The original publication is available at www.springerlink.com DOI 10.1007/s00138-007-0087-x
Machine Vision and Applications

W.W. Mayol · D.W. Murray

Tracking with General Regression

Received: 23 November 2005 / Accepted: 27 February 2007

Abstract A method is developed to track planar and near-planar objects by incorporating a model of the expected image template distortion, and fitting the sampling region to pre-trained examples with general regression. The approach does not assume a particular form of the underlying space, allows a natural handling of occluding objects, and permits dynamic changes of the scale and size of the sampled region. The implementation of the algorithm runs comfortably in modest hardware at video-rate.

Keywords Real-time tracking · non-parametric regression · kernel methods

1 Introduction

Tracking is a core competence in computer vision, underpinning many visual applications. Amongst the wide variety of approaches adopted, statistical methods, and in particular kernel techniques, have received particular attention recently, as they offer a way to accommodate non-linearities such as occlusions and perspective changes that make tracking difficult. The robustness against fast motions and pose changes achieved with kernel methods is evident in the work of, for example, Comaniciu et al. [1], Perez et al. [2], and Yang et al. [3].

When the interest is in recovering not only object position but also projective parameters, linear approximations such as least squares have been widely used, e.g. [4–6]. For planar objects, this has resulted in a number of highly practical and transferable strategies for handling different motion models and for generating training sets. In this paper however, we do not restrict the method to the linear approximation. Instead we pose the problem of recovering motion parameters as a general regression problem [7,8]. The result is a simple and modular method that uses template matching and learning of expected distortions to recover anisotropic scale and orientation in real-time, along with, of course, object position.

Research supported by Grants GR/N03266 and GR/S97774 from the UK Engineering and Physical Science Research Council, and by a Mexican CONACYT scholarship to WWM.

W.W. Mayol (Corresponding Author)
Department of Computer Science, University of Bristol, Woodland Road BS8 1UB, UK
E-mail: wmayol@cs.bris.ac.uk

D.W. Murray
Department of Engineering Science, University of Oxford, Parks Road OX1 3PJ, UK

2 Problem formulation and the linear approach

Let $\mathtt{I}_0$ be the image in which tracking is initialised, and let $\mathtt{T}_0 = [\mathbf{x}_0, \mathbf{y}_0, \mathbf{c}_0]$ be a template composed of n samples of $\mathtt{I}_0$. Each element $\mathtt{T}_{0i} = [x_{0i}, y_{0i}, c_{0i}]$ of the template comprises the image coordinates $[x_{0i}, y_{0i}]$ and a pixel attribute c_{0i} (we use grey values, but colour could be used) at that location. The problem of tracking $\mathtt{T}_0$ through a sequence of images can be regarded as one of finding the vector of free parameters $\boldsymbol{\mu}$ of a 2D motion model $\Phi(\boldsymbol{\mu})$ that minimises some measure of the difference between the original template and distorted template $\mathtt{T} = [\mathbf{x}, \mathbf{y}, \mathbf{c}]$

$$\boldsymbol{\mu} = \arg\min_{\boldsymbol{\mu}'} ||\mathtt{T}_0 - \mathtt{T}(\Phi(\boldsymbol{\mu}'), \mathtt{I})|| \,. \tag{1}$$

The motion model $\Phi(\boldsymbol{\mu})$ might range from a 2D translation with two parameters, to a collineation with eight.

In the absence of a deterministic algorithm, some form of numerical optimisation is typically used to find $\boldsymbol{\mu}$. Here, however, instead of placing effort in the search for $\boldsymbol{\mu}$ during tracking, effort is invested in training the process before tracking starts, so that template differences automatically generate good $\boldsymbol{\mu}$'s. One aim is, of course, to reduce the number of iterations during the search for the optimum.

