A Sparse Probabilistic Learning Algorithm for Real-Time Tracking

Oliver Williams
Department of Engineering
University of Cambridge
omcw2@cam.ac.uk

Andrew Blake
Microsoft Research Ltd.
Cambridge, UK

Roberto Cipolla
Department of Engineering
University of Cambridge

Abstract

This paper addresses the problem of applying powerful pattern recognition algorithms based on kernels to efficient visual tracking. Recently Avidan [1] has shown that object recognizers using kernel-SVMs can be elegantly adapted to localization by means of spatial perturbation of the SVM, using optic flow. Whereas Avidan's SVM applies to each frame of a video independently of other frames, the benefits of temporal fusion of data are well known. This issue is addressed here by using a fully probabilistic 'Relevance Vector Machine' (RVM) to generate observations with Gaussian distributions that can be fused over time. To improve performance further, rather than adapting a recognizer, we build a localizer directly using the regression form of the RVM. A classification SVM is used in tandem, for object verification, and this provides the capability of automatic initialization and recovery.

The approach is demonstrated in real-time face and vehicle tracking systems. The 'sparsity' of the RVMs means that only a fraction of CPU time is required to track at frame rate. Tracker output is demonstrated in a camera management task in which zoom and pan are controlled in response to speaker/vehicle position and orientation, over an extended period. The advantages of temporal fusion in this system are demonstrated.

1. Introduction

Systems for tracking moving objects may be feature-based or model-based. Feature-based tracking can rely on persistence of image curves [10, 4] or image appearance [8]. Likewise model-based tracking can use curves, either in 3D [12] or in 2D [20], or appearance [2]. Furthermore, statistically-based object recognizers using support vector machines (SVMs) [13, 15] and boosting [18] have now become fast enough to run at around video rate, despite having to search each video frame. Most recently, the distinction between localization and tracking has been broken down still further by the introduction of 'support vector tracking' [1], and it is this paradigm that we seek to extend here.

1.1 The support vector tracker

The SVM is a powerful algorithm for binary classification [5], in which the class of a test image-patch $\mathbf{q}$ is determined according to the sign of a classification function

$$f(\mathbf{q}) = \sum_{i=1}^{N} y_i \alpha_i k(\mathbf{q}, \mathbf{x}_i) + b \tag{1}$$

in which $\mathbf{x}_i$, $i \in [1, N]$ are vectors from a training set with labels $y_i \in \{-1, +1\}$, α_i and b are real-valued coefficients which must be determined by a learning algorithm, and $k(\cdot, \cdot)$ is a 'kernel' function which computes a generalised inner product. A great attraction of the SVM is the 'sparsity' conferred by the SVM training algorithm which typically sets most of the α_i to zero, leaving just a few active terms in the sum (1), involving a subset of the $\mathbf{x}_i$ which are known as 'support vectors'.

A classifier trained to recognize a particular class of object, can be used for tracking such an object by applying the classifier on each neighbourhood in a tessellation over some configuration space such as the Euclidean similarities (image translation, rotation and zoom). Such a search is labour intensive and Avidan's [1] support vector tracker seeks to mitigate this (in the space of translations) by perturbing the classification function f in (1) with respect to image translation. Given an object $\mathbf{q}$, perturbation analysis allows the translation $T_{\mathbf{u}}$, by a vector $\mathbf{u}$, to be found such that the value $f(T_{\mathbf{u}}\,\mathbf{q})$ of the classification function is maximized. The perturbed classification function, Avidan shows, is expressed in terms of image gradient as

$$f(T_{\mathbf{u}}\,\mathbf{q}) = f(\mathbf{q} + \mathbf{u} \cdot \nabla \mathbf{q}). \tag{2}$$

Using (2) to compute approximately the displacement $\mathbf{u}$ that maximizes the classification function f can save computation by reducing the density of tessellation required to achieve a given degree of tracking precision. In principle

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

this perturbation can be extended beyond the space of translations, for example to Euclidean similarities or affine transformations, in which case the gradient operator ∇ in (2) is extended to the full set of affine flow operators.

