Sparse Bayesian Learning for Efficient Visual Tracking

Oliver Williams, Andrew Blake, *Member*, *IEEE*, and Roberto Cipolla, *Member*, *IEEE*

Abstract—This paper extends the use of statistical learning algorithms for object localization. It has been shown that object recognizers using kernel-SVMs can be elegantly adapted to localization by means of spatial perturbation of the SVM. While this SVM applies to each frame of a video independently of other frames, the benefits of temporal fusion of data are well-known. This is addressed here by using a fully probabilistic *Relevance Vector Machine* (RVM) to generate observations with Gaussian distributions that can be fused over time. Rather than adapting a recognizer, we build a *displacement expert* which directly estimates displacement from the target region. An object detector is used in tandem, for object verification, providing the capability for automatic initialization and recovery. This approach is demonstrated in real-time tracking systems where the *sparsity* of the RVM means that only a fraction of CPU time is required to track at frame rate. An experimental evaluation compares this approach to the state of the art showing it to be a viable method for long-term region tracking.

Index Terms—Probabilistic algorithms, robust regression, tracking, object recognition.

1 Introduction

Real-time tracking is an important component of many modern applications, including: human-computer interaction [5], surveillance [29], vehicle automation [1], and augmented reality [26]. Systems for tracking moving objects may be feature-based or model-based. Feature-based tracking can rely on persistence of image curves [16], [4] or image appearance [6]. Likewise, model-based tracking can use curves, either in 3D [19] or in 2D [30], or appearance [2]. Furthermore, statistically-based object detectors using support vector machines (SVMs) [20], [21] and boosting [28] are now fast enough to run at video rate, despite having to search every video frame.

Recently, the distinction between detection and tracking has been broken down further by the introduction of *support vector tracking* [1] in which the output of an SVM object detector is locally optimized to locate a target (see Section 1.1). This paper seeks to extend the use of statistical learning algorithms for object localization. The *relevance vector machine* (RVM) [25] is a sparse learning algorithm, similar to the SVM in many respects but capable of delivering a fully probabilistic output (see Section 2). Section 3 describes how an RVM is trained to learn the relationship between the local motion of an object and its appearance in an image. This *displacement expert* is trained online and can then be used to infer unknown motion from an image and thereby track a single target object. Owing to the sparsity properties of the RVM, the displacement expert makes estimates very efficiently meaning the object is tracked in real time using only a fraction of the processing power of a desktop PC.

While it is highly desirable to have such fast local search, there are circumstances in which it will not suffice, such as initialization. Section 4 explains how an object detection algorithm is used in tandem with the displacement expert to provide robustness on occasions that local search fails. The result is a system that is not only efficient but highly robust in the very long term. Section 5 describes an experimental evaluation of this system including comparison to existing methods. These experiments show that a system exploiting the strengths of both tracking and detection is efficient, accurate, and robust. Examples are given for the affine tracking of faces, hands, and cars.

1.1 Previous Work

The SVM is a powerful algorithm for binary classification [27], in which the class of a vectorized subimage, $\mathbf{x}$ is determined according to the sign of a classification function

$$f(\mathbf{x}) = \sum_{i=1}^{N} t_i w_i k(\mathbf{x}, \mathbf{z}_i) + w_0, \tag{1}$$

where $\mathbf{z}_i$, $i \in [1, N]$ are vectors from a training set with labels $t_i \in \{-1, +1\}$; w_i are real-valued coefficients which must be determined by the learning algorithm, and $k(\cdot, \cdot)$ is a *kernel* function [22] which computes a generalized inner product. A great attraction of the SVM is the *sparsity* conferred by its training algorithm which typically sets most of the w_i to zero, leaving just a few active terms in (1); the active subset of the $\mathbf{z}_i$ are known as *support vectors*. A classifier such as this, trained to recognize a particular class of object, can be used for detection by applying it to subimages extracted in a tessellation over some state space (e.g., translation and scale) [20], [21].

Such a search is labor intensive and Avidan's support vector tracker [1] seeks to mitigate this by perturbing the classification function f with respect to image translation.

- *O. Williams is with the Machine Intelligence Laboratory, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK. E-mail: omcw2@cam.ac.uk.*
- *A. Blake is with Microsoft Research, 7 J.J. Thompson Ave., Cambridge, CB3 0FB, UK. E-mail: ablake@microsoft.com.*
- *R. Cipolla is with the Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK. E-mail: cipolla@eng.cam.ac.uk.*

Manuscript received 1 Mar. 2004; revised 28 Oct. 2004; accepted 29 Oct. 2004; published online 13 June 2005.
Recommended for acceptance by S. Sclaroff.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0110-0304.

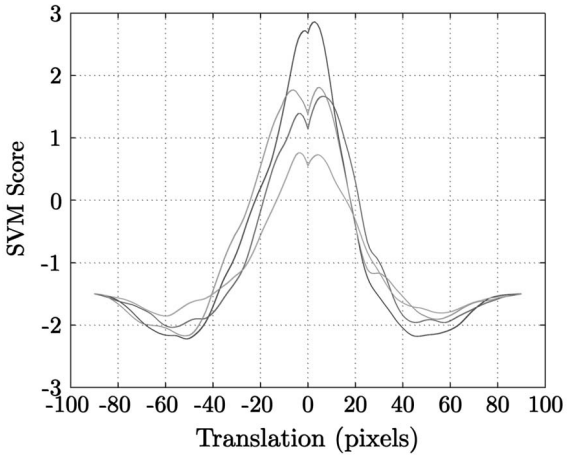


Fig. 1. Classification function of an SVM trained for recognition. An SVM trained on a face database was used to classify four unseen examples, misregistered by a given horizontal translation. A maximum is present at the true position, though the function has additional local maxima. The dip at zero-translation is due to the many training images showing a small out-of-plane rotation to the left or right.

Given an object $\mathbf{x}$, perturbation analysis allows the translation $T_u\mathbf{x}$, by a vector $\mathbf{u}$, to be found such that the value $f(T_u\mathbf{x})$ of the classification function is maximized (see Fig. 1). The perturbed classification function, Avidan shows, is expressed in terms of the image gradient as:

$$f(T_u\mathbf{x}) \approx f\left(\mathbf{x} + \frac{\partial \mathbf{x}}{\partial \mathbf{u}}\delta\mathbf{u}\right). \quad (2)$$

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

Fig. 1 illustrates the difficulties of tracking via an optimization: What is a suitable cost function and how effectively or efficiently can it be optimized? This has been a difficult question to answer and methods such as [10], [15], [7] avoid it by inferring local motion directly from images using a linear transformation of the image gradient. The work described in the following sections extends these ideas by using the relevance vector machine (RVM) [25] to learn a *nonlinear* mapping from images $\mathbf{x}$ to displacements $\delta\mathbf{u}$. The resulting *displacement expert* requires neither the optimization step of [1] nor the smooth image gradients of [10], [15]. Section 2 briefly describes the RVM and Section 3 explains how the displacement expert is created. Support vector tracking applies to each frame of a video independently of other frames and a further benefit of the fully probabilistic RVM is temporal fusion [9] (improved efficiency and enhanced ability to deal with clutter in the background) which can be accomplished in an effective and principled way in a probabilistic setting.

The price of eschewing optimization as a method of tracking is that we relinquish a direct mechanism for detecting loss of lock.[1] To overcome this, the approach of [13] is adopted. Tracker estimates are checked by a binary classifier trained to detect the object of interest. A negative result triggers a full-image search. Other approaches, capable of recovering from failure, use a gating signal [4] to accept only measurements with a high predicted accuracy and particle filtering [11] to maintain multiple hypotheses. The validation and restart mechanism is described in Section 4.

In summary, the system developed and demonstrated in this paper has the following properties:

1. Fully probabilistic regression for displacement, using RVMs.
2. Displacement is modeled as Euclidean and Affine similarity transformations.
3. Tracker is trained online from labeled images.
4. Observations of displacement are fused temporally with motion prediction.
5. Initialization and recovery are handled by an object detector running in tandem.
6. Tracking is efficient—better than real-time (i.e., leaving processor cycles free for other processes)—and this is demonstrated in a variety of scenarios (Section 5).

2 Sparse Bayesian Regression

The SVM has several desirable properties, including:

1. It fits functions in high-dimensional feature spaces, through the use of kernels.
2. Despite a possibly large space of functions available in feature space, good generalization performance is nevertheless achieved by *margin maximization* [22].
3. It is *sparse*: Only a subset of training examples is retained at runtime, thereby improving computational efficiency (Section 1.1).

Its principal disadvantage is that estimates are made with no measure of uncertainty, i.e., the output is not a probability distribution, hence, precluding the use of a statistical filter on estimates made as part of a temporal sequence.

The Relevance Vector Machine, or RVM, was proposed by Tipping [24] as a Bayesian treatment of the sparse learning problem. The RVM preserves the generalization and sparsity properties of the SVM, yet it also yields a probabilistic output, as well as circumventing other limitations of the SVM.[2] As proposed in Section 1.1, this machine learns a *regression* from images (rather than classification) and, hence, the training set will comprise N pairs of example vectors, $\mathbf{z}_i \in \mathbb{R}^M$ and *target* values, $t_i \in \mathbb{R}$.

2.1 Relevance Vector Machine

The first consideration is the output functional, which is almost identical to that of the SVM (1) [27]:

$$g(\mathbf{x}) = \sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{z}_i) + w_0, \quad (3)$$

where $\mathbf{x}$ is an input vector and $g : \mathbb{R}^M \to \mathbb{R}$ is the scalar-valued output function, modeled as a weighted sum of kernel evaluations between the test vector and the training examples. The kernel function, $k : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ can be considered either as a similarity function between vectors, or as a basis function centered at $\mathbf{z}_i$. Training determines the

1. If an optimization method converges to an inadequate value of the objective function, it can be inferred that the tracker has lost lock.

2. There is no necessity for *Mercer* kernels [22] and no error/margin trade-off parameter: C.

weights, $\mathbf{w} = [w_0, \ldots, w_N]$ and the sparsity property will arise if some of the w_i are set to zero.

As this is Bayesian inference, a prior distribution is specified over $\mathbf{w}$ and, to encourage sparsity, Tipping stipulates a zero-mean Gaussian distribution [25]:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, A), \tag{4}$$

where $\mathcal{N}(\mathbf{d}; \mathbf{e}, F)$ denotes the multivariate Gaussian distribution of a random vector $\mathbf{d}$ with mean $\mathbf{e}$ and covariance matrix F. The covariance $A = \mathrm{diag}(\alpha_0, \ldots, \alpha_N)$: There is an individual and independent hyperparameter for every weight.

For the likelihood term, it is assumed that the training targets are sampled with additive noise [25]:

$$t_i = g(\mathbf{z}_i) + \epsilon_i, \tag{5}$$

where $\epsilon_i \sim \mathcal{N}(\epsilon; 0, \sigma^2)$. Thus,

$$p(t_i|\{\mathbf{z}_i\}) = \mathcal{N}\big(t_i; g(\mathbf{z}_i, \sigma^2)\big). \tag{6}$$

σ^2 is another hyperparameter and global across the training set.

Without considering hyperpriors (the general case is discussed in [25]), the posterior is also Gaussian:

$$p(\mathbf{w}|\{\mathbf{z}_i, t_i\}, \boldsymbol{\alpha}, \sigma^2) = \mathcal{N}(\mathbf{w}; \hat{\mathbf{w}}, \Sigma), \tag{7}$$

with [25]

$$\Sigma^{-1} = \sigma^{-2}\Phi^T\Phi + A$$
$$\hat{\mathbf{w}} = \sigma^{-2}\Sigma\Phi^T\mathbf{t}.$$

Φ is called the *design matrix* and contains the intratraining set kernel values: $\Phi_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$.

2.2 Training

As shown in (7), $\hat{\mathbf{w}}$ and Σ are determined by the hyperparameters, $\boldsymbol{\alpha}, \sigma^2$ which characterize different models and training involves finding these values via Bayesian model selection. To do this, the *evidence*, $p(\{\mathbf{z}_i, t_i\}|\boldsymbol{\alpha}, \sigma^2)$ [25] is maximized using a variant of gradient ascent. The algorithm begins optimizing over the entire training set, but examples are *pruned* whenever the associated α_i falls below a threshold,[3] leading to the final sparse solution. Those examples remaining with $w_i \neq 0$ are termed *relevance vectors*. Each iteration of the training algorithm evaluates the posterior (7) which involves matrix inversion: an $\mathcal{O}(N^3)$ operation. The RVM therefore takes longer to train than the SVM, although faster training algorithms are being developed for large training sets [23].

The kernel function used for the experiments described in this paper was the Gaussian radial basis function [22]:

$$k(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{1}{2M\kappa^2}\|\mathbf{z}_i - \mathbf{z}_j\|^2\right), \tag{8}$$

which is defined here to compensate for the input dimensionality M. The choice of the width parameter, κ^2 has a dramatic effect on the outcome of training: too small and the RVM will overfit, too large and it will underfit (large σ^2 and poor accuracy at runtime). The value used here was set by experiment (see Section 5).

3. $\alpha_i = 0$ implies that $p(w_i|\alpha_i)$ is infinitely peaked at zero.

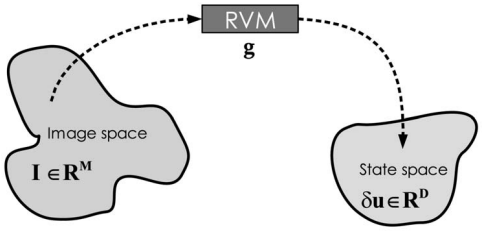


Fig. 2. Mapping from image space to state space. An RVM displacement expert maps a high-dimensional vector, corresponding to an image subregion, into a vector with dimensionality given by the degrees of freedom of the motion being modeled. M is the number of pixels in an image patch and D is the number of degrees of freedom of the motion being learned.

3 Building a Displacement Expert

The aim of the work described in this section is to train an RVM such that it learns the relationship between images and motion. Specifically, we want to convert an image subregion into a vector, $\mathbf{x}$ and for an RVM expert to return the displacement, $\delta\mathbf{u} \in \mathrm{GE}(2)$ of that subregion from the target's true location:

$$\delta\mathbf{u} = \mathbf{g}(\mathbf{x}). \tag{9}$$

The RVM learns a mapping $\mathbf{g}$ from the high-dimensional space of subimages to the low-dimensional motion state space (see Fig. 2).

3.1 Sampling and normalization

A subimage is considered as a $p \times q$ rectangular region of a complete gray-scale image, I. The following procedure is used to sample from the image into a vector:

1. Sample the required pixels from the image into a patch at the origin:

$$P(i,j) = I(\boldsymbol{\psi}_{\mathbf{u}}(i,j)) \quad i = 1 \ldots p \quad j = 1 \ldots q, \tag{10}$$

where $\boldsymbol{\psi}_{\mathbf{u}} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^2$ is a *warp* function that transforms a point by the transformation parameterized by $\mathbf{u}$.

2. Raster scan the transformed region into a vector:

$$r_{i+(j-1)p} = P(i,j) \quad i = 1 \ldots p \quad j = 1 \ldots q. \tag{11}$$

3. Finally, histogram equalize this vector to provide a degree of illumination invariance [12]:

$$\mathbf{x} = h(\mathbf{r}), \tag{12}$$

where h denotes the histogram equalization operation.

3.2 Creating a Training Set

Initially, the algorithm requires at least one *seed* image I containing the labeled region of interest $\boldsymbol{\lambda}$. Such images are labeled either by hand or automatically. Fig. 3 shows how multiple training examples, $\{\mathbf{z}\}$, are excised from a single seed image.

For this paper, experiments were mostly carried out with a motion state space of dimension $D = 4$ (the Euclidean similarities). The RVM, as described in Section 2, only learns a scalar valued function, thus multiple RVMs are necessary, each mapping to a different state space dimension. The RVMs are trained on an identical set of vectors, $\{\mathbf{z}\}$, but different targets, $\{t\}$. Each machine will see examples displaced along

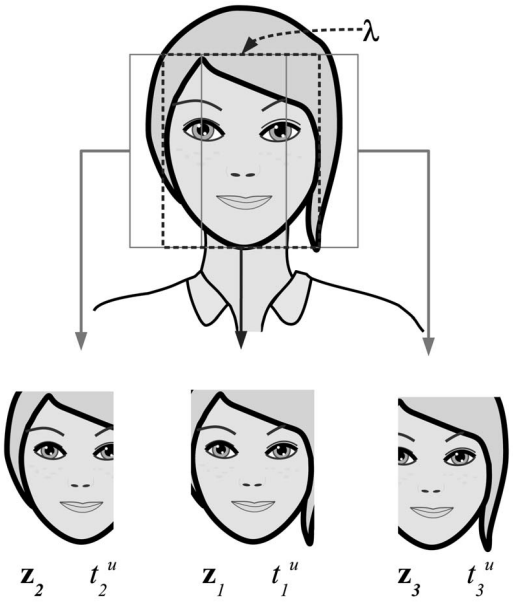


Fig. 3. Example Generation. From a labeled region of interest, many training examples may be sampled and converted to vectors $\mathbf{z}_i$, accompanied by target displacements t_i. As a tutorial example this expert is estimating horizontal translation only.

state space dimensions for which they are not experts and this beneficially engenders insensitivity to the orthogonal degrees of freedom. Examples are generated with random displacements sampled from a uniform distribution

$$\mathbf{t}_i \sim \mathcal{U}(-\mathbf{\Delta}, +\mathbf{\Delta}), \tag{13}$$

where $\mathcal{U}$ is the multivariate uniform distribution between limits $\pm\mathbf{\Delta}$. An example $\mathbf{z}_i$ is then excised from the seed image as described in Section 3.1 with warp parameters $\mathbf{u} = \mathbf{t}_i + \boldsymbol{\lambda}$.

Fig. 4 shows some subimages from a real training set for a face tracking application. This highlights a problem with generating examples from only one seed image: the background is being learned along with the foreground target. Using more seed images showing the target against different backgrounds mitigates this problem and also provides some invariance to appearance changes in the target (Section 5 shows some experiments assessing the efficacy of using multiple seed images).

The choice of the range of displacements and the number of examples affects three aspects of training and runtime performance:

1. The Gaussian RBF kernel (8) employs the L_2 norm between vectors. If examples are more tightly spaced, the width of the kernel κ can be made smaller resulting in more sharply peaked basis functions capable of more precise localization.
2. The range of displacements used in generating training examples $\mathbf{\Delta}$ dictates the range over which the expert can be relied on at runtime (see Section 5) and thereby the maximum interframe motion that can be tracked.
3. RVM training time scales as $\mathcal{O}(N^3)$ and both a larger capture range and closer packed examples result in more relevance vectors retained by the expert, incurring more kernel evaluations and reduced speed at runtime.

Section 5 contains experimental results showing the effects of these settings on real tracking performance. Algorithm 1 summarizes the procedure for training a displacement expert:

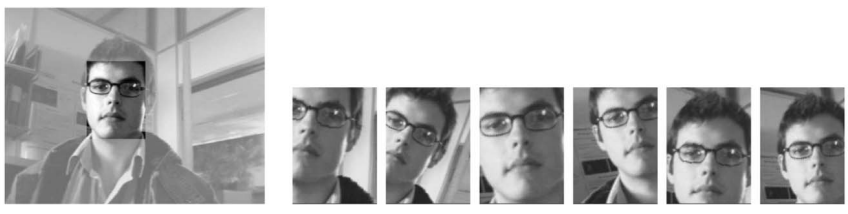
Algorithm 1 Learn displacement expert $\delta\mathbf{u} = \mathbf{g}(\mathbf{x})$
Require: Seed images: : $\{I\}$
Require: Labels: $\{\boldsymbol{\lambda}\}$
Require: Displacement Range: $\mathbf{\Delta}$
Require: Number of examples: N
  for $i = 1$ to N **do**
    Generate random displacement
    $\mathbf{t}_i \leftarrow \text{Uniform}(-\mathbf{\Delta}, +\mathbf{\Delta})$
    Sample from random seed image, I_r (Section 3.1)
    $\mathbf{z}_i \leftarrow \text{Sample}(I_r, \boldsymbol{\lambda}_r + \mathbf{t}_i)$
  end for
  Train an RVM for each state space dimension
  for $j = 1$ to D **do**
    Select correct target data
    $\{\tau\} \leftarrow j\text{th dimension of}\{\mathbf{t}\}$
    RVM training algorithm as described in [25]
    $\hat{\mathbf{w}}_j, \Sigma_j \leftarrow \text{RVMTrain}(\{\mathbf{z}\}, \{\tau\})$
  end for
  Delete from memory all examples $\mathbf{z}_i$ for which $\hat{w}_i = 0$

Once trained, what is special about the training examples kept as "relevant" (i.e., $w_i \neq 0$)? Fig. 5 shows the displacements of training examples for an expert working with two degrees of freedom (2D translation). Those chosen as relevant appear to be the most extreme examples and as such are prototypical of horizontal or vertical translation.

3.3 Tracking with a Displacement Expert

Fig. 6 shows the process by which the displacement of a subimage from the true target position is estimated. The displacement along state-space dimension i is calculated as

$$\delta u_i = g_i(\mathbf{x}_t) = \mathbf{w}_i^T \mathbf{k}, \tag{14}$$



(a) (b)

Fig. 4. Real training examples. (a) A labeled seed image. (b) Some typical examples used to train the relevance vector machines on displacements in translation, rotation and scaling.

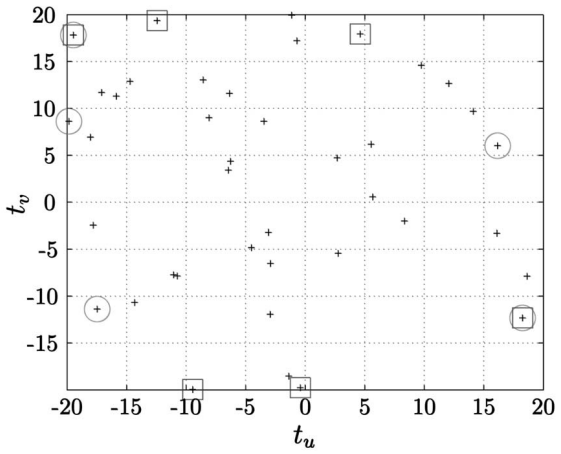


Fig. 5. The relevance vectors span the state space. As a tutorial example, this figure shows the positions ($\mathbf{t}_i$) of examples used to train a tracker in a space of 2D translations. Relevance vectors are indicated for horizontal translations (circles) and vertical ones (squares).

where $k_j = k(\mathbf{x}_t, \mathbf{z}_j)$ forms a vector of kernel evaluations and $\mathbf{x}_t$ is the queried subregion at time t sampled into a vector using $\mathbf{u}_t$. From (7), $\mathbf{w}_i$ is Gaussian distributed as $\mathcal{N}(\mathbf{w}_i|\hat{\mathbf{w}}_i, \Sigma_i)$ meaning that δu_i is also Gaussian with mean $\hat{\mathbf{w}}_i^T\mathbf{k}$ and variance

$$s_i = \mathbf{k}^T\Sigma_i\mathbf{k} + \sigma_i^2, \tag{15}$$

where σ_i^2 is the variance from (6) [25]. To secure the benefits of temporal fusion of data, observations must be obtained in a probabilistic setting. This is one of the principal advantages of the RVM over the SVM: The RVM doesn't just estimate the change in state, but generates an output probability distribution:

$$\delta\mathbf{u} \sim \mathcal{N}(\delta\mathbf{u}|W\mathbf{k}, S). \tag{16}$$

W is a matrix whose rows are the weight vectors of the D RVMs,

$$W = \begin{bmatrix} \hat{\mathbf{w}}_1^T \\ \vdots \\ \hat{\mathbf{w}}_D^T \end{bmatrix} \tag{17}$$

and S is a diagonal covariance matrix (the cross-covariance terms are assumed to be zero) containing the scalar variances from (15):

$$S = \mathrm{diag}(s_1, \ldots, s_D). \tag{18}$$

This probabilistic output can be treated as a Gaussian *innovation* and incorporated into a Kalman-Bucy filter [9].

Equations for the evolution of the state and observation are established by modeling the dynamics as a second order autoregressive process (ARP) [3]. The state equations are augmented to account for the *two* previous observations:

$$p(\mathbf{u}'_{t+1}|\mathbf{u}'_t) \propto \exp\left\{-\frac{1}{2}\left(\mathbf{u}'_{t+1} - \Theta\mathbf{u}'_t\right)^T Q^{-1}\left(\mathbf{u}'_{t+1} - \Theta\mathbf{u}'_t\right)\right\}, \tag{19}$$

where $\mathbf{u}_t$ is a state estimate at time t and $\mathbf{u}'_t = [\mathbf{u}_t\mathbf{u}_{t-1}]^T$ is its augmented form. The coefficients, Θ and Q are learned using maximum likelihood from a sequence capturing typical motion of the target region [4].

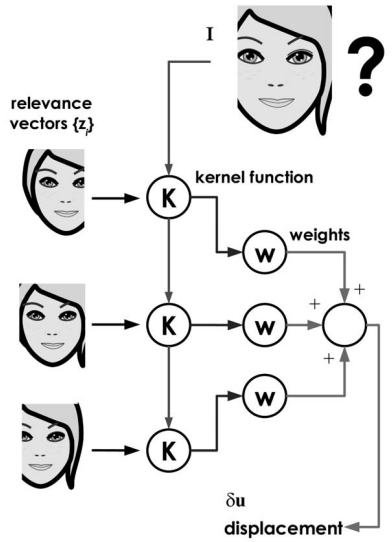


Fig. 6. Expert prediction. A vector, $\mathbf{x}$ that has been sampled from an image, is compared to the relevance vectors using the kernel function. These values are then summed via the weights found by Algorithm 1 to give an estimated displacement $\delta\mathbf{u}$ of $\mathbf{x}$ from the target.

The innovation term, $\delta\mathbf{u}$, is then used to update the state estimate, $\mathbf{u}_t$, by fusing this observation with the motion model [9]. Algorithm 2 outlines the continuous tracking process:

Algorithm 2 Tracking loop
Require: Initial state estimate: $\mathbf{u}_0$
Require: Initial state covariance: U_0
  $t \leftarrow 0$
  loop
    $I \leftarrow$ New image from capture device
    $\mathbf{x}_t \leftarrow \mathrm{Sample}(I, \mathbf{u}_t)$
    Extrapolate state estimate using motion model (19)
    $\mathbf{u}_{t+1} \leftarrow \Theta\mathbf{u}_t$
    $U_{t+1} \leftarrow \Theta U_t\Theta^T + Q$
    Get innovation from displacement expert (16)
    $\delta\mathbf{u}, S \leftarrow \mathbf{g}(\mathbf{x}_t)$
    Compute Kalman gain [9]
    $G_k \leftarrow U_{t+1}[U_{t+1} + S]^{-1}$
    Fuse prediction and innovation
    $\mathbf{u}_{t+1} \leftarrow \mathbf{u}_{t+1} + G_k\delta\mathbf{u}$
    $U_{t+1} \leftarrow U_{t+1} - G_kU_{t+1}$
    $t \leftarrow t + 1$
  end loop

4 Initialization and Recovery

For efficient operation, the RVM tracker fully exploits temporal coherence. However, a robust tracker capable of operating for an indefinite period also needs a detection system, running in tandem, for initialization and recovery. This object detector operates in two distinct modes. During continuous tracking, the identity of the tracked object is verified by warping it according to the current state estimate (Section 3.1) and testing it with a classifier.[4] Absence of

4. Because image patches are warped before verification, rotated objects are successfully verified. The restart mechanism, however, does not model rotations and, thus, will not locate an object rotated by a significant amount from its canonical orientation.

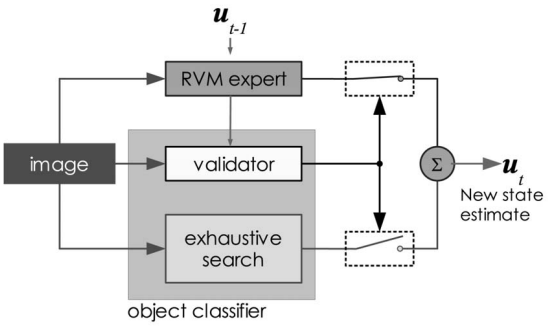


Fig. 7. Overall system design. At each time step, the validator uses the object classifier to examine the subregion described by the state estimate, $\mathbf{u}_t$. If the classifier gives a positive response, the RVM displacement expert is used to generate estimates as outlined in Algorithm 2. A negative response from the validator indicates loss of lock requiring the classifier to reacquire the target by exhaustive search. When search completes successfully, control is returned to the displacement expert.

verification triggers a search in which the classifier tests a tessellation of neighborhoods in a grid over the image and this continues until verification is obtained, after which RVM tracking resumes. The same mechanism is used to initialize from the first frame (i.e., provide $\mathbf{u}_0$ in Algorithm 2).

4.1 Overall System Design

The complete tracking system now has three components which are shown in Fig. 7:

1. The RVM displacement expert as described in Section 3.
2. A validator which uses a classifier to test the region of the image described by the current state estimate.
3. A full-frame search which uses the chosen classifier to search an entire image exhaustively, independent of previous estimates.

The search mechanism can also be used to build the tracking component by automatically labeling the *seed* images required by the training algorithm (Algorithm 1). This is only possible when the type of object to be tracked is known beforehand and the classifier suitably trained. The overall algorithm is summarized in Algorithm 3:

Algorithm 3 Complete tracking system
Require: Displacement Range: $\mathbf{\Delta}$, # examples: N, # seed images: $nSeed$
Expert Training
$n \leftarrow 0$
while $n < nSeed$ **do**
  while No object detected **do**
    $I_n \leftarrow$ New image from capture device
    $\boldsymbol{\lambda}_n \leftarrow \text{Search}(I)$
    $n \leftarrow n + 1$
  end while
end while
Train expert on $(\{I\}, \{\boldsymbol{\lambda}\}, N, \mathbf{\Delta})$ from Algorithm 1
Initialization
while No object detected **do**
  $I \leftarrow$ New image from capture device
  $\mathbf{u}_0 \leftarrow \text{Search}(I)$
end while
Tracking Loop
$t \leftarrow 0$
mode$\leftarrow$ EXPERT
loop
  $I \leftarrow$ New image from capture device
  if mode $==$ EXPERT **then**
    Update state according to Algorithm 2
    Test new estimate with validator
    if Test passed **then** mode $\leftarrow$ EXPERT
    else mode $\leftarrow$ SEARCH
  else
    $\mathbf{u}_{t+1} \leftarrow \text{Search}(I)$
    if Target found **then** mode $\leftarrow$ EXPERT
    else mode $\leftarrow$ SEARCH
  end if
  $t \leftarrow t + 1$
end loop

4.2 Algorithmic Complexity

As described in Section 2, RVM training time scales as $\mathcal{O}(N^3)$, where N is the number of training examples. In the case of the displacement expert, $N < 100$ examples in total produces satisfactory results (see Section 5) and training takes a few seconds. During continuous tracking (following Algorithm 2), the operation count for generating innovations from the expert scales linearly with three terms:

$$\text{ops}_{\text{expt}} \propto [\#\text{ pixels in patch}] \times [\#\text{ relevance vectors}] \times [\text{DOF}]. \tag{20}$$

In a particular application, the number of degrees of freedom is fixed, however, the strength of the RVM learning algorithm is that it can learn a displacement rule from a training set of subsampled images (typically comprising 400-1,000 pixels), retaining a fraction of the training set as relevance vectors (typically around 75 percent in our experiments, see Section 5).

For a typical object detector, which searches an image exhaustively (in translation and scale space), the number of operations is also linear:

$$\begin{aligned}\text{ops}_{\text{search}} \propto\ & [\#\text{ features}] \times [\text{Image width}] \times [\text{Image height}] \\ & \times [\text{Tessellation density}] \times [\#\text{ scales}].\end{aligned} \tag{21}$$

Fast object detection algorithms achieve computational efficiency in part by adapting the number of features used for each test. In the case of [28], [21] this is done via a detection cascade wherein many candidates are rejected without the need to examine all of the features. Importantly, however, the cost of a detector of this kind still scales with the size of an input image whereas the cost of the displacement expert does not. The validator, which tests just one location, also escapes any dependency on image size and thereby imposes negligible computational cost.

5 Experimental Results

Experiments were conducted to investigate the performance and characteristics of the tracking system described in this paper. These are divided into experiments on the displacement expert alone and on the complete hybrid system of Section 4:

Displacement expert

1. Sensitivity to parameter settings,
2. Effect of multiple seed images,

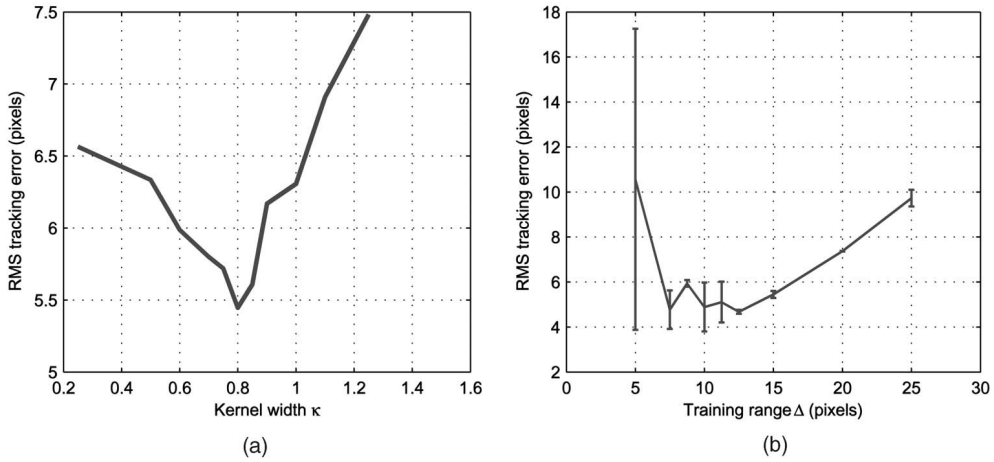


Fig. 8. Parameter sensitivity. (a) Effect of kernel width parameter, κ on tracking accuracy. (b) Effect of training displacement range, Δ (only translation shown), on tracking accuracy. Error bars indicate $\pm$ one standard deviation in results.

3. Tolerance to unmodeled 3D motion and deformation,
4. Tolerance to lighting variation and occlusion,
5. Tolerance to varying background,
6. Tracking with 6 degrees of freedom, and
7. Comparison to other 2D tracking approaches.

Complete System

1. Comparison to plain object detector and
2. Long-term performance.

5.1 Test Data

The experiments were carried-out on video sequences captured with a variety of equipment and showing various objects, including: Five videos of between 25 and 100 seconds, captured with Web cams and showing the faces of three individuals moving against office backgrounds. A 60 second video showing a face under severe lighting variation. Two short sequences showing a hand and a planar CD case, respectively. Two videos of passing cars, captured with a camcorder. A one hour-long sequence captured online by a Web cam.

All experiments detailed in this paper were carried out on a desktop PC with a 2.54GHz Intel Pentium IV processor and 512MB of RAM.

5.2 Performance Measures

The two properties of the algorithm we need to measure are speed and accuracy. Speed is measured here as both time taken to track each frame and CPU utilization. This second measure is meaningful when video frames are only available at a maximum rate (e.g., 15 or 25 frames per second) and the tracker leaves processor cycles free.

Accuracy is reported as the root-mean-square (RMS) error between tracking estimates and a ground-truth state. The ground-truth was obtained by hand-labeling each frame with the horizontal and vertical position of the target object. The RMS error is defined as:

$$e = \sqrt{\frac{1}{F}\sum_{f=1}^{F}\sum_{d\in\{u,v\}}(u_{df} - v_{df})^2}, \qquad (22)$$

where F is the number of frames in a sequence. v_{df} is the ground-truth state for degree of freedom d in frame f and u_{df} is the mean of the corresponding estimate made by the displacement expert. In some of the experimental sequences, the displacement expert loses track. On these occasions, the RMS error becomes meaningless (being wrong by 100 pixels is no better or worse than 200 pixels). Therefore, an "adjusted" RMS value is also reported for some experiments. This is found by not including any frames with loss of lock in (22). To make this fair, a second figure is reported stating what proportion of the frames (inliers) were included in this estimate. Inliers are defined as frames for which an exhaustive object detector gives at least one positive hit.

5.3 Displacement Expert

The first set of experiments were conducted on the displacement expert alone. Unless otherwise stated, all experiments were performed *without* the use of a Kalman filter (Algorithm 2) to eliminate any advantages introduced by a motion model. Without the Kalman filter, tracking is performed by updating a point state estimate by the mean of the output of the displacement expert.

5.3.1 Parameter Settings

To train the displacement expert, Algorithm 1 requires at least one labeled seed image along with parameters for the range of displacements to generate training examples from (Δ) and and number of examples to create (N). A third free parameter is the width of the RBFs used in the RVM (κ in (8)). Fig. 8 shows how the RMS tracking error varies with κ and Δ (when trained from a single seed image).

Fig. 9a shows the tracking error and average tracking time per frame against N (including training time). Fig. 9b shows the number of *relevance vectors* ($w_i \neq 0$) retained for a given training set size. In the linear part of this graph, the number of relevance vectors is approximately $0.75N$.

In subsequent experiments, the values $\kappa = 0.8$, $N = 50$ were used. A training range of $\Delta_{u,v} = 15$ pixels was used for translation and $\Delta_s = 0.1$, $\Delta_\theta = 10$ were used for scale and rotation, respectively.

By using a single seed image, the performance of the displacement expert is dependent on what can be learned from that one image. To capture greater invariance to deformation, 3D rotation, lighting, and background, it is favorable to train the expert on examples gathered from

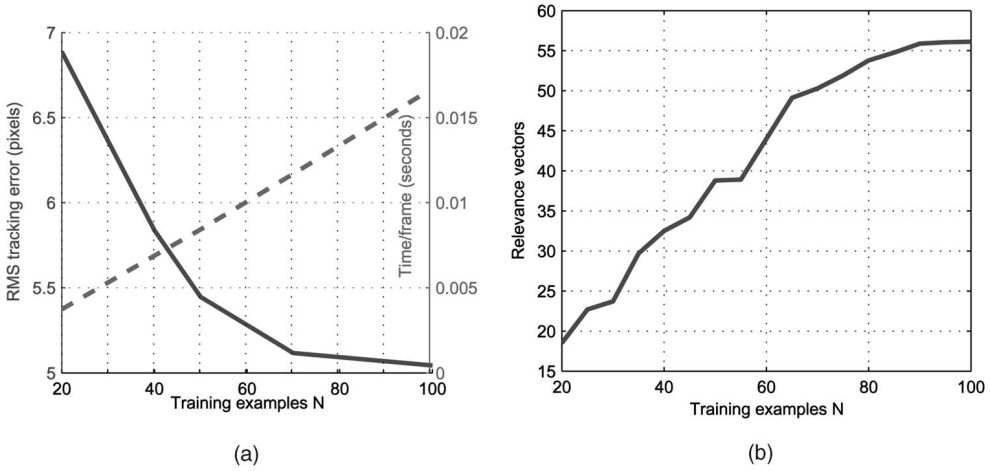


Fig. 9. Effect of training set size N. (a) Tracking accuracy and tracking time per frame (inclusive of training) as a function of training set size N. (b) Average number of relevance vectors against N.

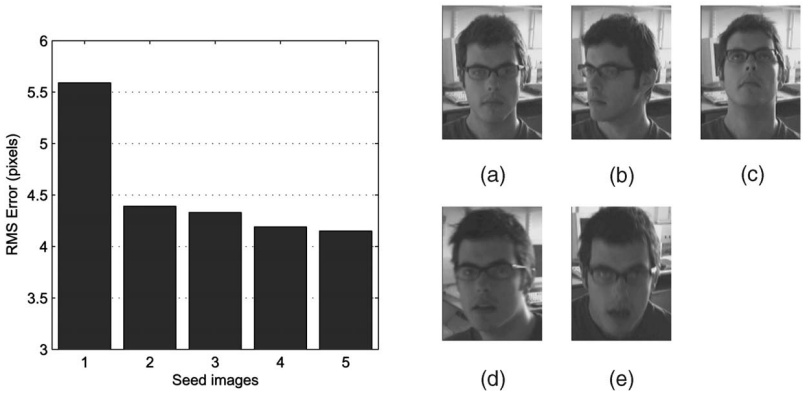


Fig. 10. Benefits of using more than one seed image. This plot shows how RMS tracking error decreases as additional seed images are used to create the displacement expert. The images on the right are the five seed images used in this experiment given in the order in which they were used.

several different seed images. Fig. 10 shows how accuracy is improved by using more than one seed image.

5.3.2 Tolerance to Unmodeled Appearance Changes

The displacement expert, trained to track the Euclidean similarities, does not model three-dimensional rotation (pitch and yaw) and as such must have a certain degree of tolerance to these transformations. When trained with a single seed image (Fig. 10a), the maximum yaw (rotation about vertical axis) tolerated before loss of lock was 37 degrees and the maximum pitch (rotation about horizontal axis) was 53 degrees. When trained on three seed images (Figs. 10a, 10b, and 10c), this performance improves to: maximum yaw, 48 degrees; maximum pitch 61 degrees.[5]

Other important appearance changes that can occur are due to deformation of the object (e.g., facial expression in face tracking), occlusion by some foreign object or a change in lighting conditions. The first two rows of Fig. 11 demonstrate the tolerance of the displacement expert to deformation and partial occlusion (when trained from a single seed image). Due to the use of intensity data as observations, the displacement expert fails under significant changes in lighting. Preliminary experiments have used the output of *steerable filters* [8] to give the orientation of the strongest gradient at each image location. As these features are magnitude invariant, they exhibit excellent tolerance to severe lighting variation when used by the displacement expert (row 3 of Fig. 11).

5.3.3 Effect of Background

While it is possible that the relatively small number of training examples generated by Algorithm 1 are sufficient to capture variation in the appearance of the target object, they are certainly insufficient to capture all possible backgrounds. There are two strategies available to help reduce the effect of background:

1. If the target region is cropped inside the true object boundary, the "background," as registered by the displacement expert, is actually the periphery of the target and will therefore exhibit far less variation than arbitrary background would.
2. Training on multiple seed images showing the target against different backgrounds will introduce some invariance to background. While this still will not generalize all backgrounds, true background rarely occupies more than 25 percent of a sampled region and the remaining pixels can still be used to guide the displacement expert.

5. Figs. 10a, 10b, and 10c only demonstrate pitch and yaw in one direction and in this experiment, the maximum was only measured in this one direction. To improve performance in all directions, more examples would be needed in the opposite sense to Figs. 10b and 10c.

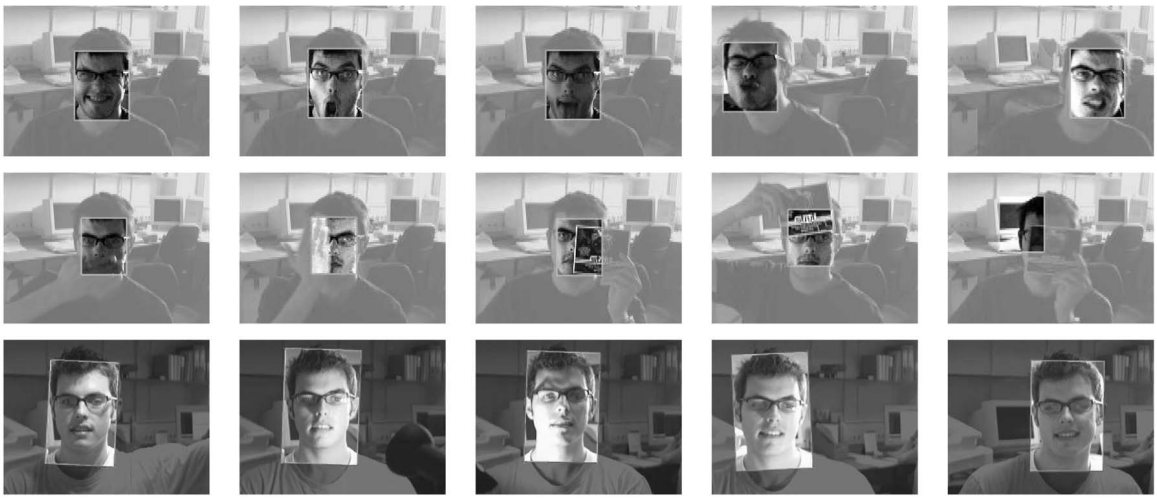


Fig. 11. Tolerance to appearance changes. Rows 1 and 2 show the displacement expert tracking a face undergoing deformation and occlusion. Partial occlusion is tolerated (although small misalignments occur), however, the last occlusion in row 2 is too severe and tracking is lost. A displacement expert using gray-scale features is very sensitive to lighting variation, but one using the orientation from steerable filters [8] is extremely tolerant to it. Row 3 shows the result of tracking such filtered image features under severe lighting variation.

To test these conjectures, two experiments were devised. The first tested how tracking accuracy changes as the target region becomes a smaller fraction of the true object size. The results of this are shown in Fig. 12. The second examines the effect of using different backgrounds in seed images. The results of this are in Fig. 13.

For objects with nonrectangular boundaries, another method to minimize the influence of background on tracking performance is to sample pixels from contoured regions. Fig. 14 shows clips from a hand tracking experiment in which pixels are sampled from a region described by a closed B-spline curve [4].[6]

As an example of tracking objects with more varying backgrounds than an office, Fig. 15 shows the results of tracking passing cars. Both cars were tracked successfully after training from a single seed image.

5.3.4 Affine Tracking

An alternative method for handling 3D rotation is to model it approximately with an affine transformation. This requires six degrees of freedom in the displacement expert, the results of which are shown in Fig. 16.

5.3.5 Benefits of Temporal Fusion

Fig. 17 shows the advantages of using probabilistic inference. The object being tracked moves progressively more rapidly 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 is an order of magnitude lower. Probabilistic inference has greatly increased both the stability and the robustness of tracking.

5.3.6 Comparison to other Approaches

Many approaches for 2D region tracking exist in the literature. We compare the displacement expert to two of them here:

1. First, we compare to normalized cross-correlation [17] as this is a well established and familiar approach and will serve as a benchmark. This was implemented by updating the state estimate to the location having the highest correlation score with an initial template. This search was performed in a region around the previous state estimate. The size of search region was chosen as the smallest that enabled the algorithm to track the entire sequence without failure.
2. Second, we compare the displacement expert to the performance of the *WSL* tracker of Jepson et al. [14]. This is an adaptive approach and as such is reported to have excellent tolerance to appearance changes as well as good accuracy.

The three methods were compared for both accuracy (RMS error) and efficiency (time per frame). The results are recorded in Fig. 18.

5.4 Complete System

Section 4 explains how a validation/restart mechanism can be used in conjunction with the displacement expert to provide long-term reliability. Recently, work such as [28] has provided highly efficient face detection algorithms and for face tracking applications of this system FloatBoost [18] was used. Experiments were carried out to measure both the CPU demand and accuracy of FloatBoost alone[7] and the hybrid tracker of Section 4 using FloatBoost as the validator and detector. Fig. 19 shows how the accuracy of frame-independent search by FloatBoost compared to those made by a hybrid system with the displacement expert. The sequences used in this experiment all include the target object becoming totally occluded for a short time. To calculate the adjusted RMS values, inliers were counted as those frames for which

6. Note that this does not model the fact that a hand is not a rigid body but is merely illustrating the use of contoured boundaries.

7. As a fairer comparison to the displacement expert, a version of the FloatBoost detector was used that is optimized for finding only one face in an image. By using a heuristic search pattern and halting after one face is found, this version is more efficient than exhaustive full-frame search.