Amongst linear methods, Gleicher [4] proposed a learning algorithm based on the difference decomposition where a base of optimal vectors is generated before the tracking starts. The aim there was to recover $\boldsymbol{\mu}$ from the difference between the current sample $\mathtt{T}$ and $\mathtt{T}_0$. That system is capable of handling rigid and some non-rigid objects by means of having several joined rigid templates accommodated in a

tessellation. Recent variations on this approach are those of Jurie and Dhôme [5] and Cootes et al. [9] who also used linear regression to estimate an interaction matrix (denoted $\mathtt{A}$ below).

The linear approach provides a useful background, and the formulation presented in [5] is reviewed here.

Recall that the colours at pixels $[x_{0i}, y_{0i}]$, $i = 1 \dots n$ in the original template are $\mathbf{c}_0 = [c_{01} \dots c_{0n}]$ at n sites. Let the original image $\mathtt{I}_0$ now be warped by some function $\boldsymbol{\Phi}(\boldsymbol{\mu}_j)$ where j is the index of the training sample, and let ψ represent a scheme for interpolating the colour c_{ji} that would be observed at the original position $[x_{0i}, y_{0i}]$. That is,

$$c_{ji} = \psi(\boldsymbol{\Phi}(\boldsymbol{\mu}_j), \mathtt{I}_0, x_{0i}, y_{0i}) \; . \tag{2}$$

Let these be held in a vector $\mathbf{c}_j$. Now assume that the changes in colour $\Delta\mathbf{c}_j = (\mathbf{c}_j - \mathbf{c}_0)$ can be represented in terms of the model parameters as

$$\Delta\mathbf{c}_j \; \mathtt{A} = \left[\Delta c_{j1} \; \Delta c_{j2} \; \dots \; \Delta c_{jn} \right] \mathtt{A} = \boldsymbol{\mu}_j{}^{\top} \; , \tag{3}$$

where, if $\boldsymbol{\mu}$ is of length p, the interaction matrix $\mathtt{A}$ has dimensions $n \times p$. Suppose now that $j = 1 \dots m$ parameter vectors $\boldsymbol{\mu}_j$ are generated, and for each one the image $\mathtt{I}_0$ is warped and the colour difference vectors measured. Then

$$\mathtt{M} \; \mathtt{A} = \begin{bmatrix} \Delta\mathbf{c}_1 \\ \Delta\mathbf{c}_2 \\ \vdots \\ \Delta\mathbf{c}_m \end{bmatrix} \mathtt{A} = [\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_m]^{\top} \; , \tag{4}$$

and $\mathtt{A}$ can be estimated in a least squares sense via the Moore-Penrose pseudo-inverse

$$\tilde{\mathtt{A}} = \left[\mathtt{M}^{\top}\mathtt{M} \right]^{-1} \mathtt{M}^{\top} \left[\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_m \right]^{\top} \; . \tag{5}$$

With $\tilde{\mathtt{A}}$ pre-learned in this way, given an input colour difference vector $\Delta\mathbf{c}$ between the template in image $\mathtt{I}_0$ and the template at the same position in some new image $\mathtt{I}$, the motion parameters for that new image can be estimated as

$$\tilde{\boldsymbol{\mu}}^{\top} = \Delta\mathbf{c} \; \tilde{\mathtt{A}} \; . \tag{6}$$

During the training stage, vectors $\boldsymbol{\mu}_j$ in Eq. 4 may be generated at random either within the entire space of motion parameters, or within a restricted space to incorporate some prior about the expected distortion. For example, Jurie and Dhôme [5] considered a restricted 5-dof specialised affine distortion, modelled with anisotropic scale, rotation around the template centre and two translations, so that

$$\boldsymbol{\mu}_j = [s_x, s_y, \theta, t_x, t_y]_j \; . \tag{7}$$

3 General regression and tracking

Tracking an image region using the above method is certainly fast, as it requires only n subtractions to find $\Delta\mathbf{c}$, where n is typically a few hundred, followed by multiplication with a $n \times p$ matrix, where p is five. However, the method comes with some impracticalities and risks. First, it is a linear approximation, requiring further strategies (some of which are suggested in [6]) to handle even the smallest occlusion. Secondly, it involves a matrix inversion which can have problems of stability as row elements can potentially be equal. Finally, the method requires that matrix $\mathtt{M}$ has $m > n$, meaning that the training phase requires a significant number of examples before any tracking can take place.

Estimating $\boldsymbol{\Phi}$ in Eq. 1 can be seen as a regression problem given a set of samples $\mathbf{c}$ and their corresponding codomain values $\boldsymbol{\mu}$. The problem of estimating from sparse samples is well know in the statistics literature, and there are plenty of methods to do so. In this particular case, we are interested in regression that is not preconceived in its form, often called non-parametric or general regression.

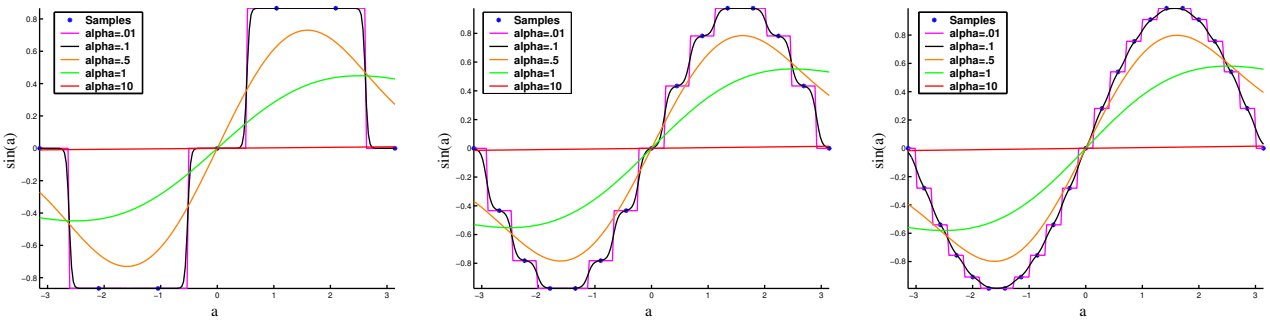
One of the best known methods for general regression estimation is the Nadaraya-Watson estimator [7,8] which belongs to the family of Parzen kernel methods. Given a sample of independent observations $[x_1, y_1] \dots [x_m, y_m]$, the approximation to $y(x)$ takes the form

$$\tilde{y}(x) = \frac{\sum_{i=1}^{m} y_i \, K\left((x - x_i)/h_m\right)}{\sum_{i=1}^{m} K\left((x - x_i)/h_m\right)} \; . \tag{8}$$

Each kernel function K has the centre of its profile located at a sample x_i and its influence at neighbouring locations x is weighted by distance scaled by the bandwidth h_m. The underlying function is assumed continuous, and convergence is guaranteed as $m \to \infty$ [7,10] provided (i) the kernel satisfies $K(x) < C < \infty$, for some positive constant C, and $\lim_{x \to \pm\infty} |xK(x)| = 0$; and (ii) the bandwidth obeys $\lim_{m \to \infty} h_m \to 0$. Within these criteria, the actual form of $K(x)$ is usually not critical, as many have observed experimentally (for example, [10,1]), and an ample variety of kernels such as Gaussian, Epanechnikov [11], and triangular, are in common use.

For illustration, Fig. 1 shows the effect of fitting to a sine function using Gaussian kernels, for different numbers of samples and different Gaussian widths. As the number of samples increases the reconstruction is certainly more accurate, but even a limited number of them reduces the mean squared error compared with a linear approximation. Here a linear estimator is equivalent to using kernels with a very large bandwidth, which of course merely reproduces the average value of zero.

To use general regression in the context of visual tracking, we first have to generate m training pairs $\{\mathbf{c}_j, \boldsymbol{\mu}_j\}$. As in the linear case, one could obtain them by warping the original image $\mathtt{I}_0$ according to motion parameters $\boldsymbol{\mu}_j$, so that the c_{ji} are given by Equation 2. As earlier, the motion parame-



(a) (b) (c)

Fig. 1 Examples of the regression of a sine function with the Nadaraya-Watson estimator using sparse samples that increase in number. Gaussian kernels with different variances (σ^2) scaled by α are used. