drift have been proposed. For example, [2] and [5] compute the pose of an object by bringing it into registration with the £rst frame in the video sequence. This approach restricts the range of appearances to be near the initial pattern unless complicated model acquisition techniques are employed. Another approach is to use subject-independent models that are re£ned over time ([1, 9]), but the accuracy of these methods is often limited by the coarseness of their models, though strong prior motion models can sometimes be used to obtain better accuracy (eg, [14]).

In this paper we show how typical differential tracking algorithms can be stabilized without changing the core structure of the tracker. We relax the restriction that only temporally adjacent frames will be used for differential tracking, allowing high-quality pose change measurements to compensate for poor quality ones. We compute pose changes between each frame and several anchor frames that are close in pose and appearance to it. These differential motion estimates are then combined to provide a robust estimate of pose for each frame. Conceptually, previous frames are used as an image-based model of the object being tracked, alleviating the need to construct an explicit model of the scene as is done in [11] and [4], for example.

The next section provides a maximum likelihood framework for differential tracking. We then augment this model to incorporate additional anchor frames. In order to £nd the maximum likelihood poses in this augmented model, it is necessary to measure the uncertainty in each pose estimate, so we develop an error measure for parametric pose estimation. We then discuss details involved in implementing our algorithm and apply our framework to a simple 2D tracking problem where camera motion is restricted to frontoparallel translation over a synthetic planar object. Experiments in sections 4.1 and 4.2 show how to augment the 6-DOF tracker of [3] with our framework and demonstrate its use in tracking heads through large rotations and computing egomotion in long sequences.

2. Differential Tracking as Maximum Likelihood

We propose a measurement model suitable for representing differential trackers. We then frame our drift-reduced tracker in this model by adding additional measurement nodes. In order to cast tracking as a maximum likelihood problem, we develop an error model for estimating parametric pose change.

2.1. A Measurement Model

Consider a sequence of images $y_0 \cdots y_t$ with associated object poses $\xi_0 \cdots \xi_T$. Let $\delta_0^1 = d(\xi_1, \xi_0)$ be the pose change between frames with pose ξ_0 and ξ_1. If the parametrization is additive, d just subtracts ξ_0 from ξ_1. In the af£ne case, d computes $A^{-1}[A(\xi_1)A(\xi_0)^{-1}]$, where A returns a 3x3 af£ne matrix given a 6 dimensional vector, and

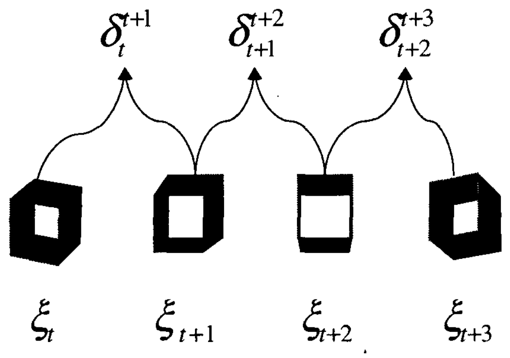

Figure 1: Independence diagram for a simple pose tracker. The tracker measures pose differences $\{\delta\}$ between adjacent frames.

A^{-1} returns the six parameters of the af£ne transformation given an af£ne matrix. We also de£ne d^{-1} such that $d^{-1}(d(\xi_1, \xi_0), \xi_0) = \xi_1$. Estimating the pose change between frames y_{t-1} and y_t results in a pose difference δ_{t-1}^t with distribution $p(\delta_{t-1}^t | y_{t-1}, y_t)$.

Assuming that pose governs everything about appearance, δ_{t-1}^t is conditionally independent of y_{t-1} and y_t given ξ_{t-1} and ξ_t, so $p(\delta | y_{t-1}, y_t) = p(\delta | \xi_{t-1}, \xi_t)$[1]. Figure 1 depicts the resulting independence diagram for a differential tracker. The joint density of measurements $\{\delta\}$ and poses $\{\xi\}$ is

$$p(\{\xi\}, \{\delta\}) = p(\{\xi\}) \prod_{t=1}^{T} p(\delta_{t-1}^t | \xi_{t-1}, \xi_t)$$

Finding the set of ML poses $\{\xi\}$ involves computing

$$\begin{aligned} &\arg\max_{\{\xi\}} p(\{\xi\} | \{\delta\}) \\ &= \arg\max_{\{\xi\}} \sum_{t=1}^{T} \ln p(\delta_{t-1}^t | \xi_{t-1}, \xi_t) \qquad (1) \end{aligned}$$