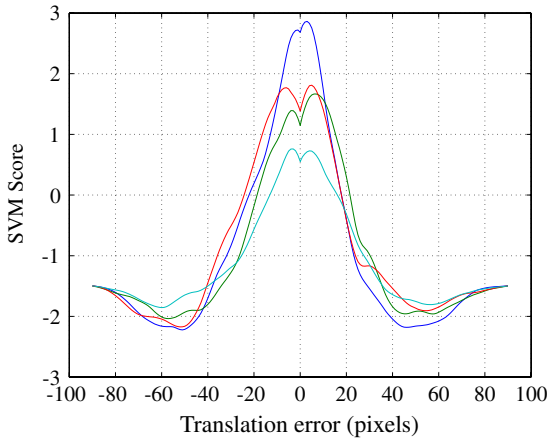

Figure 1. Classification function of an SVM trained for recognition. *A maximum is present at the true position, though the signal is noisy, producing additional local maxima.*

1.2 Probabilistic support vector tracking

Avidan's support vector tracker applies to each frame of a video independently of other frames. However, the benefits of temporal fusion [7] of data in visual tracking are well known [4]: improved efficiency and enhanced ability to deal with clutter in the background. Temporal fusion can be accomplished in an effective and principled way in a probabilistic setting. Therefore we seek to place a probabilistic construction on classifier output. One approach is to interpret SVM classifier output in probabilistic terms [11, 14]. On the other hand the 'Relevance Vector Machine' (RVM) [17] is a sparse classifier that is directly probabilistic. Moreover, it can be used for regression, as opposed to classification. That fits exactly the problem addressed here, in which regression onto the displacement $\mathbf{u}$ is required. This means that instead of training a machine to recognize known objects verses non-objects, a machine is trained to estimate the displacement of known objects alone. (Regression with SVMs is possible but relatively awkward, and not known to be susceptible to a probabilistic interpretation.) Therefore the system developed and demonstrated in this paper has the following distinctive properties.

- Fully probabilistic regression for displacement, using RVMs
- Displacement is modelled as a Euclidean similarity transformation
- Observations of displacement are fused temporally with motion prediction
- Initialization and recovery are handled by an SVM for object recognition running in tandem
- Tracking is efficient — better than real-time (i.e. leaving processor cycles free for other processes) — and this is demonstrated with a real time camera management system for teleconferencing.

2. Motion classification

Support vector tracking [1] uses an SVM optimized for recognition but for tracking it is motion, rather than classification, that is of primary interest. The SVM, with its good generalizing ability, would give more satisfactory results if it were trained to discriminate motion rather than object/non-object and, as a possibly easier generalization, the resulting machines would have fewer support vectors, making them more efficient.

2.1. Inferring state

The tracker follows a 2D image region containing an object. Intensity changes within this region are assumed to be due to motion only. Therefore, a four-dimensional state vector is used describing a Euclidean similarity transformation:

$$\mathbf{X} = [u, v, s, \theta] \in \mathrm{GE}(2) \tag{3}$$

An observed image at time t, $\mathbf{I}_t$, is a vectorized pixel array that is some unknown function of the present state: $\mathbf{I}_t = H(\mathbf{X}_t)$. However, treating the state as the dependent variable, $\mathbf{X}_t = H^{-1}(\mathbf{I}_t)$, it should be possible to find a regression inferring a state estimate, $\tilde{\mathbf{X}}$ from a given image. Several SVM are now required to cover all degrees of freedom of allowed motion. The obvious configuration is for one SVM to be assigned to each state space dimension. For example, for x translation, the training set includes a negative subset: images of the object translated right by 20 pixels, and a positive subset: images translated left 20 pixels. When a vector is tested, a score of +1/-1 would indicate that our test region should move 20 pixels right/left to realign with the true location. Translation of magnitude less than 20 pixels should yield a score between ± 1, as Figure 2 shows. The same holds for vertical translation, rotation and zoom. 20 pixels was chosen as a compromise: more than this gives the tracker a larger capture range, but the marginal response is less consistent; less than this gives a small capture range leading to a less robust tracker.

