Learning to track the visual motion of contours

Andrew Blake, Michael Isard and David Reynard
Department of Engineering Science,
University of Oxford,
Oxford OX1 3PJ, UK.

ABSTRACT

A development of a method for tracking visual contours is described. Given an "un-trained" tracker, a training-motion of an object can be observed over some extended time and stored as an image sequence. The image sequence is used to learn parameters in a stochastic differential equation model. These are used, in turn, to build a tracker whose predictor imitates the motion in the training set. Tests show that the resulting trackers can be markedly tuned to desired curve shapes and classes of motions.

Contents

1 Introduction **2**

2 Tracking framework **2**
2.1 Curve representation . . . 2
2.2 Tracking as estimation over time . . . 3
2.3 Rigid body transformations . . . 3
2.4 Curves in motion . . . 7
2.5 Discrete-time model . . . 7
2.6 Measurement model . . . 8

3 Tracking algorithms **8**
3.1 Full time-varying filter . . . 8
3.2 Steady-state filter . . . 9
3.3 Bootstrapping . . . 10

4 System identification **10**
4.1 Estimation for one-dimensional variation . . . 11
4.2 Exercising the learning algorithm . . . 11
4.3 Estimation of multivariate system parameters . . . 14
4.4 Learning zoom . . . 15
4.5 Nonrigid motion . . . 15

5 Trained filters **15**
5.1 Rigid hand-motion: configuration-space training . . . 18
5.2 Hand tracking: phase-space training . . . 21
5.3 Nonrigid motion: lips . . . 21
5.4 Connected speech . . . 25
5.5 Nonrigid hand motions . . . 25
5.6 Full Kalman filter . . . 25

6 Conclusions **25**

1 Introduction

A significant recent development in real-time visual sensing has been the invention of contour trackers [25] that can track the motion of silhouettes and surface features. This allows the camera to be treated as a sparse sensor with consequent processing efficiency that lends itself to real-time operation. There are many potential applications, e.g. for instance in biomedical image-analysis (e.g. [3]) in surveillance (e.g. [23, 33, 31]) and in autonomous vehicle navigation (e.g. [17]) and in hand-eye coordination for robots [10]. A review of existing methods in curve-tracking can be found in [13].

Real-time video tracking has been achieved for line-drawings [22] and polyhedral structures [27] and for simple, natural features such as road-edges [17]. Commercial systems (e.g. Watsmart, Oxford Metrics) are available which track artificial markers on live video. Natural point features (e.g. on a face) but not curves, have been tracked at 10Hz using a workstation assisted by image-processing hardware [4]. Curve trackers [25, 16, 12] have been demonstrated on modest workstations with some success.

Several researchers have described Kalman filter formalisms for tracking of curves and surfaces [34, 35]. This paper is based on a particular linear filter for curves which incorporates a mean shape — a "template", as used by a number of researchers [19, 21, 9, 38]. The tracker used here [11, 12] also has an affine invariance mechanism to accommodate 3D rigid transformations of planar shapes. We refer to this as the "un-trained" tracker, not yet tuned for motion.

Tuning for specific motions, it transpires, is a key ingredient in achieving robust tracking. It can be done consistently in the Kalman filter framework by defining dynamics within an appropriate state-space. Deterministic models of visual motion based on oscillatory modes have been used previously [29]. The generality of motion models is greatly increased by including a stochastic component in the dynamics, as is done commonly in control theory via *stochastic differential equations* [1]. Even in the case of planar rigid motion, the dynamics of a visual contour are potentially complex with six independent modes and many degrees of freedom for the driving noise process.

Therefore, rather than programming *ad hoc* dynamics, we have developed a procedure to *learn* them from extended motion sequences. The procedure will be explained and its effectiveness in tuning a tracker to characteristic classes of motions will be demonstrated. The disadvantage of the method is that each tuned tracker is effective only for the relatively narrow class of shapes and motions on which it was trained. The advantage is that performance is enhanced compared with an un-trained tracker, particularly in the ability to ignore background distractors and to follow rapid motions.

2 Tracking framework

The tracker consists of an estimator for a piecewise smooth image-plane curve in motion

$$\mathbf{r}(s,t) = (x(s,t), y(s,t)).$$

Following the tracking formulations of others [28, 14], the curve representation is in terms of B-splines. Splines of order d are used, with the possibility of multiple knots for vertices. In our experiments, splines are usually quadratic ($d = 3$). The tracking framework described here follows an earlier one [12] but is generalised to allow any learned motion model to be used in the prediction phase of tracking.

2.1 Curve representation

Curves are represented as parametric B-splines with N spans and N_c control points.

$$x(s,t) = \mathbf{B}(s)\mathbf{X}(t) \quad \text{and} \quad y(s,t) = \mathbf{B}(s)\mathbf{Y}(t), \quad 0 \le s \le N$$

where $\mathbf{X} = (X_1, .., X_{N_c})^T$ and similarly for $\mathbf{Y}$. The elements of $\mathbf{X}$ and $\mathbf{Y}$ are simply the x, y coordinates of the set of control points (X_m, Y_m) for the B-spline. The number of control points is equal to the number of spans ($N_c = N$) for closed curves whereas $N_c = N + d$ for open curves (with

appropriate variations, in each case, where multiple knots are used to vary curve continuity). The vector $B(s)$ consists of blending coefficients defined by

$$\mathbf{B}(s) = (B_1(s), ..., B_{N_c}(s))$$

where $B_m(s)$ is a B-spline basis function [18, 7] appropriate to the order of the curve and its set of knots.

2.2 Tracking as estimation over time

The tracking problem is to estimate the motion of some curve — in examples in this paper it will be the outline of a hand or of lips. The underlying curve — the physical truth — is assumed to be describable as a B-spline of a certain predefined form with control points $\mathbf{X}(t), \mathbf{Y}(t)$ varying over time. The tracker generates *estimates* of those control points, denoted $\hat{\mathbf{X}}(t), \hat{\mathbf{Y}}(t)$ and the aim is that the estimates should represent a curve that, at each time-step, matches the underlying curve as closely as possible. The tracker is based, in accordance with standard practice in temporal filtering [20, 5], on two models: a system model and a measurement model. These will be spelt out in detail later. Broadly, the measurement model specifies the positions along the curve at which measurements are made and how reliable they are. The system model specifies the likely dynamics of the curve over time, relative to a "template" [19] whose control points, $(\overline{\mathbf{X}}, \overline{\mathbf{Y}})$, are generated by interactive software, used to draw spline-curves over a single image captured from live video.

2.3 Rigid body transformations

A tracker could conceivably be designed to allow arbitrary variations in control point positions over time. This would allow maximum flexibility in deforming to accommodate moving shapes. However, particularly for complex shapes requiring many control points to describe them, this is known to lead to instability in tracking [12]. It occurs when features are temporarily obscured and the tracker is bumped out of its steady state. The more complex the shape to be tracked, the worse is its instability when lock is lost, and this is illustrated in figure 1.

Fortunately, it is not necessary to allow so much freedom. A moving hand, for instance, provided the fingers are not flexing, is a rigid, approximately planar shape. Provided perspective effects are not too strong, a good approximation to the curve shape as it changes over time can therefore be obtained by specifying $\mathbf{Q}$, a linear vector-valued function of the B-spline coordinates $(\mathbf{X}, \mathbf{Y})$. The $\mathbf{Q}$-parameterisation of the curve embodies the reduced degrees of freedom for motion, which vary online, leaving intact the full set $(\mathbf{X}, \mathbf{Y})$ of geometrical parameters to do justice to the detail of complex shapes and to be varied offline only.

The relationships $\mathbf{Q} \leftrightarrow (\mathbf{X}, \mathbf{Y})$ between parameterisations are expressed in terms of two matrices M, W:

$$\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = W\mathbf{Q} + \begin{pmatrix} \overline{\mathbf{X}} \\ \overline{\mathbf{Y}} \end{pmatrix} \quad \text{and} \quad \mathbf{Q} = M\left[\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} - \begin{pmatrix} \overline{\mathbf{X}} \\ \overline{\mathbf{Y}} \end{pmatrix}\right].$$

The matrices M, W are defined in terms of the shape template $(\overline{\mathbf{X}}, \overline{\mathbf{Y}})$. It is known that for a planar shape, for instance, just six *affine* degrees of freedom are required [37, 26] to describe the possible shapes of the curve and this is illustrated in figure 2. The space of possible $\mathbf{Q}$-vectors is expressible as a 6-dimensional linear subspace of $\mathbf{Q}$-vectors, and a basis for this subspace is:

$$\mathcal{B} = \left\{ \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix}, \begin{pmatrix} \overline{\mathbf{X}} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \overline{\mathbf{Y}} \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \overline{\mathbf{X}} \end{pmatrix}, \begin{pmatrix} \overline{\mathbf{Y}} \\ \mathbf{0} \end{pmatrix} \right\}$$

where the N_c-vectors $\mathbf{0}$ and $\mathbf{1}$ are defined by:

$$\mathbf{0} = (0, 0, .., 0)^T \quad \mathbf{1} = (1, 1, .., 1)^T.$$

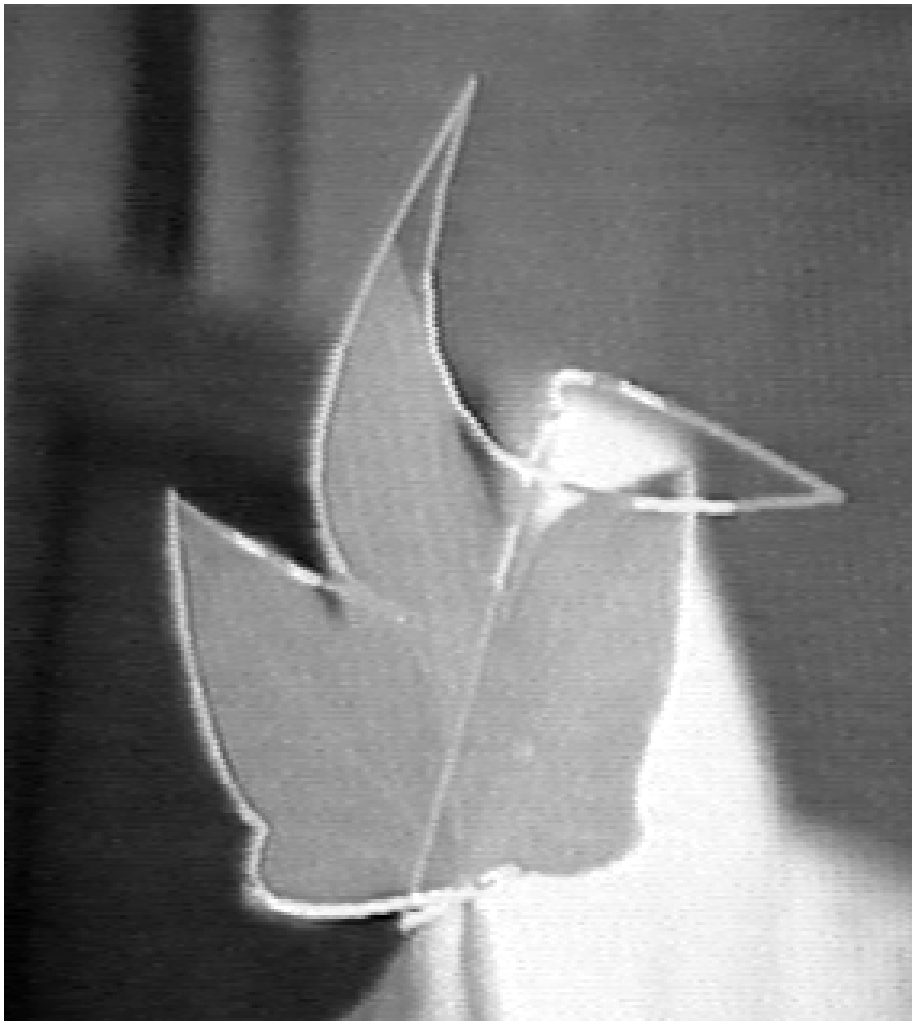


Figure 1: *Tracking can become unstable when the tracked outline is geometrically complex. This problem can be overcome however by introducing certain constraints into the configuration space of the tracker — see text.*

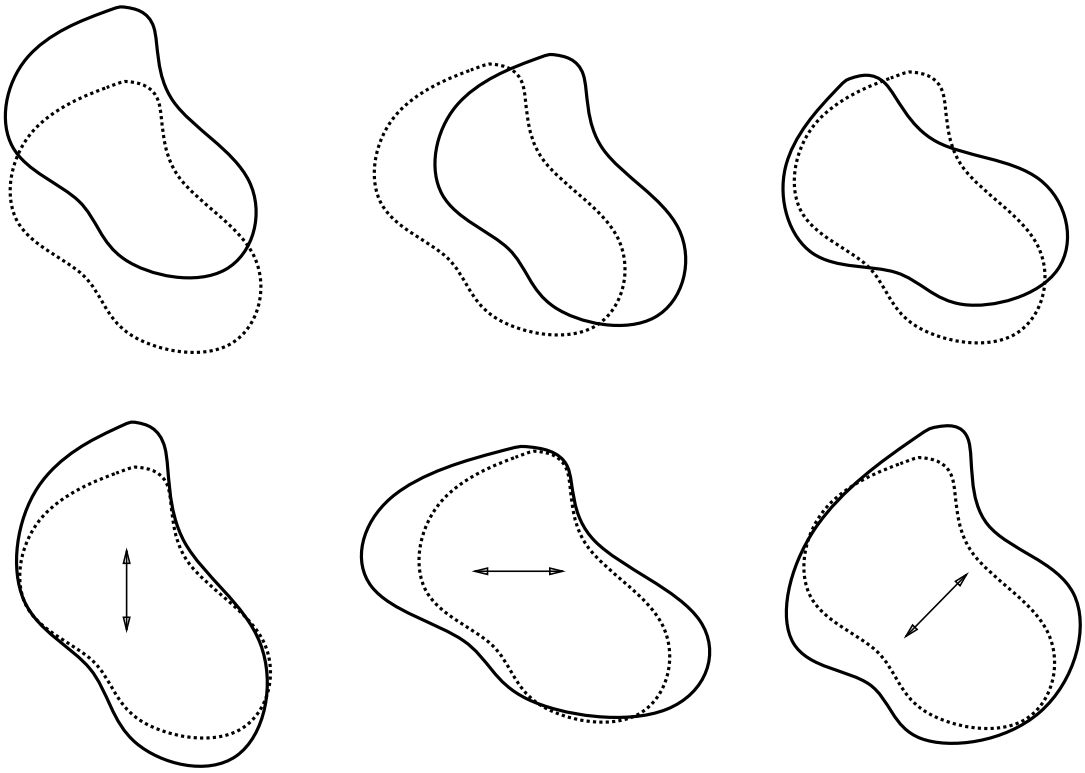


Figure 2: *The six degrees of freedom of a 2D affine transformation are illustrated here: translation vertically and horizontally, rotation and scaling vertically, horizontally and diagonally.*

In that case, the matrices M, W converting B-spline control points $(\mathbf{X}, \mathbf{Y})$ to and from the 6-vector $\mathbf{Q}$ can be defined, in terms of the template, as follows:

$$W = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \overline{\mathbf{X}} & \mathbf{0} & \mathbf{0} & \overline{\mathbf{Y}} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \overline{\mathbf{Y}} & \overline{\mathbf{X}} & \mathbf{0} \end{pmatrix} \tag{1}$$

and

$$M = (W^T \mathcal{H} W)^{-1} W^T \mathcal{H} \tag{2}$$

where

$$\mathcal{H} = \int_0^N \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes (\mathbf{B}(s)\mathbf{B}(s)^T)\, \mathrm{d}s,$$

in which $\otimes$ denotes the "Kronecker product"[1] of two matrices. Note that $\mathcal{H}$ is the "metric" matrix that arises from the "normal equations" [30] for the problem of least-squares approximation with B-splines. Examples of such matrices are given in [12].

Shape models other than planar-affine can be treated by using an appropriate $\mathbf{Q}$ vector of length N_Q with a W matrix of size $2N_c \times N_Q$. In the planar-affine case above we had $N_Q = 6$. Restricted affine motion would call for $N_Q < 6$, for instance rigid 2D translation for which $N_Q = 2$. Non-planar rigid shapes can be treated in a similar way to planar ones except that then $N_Q = 8$. For smooth silhouette curves, it can be shown, $N_Q = 11$ is appropriate[2]. Finally, nonrigid motion can be treated in the same framework. In each case, the M matrix continues to be derived from the W matrix via the formula (2) above. The table in figure 3 summarises the hierarchy of models.

Transformations	**generated by:**	Dimension	no. views
Planar translation	X, Y translation	2	1
Planar similarity	$+X, Y$ rotation, scaling	4	1
Planar affine	3D Euclidean, planar curve	6	1
3D affine	3D Euclidean, space curve	8	2(3)
3D affine+	3D Euclidean, silhouette	11	3(6)
Constrained nonrigid	+ linear deformations	$+n$	$+n$ (key-frames)
Unconstrained nonrigid	B-spline configurations	$2N_c$	

Figure 3: *Configuration sub-spaces — hierarchy for increasing complexity of object and motion.*

Given the compact representation of a curve in terms of configuration-space vector $\mathbf{Q}$, the parametric B-spline can be fully reconstructed from it. The reconstruction formula is:

$$\mathbf{r}(s,t) = U(s)\mathbf{Q}(t) \tag{3}$$

where

$$U(s) = \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \mathbf{B}(s) \right] W. \tag{4}$$

[1]The Kronecker product $A \otimes B$ of two matrices A, B is obtained by replacing each element a of A with the submatrix aB. The dimensions of this matrix are thus products of the corresponding dimensions of A, B.

[2]In the case of silhouettes, however, the $\mathbf{Q}$-vector representation of the curve is an approximation, valid for sufficiently small changes of viewpoint.

2.4 Curves in motion

Now a state vector can be chosen according to the order of the differential equation that will describe the motion. Here second order motion is assumed so the state $\mathcal{X}$ is defined in the following form:

$$\mathcal{X} = \begin{pmatrix} \mathbf{Q} \\ \mathbf{Q}' \end{pmatrix}.$$

The dynamics of the object are then defined by the stochastic differential equation

$$\dot{\mathcal{X}} = F(\mathcal{X} - \overline{\mathcal{X}}) + \begin{pmatrix} \mathbf{0} \\ \\ G\dot{\mathbf{w}} \end{pmatrix}. \tag{5}$$

where F is a $2N_Q \times 2N_Q$ matrix defining the deterministic part of the dynamics. Its eigenvectors represent modes of oscillatory motion, and the corresponding eigenvalues give natural frequencies and damping constants for those modes [2]. The random part of the dynamics are modelled by a driving noise source $\mathbf{w}(t)$ which is a unit N_Q-dimensional Wiener process — Brownian motion in continuous time [1] defined by its statistical properties:

$$E[\mathbf{w}(t)] = \mathbf{0} \quad \text{and} \quad E[\mathbf{w}(t)\mathbf{w}(t)^T] = tI_{N_Q}.$$

(I_r denotes the $r \times r$ identity matrix.) The form of the noise process corresponds to the modelling assumption that random processes enter the system only as accelerations, with no direct coupling to velocities. The covariance coefficient of random acceleration is then GG^T and is the observable part of G that, in principle, can be estimated by a learning algorithm. The mean state $\overline{\mathcal{X}}$ of the contour incorporates a mean positional displacement (relative to the template), and a mean velocity which may be taken to be zero or could, in principle, be learned from data.

2.5 Discrete-time model

It is natural to model the motion of an object in continuous time, as we have so far. Discrete time enters only when the measurement process is considered. In the case of video imagery, measurements are made synchronously at a sample interval Δ. In between sampling epochs, the continuous equations of motion (5) can be integrated [20], to give $\mathcal{X}_{n+1} \equiv \mathcal{X}((n+1)\Delta)$ in terms of $\mathcal{X}_n \equiv \mathcal{X}(n\Delta)$. Note that no *approximation* is introduced here in integrating the continuous dynamics; the discrete representation is simply a *restriction* of the continuous one to the sampling epochs.

The discrete model has the form

$$\mathcal{X}_{n+1} - \overline{\mathcal{X}} = A(\mathcal{X}_n - \overline{\mathcal{X}}) + \begin{pmatrix} \mathbf{0} \\ B\mathbf{w}_n \end{pmatrix}. \tag{6}$$

The matrix coefficient A is a $2N_Q \times 2N_Q$ matrix, defining the deterministic part of the dynamics. Its eigenvectors represent modes, as before, in the continuous case. In fact the discrete dynamics matrix A is simply related to the continuous one by

$$A = \exp F\Delta \tag{7}$$

so that for an eigenvalue λ of A, there is a corresponding (complex) eigenvalue of F, from which frequency ω and damping constant β can be computed:

$$-\beta + i\omega = \frac{1}{\Delta}\log\lambda. \tag{8}$$

Without loss of generality, A can be standardised to the form

$$A = \begin{pmatrix} 0 & I \\ \\ A_0 & A_1 \end{pmatrix} \tag{9}$$

so that $\overline{\mathcal{X}}$ must be of the form

$$\overline{\mathcal{X}} = \begin{pmatrix} \overline{\mathbf{Q}} \\ \overline{\mathbf{Q}} \end{pmatrix}$$

depending on mean position in the form of the displacement $\overline{\mathbf{Q}}$ but not on mean velocities (which now, in the discrete case, are carried by the A-matrix).

In that case, the equations of motion are simplified to the standard form:

$$\mathbf{Q}_{n+2} = A_0\mathbf{Q}_n + A_1\mathbf{Q}_{n+1} + (I - A_0 - A_1)\overline{\mathbf{Q}} + B\mathbf{w}_n. \tag{10}$$

As for the stochastic component of the discrete dynamics, at each time n, the $\mathbf{w}_n$ are independent vectors of independent unit normal random variables. This noise process is then transformed by the matrix B which couples the driving noise into the various natural modes of the deterministic dynamics. In fact B is not completely observable, only the covariance $C = BB^T$ can be computed; this is because the probability distribution for $\mathbf{w}$, being a vector of i.i.d. normal variables, is (it is easily shown) invariant to orthogonal transformations of $\mathbf{w}$.

2.6 Measurement model

In an earlier framework [12], measurement was represented as a continuous time process, not because this is a realistic model — after all, video is a synchronous sampled data stream — but because it is tractable in the sense of facilitating analysis of the tracker's performance as a control system. Here however, the learned motion models are too varied to allow the kind of performance analysis that was possible for constant velocity models. Therefore, in this paper, a synchronous, discrete-time measurement model is used throughout.

Given an estimator (see next section) $\hat{\mathbf{r}}(s,t)$ for the contour $\mathbf{r}(s,t)$, the visual measurement process at time t consists of casting rays along normals $\hat{\mathbf{n}}(s,t)$ to the estimated curve and, simultaneously at certain points s along the curve, measuring the relative position $\nu(s,t)$ of a feature (typically a high contrast edge) along the ray, so that

$$\nu(s,t) = [\mathbf{r}(s,t) - \hat{\mathbf{r}}(s,t)].\hat{\mathbf{n}}(s,t) + v(s,t) \tag{11}$$

where $v(s,t)$ is a scalar noise variable, assumed gaussian, with a variance σ^2 that will be taken to be constant, both spatially and temporally. Defining measurement to be along the normal *only* is essential as displacement tangential to the curve is unobservable, the well-known "aperture problem" of visual motion [24].

Each measurement ν is actually an "innovation" measure [5] because it is taken relative to the estimated position $\hat{\mathbf{r}}(s,t)$. This is convenient for tracking because innovation is a form of error signal, ready to be added in, via an appropriate gain, to the estimated state. In order to compute this gain, we need first to express ν in terms of the state vector $\mathcal{X}$, by means of a measurement matrix $H(s)$:

$$\nu(s,t) = H(s,t)(\hat{\mathcal{X}} - \mathcal{X}) + v(s,t) \tag{12}$$

where, from (3), (4) and (11),

$$H(s,t) = \begin{pmatrix} \hat{\mathbf{n}}(s,t)^T \otimes B(s) & \mathbf{0} \end{pmatrix} W \tag{13}$$

3 Tracking algorithms

3.1 Full time-varying filter

Following standard practice in filtering, the estimator $\hat{\mathcal{X}}(t)$ in state-space evolves in two phases [20]: *prediction* and *measurement*. For each phase there is a 1st order (mean) and 2nd order (covariance) update equation.

The prediction phase is applied once at each time step $t - \Delta \rightarrow t$. The 1st and 2nd order update equations are respectively the mean and variance from a simulation of the system dynamics (6):

$$\hat{\mathcal{X}}_n = A\hat{\mathcal{X}}_{n-1} + (I - A)\overline{\mathcal{X}} \tag{14}$$

and

$$P_n = AP_{n-1}A^T + \begin{pmatrix} 0 & 0 \\ & \\ 0 & C \end{pmatrix}. \tag{15}$$

where P_n is the covariance of the estimate $\hat{\mathcal{X}}$ at time $t = n\Delta$. In general, the evolution of P over time has been found to be valuable for contour tracking both in exploiting continuity of motion via the time-varying Kalman gain (see below), and in recovering from sensing failures using a *validation gate* [12].

Following the prediction step for a given time-step $t - \Delta \rightarrow t$, a number of measurements can be made, taken by casting normals on the image field or frame for that epoch. For each measurement, the curve estimate is updated as follows:

$$\hat{\mathcal{X}}_n \rightarrow \hat{\mathcal{X}}_n + K\nu$$

where $\nu(s)$ is innovation along normal at s as before. The Kalman gain for that measurement [20] is

$$K = P_n H^T \left(H P_n H^T + \sigma^2 \right)^{-1}.$$

After the measurement has been applied via the Kalman gain to the estimated state $\hat{\mathcal{X}}_n$, its covariance must be updated:

$$P_n \rightarrow (I - KH)P_n.$$

3.2 Steady-state filter

For many of the experiments reported in this paper we maintain steady state Kalman gains for each measurement in a fixed sequence of measurements. This is efficient because the computational cost of recomputing P is avoided, and in practice allows more complex spline curves within the constraint of full field-rate (50Hz) tracking. Of course the benefits of variable gain are lost and practically this limits the recovery performance of the tracker after lock is lost, as the last experiment of the paper shows. The steady state tracker is the restriction of the full tracker (assuming a stable system) to the case that

1. the sequence of curve positions s at which measurements are made is fixed from iteration to iteration

2. the contour is fully locked

3. all transients in the filter have settled out

4. curve normals do not rotate — in practice we require deviations from 2D translational motion to be small

Steady state gains for the measurement sequence are computed by allowing the filter to run fully locked for a few seconds — it is typically most convenient to do this simply by leaving the tracked object in its starting position, against a plain background — then freezing the gains, one gain for each curve normal in the fixed measurement sequence.

3.3 Bootstrapping

In order to collect a training set, some form of un-trained tracker is required. Since the dynamics of the moving object are unknown at this stage it is necessary to use a tracker with reasonable default dynamics. Such a tracker is less powerful than a trained tracker, particularly in clutter and with rapid motions, as this paper will show. To make up for that loss of performance, the training set can be gathered from gentle motion against an uncluttered background. To learn rapid motions, it is necessary to bootstrap, that is: first to use a default tracker to learn a relatively slow motion, then incorporate that learnt motion into a tracker which should then be more agile than the default, and use it to learn a somewhat faster motion, and so on, repeating the cycle if necessary.

The default trackers are based on constant velocity translational dynamics, with a damped elastic coupling of shape parameters to those of the template. The stochastic component of the dynamics is a homogeneous, isotropically distributed random acceleration, as described previously in some detail [12]. The deterministic component of the object-dynamics for a usable default tracker is given by:

$$F = \begin{pmatrix} 0 & I \\ -(\omega^2 E + \omega'^2 E') & -2(\beta E + \beta' E') \end{pmatrix} \tag{16}$$

where E' is a projection matrix onto a certain *privileged subspace* of curve motions and E projects on the remainder of the state-space left when the image of E' and the subspace of rigid translations are removed. Therefore $I - E - E'$ is a matrix that projects the state $\mathbf{Q}$ onto the subspace of rigid translations, and $EE' = 0$. For example, E' could be a projection onto the space of rotations/expansions of the template. This would allow unconstrained, constant-velocity translation and also independent control over the strength of the template both for rotation/expansion (via β, ω) and for other deformations of shape (via β', ω'). Further elastic damping parameters can be added to control translational motion, if desired, but we have found, in practice, that it is usually satisfactory to leave translational motion unconstrained.

In the discrete implementation, there is a limit to how large an effective value of the β, β' parameters can be achieved [2]. However it is, of course, possible to achieve "hard" settings in which β, ω or β', ω' are effectively infinite, simply by using a smaller $\mathbf{Q}$-space that excludes the degrees of freedom that are to be set "hard". This reduced space is specified by removing columns of the W-matrix in (1) corresponding to those degrees of freedom.

In later learning experiments reported in this paper, both default and trained trackers are full Kalman filters with time-varying gains and spatially distributed validation gate. Earlier experiments used steady-state trackers, obtained by allowing the full filter to settle down under conditions of full feature-lock. The steady-state gains for the default tracker could, in principle, have been obtained in the same way; instead we use something a little simpler, a form of $\alpha - \beta$ tracker [20].

4 System identification

The learning task is now to estimate the coefficients A_0, A_1, B from a training sequence $\mathbf{Q}_1, ..\mathbf{Q}_m$, gathered at the image sampling frequency of $1/\Delta = 50$Hz. As mentioned earlier, it is impossible in principle to estimate B uniquely, but the covariance coefficient $C = BB^T$ can be determined, from which a standard form for B can be computed as:

$$B = \sqrt{C}, \tag{17}$$

applying the square root operation for a positive definite square matrix [6].

The learning process is outlined below. First, to avoid overfitting, the state space must be restricted, during learning, to a low-dimensional subspace. This could be the six-dimensional "affine" subspace of the state space or some other space spanning an appropriate combination of rigid and non-rigid degrees of freedom. Secondly, Maximum Likelihood Estimation (MLE) is implemented via least-squares minimisation to obtain discrete-time system parameters. The result is a tracker tuned to those affine motions that occurred in the training-set.

4.1 Estimation for one-dimensional variation

First the estimation algorithm is described for the simple case of a one-dimensional particle — no curve or splines are involved here for the sake of tutorial simplicity, just one number describing the position of the particle along a rail. The algorithm estimates a continuous time, second order model for particle position $x(t)$ so the state-space is defined in terms of $\mathbf{Q}(t) \equiv x(t)$. From (6), for the sampled system with variables $x_n \equiv x(n\Delta)$, the coefficients A_0, A_1, B become scalars denoted a_0, a_1, b. For simplicity we will take the mean (template) $\overline{x} = 0$ so that the quantity $x_{n+2} - a_0 x_n - a_1 x_{n+1}$ is a zero-mean scalar normal variable bw_n, for each n, with unknown variance b^2. The log-likelihood function is defined up to an additive constant by

$$L(x_1, ..x_n | a_0, a_1, b) \equiv \log p(x_1, ..x_n | a_0, a_1, b) \quad + \quad \text{const}$$

but

$$p(x_1, ..x_n | a_0, a_1, b) = \prod_n p(w_n = (x_{n+2} - a_0 x_n - a_1 x_{n+1}) b^{-1})$$

so, using the normality of the distribution $p(..)$,

$$L(x_1, ..x_n | a_0, a_1, b) = -\frac{1}{2b^2} \sum_{n=1}^{m-2} (x_{n+2} - a_0 x_n - a_1 x_{n+1})^2 \quad - \quad (m-2) \log b,$$

up to an additive constant. The MLE for the coefficients a_0, a_1, b is obtained from the maximisation of the quadratic likelihood function. It is clear in this univariate case that the minimisation over b factors out, so that estimates $\hat{a}_1, \hat{a}_2$ are determined by

$$(\hat{a}_0, \hat{a}_1) = \arg \min_{a_0, a_1} \sum_{n=1}^{m-2} (x_{n+2} - a_0 x_n - a_1 x_{n+1})^2 ,$$

obtained as the solution of the simultaneous equations

$$s_{20} - \hat{a}_0 s_{00} - \hat{a}_1 s_{10} = 0 \tag{18}$$
$$s_{21} - \hat{a}_0 s_{01} - \hat{a}_1 s_{11} = 0, \tag{19}$$

where

$$s_{ij} = \sum_{n=1}^{m-2} x_{n+i} x_{n+j}, \quad i, j = 0, 1, 2$$

are discrete auto-correlation measures for the time sequence time-sequence. Now a_0, a_1 are regarded as constants, fixed at their estimated values, in the likelihood function, which can be maximised with respect to b to obtain $\hat{b}$:

$$\hat{b}^2 = \frac{1}{m-2} \sum_{n=1}^{m-2} (x_{n+2} - \hat{a}_0 x_n - \hat{a}_1 x_{n+1})^2 . \tag{20}$$

4.2 Exercising the learning algorithm

Now the learning of univariate motions is illustrated by the example of a tracked hand in motion. The oscillation is largely horizontal (figure 4) but in any case, for the purposes of this illustration, the horizontal component of motion of the centroid of the tracked curve will be extracted. Once the learning is done, the learned model $(\hat{a_0}, \hat{a_1}, \hat{b})$ can be simulated by generating pseudo-random gaussian noise — a synthetic random process — with variance $\hat{b}^2$ that drives discrete dynamics with coefficients $\hat{a}_0, \hat{a}_1$. The sample of random dynamics shown appears plausibly to be drawn from the same family of random signals as the training set. In this case a model of motion as 2nd order dynamics driven by noise seems to be adequate.

However, not all data-sets can be represented faithfully by such a model. For example the oscillatory motion in figure 5 is not a perfect sinusoid — it was generated by hand. A perfect

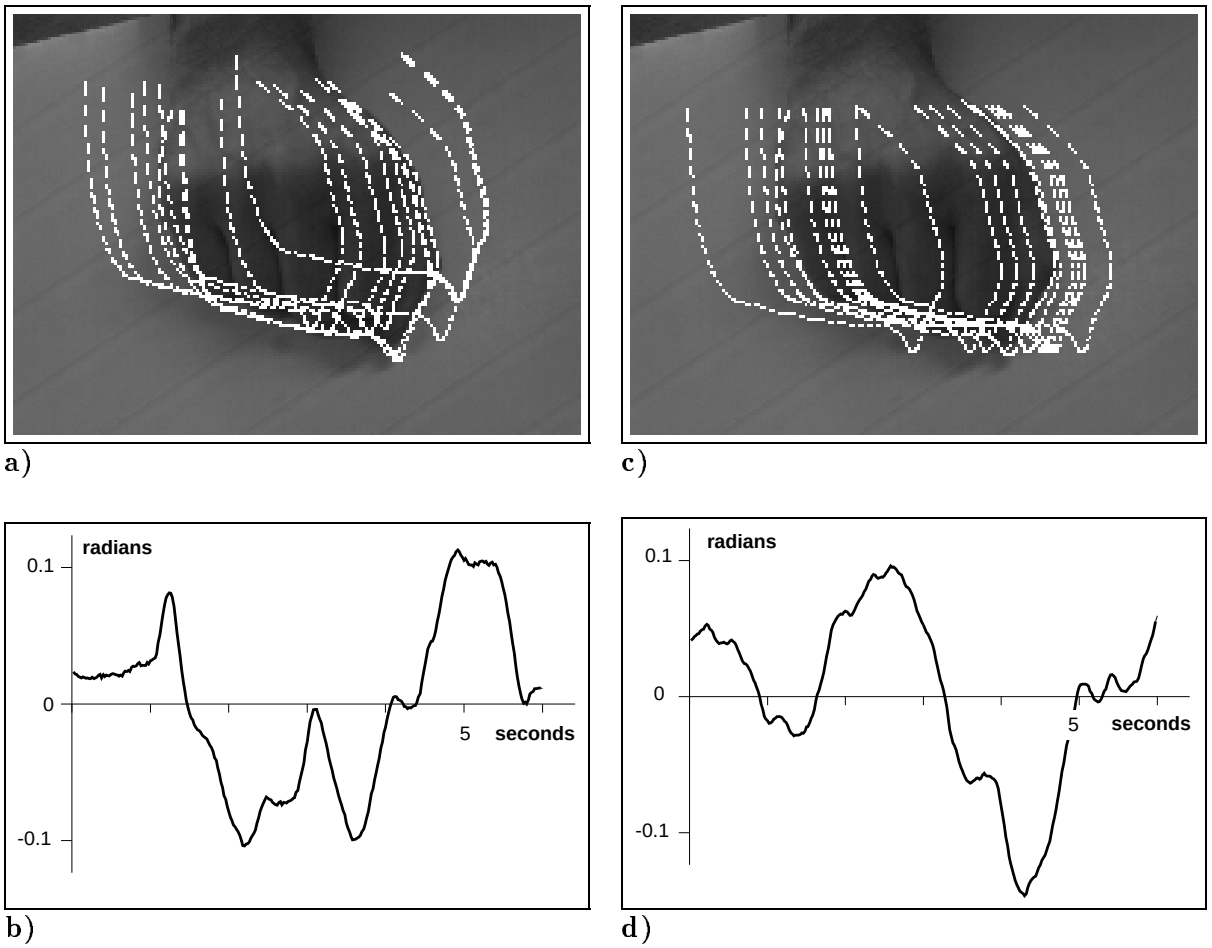


Figure 4: **Learning horizontal motion.** *a) Training sequence of a tracked hand captured at 50Hz (full video field-rate) in oscillatory motion. The initial frame of the hand is shown with a subsequence of the positions of the tracked curve overlaid. The centroid's horizontal position is displayed as a time-sequence in b). c) Simulation of the learned system: note that the motion swept out when the learned dynamics are simulated is similar to the motion in the training set. d) Centroid's horizontal position for the sequence in c). The synthesised motion appears to have reproduced a signal similar to that in the training set — observe particularly the width of the half-cycles and the amplitude of the signal.*

sinusoid could of course be represented completely by the model, using the second order dynamics and without recourse to the stochastic component of the model. The imperfect sinusoid cannot, it turns out, be represented completely; its long-term behaviour eludes the model but its short-term behaviour, which is of particular use for prediction in trackers, is captured. The horizontal motion training set, shown as a graph in figure 5b, is used in the estimation procedure of the previous section, to obtain discrete parameters a_0, a_1, b. They can be interpreted as a damped oscillation $\exp -\beta t \exp i\omega t$ in continuous time, using equation (8), where $\beta = 1.67$ and $\omega = 3.68$ giving a period $(2\pi/\omega)$ of 1.7 seconds. This is close to the apparent periodicity of the data set in which there are between 12 and 13 cycles over 20 seconds, a period of 1.54–1.67 seconds. The decay time constant of $1/\beta = 0.6$ s is not, however, apparent in the data set. It should be thought of as a coherence time constant arising because the training set does not fit our ideal model very closely: it contains the wrong kind of noise, probably closer to white noise than the integrated Brownian motion inherent in our model. Lastly, b also has a natural physical interpretation, in conjunction with the other parameters. It is the measure of the amplitude of the noise that drives the process and it can be shown that the RMS (root-mean-square) amplitude of the modelled process, in the steady state, is a function of a_0, a_1, b. For the model learned from this training set, the RMS amplitude is 0.06 rad,

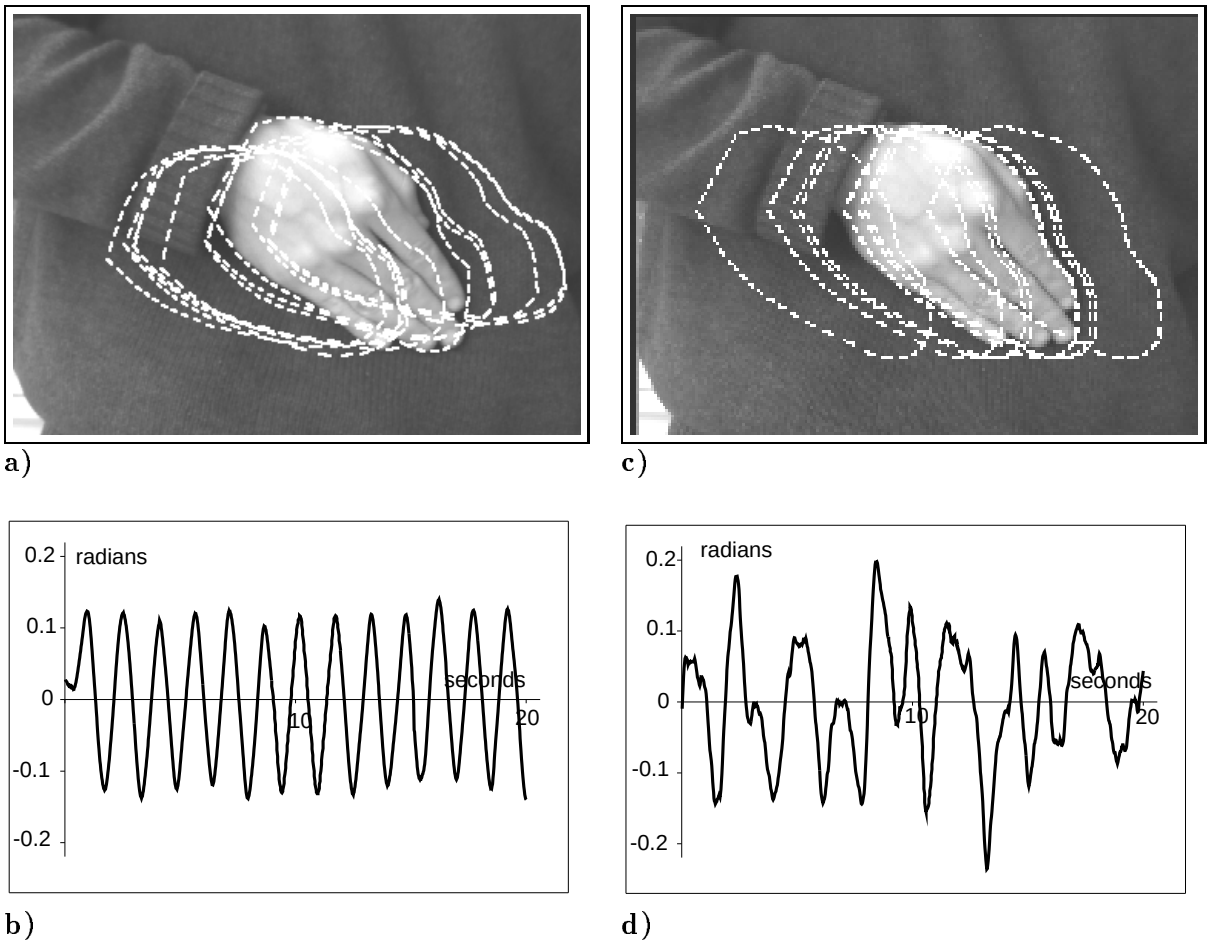


Figure 5: **Learning horizontal oscillation.** *As for figure 4, but now for an oscillatory data-set. a) Training sequence b) Centroid's horizontal position. c) and d) as a) and b) but for simulation of system learnt from the training set. The synthesised motion has captured the short term behaviour of the training set (e.g. half-cycles) but, because the motion model is limited to 2nd order, has not captured the phase-coherence that exists in the training set over longer time-scales.*

which appears close to the RMS amplitude of the training set in figure 5.

To illustrate the learned model, we can simulate, as before. This is done with static initial conditions, not that the initial conditions have much influence beyond the first few seconds, given that the exponential-decay time-constant of the system is less than one second. A sample signal, generated in this way, is shown in figure 5c,d. To check for consistency, we can re-learn parameters from this synthesised signal, treating it as a training set, which gives

$$\omega_0 = 3.69\,\mathrm{s}^{-1}, \quad \beta = 1.75\,\mathrm{s}^{-1}, \quad \text{and RMS amplitude} = 0.08\mathrm{rad},$$

which is close to the parameter set for the learned model. This indicates that the training procedure is valid, particularly that the sequence is sufficiently long to give repeatable results.

The synthesised signal illustrates a typical sample path from our family of second order stochastic differential equations. It is not, of course, exactly like the training set — it is unlikely that a signal like the training set would have been produced by the simulation. However the system is the member of the family of systems that is "closest" to the training set. Broadly, the small scale properties of the training set have been captured — the period of oscillations and amplitude — but not the large scale property of signal coherence. Clearly the model that has been learned is not a complete one but is adequate for tracking, in which the purpose of the dynamical model is to make predictions over relatively short time-scales whilst relying on data over longer time scales. Such a model, it can

be argued, is actually *more* appropriate than a more complete one which, while it may be suitable for signal *recognition*, would be too narrowly tuned — insufficiently general — to track a broad class of signals which is what is required here.

4.3 Estimation of multivariate system parameters

The general multivariate estimation algorithm follows broadly the line of the univariate one, but the separability of the estimation of deterministic and stochastic parameters, whilst it still holds, is no longer so obvious. For simplicity of notation we assume that the mean $\overline{\mathbf{Q}}$ has been subtracted off:

$$\mathbf{Q} \to \mathbf{Q} - \overline{\mathbf{Q}}.$$

The log-likelihood function for the multivariate normal distribution is then, up to a constant:

$$L(\mathbf{Q}_1, ..\mathbf{Q}_n | A_0, A_1, B) = -\frac{1}{2}\sum_{n=1}^{m-2} \left| B^{-1}\left(\mathbf{Q}_{n+2} - A_0\mathbf{Q}_n - A_1\mathbf{Q}_{n+1}\right)\right|^2 - (m-2)\log\det B. \tag{21}$$

Now the problem is to estimate A_0, A_1 and $C = BB^T$ by maximising the log-likelihood L. Maximising first with respect to A_0, A_1, it will be shown that separability holds — maxima with respect to A_0, A_1 turn out to be independent of the value of C. Equivalently to maximising L we minimise

$$f(A_0, A_1) = \sum_{n=1}^{m-2} \left| B^{-1}\left(\mathbf{Q}_{n+2} - A_0\mathbf{Q}_n - A_1\mathbf{Q}_{n+1}\right)\right|^2 \tag{22}$$

with respect to A_0, A_1. Now this function can be expanded as

$$f(A_0, A_1) = \mathrm{tr}(ZC^{-1})$$

where

$$\begin{aligned} Z &= S_{22} + A_1 S_{11} A_1^T + A_0 S_{00} A_0^T - S_{21}A_1 - S_{20}A_0^T + A_1 S_{10} A_0^T \\ &\quad - A_1 S_{12} - A_0 S_{02} + A_0 S_{01} A_1^T \end{aligned} \tag{23}$$

and

$$S_{ij} = \sum_{n=1}^{m-2} \mathbf{Q}_{n+i}\mathbf{Q}_{n+j}^T, \quad i,j = 0,1,2, \tag{24}$$

the second-order moment matrices for the multivariate time-sequence $\mathbf{Q}_n$, $n = 1, .., m$. Now, completing the square in Z, with respect to both A_0 and A_1, we can rewrite Z as

$$Z(A_0, A_1) = Z' + Z_0$$

where Z_0 is a constant matrix, and where

$$\begin{aligned} Z' &= (A_1 - \hat{A}_1)S_{11}(A_1 - \hat{A}_1)^T + (A_0 - \hat{A}_0)S_{00}(A_0 - \hat{A}_0)^T \\ &\quad + (A_1 - \hat{A}_1)S_{10}(A_0 - \hat{A}_0)^T + (A_0 - \hat{A}_0)S_{01}(A_1 - \hat{A}_1)^T \end{aligned}$$

and $\hat{A}_0, \hat{A}_1$ are the solutions of the simultaneous equations

$$\begin{aligned} S_{20} - \hat{A}_0 S_{00} - \hat{A}_1 S_{10} &= 0 \\ S_{21} - \hat{A}_0 S_{01} - \hat{A}_1 S_{11} &= 0. \end{aligned} \tag{25}$$

Now a minimum of f exists because it is bounded below by 0. It must therefore be a positive definite quadratic form, achieving its minimum when

$$A_0 = \hat{A}_0 \text{ and } A_1 = \hat{A}_1.$$

These conditions are independent of the value of C — the separability condition as required.

Having obtained estimators for A_0 and A_1, now C can be estimated. Rewriting (21) as

$$L = -\frac{1}{2}\mathrm{tr}(ZC^{-1}) + \frac{1}{2}(m-2)\log\det C^{-1},$$

fixing $A_0 = \hat{A}_0,\ A_1 = \hat{A}_1$, and extremising with respect to C^{-1} [using the identity $\partial(\det M)/\partial M \equiv (\det M)M^{-T}$] we obtain

$$\hat{C} = \frac{1}{m-2}Z(\hat{A}_0, \hat{A}_1), \tag{26}$$

which can be computed efficiently from (23).

4.4 Learning zoom

As an illustration of multivariable learning, a sequence of zooming motions of a box is inferred from training data by the algorithm described above, operating over the 6-dimensional planar-affine space (figure 6). It is instructive to examine the estimated model parameters. The zoom component of the sequence is oscillatory with a period of around 2.5s. The modes have complex frequencies (rad/s) of

$$-82,\ -57.9,\ (-54.4 \pm 1.55i),\ -39.8,\ -27.2, -9.61,\ -6.9,\ (-4.69 \pm 2.9i),\ (-0.54 \pm 2.76i)$$

of which the last, with the slowest decay time of around 2 seconds, has a period of 2.3s which corresponds closely to the period of the training set, around 2.5s. The corresponding mode, expressed in the basis $\mathcal{B}$, has approximately (within 5%) the form $(0, 0, 1, 1, 0, 0)$ corresponding to pure zoom. Similarly the 6×6 covariance matrix C, when expressed in the affine basis $\mathcal{B}$, is dominated by its central 2×2 block, corresponding to noise in the zoom process.

4.5 Nonrigid motion

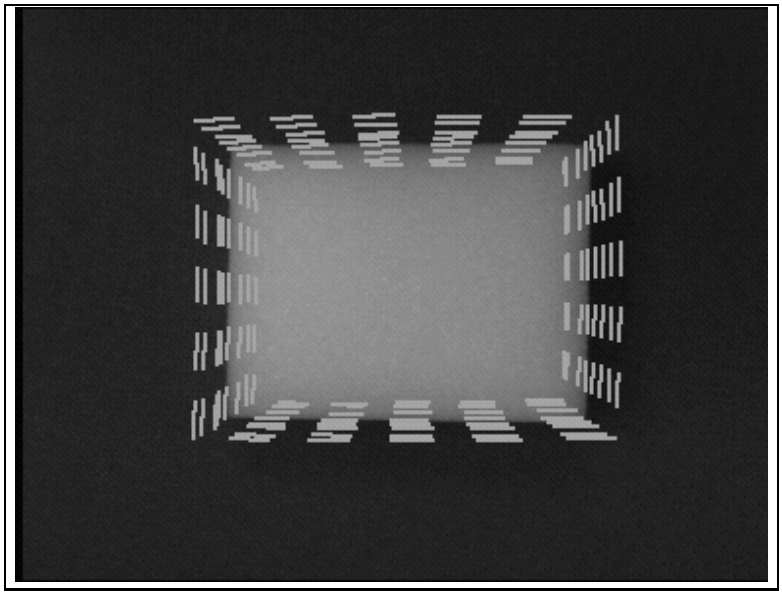
The methods illustrated so far for learning and tracking rigid motion can be extended to non-rigid motion. The tracking and learning algorithms described earlier continue to apply, but now the $\mathbf{Q}$-space must be redefined to parameterise non-rigid motion of a curve.

The $\mathbf{Q}$-space is extended by redefining the transformation matrices M, W of section 2.3. Augmenting the earlier definition, in which permutations of the template shape $(\overline{\mathbf{X}}, \overline{\mathbf{Y}})$ make up the W matrix, the W matrix must be given additional columns reflecting additional degrees of freedom for non-rigid motion. This then increases the dimension of the $\mathbf{Q}$ vector from 6 to some larger number. The extra degrees of freedom are derived from *key-frames*, typical non-rigid deformations, on which the tracked contour is positioned interactively, as in figure 7. Keyframes can be added directly to the subspace by appending each keyframe vector as a column of the W-matrix. Alternatively, a large set of keyframes can be used and economised to a smaller set using Principal Components Analysis (PCA), as Cootes et al. [15] and Baumberg and Hogg [8] do. During tracking, the first 6 components of the $\hat{\mathbf{Q}}_n$-vector report on the rigid motion, and the remaining components report on non-rigid motion, so rigid and non-rigid motion can be monitored somewhat independently.

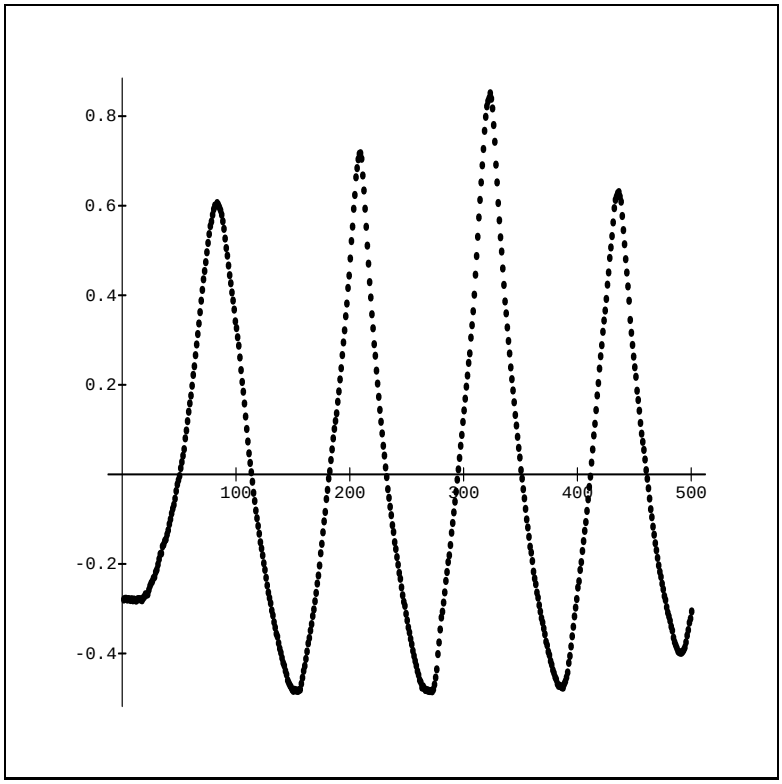
This linear mechanism for handling nonrigidity is in marked contrast to the detailed, nonlinear kinematic model used by Rehg and Kanade [32]. The advantage of retaining linearity is reduced complexity (nonlinear models require the computationally more complex Extended Kalman Filter for tracking [20]) and possibly improved stability. The disadvantage is that the linearly approximated kinematics are appropriate only for small displacements. Linear approximations are possible for larger displacements but produce underconstrained kinematic models.

5 Trained filters

This section investigates the effect on tracking performance of replacing default dynamics in the predictor with specific dynamics learned from a training set. The tracking procedure for learned



a)



b)

Figure 6: **Learning zooming motion.** *a) A training sequence of 10 seconds duration, captured at 50 Hz by the "default" tracker, tracks an object as it zooms inwards and outwards, spanning a family of contours as shown. This is used to estimate a system model and then to construct a tracker trained for the zooming motion. The zoom component of the training sequence in a) is plotted in b) as change in scale factor vs. field number (at 50Hz). It shows a dominant period of about 2.5 seconds.*

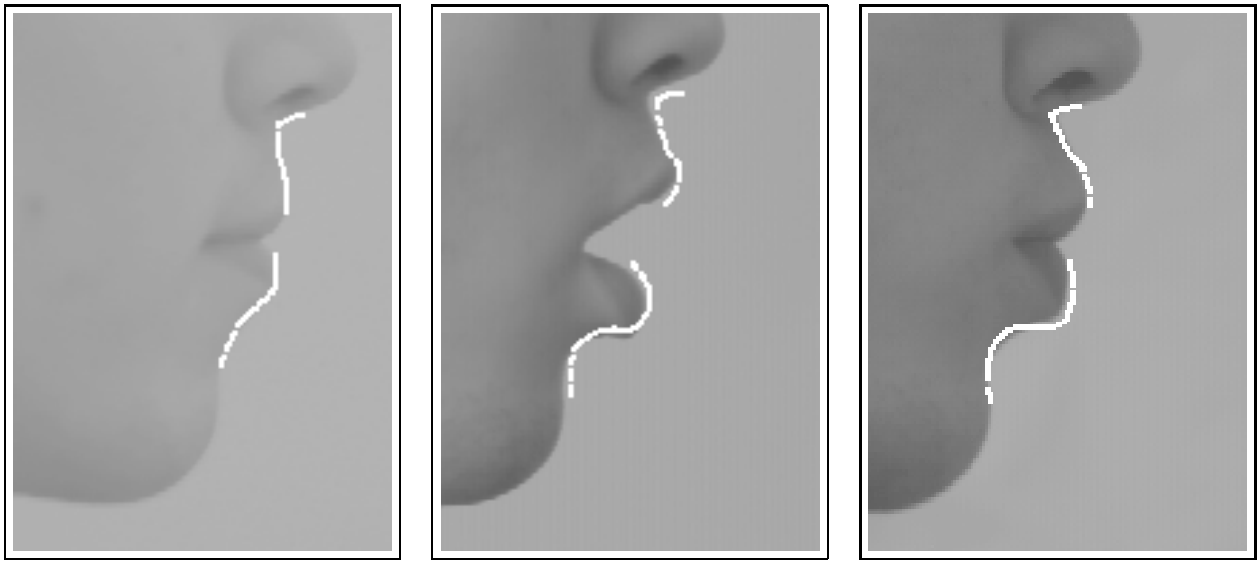


Figure 7: **Keyframes.** *Lips template followed by two key-frames, representing interactively tracked lips in characteristic positions. The key-frames are combined linearly with appropriate rigid degrees of freedom, to give a* **Q***-space suitable for use in a tracker for non-rigid motion.*

motions has been implemented for real-time (50 Hz) tracking on a SUN Sparc-II computer with Datacell S2200 framestore, without the need for any additional hardware. It has been applied to a variety of training sequences with different objects, involving various motions, both rigid and non-rigid. Trackers have been trained for the whole affine subspace corresponding to 3D rigid motion of a hand, including translation parallel to the image plane, longitudinal motion (zoom), image plane rotation, and rotation about an arbitrary three-dimensional axis. They have also been trained to follow nonrigid motions both of hands and of lips.

For both rigid and nonrigid motions, tracking performance can be greatly enhanced by training. There are two contributory effects here, one static and one dynamic — one in configuration-space and one in phase-space. The static effect is that learning characterises the likely configurations — both shapes and positions — of the visible contour; in the tracker this helps maintain a closer match between the tracked and the actual contour. Algorithms to learn static models have been demonstrated previously [21, 15] but such algorithms are unable to exploit training sets that are gathered as time-sequences. For static algorithms, permuting the order of elements in a training set has no effect on the model learned. Our algorithm exploits the time-sequence structure by simultaneously learning static and dynamic components of an object model. Learning is in phase-space, rather than merely in configuration-space. Once learned, the dynamic component of the model is used to great effect in tracking to bridge any hiatus in the measurement process, either when the target is temporarily obscured or when spatial lag is so great that the target momentarily falls outside the measurement window.

5.1 Rigid hand-motion: configuration-space training

Three training sets are used here, each involving one component of rigid body motion, as in figure 8. The hand is almost planar, so the motions result in 2D affine deformations in the image plane. After training, the tracker follows motions similar to the ones in the training set, but will not follow other rigid body motions. This is illustrated in figure 9. Component motions of this kind can be built up individually and combined to form the predictor for a single tracker. The tracker is then tuned to follow not merely the disjunction of the component motions, but also any linear combination of them. Thus learning can be achieved in a *modular* fashion, and the result exhibits some degree of generalisation. How can combination of learned models be achieved? Mere concatenation of data-sets for each of the individual components does not quite work because the resulting learned model would be influenced by the abrupt transitions between data-sets, which are spurious. Instead, maximum likelihood estimates of system parameters are achieved by *adding moments* $S_{i,j}$ (24) from each of the datasets:

$$S_{i,j} = \sum_k S_{i,j}^{(k)} \quad \forall i,j$$

to achieve combined moments for use in the estimation algorithm (25).

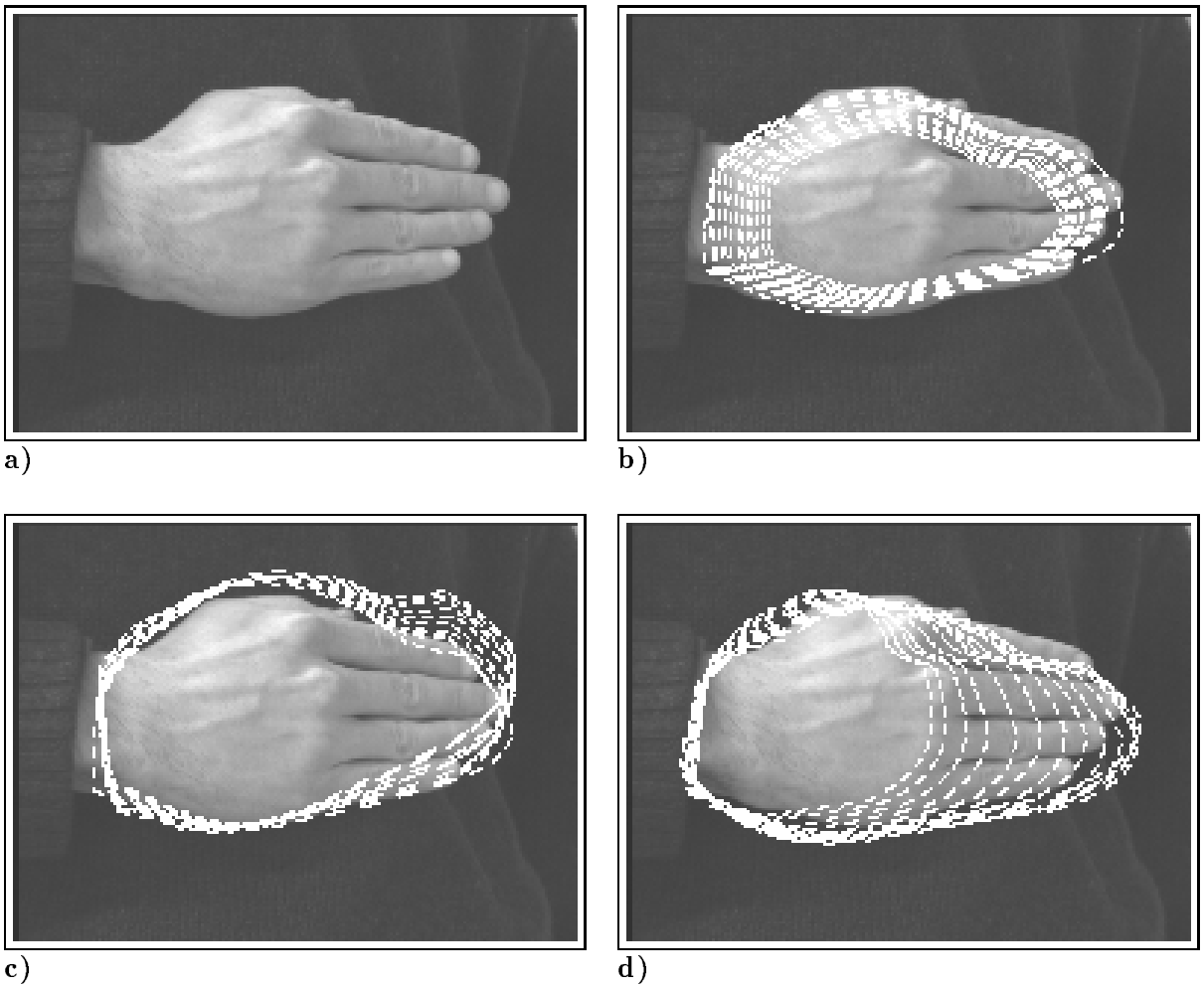


Figure 8: **Training single components of affine deformation**. *a) Hand in rest position. Training sets are shown for: b) zoom — hand moves towards and away from camera; c) rotation about the line of sight; d) flapping — rotation about an axis parallel to the image plane.*

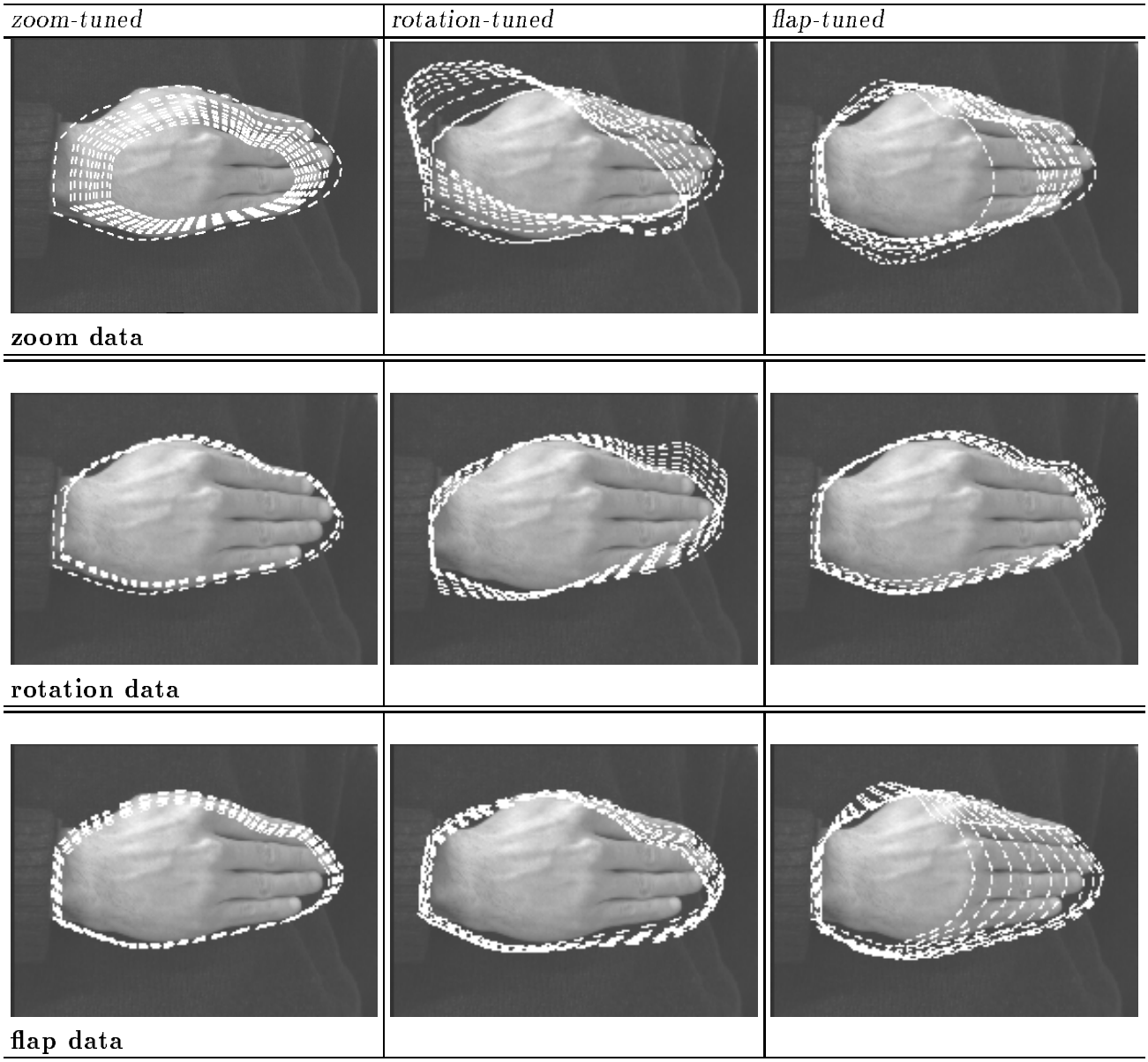


Figure 9: **Filtering single components of affine deformation** *The three training sets from figure 8 are used here to generate filters sensitive specifically to zoom, rotation and flapping. Each filter is tested on the zoom, rotation and flapping training sequences to illustrate the specificity of training. Accurate tracking is shown in images along the diagonal in which each filter is run on its own training sequence. Off the diagonal, when a tuned tracker is applied to an inappropriate test motion sequence, tracking fails.*

5.2 Hand tracking: phase-space training

Configuration-space training, as demonstrated above, captures only the static component of the object-model. The next experiment demonstrates the power of incorporating learned dynamics into a curve tracker. This time, a training set of *vertical*, oscillatory, rigid motion has been generated and used to learn motion coefficients A, B, similar to the earlier example of learned horizontal motion (section 4.2). The training set is shown in figure 10.

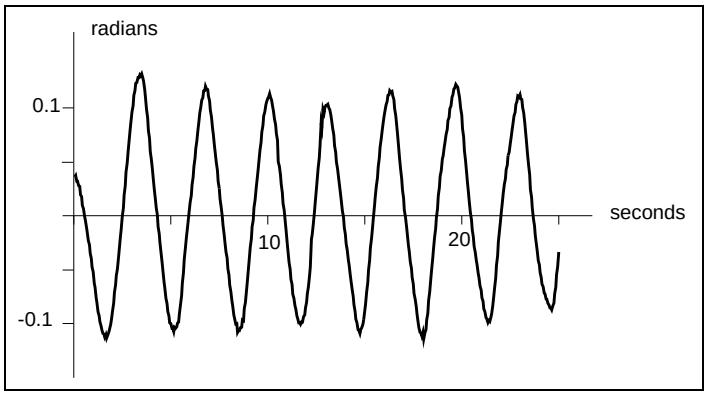


Figure 10: **Training set for rigid, oscillatory, vertical motion.** *The centroid's vertical position is displayed as a time-sequence, over the 25 second duration of the training sequence.*

Testing of the trained tracker, incorporating the learned motion, is done against two forms of the un-trained, default tracker. In both forms, translational motion is allowed freely. The first form is "stiff", indicating that motion other than translation is resisted strongly — the values of $\beta, \omega, \beta', \omega'$ in (16) are large ($\beta = 45\text{s}^{-1}, \omega = 31\text{s}^{-1}, \beta' = 50\text{s}^{-1}, \omega' = 37\text{s}^{-1}$ where β, ω govern non-affine deformations and β', ω' govern affine ones). The second form is "hard" in which those values are effectively infinite; then the shape of the estimated curve is truly rigid, having only translational degrees of freedom.

The test sequences consist of rapid, vertical, oscillatory motions of a hand. The sequences are stored on video so that fair comparisons can be made, using the standard sequences, of the performance of different trackers. Two sequences are made: one of regular oscillation against distracting background features — "clutter", the other of oscillatory motion with gradually increasing frequency — a "chirp". Results of the clutter test are shown in figure 11. In this case the stiff-untrained tracker fails but the trained tracker succeeds. However the hard-untrained tracker also succeeds because it has a stronger shape-memory than the stiff-untrained tracker. This indicates that it is the trained tracker's learned *shape*, more than its learned *dynamics*, that enables it to track successfully in clutter. In the case of the chirp test, the results which are shown in figure 12 are rather different. In this test it is the rapidity of the oscillations that make tracking difficult. The trained tracker follows the motion successfully, right up to a rate of around 3 oscillations per second. Neither of the untrained trackers can achieve this which indicates that it is the learned *dynamics* of the trained tracker that enables it to succeed here. This is confirmed by the snapshots in figure 12 which show that when the measurement process fails due to excessive lag in the tracker, it is the learned dynamical model that effectively bridges the hiatus and allows lock subsequently to be recovered.

5.3 Nonrigid motion: lips

The feasibility of tracking lip movements frontally when lip high-lighter is worn (figure 13), was demonstrated by Kass et al [25] and our system can do this at video-rate. This paradigm can be extended by using highlighter on a number of facial features, as Terzopoulos and Waters [36] did. It is expected that this could be used with our real-time trainable tracker to build an effective front-end for actor-driven animation, without recourse to expensive virtual-reality input devices.

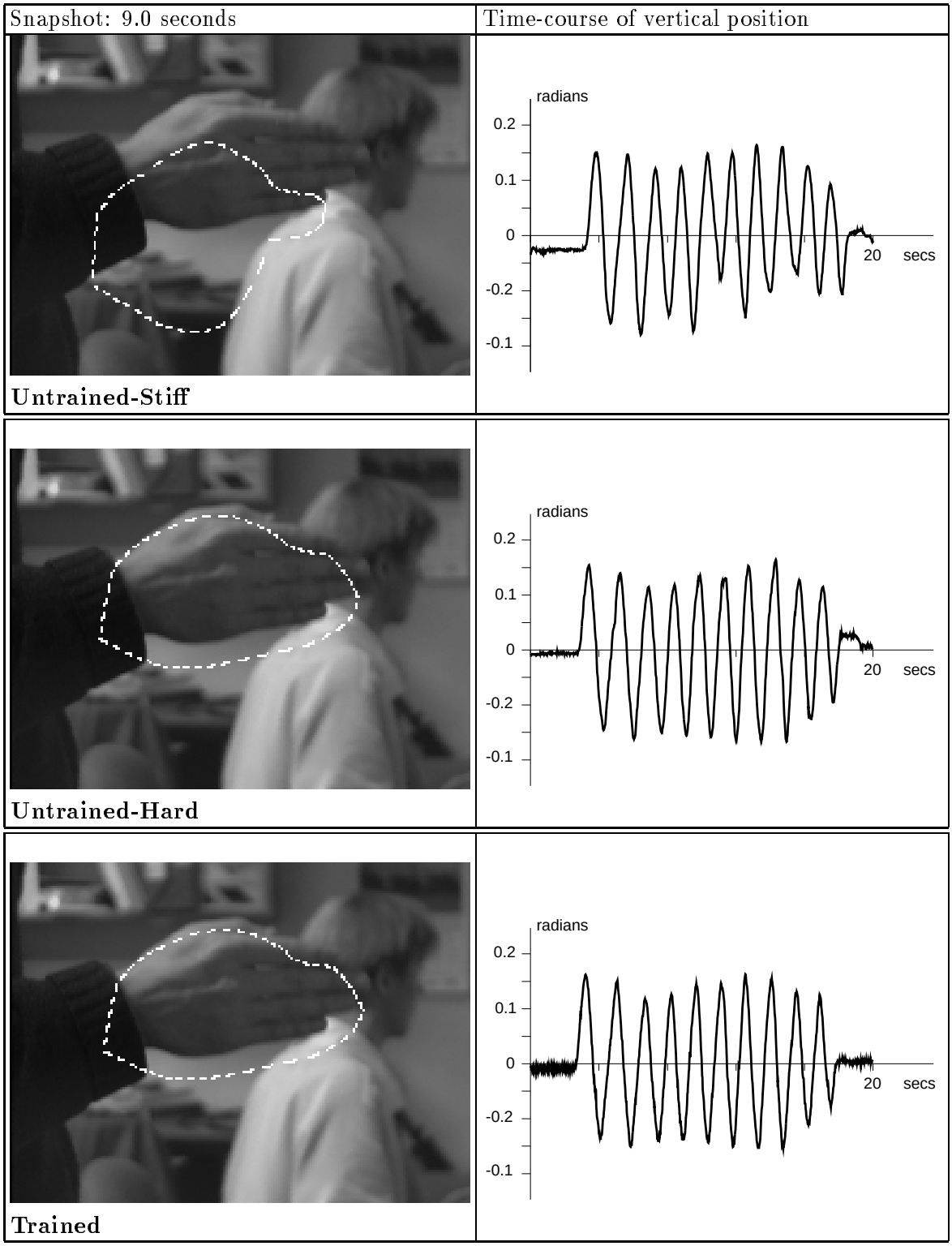


Figure 11: **Trained tracker for oscillatory rigid motion, tested against clutter**. *The two untrained trackers and the trained tracker are compared here using a "clutter" test-sequence. After 9.0 seconds, the stiff-untrained tracker is distracted by background clutter and loses lock, but the hard-untrained and the trained trackers continue to track successfully.*

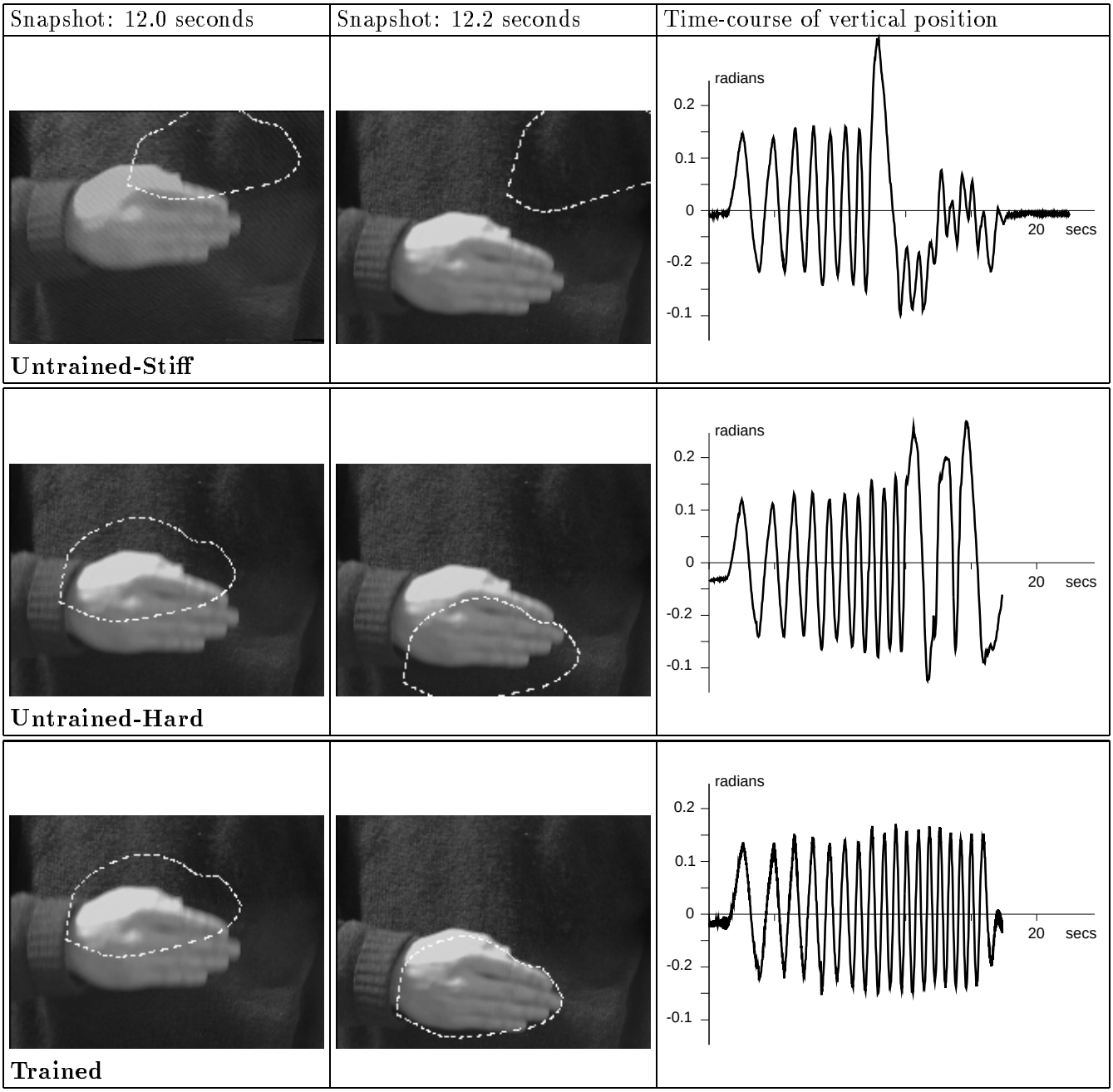


Figure 12: **Trained tracker for rigid motion, tested with rapid oscillations**. *A "chirp" test motion, consisting of vertical oscillations of progressively increasing frequency, is tracked by each of three trackers: untrained-stiff, untrained-hard and trained. For the untrained trackers, lock is lost after about 12 seconds and is unrecoverable another 0.2 second later. The trained tracker is still tracking after 12 seconds, though it is lagging sufficiently (more than 40 pixels, the size of the measurement window) to have lost lock. The learnt model takes over tracking temporarily, in the absence of measurements, and by 12.2 seconds lock has been recovered.*

Figure 13: *Tracking a frontal view of lips is possible if lip highlighter is worn, to ensure adequate contrast with skin.*

Tracking lips side-on, whilst arguably less informative, has the advantage of working in normal lighting condition without cosmetic aids. This could be used to turn the mouth into an additional workstation input-device. Deformations of the lips for two sounds are shown in figure 14. Two

Figure 14: **Single-syllable training.** *Deformations of the mouth are shown corresponding to the sounds a) "Pah" and b) "Ooh".*

trackers are trained individually for the sounds "Ooh" and "Pah", and the resulting selectivity is shown in figures 15 and 16. It is clear from these results that the tuning effect for individual sounds is strong. Again, as in the earlier flap/zoom/rotate demonstration for rigid motion, the point is not that this particular tuning is operationally desirable, but that it demonstrates the power of the tuning process. Operationally, to render assistance to speech analysis, it is necessary to learn the repertoire of lip-motions that occurs in typical connected speech.

5.4 Connected speech

Now, complexity is increased by training on connected speech. Two-stage training is used. In the bootstrap stage, the default tracker followed a slow-speech training sequence which is then used, via the learning algorithm, to generate a tracker. This tracker is capable of following speech of medium speed and is used to follow a medium-speed training sequence, from which dynamics for a full-speed tracker are obtained. The trained tracker is then tested against the default tracker, using a test sequence entirely different from the training sequences. Two deformation components of lip motion are extracted, at 50Hz, as "lip-reading" signals. The more significant one, in the sense that it accounts for the greater part of the lip motion, corresponds approximately to the degree to which the lips are parted. This component is plotted both for the default tracker and the partly and fully trained ones in figure 17. It is clear from the figure that the trained filter is considerably more agile. In a demonstration in which the signal is used to animate a head, the untrained filter is able to follow only very slow speech, whereas the trained filter successfully follows speech delivered at a normal speed.

5.5 Nonrigid hand motions

Both rigid and nonrigid motion of a hand can be used as a 3D input device. The freedom of movement of the hand is illustrated in figure 18, with rigid motion picked up to control 3D position and attitude, and nonrigid motion signaling button pressing and "lifting". The tracker output — the components of $\hat{\mathbf{Q}}$ varying over time — have successfully been used to drive a simulated object around a 3D environment.

5.6 Full Kalman filter

Finally, for the case of rigid motion of the hand, the effect of training is shown in the full Kalman filter with time-varying gains and validation gates operating on the measurement process. Allowing the validation gate to operate enhances the tracker's powers of recovery from loss of lock. Performance is reduced, not only when training is omitted (as we have already seen in numerous examples), but also when Kalman gains are fixed. This is clearly demonstrated in figures 19 and 20.

6 Conclusions

A new learning algorithm has been described for live tracking of moving objects from video. It supplies particular dynamics, modelled by a stochastic differential equation, to be used predictively in a contour tracker. The process is bootstrapped by a default tracker which assumes constant velocity rigid motion driven randomly. It is crucial that the constraints of rigid-body motion are incorporated — represented in our algorithm by the $\mathbf{Q}$-space. This is what allows stable tracking, for which the number of free parameters must be limited, to be combined with the apparently conflicting requirement that a large number of control points are needed for accurate shape representation. Rather than using arbitrarily chosen dynamics in the tracker they are acquired by a learning algorithm that allows dynamical models to be built from examples. When such a model is incorporated into a tracker, agility and robustness to clutter are considerably increased. In the case of non-rigid motion, the learning algorithm has proved, so far, to be essential to obtaining any satisfactory tracking performance.

The advent of workstations with integral cameras and framestores (designed to facilitate teleconferencing) brings an opportunity for these algorithms to be put to work. Unadorned body parts become usable input devices for graphics. This has potential applications in user-interface design, automation of animation, virtual reality, surveillance, the design of computer aids for the handicapped and perhaps even low-bandwidth teleconferencing.

This exploration of the learning of visual motion raises many issues. For example: how can the acuity of the tuning of a model, or conversely its generality, be controlled? Can the learning paradigm be developed to allow model-based recognition based on visual motion, using tuned trackers

as a bootstrap stage? Is it possible to learn disjunctions of motion models, either by raising the model order or by explicit model switching?

Acknowledgements

We are grateful for the use of elegant software constructed by Rupert Curwen, Nicola Ferrier, Simon Rowe, Henrik Klagges and for discussions with Mike Brady, Roger Brockett, Yael Moses, David Mumford, Brian Ripley, Richard Szeliski, Lionel Tarassenko, Andrew Zisserman. We acknowledge the support of the AFRC (DR) and the SERC (AB), and of the Newton Institute, Cambridge.

References

[1] K. J. Astrom. *Introduction to stochastic control theory.* Academic Press, 1970.

[2] K. J. Astrom and B. Wittenmark. *Computer Controlled Systems.* Addison Wesley, 1984.

[3] N. Ayache, I. Cohen, and I. Herlin. Medical image tracking. In A. Blake and A. Yuille, editors, *Active Vision*, pages 285–302. MIT, 1992.

[4] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Trans. Pattern Analysis and Machine Intell.*, 15(6):602–604, 1993.

[5] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[6] S. Barnett. *Matrices: Methods and Applications.* Oxford University Press, 1990.

[7] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, 1987.

[8] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In J-O. Eklundh, editor, *Proc. 3rd European Conference on Computer Vision*, pages 299–308. Springer–Verlag, 1994.

[9] A. Bennett and I. Craw. Finding image features for deformable templates and detailed prior statistical knowledge. In P. Mowforth, editor, *Proc. British Machine Vision Conference*, pages 233–239, Glasgow, 1991. Springer-Verlag, London.

[10] A. Blake, J.M. Brady, R. Cipolla, Z. Xie, and A. Zisserman. Visual navigation around curved obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 3, pages 2490–2499, 1991.

[11] A. Blake, R. Curwen, and A. Zisserman. Affine-invariant contour tracking with automatic control of spatiotemporal scale. In *Proc. 4th Int. Conf. on Computer Vision*, pages 66–75, 1993.

[12] A. Blake, R. Curwen, and A. Zisserman. A framework for spatio-temporal control in the tracking of visual contours. *Int. Journal of Computer Vision*, 11(2):127–145, 1993.

[13] A. Blake and A. Yuille, editors. *Active Vision.* MIT, 1992.

[14] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *Proc. 3rd Int. Conf. on Computer Vision*, pages 616–625, 1990.

[15] T.F. Cootes, C.J. Taylor, A. Lanitis, D.H. Cooper, and J. Graham. Buiding and using flexible models incorporating grey-level information. In *Proc. 4th Int. Conf. on Computer Vision*, pages 242–246, 1993.

[16] R. Curwen and A. Blake. Dynamic contours: real-time active splines. In A. Blake and A. Yuille, editors, *Active Vision*, pages 39–58. MIT, 1992.

[17] E.D. Dickmanns and V. Graefe. Applications of dynamic monocular machine vision. *Machine Vision and Applications*, 1:241–261, 1988.

[18] I.D. Faux and M.J. Pratt. *Computational Geometry for Design and Manufacture*. Ellis-Horwood, 1979.

[19] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE. Trans. Computers*, C-22(1), 1973.

[20] A. Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

[21] U. Grenander, Y. Chow, and D. M. Keenan. *HANDS. A Pattern Theoretical Study of Biological Shapes*. Springer-Verlag. New York, 1991.

[22] C. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*, pages 59–74. MIT, 1992.

[23] D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[24] B.K.P. Horn. *Robot Vision*. McGraw-Hill, NY, 1986.

[25] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, pages 259–268, 1987.

[26] J.J. Koenderink and A.J. Van Doorn. Affine structure from motion. *J. Optical Soc. of America A.*, 8(2):337–385, 1991.

[27] D.G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *Int. Journal of Computer Vision*, 8(2):113–122, 1992.

[28] S. Menet, P. Saint-Marc, and G. Medioni. B-snakes: implementation and application to stereo. In *Proceedings DARPA*, pages 720–726, 1990.

[29] A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:730–742, 1991.

[30] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1988.

[31] B.S.Y. Rao, H.F. Durrant-Whyte, and J.A. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *ijrr*, 12(1):20–44, 1993.

[32] J.M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In J-O. Eklundh, editor, *Proc. 3rd European Conference on Computer Vision*, pages 35–46. Springer–Verlag, 1994.

[33] G.D. Sullivan. Visual interpretation of known objects in constrained scenes. *Phil. Trans. R. Soc. Lond. B.*, 337:109–118, 1992.

[34] R. Szeliski and D. Terzopoulos. Physically-based and probabilistic modeling for computer vision. In B. C. Vemuri, editor, *Proc. SPIE 1570, Geometric Methods in Computer Vision*, pages 140–152, San Diego, CA, July 1991. Society of Photo-Optical Instrumentation Engineers.

[35] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7), 1991.

[36] D. Terzopoulos and K. Waters. Analysis of facial images using physical and anatomical models. In *Third International Conference on Computer Vision*, pages 727–732, Osaka, Japan, December 1990.

[37] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[38] A. Yuille and P. Hallinan. Deformable templates. In A. Blake and A. Yuille, editors, *Active Vision*, pages 20–38. MIT, 1992.

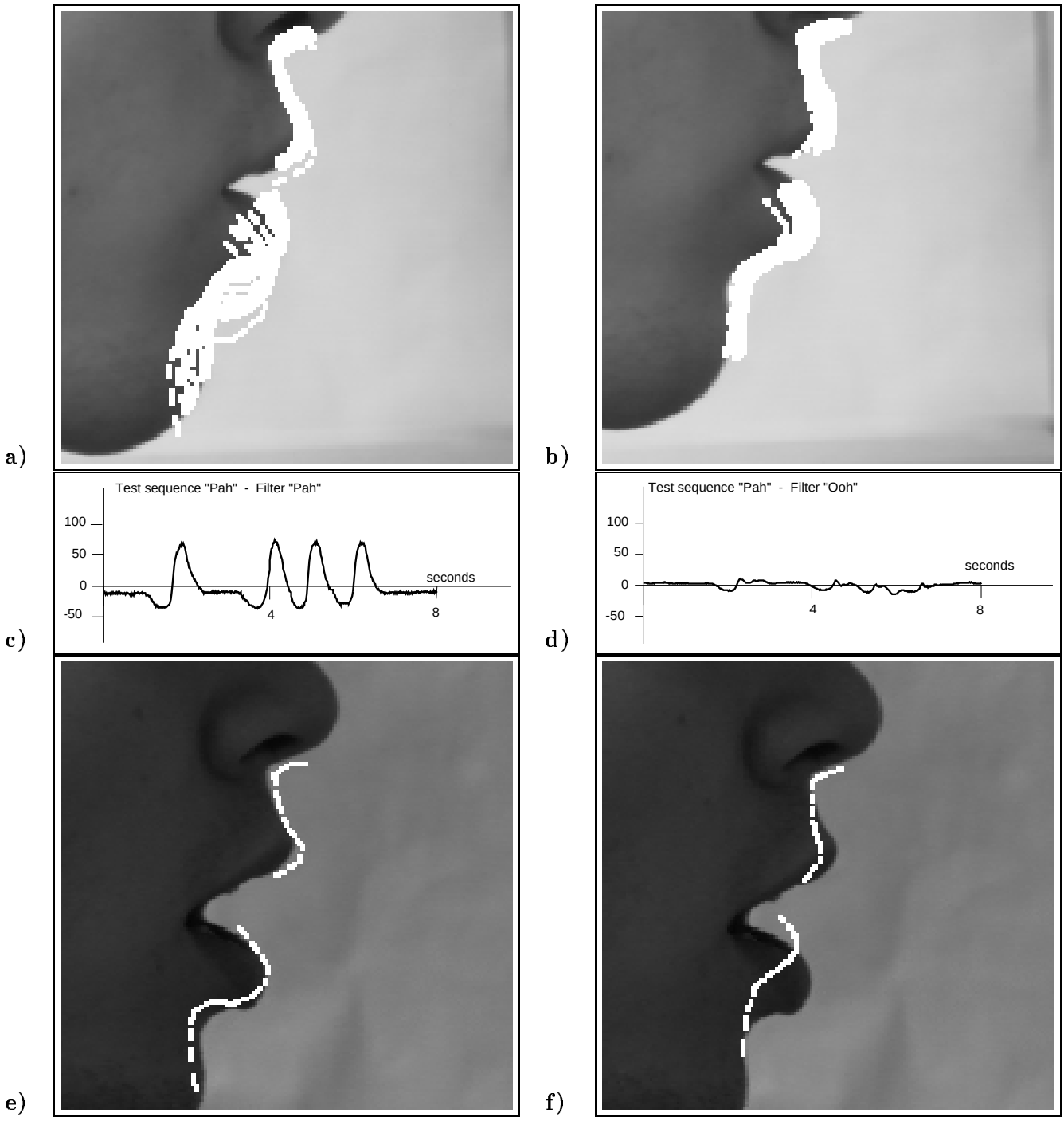


Figure 15: **Filter selectivity for sounds**. *A test sequence in which the sound "pah" is repeated, is tracked by filters trained on the sounds "pah" a) and "ooh" b) — the pictures show the area swept out by successive positions of the tracked contour. Note that the "pah" filter successfully tracks the opening/shutting deformation of the mouth whereas only lateral translation of the head is tracked by the "ooh" filter. The nonrigid deformation containing the speech information is lost because a tracker trained on "ooh" cannot accommodate deformations associated with "pah". Motion signals corresponding to a), b) are plotted in terms of an appropriate scalar component of estimated motion in c) and d) respectively. Tracked contours approximately 4.1s after the start of the signal are shown in e) and f) respectively. The signal c) shows clear pairs of troughs (shutting of the mouth) and peaks (opening), one pair for each "pah" but in d) (tracker trained on "ooh") there is minimal opening/shutting response.*

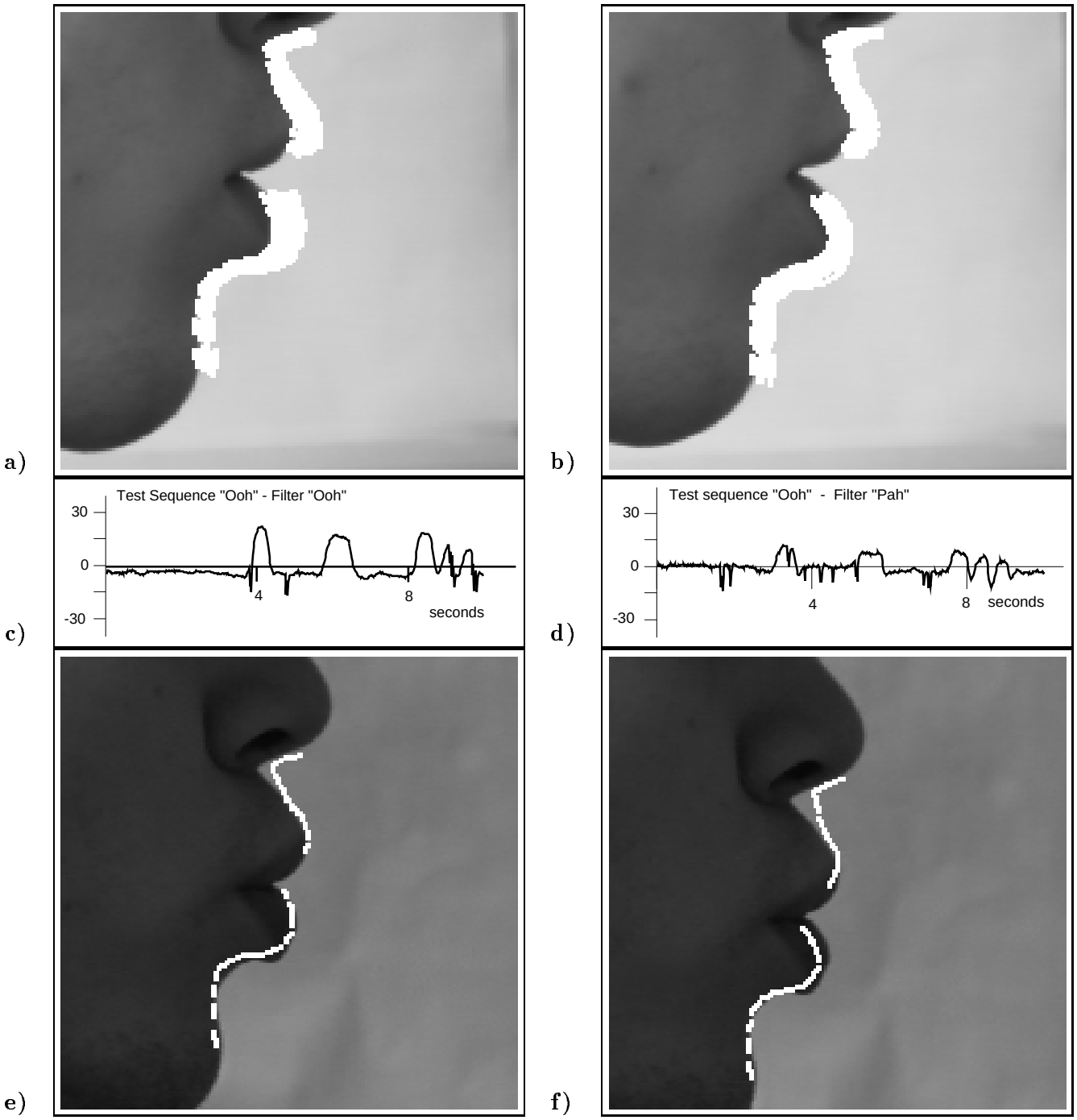


Figure 16: **Filter selectivity for sounds**. *As a companion to figure 15, repeated "ooh" sounds are tracked by filters trained on "ooh" a) and "pah" b), with corresponding motion signals in c) and d) and tracked contours, approximately 5.7s after the start of the signal, in e) and f) respectively. The component of motion plotted here is different to the one in figure 15, now being appropriate to the "ooh" signal. As before, the appropriately trained filter shows a greater response. The swept motion in b) is pure translation whereas in a) the central bulge in the white region indicates the deformation that accompanies the "ooh" sound. The signal in d) is attenuated compared with the signal in c) and is also noisier.*

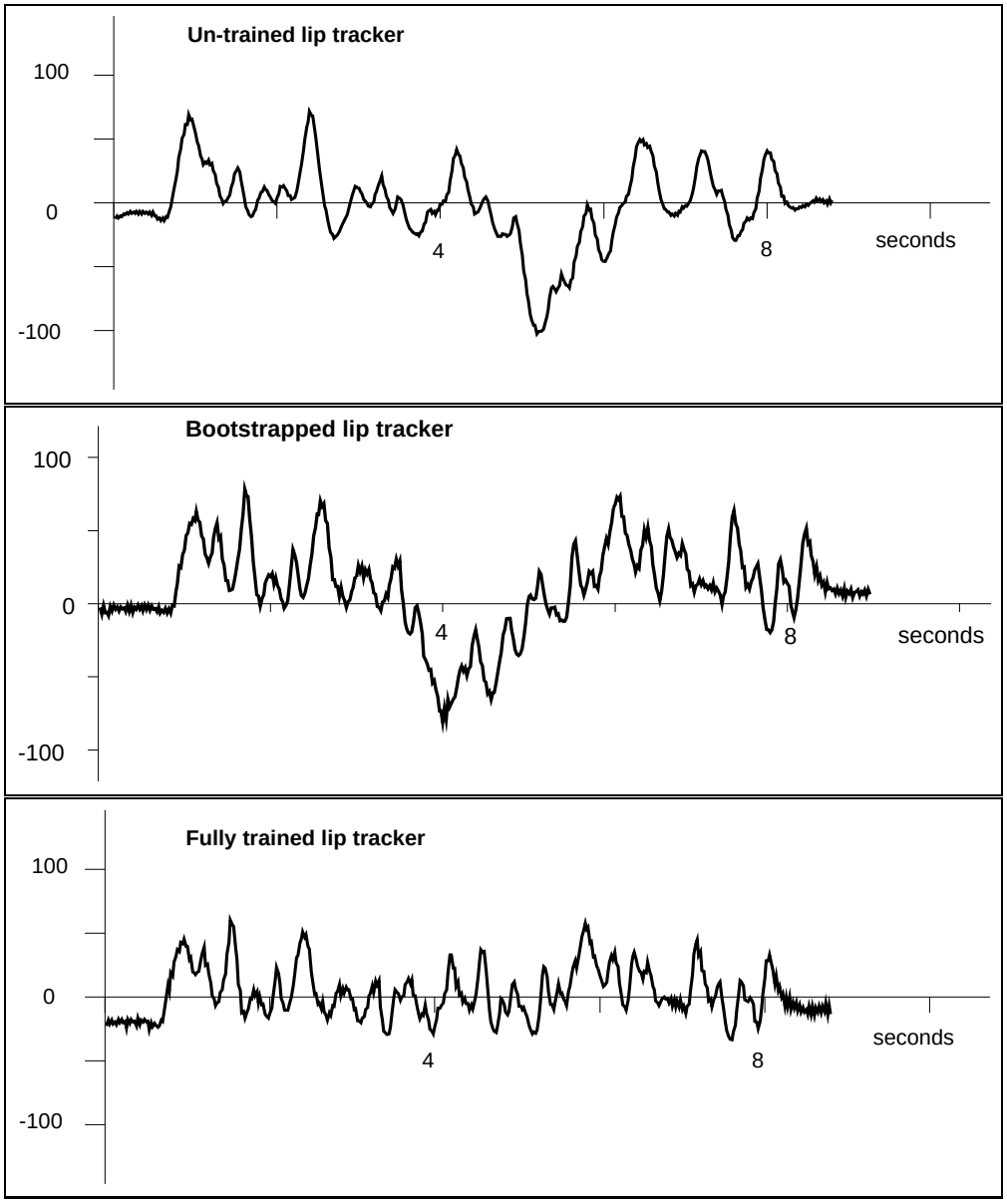


Figure 17: **Trained lip tracker.** *Training a tracker for side-on viewing of speaking lips greatly enhances tracking performance. The graphs show plots from the un-trained, default filter, the bootstrapped filter after one training cycle and lastly the filter after a second training cycle. One component of deformation of the lips is shown, corresponding to the degree to which the mouth is open — the space of deformations spanned by the first two templates in figure 7. Note the considerable loss of detail in the default filter and the overshoots in both default and bootstrapped filters, compared with the fully trained filter. (The sentence spoken here was "In these cases one would like to reduce the dependence of a sensory information processing algorithm on these constraints if possible.".)*

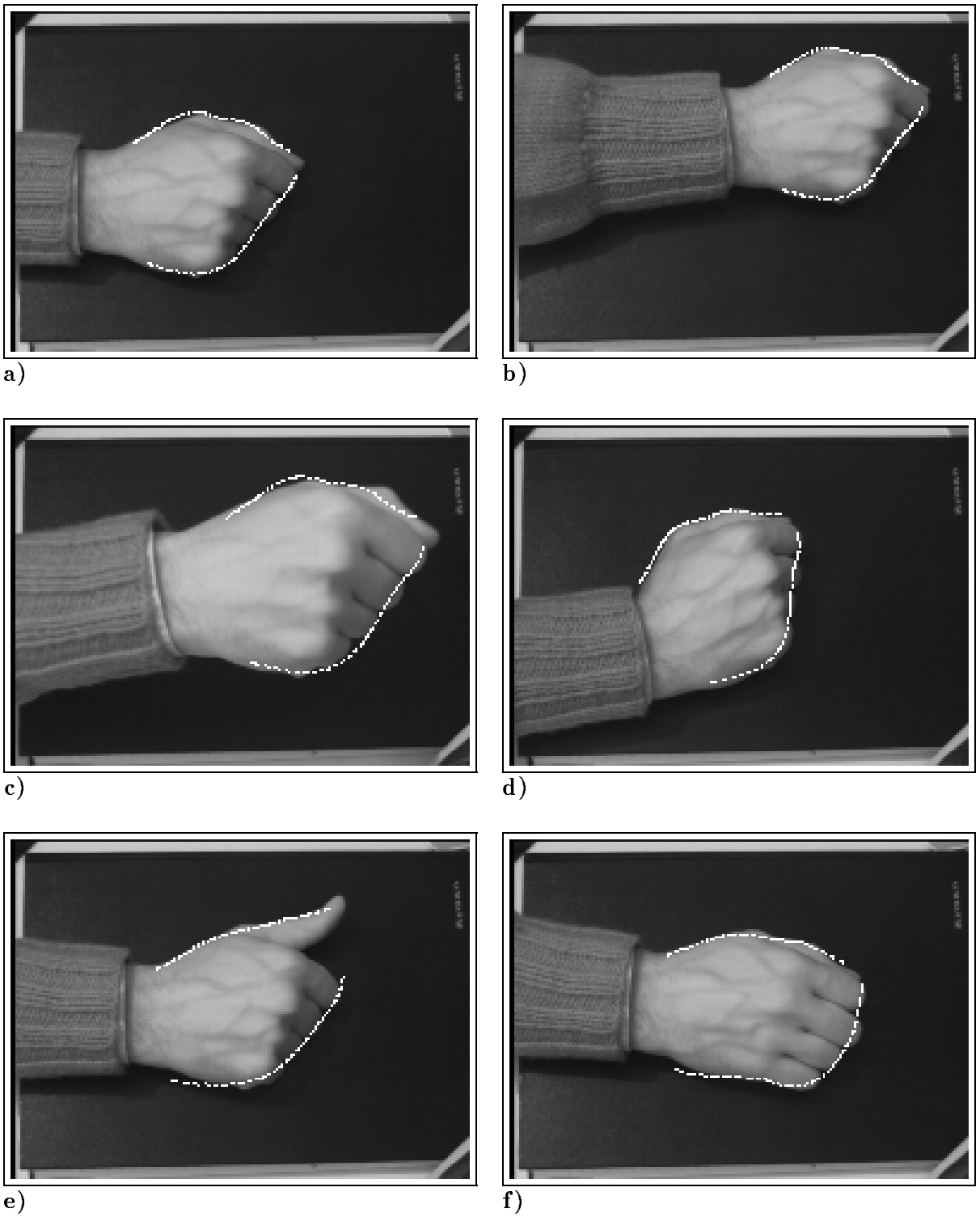


Figure 18: **The unadorned hand as a 3D mouse.** *A hand in its home position (a) can move on the xy-plane of the table (b) to act as a regular mouse but can also rise in the z direction (c) and the zooming effect is picked and used to compute z. Rotation can also be tracked (d). Nonrigid motion tracking can be used to pick up signals. For instance (e) signals a button-press and (f) signals the analogue of lifting a conventional mouse to reposition it.*

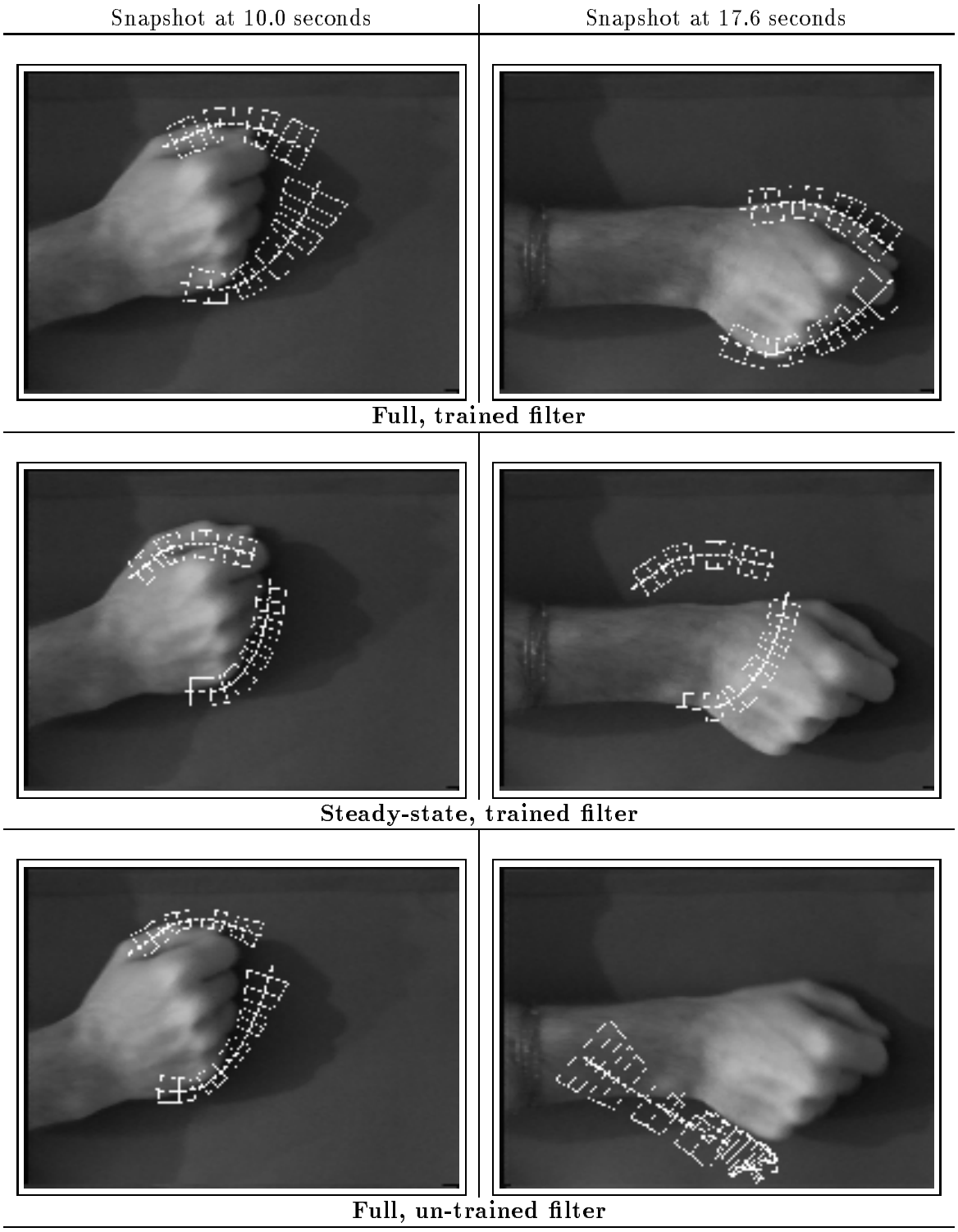


Figure 19: **Trained full Kalman filter.** *Performance in tracking of a test sequence is shown for the full (time-varying), trained Kalman filter versus a steady state version and an un-trained version of the same filter. Snapshots clearly show eventual failure of tracking, on a particular fixed test motion sequence, for all except the full, trained filter — see also the next figure.*

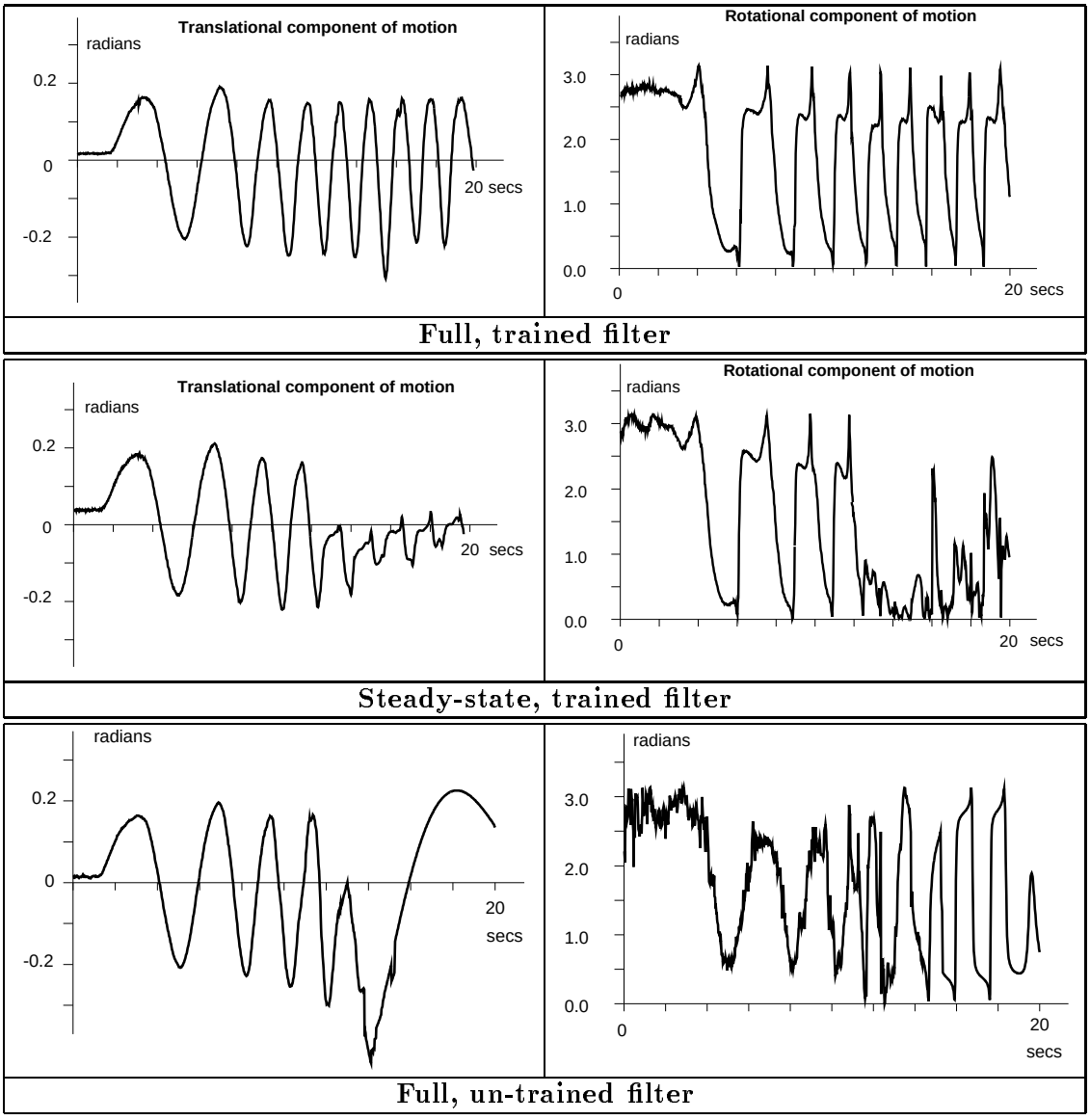


Figure 20: **Trained full Kalman filter.** *Tracking failures for the untrained and steady-state filters, shown in the snapshots of the previous figure, are displayed graphically here.*