We can show that the traditional method of computing pose changes and updating pose estimates is in fact the ML solution by assuming that the performance of the tracker depends only on pose change and not on absolute pose. As a result, $p(\delta | y_{t-1}, y_t) = p(\delta | d(\xi_t, \xi_{t-1}))$. Making a £nal Gaussianity assumption on the posterior, we obtain:

$$p(\delta | \xi_t, \xi_{t-1}) = N(\delta_{t-1}^t; d(\xi_t, \xi_{t-1}), \Lambda_{t,t-1}). \qquad (2)$$

Equation (1) can now be rewritten as

$$\arg\min_{\{\xi\}} \sum_{t=1}^{T} \|\delta_{t-1}^t - d(\xi_t, \xi_{t-1})\|_{\Lambda_{t,t-1}}. \qquad (3)$$

[1]This implies that given the pose, there is no other source of uncertainty in the appearance of a frame. As will be shown later, imager noise is funnelled into $p(\delta | y_{t-1}, y_t)$ by other means, alleviating the need for a cumbersome integration step here.

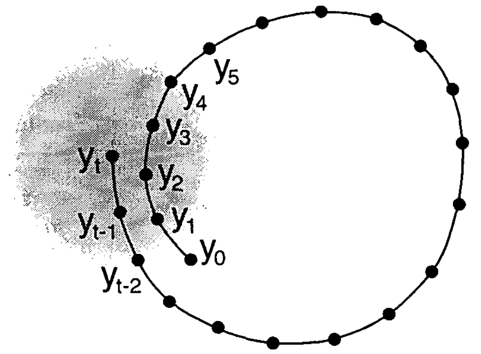

Figure 2: When estimating the pose of frame y_t, we should take into account the pose change between y_t and y_{t-1} as well as all other frames which are in the shaded region.

The minimum value for this problems is 0, and occurs when

$$\begin{aligned} \delta_{t-1}^t &= d(\xi_t, \xi_{t-1}) \\ \xi_t &= d^{-1}(\delta_{t-1}^t, \xi_{t-1}), \end{aligned} \qquad (4)$$

con£rming that the traditional update equation does indeed maximize likelihood given the simplifying assumptions we've made. Note that $\Lambda_{t,t-1}$ drops out of the optimization, and so it is not necessary to compute the error in pose changes.

2.2. Using multiple base frames to reduce drift

To improve pose estimation, we invoke two principal insights:

1. When the trajectory comes close to crossing itself (ie, $\xi_t \approx \xi_s, t > s$), tracking should be performed between frames y_t and y_s as well.

2. Information about the pose of future frames can be used to adjust the pose estimate of past frames.

Proposition 1) provides redundant reliable information which allows us to better estimate pose. Proposition 2) is appealing since returning near a previously visited point can disambiguate measurements if information from the future is allowed to affect the past. Hence, in £gure 2, we would do well to compute a pose change estimate between y_t and all frames that lie in the shaded region, and allow these measurements to in¤uence the pose of frames $y_1 \cdots y_t$.

We augment the measurement model laid out in the previous section to incorporate these additional measurements. To improve performance, we can also incorporate knowledge about the dynamics of the pose parameters. Figure 3 shows how to update the graphical model of the differential

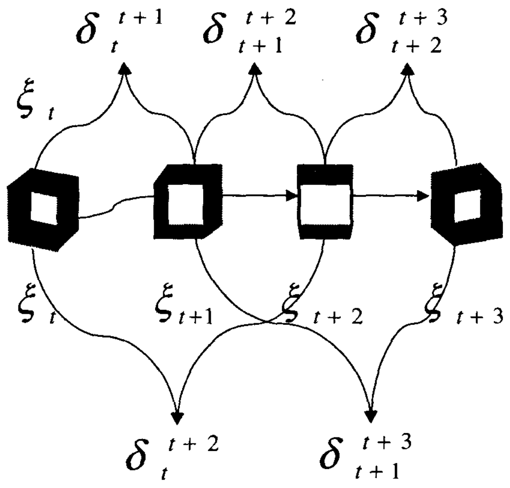

Figure 3: The measurement model when multiple base frames are used. A dynamical model for pose change is also added (horizontal arrows).

tracker to incorporate the added information. The joint of the poses and observations becomes

$$p(\{\xi\},\{\delta\}) = p(\xi_0)\prod_{t=1}^{T} p(\xi_t|\xi_{t-1}) \prod_{(f,g)\in D} p(\delta_f^g|\xi_f,\xi_g)$$

where D is the set of pairs of frames between which we have calculated the pose change. Using the Gaussian uncertainty model of (2), the ML poses are

$$\begin{aligned}\arg\min_{\{\xi\}} \quad & \sum_{(f,g)\in D} \|\delta_{t-1}^t - d(\xi_t,\xi_{t-1})\|_{\Lambda_{t,t-1}} \\ + & \sum_{t=1}^{T} \|d(\xi_t,\xi_{t-1})\|_{\Lambda_d} \end{aligned} \tag{5}$$

where we have assumed that the pose dynamics are Brownian with covariance Λ_d. The optimization problem can be thought of as relaxing a spring system where the natural length of a spring between nodes ξ_f and ξ_g is δ_f^g and its stiffness is $\Lambda_{f,g}^{-1}$.

Unlike the minimization problem of the traditional tracker, we now need to know $\Lambda_{f,g}$. An approximation to $\Lambda_{f,g}$ is derived in the following two sections.

2.3. Estimating Pose Change

The simplest pose change tracker computes the maximum likelihood pose difference $\hat{\delta}_{t-1}^t$ by assuming that y_t can be warped back to y_{t-1}. Camera noise and any change in appearance that is not modelled by warping is modelled with identically distributed and independent Gaussian noise of unspeci£ed variance added to every pixel. The generative model of y_{t-1} is then:

$$\begin{aligned} y_{t-1}(x) &= y_t(x-u(x;\delta_{t-1}^t)) + \mathbf{w}(x) \\ p(y_{t-1}(x)|y_t(x),\delta_{t-1}^t) &= \mathcal{N}(y_{t-1}(x); \\ & \quad y_t(x-u(x;\delta_{t-1}^t)),\sigma_w^2) \end{aligned} \tag{6}$$

where $\mathbf{w(x)}$ is Gaussian and white over space and time, and has constant variance σ_w^2 over the image. $\mathcal{N}(x;\mu,\sigma^2)$ is a Gaussian distribution with means μ and variance σ^2. $u(x;\delta)$ is the warping function: it is used to displace a pixel at location x to location $x+u(x;\delta)$ in the target image. The ML estimate, $\hat{\delta}$, maximizes the posterior $p(\delta|y_t,y_{t-1})$. This is equivalent to minimizing a sum-of-squared error function over δ:

$$\begin{aligned}\hat{\delta} &= \operatorname*{argmax}_{\delta} p(\delta|y_t,y_{t-1}) \\ &= \operatorname*{argmin}_{\delta} \sum_x [y_{t-1}(x) - y_t(x-u(x;\delta))]^2 \end{aligned} \tag{7}$$

This is the traditional least squares formulation for tracking, derived in a probabilistic framework. Various total-least squares formulations which allow y_t to be noisy as well have been proposed [13, 8]. We have demonstrated that pose change estimation computes the mode of the distribution $p(\delta|y_t,y_{t-1})$. To fully qualify this distribution, we still need to compute its covariance $\Lambda_{t,t-1}$.

2.4. Uncertainty in motion estimates

Probabilistic methods for computing uncertainty in optical ¤ow have been proposed in [12, 8]. We approximate the posterior $p(\delta|y_t,y_{t-1})$ by £tting a Gaussian distribution at the mode $\hat{\delta}$ computed by the pose estimator. The derivation is based on the approximation made in Laplace's method (see [6] for a note on the subject).

Using Bayes rule, we can rewrite the log-posterior:

$$\begin{aligned}\log p(\delta|y_t,y_{t-1}) &= \log p(y_{t-1}|\delta,y_t) \\ &+ \log p(\delta|y_t) - \log p(y_{t-1}|y_t)\end{aligned} \tag{8}$$

Since $\hat{\delta}$ is taken to be the ML estimate, the £rst derivative of (8) vanishes at $\hat{\delta}$. Assuming uniform $p(\delta|y_t)$ (this is the case if $p(\delta)$ is itself uniform, since we can glean nothing about future poses from a single image), the Hessian of (8) becomes

$$\mathbf{H} = \frac{\partial^2}{\partial\delta^2}\log p(\delta|y_t,y_{t-1}) = \frac{\partial^2}{\partial\delta^2}\log p(y_{t-1}|y_t,\delta)$$

The Taylor expansion of (8) about its mode is therefore:

$$\begin{aligned}\log p(\delta|y_t,y_{t-1}) &\approx \log p(\hat{\delta}|y_t,y_{t-1}) \\ &+ \frac{1}{2}\Delta\delta^T\frac{\partial^2}{\partial\delta^2}\log p(y_{t-1}|y_t,\hat{\delta})\Delta\delta \\ &+ \text{H.O.T.,}\end{aligned}$$

where $\Delta\delta = \delta - \hat{\delta}$. Dropping high order terms and exponentiating, we obtain a Gaussian approximation to the posterior:

$$p(\delta|y_t, y_{t-1}) \approx \alpha \exp\left(\frac{1}{2}(\delta - \hat{\delta})^T \mathbf{H}(\delta - \hat{\delta})\right)$$

This Gaussian has mean $\hat{\delta}$ as expected, and its variance is $\Lambda_{t,t-1} = -\mathbf{H}^{-1}$. In this case, $\mathbf{H}$ is the Hessian of the log of

$$\begin{aligned} p(y_{t-1}|y_t, \hat{\delta}) &= \prod_x p(w_t(x)) \\ &= \prod_x N(y_{t-1}(x); y_t(x - u(x; \hat{\delta})), \sigma_w^2), \end{aligned}$$

which is found to be

$$\mathbf{H} = \frac{1}{\sigma_w^2} \sum_x \frac{\partial u}{\partial \delta}^T (\tilde{y}_{t-1} \nabla_x^2 \hat{y}_{t-1} - \nabla \hat{y}_t \nabla \hat{y}_t^T) \frac{\partial u}{\partial \delta},$$

where $\hat{y_{t-1}} = y_t(x - u(x; \hat{\delta}))$ is the reconstructed y_{t-1} and $\tilde{y}_{t-1}(x) = y_{t-1}(x) - \hat{y}_{t-1}(x)$ is the reconstruction residual[2]. Since in practice the reconstruction error is small, we can further approximate $\mathbf{H}$ by:

$$\mathbf{H} = -\frac{1}{\sigma_w^2} \sum_x \frac{\partial u}{\partial \delta}^T \nabla y_{t-1}(x) [\nabla y_{t-1}(x)]^T) \frac{\partial u}{\partial \delta},$$

Finally, σ_w^2 can be estimated as

$$\hat{\sigma}_w^2 = \frac{1}{N} \sum_x [y_{t-1}(x) - y_t(x - u(x; \hat{\delta}))]^2. \qquad (9)$$

Our final estimate of the variance of $p(\delta|y_t, y_{t-1})$ is:

$$\Lambda_{t,t-1} = \hat{\sigma}_w^2 \left[\sum_x \frac{\partial u}{\partial \delta}^T \nabla y_{t-1}(x) \nabla y_{t-1}(x)^T \frac{\partial u}{\partial \delta} \right]^{-1}.$$

This expression has an intuitive interpretation which makes it suitable as an approximation of the posterior covariance. $\hat{\sigma}_w^2$ can be interpreted as the RMS reconstruction error after warping according to the recovered pose change. $\mathbf{H}$ can be interpreted as the average sensitivity of each component of δ, weighted by the strength of the features in the image. This is because $\nabla y(x) \nabla y(x)^T$ represents the strength of a feature at location x (see [10]), and $\frac{\partial u}{\partial \delta}(x; \delta)$ is a measure of the sensitivity of δ at various points in the image.

To illustrate this point, we compute the sensitivity of a translational and an affine tracker. In the translational case, $u(x; \delta) = \delta$. So $\frac{\partial}{\partial \delta} u(x; \delta) = \mathbf{I}$. The covariance becomes

$$\Lambda_{translation} = \hat{\sigma}_w^2 \left[\sum_x \nabla y \nabla y^T \right]^{-1}, \qquad (10)$$

[2]In deriving this expression, we have assumed that $\partial^2 u / \partial \delta^2 = 0$. ie, u is linear wrt δ.

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:

$$\frac{\partial}{\partial \delta} u(x; \delta) = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}.$$

If we set $\nabla y_{t-1}(x) \nabla y_{t-1}(x)^T = \mathbf{I}$, effectively assigning to all points the same feature properties, the covariance becomes

$$\Lambda_{affine} = \hat{\sigma}_w^2 \left[\sum_x \begin{bmatrix} \begin{matrix} x^2 & xy & x \\ xy & y^2 & y \\ x & y & 1 \end{matrix} & \mathbf{0} \\ \mathbf{0} & \begin{matrix} x^2 & xy & x \\ xy & y^2 & y \\ x & y & 1 \end{matrix} \end{bmatrix} \right]^{-1}.$$

According to this expression, 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. Both observations are consistent with our expectation.

3. Results: A simple 2D tracker

We first show results when tracking the position of an aperture moving over an image. ξ_t represents the current pixel location of the aperture and y_t denotes the image captured through the aperture. Since ξ only parametrizes translation, a simple motion model with $u(x; \delta) = \delta$ is adequate. Figure 4 shows the pose estimates from a differential tracker which finds pose changes by minimizing (8) using gradient descent. The update is according to (4) and is additive.

The algorithm estimates the pose change between consecutive 50x50 pixel windows which translate by an average of 5.6 pixels each step along a spiral path. The average error in estimating δ is around 0.66 pixels, which after 626 iterations, results in approximately 55 pixels of drift.

To measure the uncertainty of the pose change estimator, we used the pose covariance from equation (10). Figure 4 displays tracking performance on the same aperture trajectory. The previous frame was always used as an anchor frame, along with the 3 past frames which were closest in pose to the previous frame. In 626 frames, tracking drifts by at most 2.44 pixels and is off by 0.11 pixels at frame 623. Figure 5 compares the pose error of the two trackers over time. The the drift-reduced tracker stops accumulating error after about 50 frames, while the unenhanced tracker continues drifting.

To find the poses which maximize equation (5), we computed the derivatives of the log-likelihood with respect to

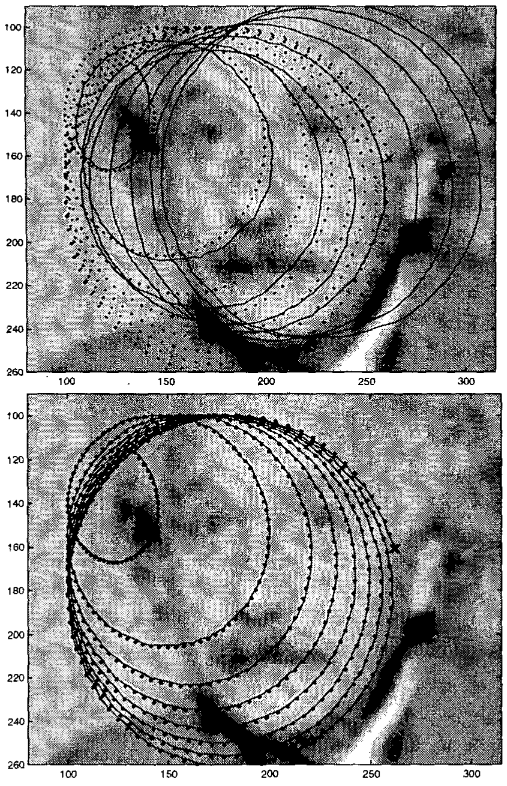

Figure 4: Estimates of the position of a 50x50 pixel aperture as it follows a spiral path on the image. The position estimate is based solely on the image acquired through the aperture. Top: traditional tracker. The estimated trajectory (solid) terminates (marked by 'x') with more than 55 pixels of error relative to ground truth (dotted). Bottom: drift-reduced tracker, using at most 4 past frames. The estimated trajectory ends less than 1 pixels from the ground truth.

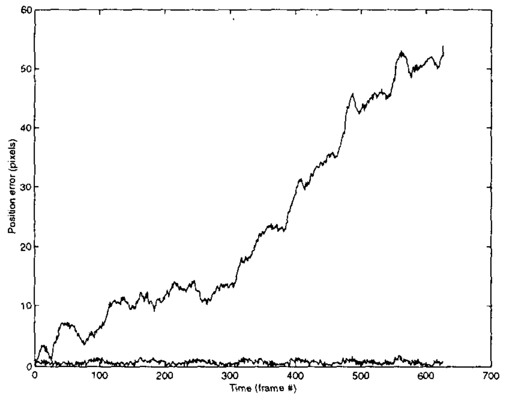

Figure 5: Comparison of position error between simple tracker and drift-reduced tracker.

each pose:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial \xi_i} p(\{\delta\}|\{\xi\}) \\
&= -\sum_{(f=i,g)\in D} \Lambda_{i,g}(\delta_i^g - \xi_i + \xi_g) \\
&+ \sum_{(f,g=i)\in D} \Lambda_{f,i}(\delta_f^i - \xi_f + \xi_i) \qquad (11)
\end{aligned}
$$

Equation (11) is a sparse linear system in terms of the poses. Given a £xed value for ξ_0, this system can be solved very ef£ciently (Matlab's backslash operator, which uses simple Gaussian elimination solves the above 626 frame problem in less than a second).

4. Stabilized 3D Tracking

Our method can also be applied to 3D tracking. We show results using a rigid motion tracker with integrated intensity and depth constraints, but our method is applicable to any parametric motion formulation, with or without depth constraints.

Depth constraints have been shown to increase the accuracy of gradient-based rigid motion tracking [3]. A depth constancy constraint analogous to the traditional brightness constancy constraint can be derived and yields:

$$
\begin{aligned}
-I_t &= \begin{bmatrix} I_x & I_y \end{bmatrix} u(x;\delta) \\
-Z_t &= \begin{bmatrix} Z_x & Z_y \end{bmatrix} u(x;\delta) - V_z(x,\delta) \qquad (12)
\end{aligned}
$$

where I_x, Z_x, etc, are the partials of y_t or y_{t-1}. Together, these equations constrain the local motion δ_{t-1}^t by using the image gradients. When the camera model is perspective, a velocity $[V_X, V_Y, V_Z]^T$ at a location in the real world results in image ¤ow

$$
\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix}\begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix}.
$$