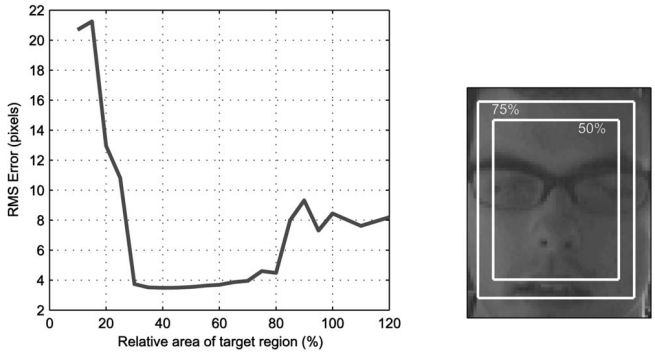


Fig. 12. Effect of cropping. By setting the target region to be inside the true target, the displacement expert is less dependent on viewing the object against one particular background. In this example, the 100 percent area is of width 59 pixels and height 74 pixels.

the boosting algorithm supplied an estimate (the boosting RMS and adjusted RMS columns were therefore computed on the same number of data).

Fig. 20 shows the mean CPU utilization of each algorithm as well as the "steady-state" CPU usage during tracking alone (i.e., ignoring exhaustive searches and training).

5.4.1 Long-Term Performance

The aim of combining an object detector with a tracker is a system that is efficient and can recover from failure of the tracking component yielding long-term (indefinite) reliability. Fig. 21 shows snapshots from a 60 minute face tracking sequence during which the target (observed by a Web cam) carries on working as normal, including leaving and returning to the scene several times. The tracker was trained from three seed images and the mean CPU utilization for this period, including online learning of the displacement expert was 32 percent. Of all the occasions that the system changed from tracking to restart mode, approximately 12 percent were due to tracker failure rather than the target simply leaving the field of view.

Fig. 14. Hand tracking. When tracking a nonrectangular object pixels can be sampled from an arbitrarily contoured region to exclude as much background as possible.

6 Discussion

Fig. 8 shows the sensitivity of tracking error to the input parameters of the displacement expert training algorithm. The error bars of Fig. 8b give insight into the behavior of the expert trained on small displacement ranges Δ: Good tracking accuracy is possible as the expert has been trained on many examples very close to the target object and predictions within this local region are very accurate. However, the expert has not been trained on any large displacements and rapid interframe motion will cause loss of lock.

Choosing the size of the training set N (Fig. 9) involves a compromise. Clearly, the more training examples included, the better the tracking performance, however, larger N incurs a larger computational penalty, for both training and tracking. During tracking, the extra burden is imposed by an increased number of relevance vectors (Fig. 9b). This number increases linearly with N up to $N = 80$ examples, after which the number of relevance vectors saturates at about 56. This implies that adding examples above $N = 80$ does not provide any extra information about displacements.

The advantages of training the displacement expert on more than one seed image are illustrated in Fig. 10. It is worth noting that, while the improvement of using two seed images over one is dramatic, subsequent improvements are far smaller. It is possible that for this face-against-office scenario two seed images captures sufficient variability in appearance. This could increase for sequences which show more dramatic variation, for example in lighting conditions (see below).

# backgrounds trained on	Avg. error on host BGs	Avg. error on unseen BGs
1	6.2	17.2
2	5.8	8.1
3	6.1	7.6

Fig. 13. Advantages of using different backgrounds in seed images. (top) Stills showing the three backgrounds used in this experiment. (bottom) Table showing the accuracy of the displacement expert when trained on one or more seed images showing different backgrounds. Results are given for tracking the object on the "host" backgrounds the expert was trained against and on unseen backgrounds.

Fig. 15. 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.

Fig. 16. Tracking with six degrees of freedom. These clips were generated by training an RVM displacement expert to work with six degrees of freedom in the affine similarity space. This approximately models the projection of a planar object undergoing out-of-plane rotations.

Fig. 11 shows that the displacement expert is robust to object deformations and partial occlusions yet, despite the use of histogram equalization (Section 3.1), it is unstable under significant lighting variation. One way to overcome this is to provide multiple seed images showing various illuminations. When being used for real-time tracking, however, the user cooperation required to gather multiple illuminations is unacceptable and an alternative approach is sought. Replacing the raw pixel features with magnitude invariant orientation information dramatically improves stability under illumination changes, however, there is the additional computational cost of filtering each incoming image. This undermines the principal advantage of the displacement expert over some other methods (e.g., WSL [14]), and a topic for future work will be to incorporate this type of invariance in a computationally efficient way.

The comparison to normalized cross-correlation in Fig. 18 shows that the displacement expert achieves greater accuracy at around a third of the computational cost. The WSL algorithm [14] is shown to be significantly more accurate than the displacement expert, however, it takes over 40 times as long to track each frame, exposing the trade-off between accuracy and speed in tracking algorithms.

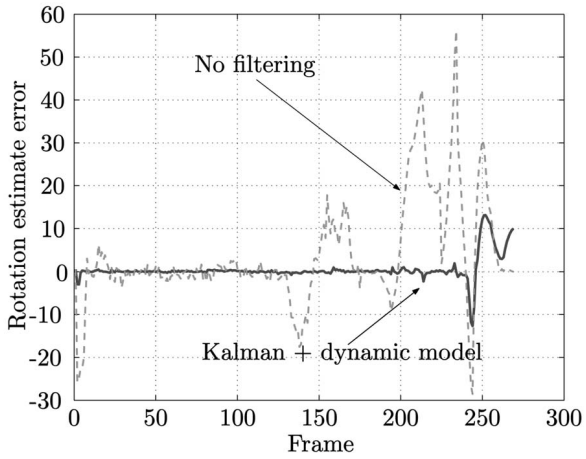


Fig. 17. Probabilistic inference improves performance. This figure shows the error performance of the RVM tracker with Kalman Filter (solid), compared with raw, unfiltered RVM output (dashed).

Fig. 12 demonstrates that setting the target region to be inside the overall object being tracked improves performance by reducing the variation in what the displacement expert sees as "background." If this is taken too far (to below around 50 percent in face tracking), the number of features available to help track the object has been reduced to such a point the accuracy degrades again. For face tracking, the optimal area is around 75 percent of the total object. Fig. 13 shows again that training on more than one seed image improves accuracy: this time for resistance to background appearance.

Figs. 19 and 20 compare the accuracy and efficiency of the displacement expert-detector hybrid with search by FloatBoost [18]. For tracking a single object, the expert shows similar accuracy to boosting (when taking the adjusted RMS value). With respect to CPU usage, the hybrid system takes 75 percent the number of cycles required by boosting for these short test sequences (including training the expert). In other, longer, scenarios with less attempts to deliberately instigate restarts, this gets closer to the 40 percent suggested by the steady-state results. It is worth mentioning, however, that the version of FloatBoost being used is optimized for finding a single face. Using a version with exhaustive search will require more CPU time, but will have the advantage of finding *all* faces present in the image.

The final experiment over 60 minutes proves the hybrid strategy to be viable for tracking an object for extended periods of time while still only using a fraction of available CPU cycles.

Method	RMS error (pixels)	Time/frame (ms)
Displacement Expert	5.6	11.1 (15.2 including training)
Cross-correlation	6.6	38.3
WSL	3.7	675

Fig. 18. Comparison to normalized cross-correlation and WSL tracking. These results were taken from tracking a 200 frame sequence of a head passing in front of the camera at approximately constant depth.

	Boosting	RVM		
Video	RMS	RMS	Adjusted RMS	Inliers
A	9.3	9.3	5.3	88%
B	10.1	17.1	11.3	70%
C	11.8	5.3	9.5	80%
D	22.0	31.4	15.6	78%

Fig. 19. RMS tracking errors. This table shows the RMS error of the two algorithms compared to a ground truth. The adjusted RMS error is calculated by only using as inliers those frames for which the detector scored a hit.

	Mean		Steady
Video	Boosting	RVM	RVM
A	44%	21%	21%
B	43%	45%	16%
C	50%	39%	20%
D	53%	43%	21%

Fig. 20. CPU Usage. Mean and steady-state (during tracking alone) CPU utilization by both algorithms.

7 Summary 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-verifier which serves to both validate observations during normal operation and to search intensively for an object during initialization and recovery. The RVM can be trained from a single object instance (*seed* image) perturbed to generate a training set and training is performed online taking only a few seconds on a desktop PC. Given that estimates are generated from local image observations only, this is a real-time tracker imposing a modest computational load in comparison to approaches that search each frame exhaustively. Thanks to its recovery mechanism it runs continuously. This is a general method for affine tracking and we have demonstrated real-time operation tracking faces, hands, and cars.

As presented, this system is designed to track a single object of interest and future research will address the efficient tracking of multiple objects. Other future work includes:

1. Greater invariance to illumination changes is obtained by preprocessing of images beyond simple intensity normalization; the aim is to incorporate these invariant features with a minimum increase in computational burden.
2. Adaptive retraining in parallel with tracking.
3. Greater variation of view and articulation.
4. Enhanced inference by efficiently modeling dependencies between state space dimensions and using motion trajectories in prediction.

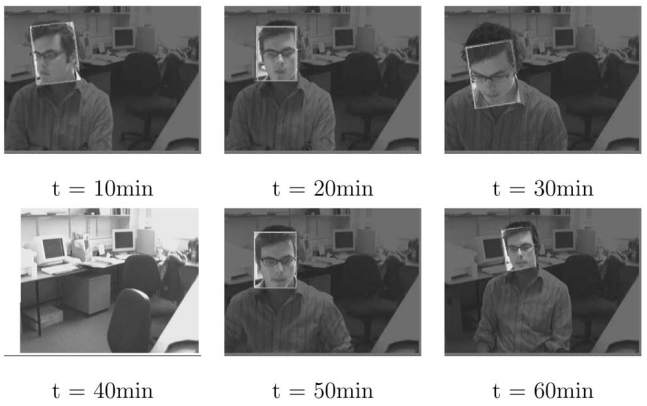


Fig. 21. Long-term tracking behavior. This figure shows tracking continuing robustly over a 60 minute period.

References

[1] S. Avidan, "Support Vector Tracking," *Proc. Conf. Computer Vision and Pattern Recognition,* 2001.
[2] M.J. Black and A.D. Jepson, "Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Proc. European Conf. Computer Vision,* vol. 1, pp. 329-342, 1996.
[3] A. Blake and M. Isard, "3D Position, Attitude and Shape Input Using Video Tracking of Hands and Lips," *Proc. Siggraph,* pp. 185-192, 1994.
[4] A. Blake and M. Isard, *Active Contours.* Springer, 1998.
[5] *Computer Vision for Human-Machine Interaction,* R. Cipolla and A. Pentland, eds. Cambridge Univ. Press, 1998.
[6] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean Shift," *Proc. Conf. Computer Vision and Pattern Recognition,* 2000.
[7] T.F. Cootes, G.J. Edwards, and C.J. Taylor, "Active Appearance Models," *Proc. European Conf. Computer Vision,* pp. 484-498, 1998.
[8] W. Freeman and E. Adelson, "The Design and Use of Steerable Filters," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 9, pp. 891-906, Sept. 1991.
[9] *Applied Optimal Estimation,* A. Gelb, ed. Cambridge, Mass.: MIT Press, 1974.
[10] G.D. Hager and P.N. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 10, pp. 1025-1039, Oct. 1998.
[11] M. Isard and A. Blake, "Contour Tracking by Stochasitc Propagation of Conditional Density," *Proc. European Conf. Computer Vision,* pp. 343-356, 1996.
[12] A.K. Jain, *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.
[13] T. Jebara and A. Pentland, "Parametrized Structure from Motion for 3D Adaptive Feedback Tracking of Faces," *Proc. Conf. Computer Vision and Pattern Recognition,* 1997.
[14] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi, "Robust On-Line Appearance Models for Visual Tracking," *Proc. Conf. Computer Vision and Pattern Recognition,* pp. 415-422, 2001.
[15] F. Jurie and M. Dhome, "Hyperplane Approximation for Template Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 996-1000, July 2002.
[16] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Proc. Int'l Conf. Computer Vision,* pp. 259-268, 1987.
[17] J.P. Lewis, "Fast Normalized Cross-Correlation," *Vision Interface,* 1995.
[18] S.Z. Li, L. Zhu, Z.Q. Zhang, A. Blake, H.J. Zhang, and H. Shum, "Statistical Learning of Multi-View Face Detection," *Proc. European Conf. Computer Vision,* 2002.
[19] D.G. Lowe, "Robust Model-Based Motion Tracking through the Integration of Search and Estimation," *Int'l J. Computer Vision,* vol. 8, no. 2, pp. 113-122, 1992.
[20] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," *Proc. Conf. Computer Vision and Pattern Recognition,* pp. 130-136, 1997.

[21] S. Romdhani, P. Torr, B. Scholköpf, and A. Blake, "Computationally Efficient Face Detection," *Proc. Int'l Conf. Computer Vision,* vol. 2, pp. 524-531, 2001.
[22] B. Scholköpf, C. Burges, and A. Smola, *Advances in Kernel Methods: Support Vector Machines.* Cambridge, Mass.: MIT Press, Dec. 1998.
[23] M.E. Tipping and A.C. Faul, "Fast Marginal Likelihood Maximisation for Sparse Bayesian Models," *Proc. Ninth Int'l Workshop on Artificial Intelligence and Statistics,* C.M. Bishop and B.J. Frey, eds., Jan. 2003.
[24] M.E. Tipping, "The Relevance Vector Machine," *Advances in Neural Information Processing Systems,* S.A. Solla, T.K. Leen, and K.R. Müller, eds., vol. 12, pp. 652-658, 2000.
[25] M.E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *J. Machine Learning Research,* vol. 1, pp. 211-244, 2001.
[26] L. Vacchetti, V. Lepetit, and P. Fua, "Stable 3D Tracking in Real-Time Using Integrated Context Information," *Proc. Conf. Computer Vision and Pattern Recognition,* 2003.
[27] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer Verlag, 1995.
[28] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. Conf. Computer Vision and Pattern Recognition,* 2001.
[29] P Viola, M.J. Jones, and D. Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance," *Proc. Int'l Conf. Computer Vision,* pp. 734-741, Oct. 2003.
[30] A.L. Yuille and P. Hallinan, "Deformable Templates," *Active Vision,* A. Blake and A.L. Yuille, eds. pp. 20-38, MIT, 1992.

Oliver Williams received the MEng/BA degree from the University of Cambridge in 2002. He is completing the PhD degree in the Department of Engineering at the University of Cambridge using Bayesian learning and inference in computer vision.

Andrew Blake has been on the faculty of computer science at the University of Edinburgh and also a Royal Society Research Fellow, until 1987, then at the Department of Engineering Science in the University of Oxford, where he became a professor in 1996, and a Royal Society Senior Research Fellow in 1998. In 1999, he was appointed senior research scientist at Microsoft Research, Cambridge, while continuing as visiting professor at Oxford. His research interests are in computer vision, signal processing, and learning. He has published a number of papers in vision, a book with A. Zisserman (*Visual Reconstruction*, MIT Press), edited *Active Vision* with Alan Yuille (MIT Press), and the book (*Active Contours*, Springer-Verlag) with Michael Isard. He was elected fellow of the Royal Academy of Engineering in 1998. He is a member of the IEEE and the IEEE Computer Society.

Roberto Cipolla received the BA degree (engineering) from the University of Cambridge in 1984 and the MSE degree (electrical engineering) from the University of Pennsylvania in 1985. In 1991, he was awarded the D.Phil. degree (computer vision) from the University of Oxford. His research interests are in computer vision and robotics and include the recovery of motion and 3D shape of visible surfaces from image sequences, visual tracking and navigation, robot hand-eye coordination, algebraic and geometric invariants for object recognition and perceptual grouping, novel man-machine interfaces using visual gestures, and visual inspection. He has authored three books, edited six volumes, and coauthored more than 200 papers. He is a member of the IEEE and the IEEE Computer Society.