3D position, attitude and shape input using video tracking of hands and lips

Andrew Blake[1] and Michael Isard[1]
Robotics Research Group, University of Oxford.

ABSTRACT

Recent developments in video-tracking allow the outlines of moving, natural objects in a video-camera input stream to be tracked live, at full video-rate. Previous systems have been available to do this for specially illuminated objects or for naturally illuminated but polyhedral objects. Other systems have been able to track non-polyhedral objects in motion, in some cases from live video, but following only centroids or key-points rather than tracking whole curves. The system described here can track accurately the curved silhouettes of moving non-polyhedral objects at frame-rate, for example hands, lips, legs, vehicles, fruit, and without any special hardware beyond a desktop workstation and a video-camera and framestore.

The new algorithms are a synthesis of methods in deformable models, B-spline curve representation and control theory. This paper shows how such a facility can be used to turn parts of the body — for instance, hands and lips — into input devices. Rigid motion of a hand can be used as a 3D mouse with non-rigid gestures signalling a button press or the "lifting" of the mouse. Both rigid and non-rigid motions of lips can be tracked independently and used as inputs, for example to animate a computer-generated face.

INTRODUCTION

The advent of general-purpose workstations with integral video-camera and real-time framestore presents an opportunity for a low-cost approach to position, attitude and shape input. This can be achieved without special hardware by using software for real-time tracking of the live video signal, processed at full video-field rate, 50Hz in systems like the one reported here.

Real-time video tracking has been achieved for line-drawings [15] and polyhedral structures [19] and for simple, natural features such as road-edges [10]. 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 [3]. Curve trackers [17] have been demonstrated on modest workstations but slower than frame-rate.

[1]Department of Engineering Science, University of Oxford, Parks Rd, Oxford OX1 3PJ, UK.

Two new algorithms are presented in this paper which allow effective, agile tracking of curves in live video data, at 50Hz, on a modest workstation (SUN IPX plus Cohu video-camera and Datacell S2200 framestore) without additional hardware. The possibility of tracking curves within a Kalman filtering framework was raised by Szeliski and Terzopoulos [23]. Kalman filters comprise two steps: prediction and measurement assimilation. Prediction employs assumptions about object dynamics to extrapolate past motion from one video frame to the next. Assimilation blends measurements from a given frame with the latest available prediction. An excellent introduction to this powerful methodology is given by Gelb [13]. Our first algorithm applies such a filter to curves represented by B-splines, to track both rigid and non-rigid motions.

The second algorithm is a "system identification algorithm" based on ideas from adaptive control theory [2] and "maximum likelihood estimation" in statistics [22]. Previous approaches to learning shape variability have used statistical models to represent a family of possible shapes [14, 9] but statically. In contrast the learning method reported here is dynamic, using and modelling *temporal* image sequences. Example motions are tracked by a general-purpose tracker based on the assumption of default object dynamics. The tracked motion is used as a training set for the new algorithm which estimates the underlying dynamics of the training motion. The learned dynamics are then used in the tracker's predictor which then enjoys enhanced tracking capability for motions similar to those in the training set. The learning process can be repeated to bootstrap trackers of successively increasing performance.

The effectiveness of the algorithms in generating agile trackers which are resistant to distraction from background clutter is demonstrated in this paper and on the accompanying video. The final section of the paper illustrates two of the possible applications for graphical input: a 3D mouse driven by natural hand movements and a lip-tracker for use in automating animation.

PROBLEM FRAMEWORK AND NOTATION

The tracker is an estimator for a moving, piecewise-smooth image-plane curve:

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

Following the tracking work of others [21, 8], the curve representation is in terms of B-splines. Quadratic splines with the possibility of multiple knots for vertices are used here.

A given curve is parameterised as a B-spline

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

where $\mathbf{X} = (X_1, .., X_{N_c})^T$ and similarly for $\mathbf{Y}$ with $N_c = N$ for closed curves and $N_c = N + d$ for open ones (with appropriate

variations where multiple knots are used to vary curve continuity). The elements of $\mathbf{X}$ and $\mathbf{Y}$ are simply vectors of x and y coordinates respectively from the set of control points (X_m, Y_m) for the B-spline. The vector $\mathbf{B}(s)$ consists of blending coefficients defined by

$$\mathbf{B}(s) = (B_1(s), ..., B_{N_c}(s))^T \quad (1)$$

where each B_m is a B-spline basis function [11, 6] appropriate to the order of the curve and its set of knots.

Tracking

The tracking problem is now to estimate the motion of some curve — 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 those estimates should represent a curve that, at each time-step, matches the underlying curve as closely as possible. The tracker consists, in accordance with standard practice in temporal filtering [13, 4], of two parts: 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 some average shape or "template" [12] whose control points are given by $(\overline{\mathbf{X}}(t), \overline{\mathbf{Y}}(t))$, and which is generated by an interactive drawing tool, drawing over a single video frame.

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 moving shapes. However, particularly for complex shapes requiring many control points to describe them, this is known to lead to instability in tracking [7]. Furthermore, it is not necessary to allow so much freedom. A moving, outstretched hand, for instance, provided the fingers are not flexing, is an approximately rigid, planar shape. It is known that, under orthographic projection, the image of such a shape has only 6 degrees of freedom [25, 18], so provided perspective effects are not too strong, a good approximation to the curve shape as it changes over time can be obtained by specifying just 6 numbers. They form a vector denoted $\mathbf{Q}$ expressing the affine transformation that is applied to the template to get to the current shape. In this representation, the template itself is simply $\overline{\mathbf{Q}} = (0, 0, 1, 1, 0, 0)^T$. Transformations between $\mathbf{Q}$ and $(\mathbf{X}, \mathbf{Y})$ can be made by applying some matrices M, W (see appendix):

$$\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = W\mathbf{Q} \quad \text{and} \quad \mathbf{Q} = M \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix}. \quad (2)$$

Non-planar rigid shapes can also be treated in a similar way to planar ones except that then $\mathbf{Q}$ must be an 8-vector, related to the control points by an appropriately defined $2N_c \times 8$ W-matrix. Alternatively, freedom for rigid motion can be restricted, for example to allow only zoom and translation, and then $\mathbf{Q}$ becomes a 3-vector, with an accompanying $2N_c \times 3$ W-matrix. Nonrigid motion can also be handled in a similar way, by choosing a $\mathbf{Q}$ representation that represents both the rigid and non-rigid degrees of freedom of the shape. This is needed, for instance, to allow finger movements in hand tracking, or to follow lip movements. It is crucial therefore to allow exactly the right degrees of freedom for nonrigidity, neither too few resulting in excessive stiffness, nor too many leading to instability. Previous snake-based methods for handling non-rigid motion [17] allowed far too many degrees of freedom which leads to instability, quite unusable for real-time tracking. Further detail on handling of non-rigid motion is given later.

TRACKER FRAMEWORK

This section describes how the dynamics of the curve are modelled, how the measurement process is modelled, and how the two are brought together to make a curve tracker.

Our model of curve motion is a second order equation driven by noise, as used widely for modelling in control theory [1]. The choice of a second order model includes constant velocity motion, decay and damped oscillation. These motions can be present independently for each of the six degrees of freedom of rigid motion. In the case of oscillation, for instance, there may be 6 independent modes present, each with its own natural frequency and damping rate. In addition to the characteristic oscillation, the "stochastic" element of the model adds in uncertainty, as a smoothed random walk, and so allows a given model to represent a whole *class* of motions. A major attraction of this type of model is compatibility with Kalman filtering. A Kalman filter, consisting of a prediction and a measurement assimilation phase, makes use of just such a model as its predictor.

Second order dynamics are conveniently written in discrete time [1] using a "state vector" $\mathcal{X}_n$, defined in terms of shape $\mathbf{Q}$ *relative* to the template $\overline{\mathbf{Q}}$:

$$\mathcal{X}_n = \begin{pmatrix} \mathbf{Q}_{n-1} - \overline{\mathbf{Q}} \\ \mathbf{Q}_n - \overline{\mathbf{Q}}. \end{pmatrix} \quad (3)$$

Successive video frames are indexed $n = 1, 2, 3, ...$

Now the dynamics of the object are defined by the following difference equation:

$$\mathcal{X}_{n+1} = A\mathcal{X}_n + \begin{pmatrix} \mathbf{0} \\ \mathbf{w}_n \end{pmatrix}. \quad (4)$$

The matrix coefficient A is a 12×12 matrix, defining the deterministic part of the dynamics; its eigenvectors represent modes, and its eigenvalues give natural frequencies and damping constants [1]. This matrix appears again later, in the tracker, which mirrors equation (4) in its prediction component. At each time n, $\mathbf{w}_n$ is an independent, normally distributed, 6-vector of random variables, with *covariance* matrix C. It is this covariance matrix C that specifies the random part of the dynamical model — the degree of randomness injected into each of the affine degrees of freedom.

Measurement of video features

Given an estimated curve $\hat{\mathbf{r}}(s)$, expressed in terms of $\hat{\mathbf{Q}}$, the measurement process at time t consists of casting rays simultaneously along several normals $\mathbf{n}(s)$ to the estimated curve. The normal vectors $\mathbf{n}(s)$ at each measurement point on the curve are calculated in the standard manner for B-spline curves [11], by differentiating to find the tangent vector and then rotating through 90°. Grey-level values in the video-input framestore are read to measure the (signed) distance $\nu(s)$ of a particular grey-level feature along the curve normal. Typically the feature is a high contrast edge, though other features can also be used [17]. In our system, the highest contrast feature along the normal is chosen within a certain window, typically of width ± 40 pixels, on either side of the estimated curve. Each position measurement is assumed to have an associated error distribution, with a root-mean-square value of σ pixels. The individual measurements made on each normal are then aggregated to form a combined "virtual" measurement $\mathbf{Z}$ in the $\mathbf{Q}$-space:

$$\mathbf{Z} = \sum_{m=0}^{m=NN_\nu} \nu(m/N_\nu)\mathbf{H}(m/N_\nu)^T$$

where N_ν is the number of measurements made (at equal intervals) along each B-spline span, N is the number of spans as before, and

$$\mathbf{H}(s) = (\mathbf{n}(s) \otimes \mathbf{B}(s))^T W,$$

where $\mathbf{n}(s)$ is the curve normal vector at a particular point of measurement and $\mathbf{B}(s)$ is the vector of B-spline blending function defined in (1). The projection matrix W that was defined in (2) has the effect here of referring 2D measurements into $\mathbf{Q}$-space, so that they can be applied, in the assimilation phase of the tracker, to update the estimated $\mathbf{Q}$-vector. (Note: the operation $\otimes$ denotes the "Kronecker product"[1].) The $\mathbf{n}(s)$ weighting in the virtual measurement $\mathbf{Z}$ has the effect (and this can be shown rigorously) of "pulling" the tracked curve *along* the normal *only*. Such a weighting along the normal reflects the fact that any displacement *tangential* to a smooth curve is unobservable, the well-known "aperture problem" of visual motion [16].

Tracking algorithm

The tracking algorithm, a standard "steady state Kalman filter" [13], consists of iterating the following equation:

$$\hat{\mathcal{X}}_{n+1} = A\hat{\mathcal{X}}_n + K \begin{pmatrix} \mathbf{Z}_{n+1} \\ \mathbf{0} \end{pmatrix},$$

The somewhat non-standard aspect of the algorithm is the 12×12 "Kalman gain" matrix K, defined in the appendix, which has to be designed to match the state-space measurement vector $\mathbf{Z}$. The gain K depends on the system's matrix coefficients A, C and on the reliability and density of measurements along the tracked contour, and is constant over time in the "steady state" case considered here. Allowing time-varying gain is useful to take account of intermittent failures in the measurement process [7], but that is outside the scope of this paper.

Default tracker

The tracking algorithm described above requires system matrices A, C to be specified. These are not known in advance and the whole purpose of the framework being set up here is to estimate them. However, for the estimation process, a stream of data from a tracked object is needed — for which a tracker is required! Clearly, estimation must be a bootstrap process, beginning with a default tracker based on default values of A, C.

A natural default setting for the deterministic part of the dynamics is to assume constant velocity for all degrees of freedom, realised by setting

$$A = \begin{pmatrix} 0 & I \\ -I & 2I \end{pmatrix}$$

where I is the 6×6 identity matrix. A natural default can also be defined for the random part of the dynamics. It is reasonable, as a default, to assume that the magnitude of the random component is uniform over the image plane, with no particular directional bias. For that case, it is known [7] that

$$C = cM\mathcal{H}^{-1}M^T$$

where the "metric" matrix $\mathcal{H}$ is defined in the appendix. This leaves just one constant — c — to be chosen to fix C.

Now the default tracker has been specified except for the values of certain constants. Typically the number of measurements per B-spline span is set to $N_\nu = 3$, a value which allows our tracker to run at video-field rate, on a SUN IPX workstation, with 20 or more control points.

The system and measurement constants c and σ also remain to be chosen. The following workable values that were used for the hand-tracking experiments reported later:

$$\sigma = 10.0 \text{ pixels} \quad \text{and} \quad c = 1.0 \text{ pixels}^2$$

— further details on choice of constants are given elsewhere [7].

LEARNING MOTION AND TRAINING TRACKERS

Training consists of using the default tracker to gather a sequence of m images at the image sampling frequency of 50Hz. The result of tracking is then a sequence of m $\mathbf{Q}$-vectors $\mathbf{Q}_1, ..., \mathbf{Q}_m$, which are the data from which system parameters A, C, are estimated. Once A, C are obtained incorporating them in a tracker in the way that was just described.

It follows from definitions in equations (3) and (4) that the 12×12 matrix A has the form:

$$A = \begin{pmatrix} 0 & I \\ A_0 & A_1 \end{pmatrix}, \tag{5}$$

where A_0, A_1 are 6×6 submatrices. Together with C, they are estimated as the solution of a least-squares algorithm. Unfortunately, details of the derivation and principles of the algorithm cannot be given here for lack of space. However it is of a type that is somewhat standard in stochastic system identification [2]. It corresponds to maximising the probabilistic "likelihood" [22] of the data with respect to the settings of A, C.

Practically, the important point is the algorithm itself, which is straightforward. It should be borne in mind that this algorithm is preparatory to filtering and is therefore done offline. Efficiency is not therefore a major consideration. The algorithm has three steps.

Step 1: matrix moments A set of matrix moments

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

is computed from the time-sequence of $\mathbf{Q}$-vectors that serve as the data for learning the parameters A, C.

Step 2: calculate A The A-matrix is calculated simply by solving a set of simultaneous equations to obtain the submatrices A_0, A_1:

$$S_{20} - A_0 S_{00} - A_1 S_{10} = 0 \tag{6}$$
$$S_{21} - A_0 S_{01} - A_1 S_{11} = 0. \tag{7}$$

Now A_0, A_1 form the full matrix A as in equation (5).

Step 3: calculate C Using the value of A just computed, C can be estimated directly:

$$C = \frac{1}{m-2} \sum_{n=1}^{m-2} (\mathbf{Q}_{n+2} - A_0\mathbf{Q}_n - A_1\mathbf{Q}_{n+1}) \cdot (\mathbf{Q}_{n+2} - A_0\mathbf{Q}_n - A_1\mathbf{Q}_{n+1})^T.$$

[1] The Kronecker product [5] $A \otimes B$ 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.