In the case of 3D motion, we de£ne $\delta = [\delta_\omega \delta_\Delta]$ where the three components of δ_ω specify in£nitesimal rotation and the three components of δ_Δ specify translation. The warping function becomes:

$$
u(x;\delta) = \frac{1}{Z}\begin{bmatrix} f & 0 & -x_1 \\ 0 & f & -x_2 \end{bmatrix}(\delta_\omega \times X + \delta_\Delta)
$$

where X is the world coordinate of the image point x. Isolating δ and plugging u back into (12):

$$
\begin{aligned}
-Z_t &= \frac{1}{Z}\begin{bmatrix} fZ_x & fZ_y & -(Z + xZ_x + yZ_y) \end{bmatrix} \mathbf{Q}\delta \\
-I_t &= \frac{1}{Z}\begin{bmatrix} fI_x & fI_y & -(xI_x + yI_y) \end{bmatrix} \mathbf{Q}\delta \qquad (13)
\end{aligned}
$$

with

$$\mathbf{Q} = [\ \mathbf{I}\ \ -\hat{X}\ ],$$

where $\hat{X}$ is the 3x3 skew symmetric matrix formed by the real-world coordinates corresponding to x and $\mathbf{I}$ is the 3x3 identity matrix. The system of equation (13) is linear and highly overconstrained and can be easily solved for δ.

For in£nitesimal 3D updates, $d(\xi_1, \xi_0)$ should be the real eigenvector of $e^{\hat{\xi}_1} e^{-\hat{\xi}_0}$ [7], but we have found that $d(\xi_1, \xi_0) = \xi_1 - \xi_0$ is adequate in practice. Drift reduction then consisted in solving equation (11) using a sparse linear system solver.

4.1. Results: 6-DOF Head Tracker

We demonstrate the performance of the drift reduction algorithm on this 3D tracker. Figure 6 describes the direction of a head as the subject looks around the room. The nose moves by at most 20 cm throughout the sequence and the head yaws by up to 80 degrees in each direction and pitches by up to a total of 55 degrees. The sequence is 800 frames long and corresponds to about 1.2 minutes of video. The face was segmented from the background using the depth information only. Pose changes were computed using the combined constraints of (13). As shown in £gure 7, after about 600 frames, the traditional tracker has accumulated noticeable drift in its estimate of rotation, whereas the drift-reduced tracker shows the pointer on the subject's nose whenever he returns to a near-frontal pose. Only appearance was used in £nding suitable anchor frames. Figure 8 plots the index of anchor frames used for each frame. The protrusions from the diagonal line are produced as the subject returns from a rotation. Note that the £rst frame is never reused. The robustness is entirely due to recovering from drift accumulated during each rotation by using frames observed while going into the rotation.

4.2. Results: Egomotion

The sequence summarized in £gure 9 demonstrates that the drift reduced tracker can also be used for computing egomotion. The task is to hold the pointer in the same location relative to the real world as the camera scans the room. Between frames 400 and 600, almost none of the original scene is visible. By frame 610, the drift-reduced tracker shows signi£cant improvement over the traditional tracker, despite the dearth of back frames before frame 630. The superior performance in the early frames demonstrates the bene£ts of the batch/non-causal nature of the drift-reduction algorithm and of allowing information in the future in¤uence the past. By frame 1050 the unenhanced tracker has drifted far enough that all subsequent pose changes throw it even further off track. Figure 10 shows a quantitive version of the results. After 600 frames, the traditional tracker starts to accumulate considerable drift. During the same period, the drift-reduced tracker keeps track of the real movement by using information prom similar previous frames as shown in £gure 12.

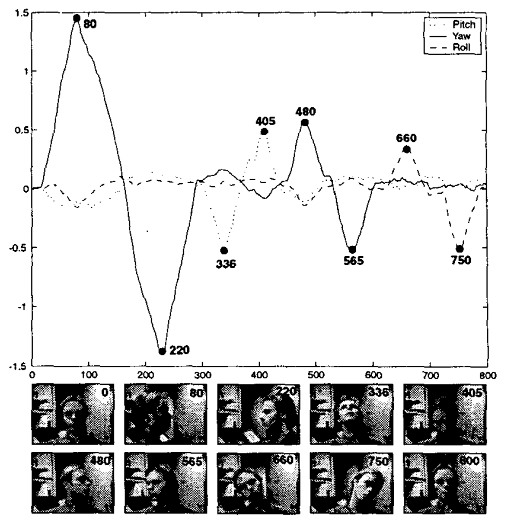

Figure 6: The sequence is 1.2 minutes long. The subject looks in all directions, by up to 80 degrees from frontal in some directions. The sequence was captured at $\sim$11 fps. The graph above provides an intuitive feel for the relative magnitude of the rotations. It plots δ_ω over time.

5. Conclusion

We have developed a framework for stabilizing parametric motion trackers in closed environments. Our method measures pose change between frames which are similar in pose and appearance, and uses these measurements to compute robust pose estimates. This improves stability since additional pose change measurements provide robustness and ground the tracking against commonly revisited sites. We derived an uncertainty model for motion estimation and used it to frame the problem of incorporating these additional measurements into a non-causal estimation framework. We demonstrated the bene£ts of using multiple base frames in our maximum likelihood framework on a synthetic 2D motion tracking problem and on 3D ego-motion computation and pose estimation.

Acknowledgement

Many thanks to Tom Minka for suggesting the Hessian approximation used in section 2.4.

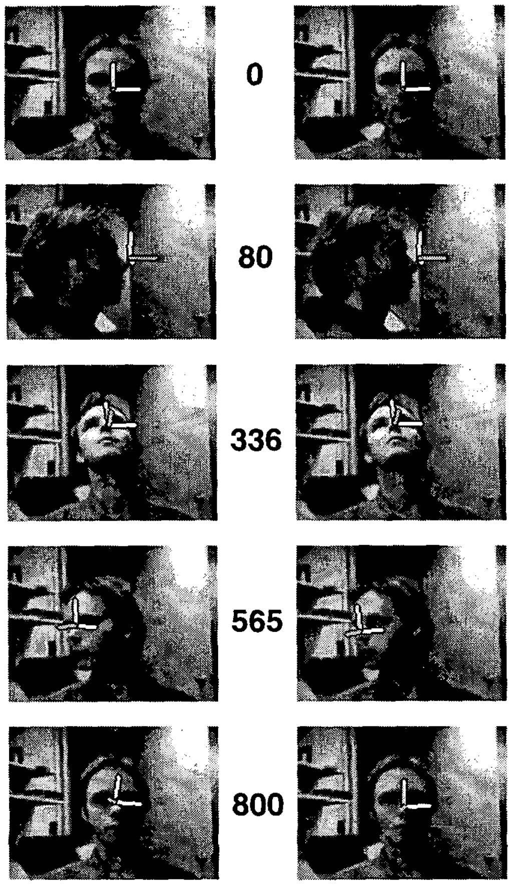

Figure 7: Left column: traditional tracker. Poses are updated according to (4). Right column: drift-reduced tracker.

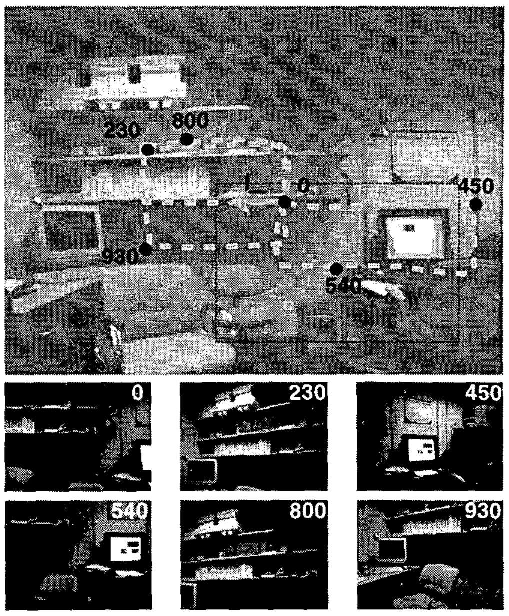

Figure 9: The camera begins panning from the center dot, in the direction of the arrow. The dashed path marks the approximate trajectory of the center of the camera (drawn by hand). Only the interior of the black rectangle is visible to the camera (approximate), so that the intial pose is completely out of view between frames 420 and 530.

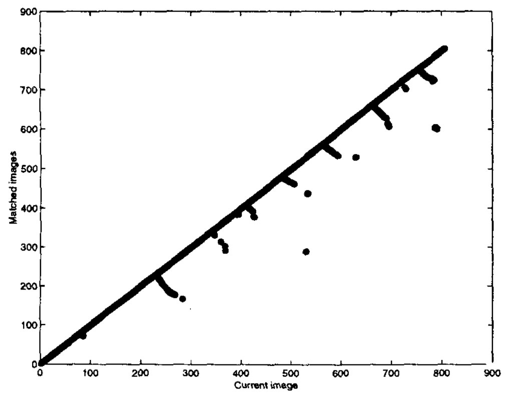

Figure 8: Anchor frames used by drift-reduced tracker. Each frame on the horizontal axis is matched by appearance with 3 previous frame. The protrusions show that as the subject returns from a rotation, frames on the way into the rotation are used as anchor.