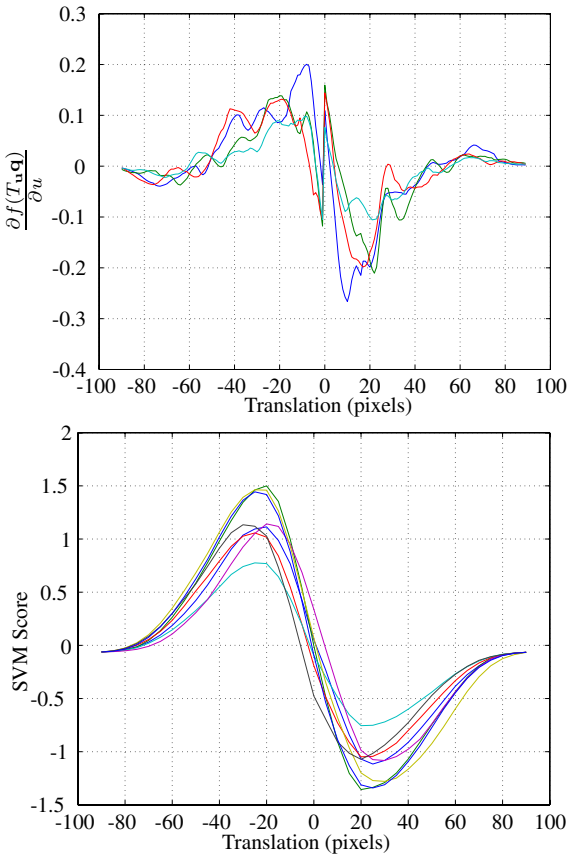

Figure 2. Advantages of an SVM trained for displacement. *(top) Avidan's displacement estimation: the derivative, with respect to translation, of the classification function for the SVM of figure 1, shown for several test examples. (bottom) Classification function for an SVM trained specifically for object displacement (as opposed to object identification). Note the superior linearity of response in the displacement-trained case.*

2.2. Linear Regression

Given a vector of SVM scores, we want to infer the state changes that created them. As the marginal response between the training sets is almost linear, it seems reasonable to fit a straight line to this data. State estimates are therefore made in two stages. The test region is passed to the SVMs which give a vector of scores, this is then interpreted via a multi-regressive function to yield the estimated change in state, $\delta\mathbf{X}$:

$$\delta\mathbf{X} = A\mathbf{f}(\mathbf{I}) + \mathbf{b} \qquad (4)$$
$$\mathbf{b} \in \mathbb{R}^4 \quad A \in \mathbb{R}^{4\times 4}$$

where $\mathbf{f}(\mathbf{I})$ is a vector of SVM scores like (1) and $\mathbf{I}$ is the vectorized image patch given by the present estimate.

2.3. Tracking with motion SVMs

Four SVMs are trained to classify the sign of each of the four dimensions of Euclidean similarity space. Each SVM requires training data with wide variations in the corresponding dimension. In addition, there must also be some variation in the other three dimensions in order to learn the desired invariance to those dimensions.

Residual cross-talk between SVMs is attenuated by the A matrix which is learned, together with $\mathbf{b}$, by conventional regression, after the SVMs have been fixed. For this, a fresh training set is used, chosen uniformly at random from a hypercubic domain in the 4-dimensional state-space. In addition, the regression delivers a covariance matrix on $\delta\mathbf{X}$ so that $\delta\mathbf{X}$ can be regarded as a random variable – observations capable of statistical fusion.

Object localization experiments were performed with the four SVMs, using the $\delta\mathbf{X}$ measure. Details of results are omitted here, but our main conclusions are given as motivation for developing a more powerful approach later. Frame-to-frame localization was frail with the 4-SVM approach and the reasons seem to be as follows. Each SVM is trained on data lying at the positive and negative extremes of variation of the relevant state space dimension. Probability distributions obtained from regression underestimate uncertainty: they reflect only the deviation from linearity of the SVM, rather than the variability found over a fully representative set of training examples.

3. Sparse probabilistic learning

The two-stage inference of SVM classification followed by linear regression is effective only up to a point, and is inelegant. It seems that some form of single stage regression might be more powerful, both for dealing with the range of variation of test examples, and for correctly modelling statistical variability.

A single-stage, probabilistic, learning paradigm is required for regression. It is possible to perform regression with an SVM; however, an approximation must be made (ϵ-SVM [16]) in order to retain a sparse solution and, as mentioned above (sec. 1.2), there have been some attempts at making this probabilistic. There is however a clear Bayesian formulation addressing all these issues.

3.1. The relevance vector machine

Tipping [17] proposed the *relevance vector machine* or RVM to recast the main ideas behind SVMs in a Bayesian

context, and using mechanisms similar to Gaussian processes [19]. The results have been shown to be as accurate and sparse as SVMs yet fit naturally into a regression framework and yield full probability distributions as their output. A brief review of Tipping's paper is presented here for those unfamiliar with the work.

An RVM is trained with a training set consisting of a vector of dependent variables, $\mathbf{Y} \in \mathbb{R}^N$ and the associated independent variable vectors $\mathbf{x}_i \; i = 1 \ldots N$. Assuming that the measured dependent variables in the training set, $\mathbf{Y}$ are corrupted by Gaussian noise with variance σ^2 the PDF can be written:

$$p(\mathbf{Y}|\boldsymbol{\alpha}, \mathbf{x}_i, \sigma^2) = (2\pi\sigma)^{-\frac{N}{2}} \exp\left\{-\frac{\|\mathbf{Y} - K\boldsymbol{\alpha}\|^2}{2\sigma^2}\right\}. \quad (5)$$

Like the SVM, the RVM is a kernel method and as such replaces instances of dot products between input vectors with the value of a kernel function [6]. K is a 'design matrix' containing the inter-training set kernel values, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Treating $\boldsymbol{\alpha}$ as a random variable, (5) is a likelihood function which could be maximized to find $\boldsymbol{\alpha}$. However, in order to emulate the sparsity properties of SVMs, some of the α_i can be set to zero, allowing corresponding training vectors to be discarded. This is expressed in a Bayesian context via a Gaussian prior over $\boldsymbol{\alpha}$, pulling it towards zero:

$$\boldsymbol{\alpha} \sim \mathcal{N}(0, \Sigma_\alpha) \quad (6)$$

with hyperparameters θ_i such that $\Sigma_\alpha = \mathrm{diag}(\theta_1, \ldots, \theta_N)$ Now the posterior PDF for $\boldsymbol{\alpha}$ is:

$$p(\boldsymbol{\alpha}|\mathbf{Y}, \mathbf{x}_i, \sigma^2, \Sigma_\alpha) \propto \mathcal{N}(\hat{\boldsymbol{\alpha}}, \hat{\Sigma}) \quad (7)$$

where

$$\hat{\boldsymbol{\alpha}} = \frac{1}{\sigma^2}\hat{\Sigma}K^{\mathrm{T}}\mathbf{Y} \quad \text{and} \quad \hat{\Sigma}^{-1} = \Sigma_\alpha^{-1} + \frac{1}{\sigma^2}K^{\mathrm{T}}K$$

and the maximum *a posteriori* estimate, $\hat{\boldsymbol{\alpha}}$ can be expressed as a function of noise variance, σ^2 and the variances over the prior, θ_i which are unknown.

The best values for these hyperparameters are determined from the *evidence*,

$$\begin{aligned} p(\mathbf{Y}|\sigma^2, \Sigma_\alpha) &= \int_{\mathbb{R}^N} \exp\left\{-\tfrac{1}{2}(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}})^{\mathrm{T}}\hat{\Sigma}^{-1}(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}})\right\}\mathbf{d}\boldsymbol{\alpha} \\ &= (2\pi)^{-N/2}|S|^{-1/2}\exp\left\{-\tfrac{1}{2}\mathbf{Y}^{\mathrm{T}}S^{-1}\mathbf{Y}\right\} \quad (8) \\ &\text{where } S = \sigma^2 I + K\Sigma_\alpha K^{\mathrm{T}} \end{aligned}$$

by optimization with respect to θ_i and σ^2 (type-II maximum likelihood). As optimization proceeds, it is found that some of the θ_i and corresponding α_i tend to zero. This is an indication that the associated training vectors are not 'relevant' (analogous to a non-support vector) and can be 'pruned' from the solution.

Common choices for the kernel function are the polynomial kernel $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + 1)^d$ and the Gaussian RBF, used here

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_k^2}\right) \text{ where } \|\mathbf{x}\|^2 \equiv \frac{1}{N}\sum_{j=1}^{N} x_j^2.$$

Being smooth and flexible it is a versatile kernel and has proved empirically to be a good choice. Note that there is now a further parameter to worry about: σ_k. If it is too large it may not be possible to fit closely to the dependent variables leading to a long training time and poor output. Conversely, if σ_k is too small over fitting will occur around each training example leading to a non-sparse solution with equally poor characteristics. σ_k is chosen by hand with $\sigma_k = 0.15$ being used throughout the experiments described in sec. 5.

3.2. Inferring state with RVMs

The aim here is to obtain an RVM which, given a sub-image excised from the input image frame according to the current state estimate, will output an estimate of the error in that estimate. To create a regression between erroneous images and the errors that generated them demands a training set of $(\delta\mathbf{X}_i, \mathbf{I}_i)$ pairs ($i \in [1, N]$). These are created by randomly perturbing several different images of the same object (although for an object with rich features just one image can suffice). In face tracking, it is useful to include some training images with out-of-plane rotations, to make the resulting tracker tolerant to such variations. Figure 3 shows some typical training examples.

This method normally provides good coverage of the state space but with a smaller training set it may be possible that a quadrant of the 4D hypercube gets neglected. Small training sets are desirable as they typically contain fewer relevance vectors making evaluation faster at runtime, however this can cause problems if the motion enters a neglected part of state space. To reinforce smaller training sets, extra examples are added, forced to lie in each quadrant (of which there are 16 in 4-space). Similar to the SVM

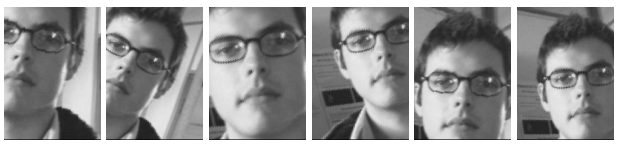

Figure 3. Training examples. *Some typical examples used to train the relevance vector machines on changes in translation, rotation and zoom.*

tracker described earlier, a vector of RVMs is required in order to obtain a vector estimate as output. Each element has one of the four state space dimensions as its dependent variable but now they all use the same training set: a continuous function is being estimated and there is no need to partition examples into positive and negative. This still means that the machine inferring x-translation, say, has been trained on images perturbed in the other three dimensions too and is insensitive to these.

An interesting question is: 'what is special about those training examples that are kept as "relevant"?' Figure 4 shows a plot of the training examples used for a tracker which, for clarity, follows x, y translation only. It can be seen that those chosen as relevant are prototypical of x or y translations being the most extreme examples. This is in contrast to SVMs where the support vectors represent the *least* prototypical — most marginal — examples in a training set.

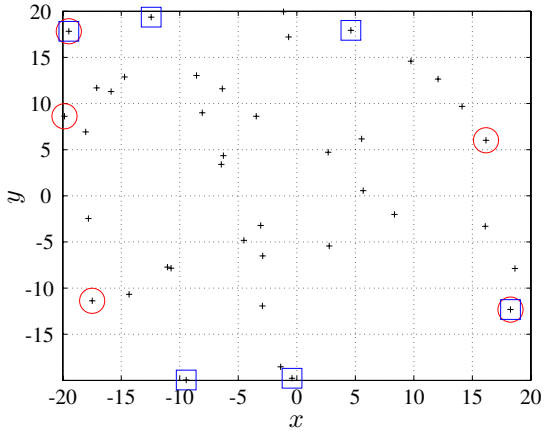

Figure 4. The relevance vectors span the state space. *As a tutorial example, this figure shows the positions of examples used to train a tracker in a space of 2D translations. Relevance vectors are indicated for horizontal (x) translations (circles), vertical (y) ones (squares).*

4. Probabilistic state inference

To secure benefits of temporal fusion of data, observations must be obtained in a probabilistic setting. This is one of the principal advantages of RVMs over SVMs: an RVM gives not only an estimate of change in state, but also generates an output probability distribution.

4.1. Kalman filtering

For each state space dimension, an estimate is made which is expressed in vector form as:

$$\delta\tilde{X}_i = \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k} \tag{9}$$

where $\mathbf{k}$ is a vector defined so that $k_0 = 1$, to allow for bias, and each

$$k_j = k(\mathbf{x}_j, \mathbf{I}(\tilde{\mathbf{X}})) \;\text{ for }\; j > 0. \tag{10}$$

is a kernel function between the j^{th} relevance vector and the current image vector. From (7) $\boldsymbol{\alpha}$ is Gaussian distributed as $\mathcal{N}(\hat{\boldsymbol{\alpha}}, \hat{\Sigma})$, making $\delta\tilde{X}_i$ also Gaussian, with mean $\hat{\boldsymbol{\alpha}}^{\mathrm{T}}\mathbf{k}$ and variance

$$v = \mathbf{k}^{\mathrm{T}}\hat{\Sigma}\mathbf{k}. \tag{11}$$

This probabilistic output can be treated as an *observation*, $\mathbf{z}$, and incorporated into a Kalman-Bucy filter [9, 7].

For each iteration of the tracker we set up state equations for the evolution of the state and the observation. The RVM actually generates an estimate of the *change* in the current state, and is hence an *innovation*, $\boldsymbol{\nu}_k = \mathbf{z}_k - \mathbf{X}_k$, with Gaussian distributed observation noise, $\mathbf{v}_k \sim \mathcal{N}(0, R_k)$. Here R_k is a diagonal covariance matrix, whose terms are the scalar variances (11) v from each of the 4 RVMs for the 4 state space dimensions.

The dynamical process is modelled as a second order auto-regressive process (ARP) [3], augmenting the state equations to take account of the *two* previous observations.

$$p(\mathbf{X'}_{k+1}|\mathbf{X'}_k) = \exp\left\{-\tfrac{1}{2}(\mathbf{X'}_{k+1} - \Phi\mathbf{X'}_k)^{\mathrm{T}}Q_k^{-1}(\mathbf{X'}_{k+1} - \Phi\mathbf{X'}_k)\right\}, \tag{12}$$

where $\mathbf{X}'$ is the augmented form of $\mathbf{X}$ for a second order process. Dynamical coefficients Φ and Q are learned using maximum likelihood from a sequence of parameters capturing typical motions of the object in question [4].

4.2 Initialization and validation

For efficient operation, the RVM tracker exploits temporal coherence fully. However, a robust tracking system capable of operating for an indefinite period also needs a recognition system, running in tandem, for initialization and recovery. Here this takes the form of an SVM recognizer, as in figure 1, operating in two distinct modes. During continuous tracking it is applied at the currently estimated state to verify the state and the identity of the tracked object. For efficiency, this test is made only every M frames (in experiments here $M = 4$ has been successful). Absence of verification triggers a search in which the SVM is applied over a tessellation of neighbourhoods in the Euclidean similarity space, and this continues until verification is obtained,

after which RVM tracking resumes. The same mechanism is used to initialize from the first frame.

It is observed, from figure 1, that as estimated object state strays from the true state, the SVM score degrades progressively, and eventually changes sign, indicating that the observation of the RVMs is invalid. Furthermore, if the object changes in ways that are not modelled within the state space (e.g. out of plane rotation), observations become invalid. For initialization, the progressive degradation of SVM score with displacement has a further significance: the recovery search does not need to cover the state space in excessive detail. The width of the peak in figure 1 indicates the density of the required tessellation of SVM locations.

5. Results

The experiments conducted here all use only one initial image region which was perturbed 45 times to create the RVM training sets as described in section 3.2.

Figure 5 shows the estimates made by the RVM concerned with x-translation. An image (not in the training set) was displaced by a known number of pixels, and the RVMs' posterior estimate of that change is plotted.

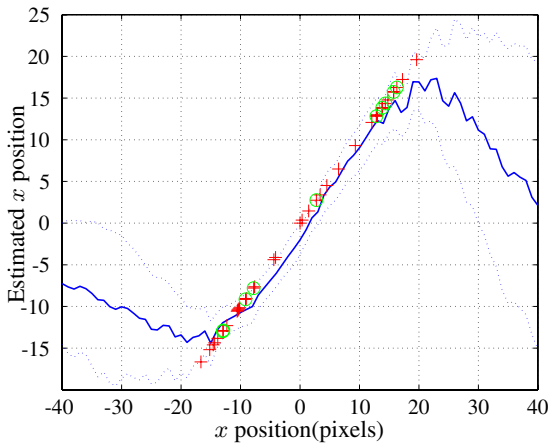

Figure 5. RVM as a statistical observer of state. *An RVM trained for regression onto horizontal translation delivers a linear mean-response as shown, together with observation variance (upper and lower dotted curves: ± 3 standard deviations). Crosses indicate the training examples, with relevance vectors marked with circles.*

Figure 6 shows how advantageous it is to use probabilistic inference. The sequence being tracked is one that becomes progressively more rapid, and without filtering the errors grow to the point where lock is lost, and the tracker requires reinitialization. With a Kalman filter in place the error[1] is an order of magnitude lower. Probabilistic inference has greatly increased both the stability and the robustness of tracking. As temporal fusion has reinforced the output, we can permit the RVMs to be more sparse by using smaller training sets: the quality of the RVM posterior estimates is correlated with the number of relevance vectors. This in turn leads to faster run-time evaluation of the RVMs and a lower computational cost per frame.

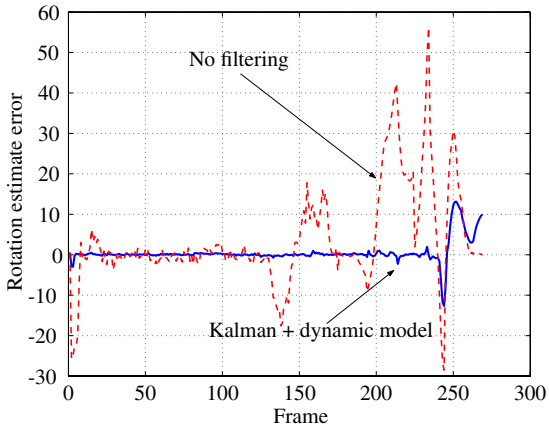

Figure 6. Probabilistic inference improves performance. *This figure shows the error performance of the RVM tracker with Kalman Filter (solid), compared with raw, unfiltered RVM output (dashed).*

Tracking might be considered redundant if it were possible to perform raw detection at frame rate. This is a popular approach [13, 15, 18] but Figure 7 confirms the substantial efficiency saving available by tracking over time. RVM tracking consumes only 15ms of CPU time per frame, but when brute-force SVM search is running consumption of CPU time rises by a factor of about 70 to 1 second per frame. (This data was gathered from code with no optimizations using double-precision floating point arithmetic throughout: it is anticipated this can be improved upon greatly.)

Figure 8 contains snapshots of the tracking of a talking head and this leads naturally to an important videoconferencing application, in which a close-up of the talker is broadcast when addressing the camera, receding to a wide-angle shot when the talker turns to a third party, or departs altogether. The camera used to create these results is a cheap, standard web cam with a plastic lens. Image patches ($\mathbf{I}$ elsewhere in this document) are sub-sampled such that $N = 704$ pixels are examined at each iteration and it is only the intensity that is tested: no colour is employed. A movie-

[1] The ground-truth position was found by allowing this tracker to iterate to convergence on each frame treated as a still. The results were then inspected by hand.

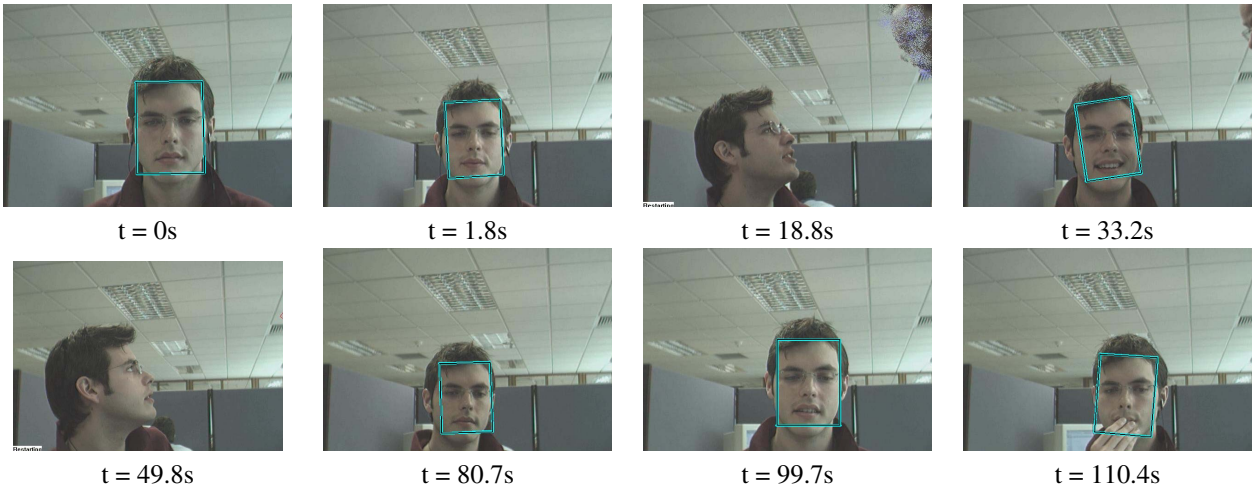

Figure 8. Long-term tracking behaviour. *This figure shows that tracking continues robustly over an extended period — 2 minutes in this example, but has run for several hours in experimental use.*

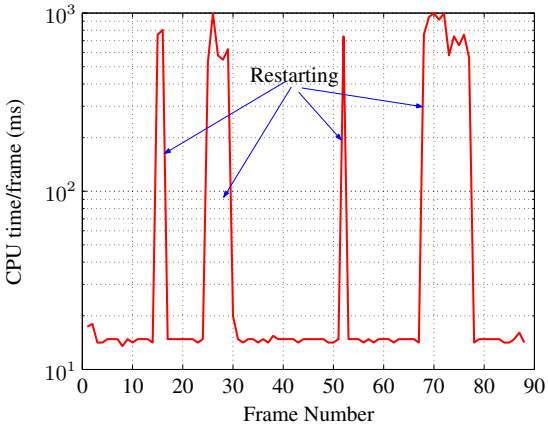

Figure 7. RVM tracking is highly efficient. *The graph shows milliseconds of CPU time required per video frame (2.6 GHz Pentium 4). During continuous tracking only around 15ms/frame of CPU time is required, thanks to the exploitation of temporal coherence in the Kalman filter. This rises dramatically to around 1s/frame when coherence is lost, in initialization or recovery modes.*

clip is available[2] showing a teleconferencing application in which the real-time tracking output is used to control zoom and pan. This was captured using a hand-held DV camera. This tracker is also capable of following a general object undergoing 2D deformation, such as the car and a license plate in figure 9.

[2] ftp://mi.eng.cam.ac.uk/pub/data/pancake.mpg

6. Discussion and conclusions

We have demonstrated a tracker using sparse probabilistic regression by RVMs, with temporal fusion for high efficiency and robustness. Further robustness is obtained by running, in tandem, an object-verification SVM which simply serves to validate observations during normal running, but also for intensive object search during initialization and recovery. Training, both of the SVM and RVM, is performed from a single object instance perturbed to generate a training set, and requires only a few minutes. The result is a reliable real-time tracker which imposes a modest computational load and, thanks to its recovery mechanism, runs continuously. We have demonstrated real time operation with a face-tracking camera-management system for teleconferencing. This is a general method for tracking and, to illustrate this, tracking of cars has also been shown.

The probabilistic framework would also permit the replacement of the Kalman filter with a particle filter [4]. Particle filters are also popular for their robustness in the presence of clutter, however the object specificity of an SVM/RVM is already a powerful filter for clutter, so there is some doubt whether a particle filter would add significant power in practice.

Future work will address a number of issues:

- greater invariance to illumination changes could be obtained by pre-processing of image intensities beyond the simple intensity normalization used here;
- adaptive re-training in parallel with tracking;
- greater variation of view and articulation.

Figure 9. Tracking cars *Digital video recordings of a passing vehicle and a license plate. The tracker was trained from a single frame and successfully follows the regions despite clutter and an unsteady camera.*

References

[1] S. Avidan. Support vector tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, Hawaii, 2001.

[2] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. European Conf. on Computer Vision*, volume 1, pages 329–342, 1996.

[3] A. Blake and M. Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *Proc. Siggraph*, pages 185–192, 1994.

[4] A. Blake and M. Isard. *Active contours*. Springer, 1998.

[5] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[6] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based methods*. Cambridge University Press, 2000.

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

[8] A. Jepson, D. Fleet, and T. El-Maraghi. Robust on-line appearance models for visual tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 415–422, 2001.

[9] R. Kalman. New methods in Wiener filtering. In *Proc. of the First Symposium on Engineering Applications of Random Function Theory and Probability*. John Wiley and Sons, Inc, 1963.

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

[11] J. Kwok. Moderating the outputs of support vector machine classifiers. In *IEEE Transactions on Neural Networks*, volume 10, pages 1018–1031, September 1999.

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

[13] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. *Proc. Conf. Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[14] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[15] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 524–531, 2001.

[16] A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, NeuroCOLT, 1998.

[17] M. Tipping. The relevance vector machine. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 652–658, 2000.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2001.

[19] C. Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In M. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer Academic Press, 1998.

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