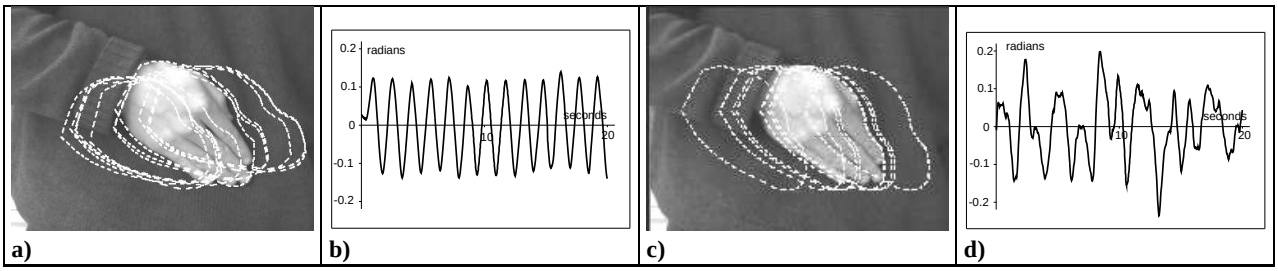

Figure 1: **Learning horizontal oscillation.** *a) Training sequence of a tracked hand captured at 50Hz (full video field-rate) in oscillatory motion. The initial frame of the hand is shown with a subsequence of the positions of the tracked curve overlaid. The centroid's horizontal position is displayed 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).*

LEARNING TRANSLATION

Model parameters are inferred from data by the statistical learning algorithm already described. The resulting tracker is specifically sensitive to a particular motion, in this case translation. This time the learnt motion will also be resynthesised and compared with the original. The horizontal motion training set, shown as a graph in figure 1b, is then used in the estimation procedure of the previous section, to obtain parameters A, C. The resulting A-matrix has several modes as in the zoom example. The mode which is most significant in the sense that it decays most slowly, represents a damped oscillation whose period is about 1.7 seconds. This is close to the apparent periodicity of the data in which there are between 12 and 13 cycles over 20 seconds, a period of about 1.6 seconds.

As a further check, taking the learned model, we can simulate it by generating pseudo-random gaussian noise and using it to "energise" the model whose coefficients are the learned values of A, C. This is illustrated in figure 1c,d. It is clear that the simulated process is similar in amplitude and natural frequency to the data set. However it is more "random" — it does not capture the phase-coherence of the training set. Such behaviour is typical of this algorithm and should be regarded as a feature rather than a bug. It indicates that the algorithm has *generalised* the example motion, capturing its broad characteristics over shorter time-scales without attempting to replicate the entire signal. A stronger model that captured the entire signal would be too strong — the model would compete, in the tracker, with the measurement process and overwhelm it. The remaining influence of the video measurements would then be too weak for effective tracking. (Note: this argument can be put rigorously, but space does not permit.)

Tracking

The following experiments demonstrate the power of incorporating learned motion into a curve tracker. This time, a training set of *vertical* rigid motion is generated and used to learn motion coefficients A, C, much as in the previous example of learned horizontal motion. The trained tracker, incorporating the learned motion is then to be tested against the un-trained, default tracker. It will be clear that the tracker has been trained to track rapid oscillation.

The test sequences are generated consisting 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 oscillatory motion of gradually increasing frequency — a "chirp" — and the other of regular oscillation against irrelevant background features — "clutter". Results of the tests are shown in figures 2 and 3. The increased agility and robustness of the trained tracker is also clear in the accompanying video.

NONRIGID MOTION

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

Key-frames

The $\mathbf{Q}$-space is extended by redefining the transformation matrices M, W from equation (2). The W matrix (see appendix for details) must be given additional columns each reflecting an additional degree of freedom for non-rigid motion. This then increases the dimension of the $\mathbf{Q}$ vector from 6 to some larger number. The extra columns of W are derived from "key-frames", typical non-rigid deformations, on which the tracked contour is positioned interactively, as in figure 4. Each of the additional components of the $\mathbf{Q}$-vector then represents the proportion of each of the key-frames in the mix. (Note that the template and key-frames do not need to be mutually orthogonal, merely linearly independent). 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.

Lips

The key-frames above generate a $\mathbf{Q}$-space that can be used in a default tracker capable of tracking slow speech and sufficient to gather data for training. As a demonstration of the effect of training, trackers were trained for the lip-motions that accompany the sounds "Pah" and "Ooh", and tested on the sound "Pah". The selectivity of the resulting tracker is shown in figure 5. It is clear from these results that the training effect for individual sounds is strong. This suggests that training for typical lip-movements that accompany speech should have a strong effect on tracking speaking lips and this is explored in the next section. The selective training effect could potentially be exploited in its own right for simple lip-reading (e.g. of commands or key-words) an idea that has been explored by others [20] using rather different mechanisms.

APPLICATIONS

There are potentially many applications for real-time position, attitude and shape input using the new algorithms for learning and tracking. Two are explored here: the use of a hand as a 3D mouse

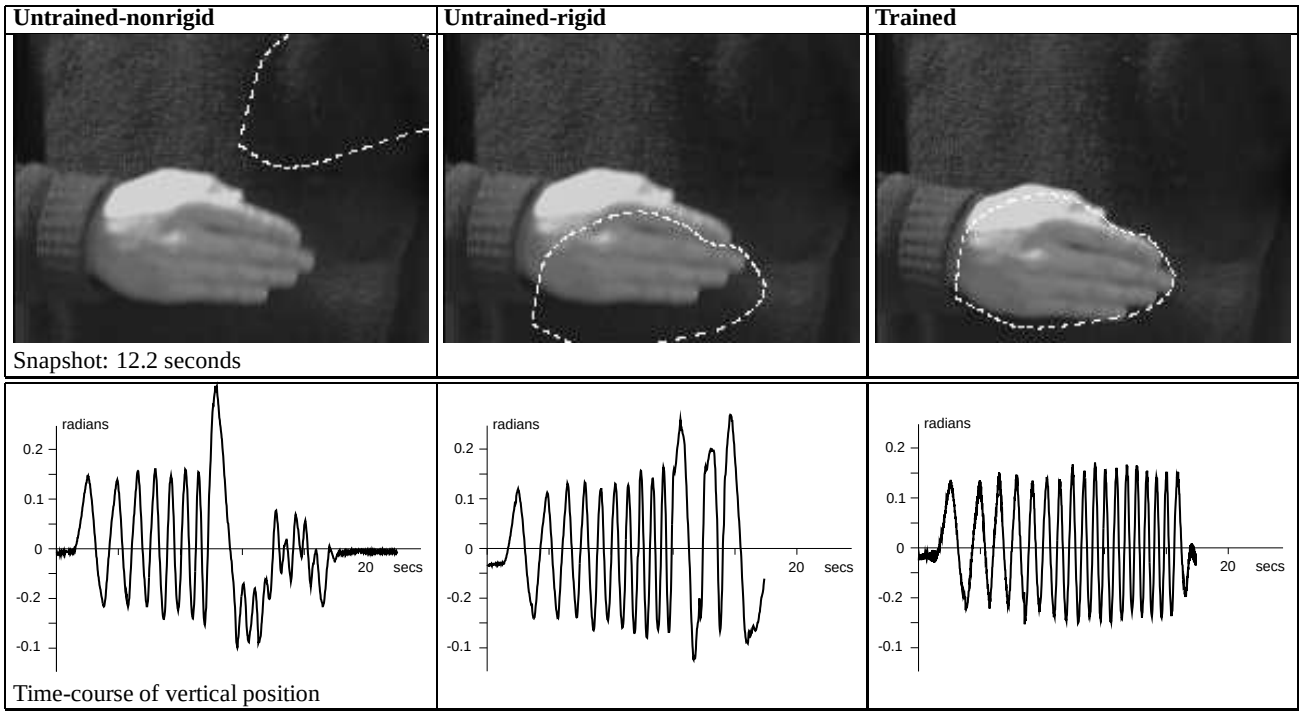

Figure 2: **Trained tracker for rigid motion, tested with rapid oscillations**. *A "chirp" test motion, consisting of vertical oscillations of progressively increasing frequency, is tracked by each of three trackers: untrained-nonrigid, untrained-rigid and trained. For the untrained trackers, lock is lost after 12.2 seconds whereas the trained tracker is still tracking. Graphs of vertical position against time clearly shows the lost lock at about 12 seconds for the untrained trackers. The trained tracker maintains lock throughout the time-sequence.*

and the tracking of lips for control of animation. Of course successful hard-ware realisations of a 3D mouse or "bat" have already been developed [26]. The advantages of the system proposed here is that on the new generation of workstations with built-in cameras it involves *no* additional hardware, and that it is software reconfigurable for example by adding finger-gestures to indicate button presses and perhaps even grasping manoeuvres of the sort used in a "dataglove". Facial animation in real time has been achieved previously using reflective markers [27] but that supplies measurements at distinct points only. The advantage of a system like the one demonstrated here is that it can measure, in real time, the motion of entire curves.

3D mouse

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 6, with rigid motion picked up to control 3D position and attitude, and nonrigid motion signalling 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 and this is illustrated on the accompanying video.

Lips

The feasibility of tracking lip movements frontally, when lip high-lighter is worn, was demonstrated by Kass et al [17]. Our system can do this at video-rate. This paradigm can be extended by using high-lighter on a number of facial features, as Terzopoulos and Waters [24] 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.

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. Learning and selectivity with single sounds were demonstrated earlier. Here, complexity is increased by training on connected speech. Two-stage training was used. In the first 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. It is then 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 components of lip motion are extracted from the tracked motion. The larger of these corresponds approximately to the degree to which the lips are parted. This component is plotted both for the default tracker and the trained one in figure 7. It is clear from the figure that the trained filter is considerably more agile. In preliminary demonstrations in which the signal is used to animate a head (frontal view), the default filter is able to follow only very slow speech, whereas the trained filter successfully follows speech delivered at a normal speed. The increased agility and robustness of the trained tracker is made clear in the accompanying video.

CONCLUSIONS

New algorithms have been described for live tracking of moving objects from video. The first algorithm is a tracker based on the control-theoretic "Kalman filter". It allows particular dynamics, modelled by a stochastic differential equation, to be used predic-

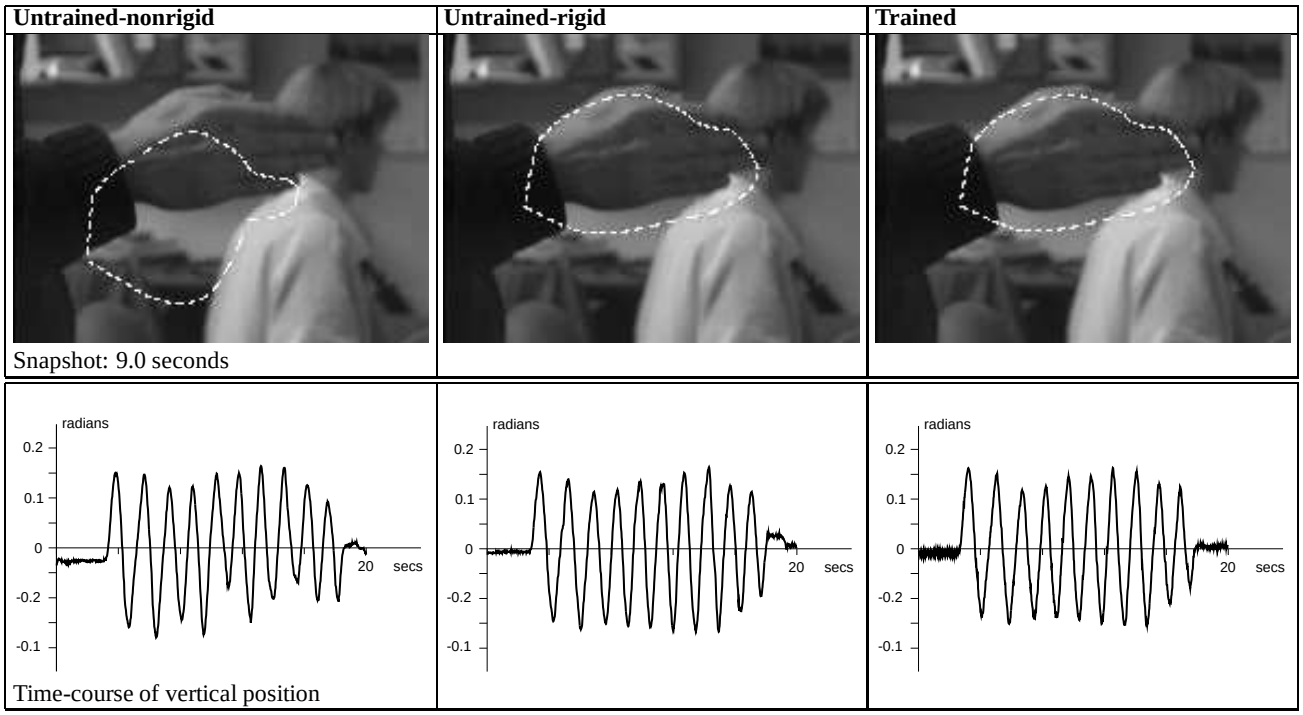

Figure 3: **Trained tracker for oscillatory rigid motion, tested against clutter**. *The two untrained trackers and the trained tracker are compared here using a "clutter" test-sequence. After 9.0 seconds, the nonrigid-untrained tracker is distracted by background clutter and loses lock, but the rigid-untrained and the trained trackers continue to track successfully.*

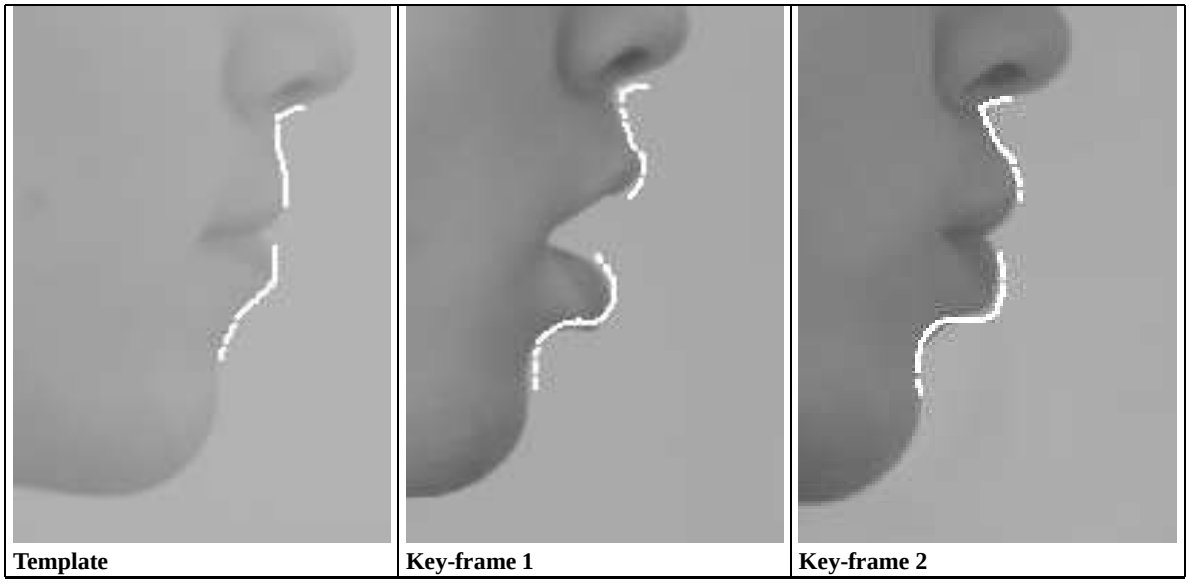

Figure 4: **Keyframes.** *A weighted sum of key-frames is added to the* **Q** *vector for use in a default tracker for non-rigid motion.*

tively in the tracker. The default tracker, for instance, 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 **Q**-space. This is what allows stable tracking, for which free parameters must be limited, to be combined with the apparently conflicting requirement of the large number of control points needed for accurate shape representation.

The second algorithm is the most original contribution. It is 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.