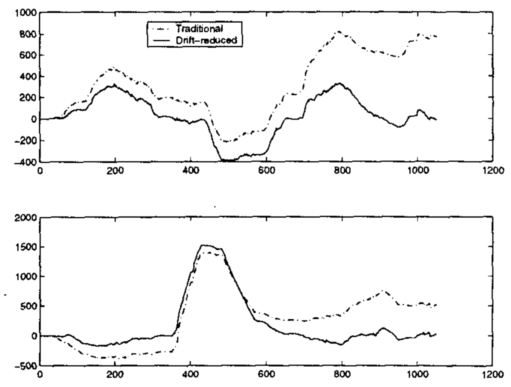

Figure 10: Top: Horizontal translation. Bottom: Vertical Translation. The traditional tracker exhibits continual drift with respect to the drift-reduced tracker.

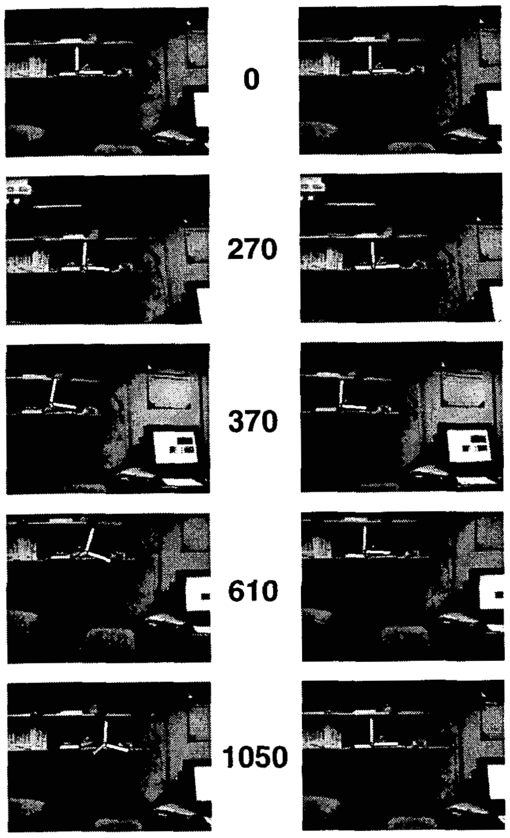

Figure 11: Left column: traditional tracker. Right column: drift-reduced tracker. Beyond 1050 frames, the traditional tracker is no longer effective.

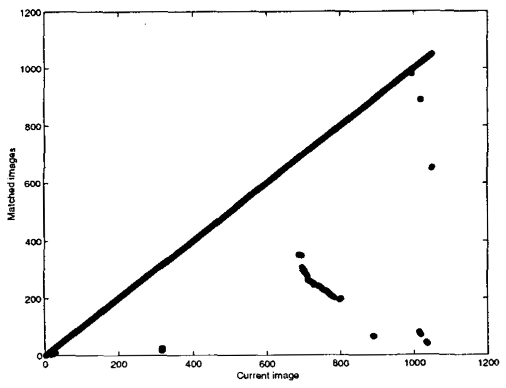

Figure 12: Anchor frames used by drift-reduced tracker. Previously visited poses are used effectively (eg, frames 300, 700, 1020).

References

[1] S. Birch£eld. Elliptical head tracking using intensity gradients and color histograms. In *CVPR*, pages 232–237, 1998.

[2] G. Hager and P. Belhumeur. Ef£cient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039, 1998.

[3] M. Harville, A. Rahimi, T. Darrell, G. Gordon, and J. Wood£ll. 3d pose tracking with linear depth and brightness constraints. In *Proceedings of CVPR 99*, pages 206–213, Corfu, Greece, 1999.

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

[5] 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.

[6] T.P. Minka. Using lower bounds to approximate integrals. Technical report, Media Lab, http://www.media.mit.edu/˜tpminka/papers/rem.html, 2001.

[7] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[8] O. Nestares, D.J. Fleet, and D.J. Heeger. Likelihood functions and con£dence bounds for total-least-squares problems. In *CVPR*, 2000.

[9] N. Oliver, A. Pentland, and F. Berard. Lafter: Lips and face real time tracker. In *Computer Vision and Patt. Recog.*, 1997.

[10] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR94*, pages 593–600, 1994.

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

[12] E.P. Simoncelli, E.H. Adelson, and D.J. Heeger. Probability distributions of optical ¤ow. In *CVPR91*, pages 310–315, 1991.

[13] J. Weber and J. Malik. Robust computation of optical ¤ow in a multi-scale differential framework. *IJCV*, 14(1):67–81, 1995.

[14] C. Wren and A. Pentland. Dynamic models of human motion. In *Proceedings of Face and Gestures*, 1998.