Finally, the advent of workstations with integral cameras and

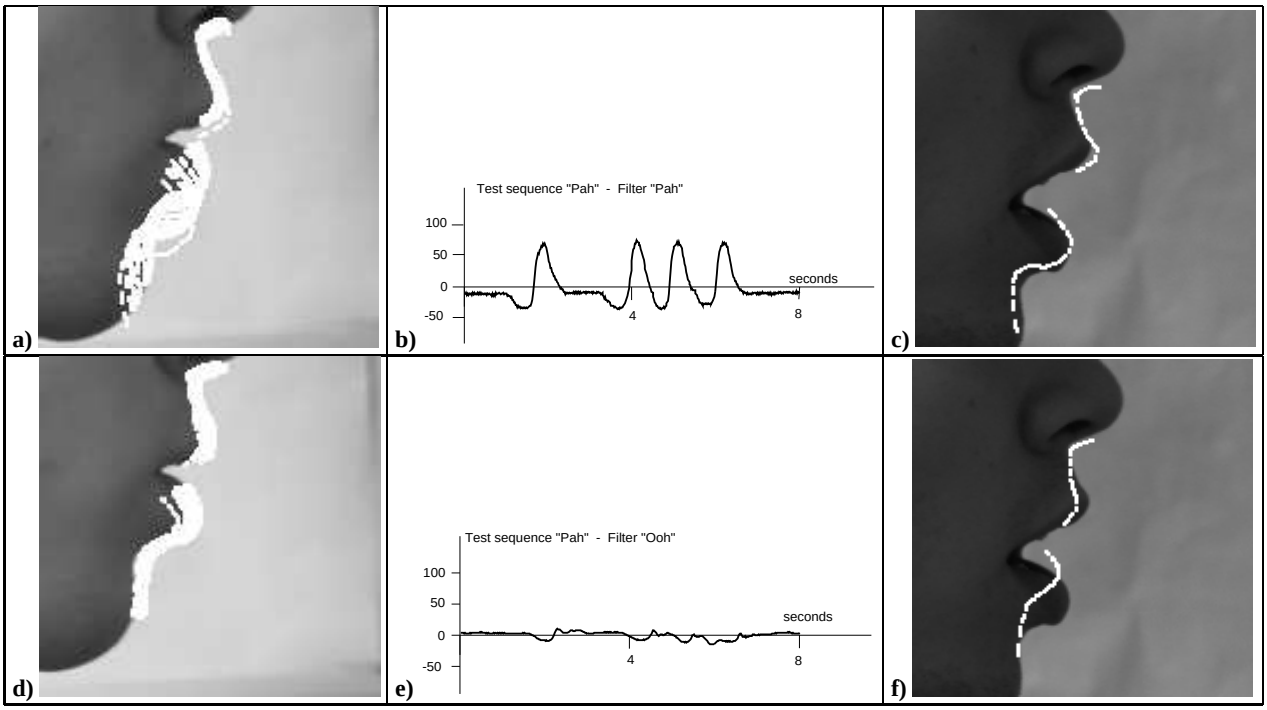

Figure 5: **Filter selectivity for sounds**. *A test sequence in which the sound "pah" is repeated is tracked by a filter trained on the sound "pah". a) shows the area swept out by successive positions of the tracked contour. Note that the filter successfully tracks the opening/shutting deformation of the mouth. The motion signal (b) is plotted in terms of an appropriate scalar component of estimated motion; it shows pairs of troughs (shutting of the mouth) and peaks (opening), one for each "pah". For instance, approximately 4.1s after the start of the signal the tracked contour (c) is clearly still locked. Corresponding pictures for a tracker trained on the sound "Ooh", fed with the same "Pah" test-sequence, are shown in d),e),f). It is clear (d) that only lateral translation is tracked — the nonrigid deformation containing the speech information is lost, as expected. There is minimal opening/shutting response (e). After 4.1s (f) lock has been lost.*

framestores 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 and perhaps even low-bandwidth teleconferencing and the design of computer aids for the handicapped.

Acknowledgements

We are grateful for the use of elegant software constructed by Rupert Curwen, Nicola Ferrier, Simon Rowe, Henrik Klagges and for discussions with Roger Brockett, Yael Moses, David Reynard, Brian Ripley, Richard Szeliski, Andrew Zisserman. We acknowledge the support of the SERC, the EC Esprit programme (SECOND) and the Newton Institute, Cambridge.

REFERENCES

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

[2] K. J. Astrom and B. Wittenmark. *Adaptive control.* Addison Wesley, 1989.

[3] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Trans. Pattern Analysis and Machine Intell.*, in press, 1993.

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

[5] Stephen Barnett. *Matrices: Methods and Applications*. Oxford University Press, 1990.

[6] 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.

[7] 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.

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

[9] 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.

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

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

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

[13] Arthur Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974.

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

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

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

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

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

[19] 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.

[20] K. Mase and A. Pentland. Automatic lip-reading by optical flow analysis. Media Lab Report 117, MIT, 1991.

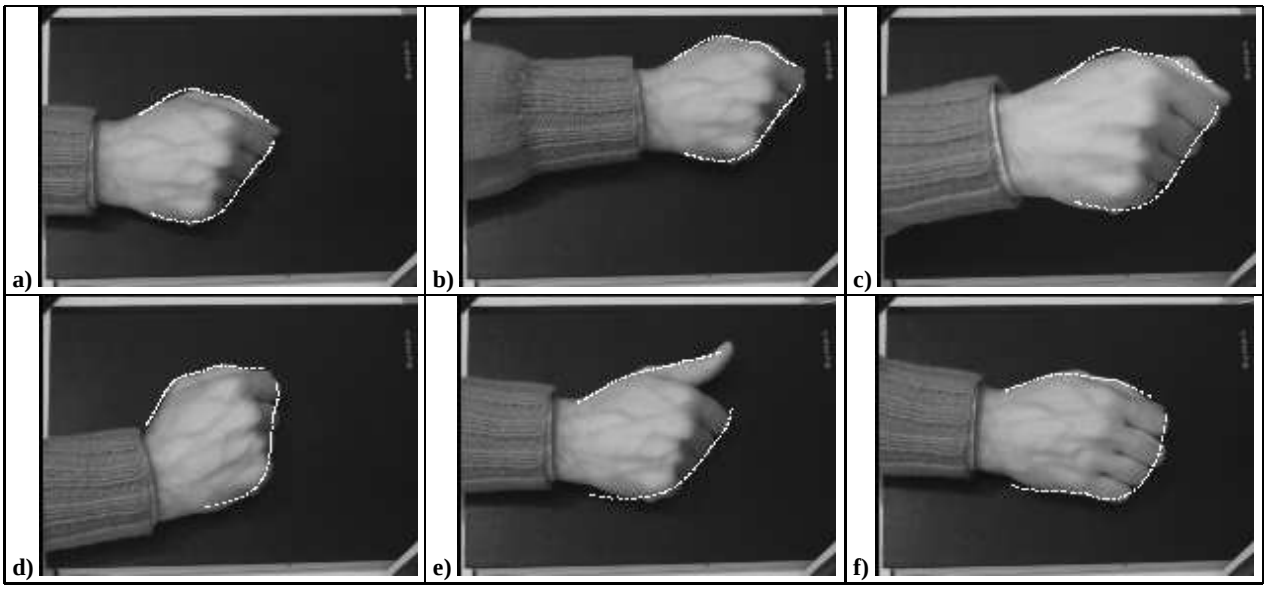

Figure 6: **The unadorned hand as a 3D mouse.** *A hand in its home position (a) can move on the xy-plane of the table (b) to act as a regular mouse but can also rise in the z direction (c) and the zooming effect is picked and used to compute z. Rotation can also be tracked (d). Note that measured affine distortions in the image plane are straightforwardly translated back into 3D displacements and rotations, as the demonstration of hand-controlled 3D motion on the accompanying video shows. 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.*

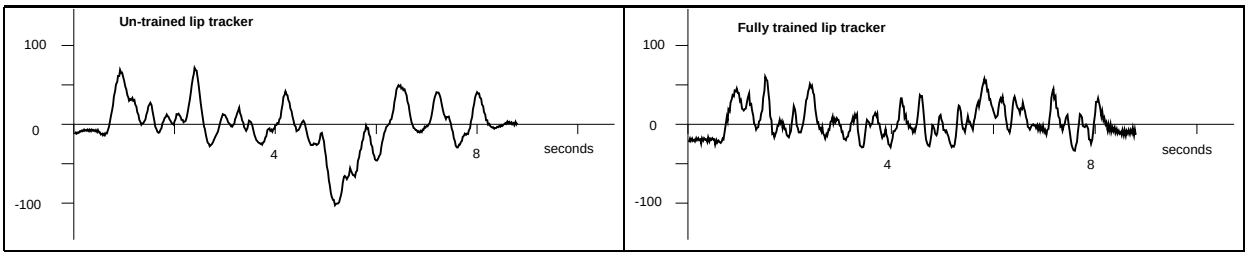

Figure 7: **Trained lip tracker.** *Training a tracker for side-on viewing of speaking lips greatly enhances tracking performance. The graphs show plots from the default and trained filters respectively of one deformation component of the lips during connected speech. The component corresponds to the degree to which the mouth is open — the space of deformations spanned by the first two templates in figure 4. Note the considerable loss of detail in the default filter, and occasional overshoots, compared with the 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.".)*

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

[22] A. Papoulis. *Probability and Statistics.* Prentice-Hall, 1990.

[23] 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.

[24] D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis and animation. *J. Visualization and COmputer Animation*, 11(2), 1990.

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

[26] C. Ware and D.R. Jessome. Using the bat: a six-dimensional mouse for object placement. In *Proc. Graphics Interface*, pages 119–124, 1988.

[27] L. Williams. Performance-driven facial animation. In *Proc. Siggraph*, pages 235–242. ACM, 1990.

A. Matrices M, W for rigid planar shape

Matrices M, W convert B-spline control points $(\mathbf{X}, \mathbf{Y})$ to and from the affine 6-vector $\mathbf{Q}$ representation, and are defined to be:

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

and

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

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

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

and

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

See also [7].

B. Calculating the gain

Filter gain K is calculated by iterating to convergence the "discrete Ricatti equation" [13] for this problem, which can be shown to be:

$$P_{n+1} = \left((A P_n A^T + C')^{-1} + \frac{1}{\sigma^2} J' \right)^{-1}$$

$$\text{where } C' = \begin{pmatrix} 0 & 0 \\ 0 & C \end{pmatrix} \text{ and } J' = \begin{pmatrix} J & 0 \\ 0 & 0 \end{pmatrix}$$

$$\text{and } J = \sum_{m=0}^{m=NN_\nu} \mathbf{H}(m/N_\nu)^T \mathbf{H}(m/N_\nu)$$

to obtain P_∞. Any reasonable initial condition, such as $P_0 = 0$ will do, and the equation is bound to converge provided the learned dynamics represented by the matrix A are stable. Finally the steady state Kalman Gain for the measurement $\mathbf{Z}$ can be shown to be $K = (1/\sigma^2) P_\infty$.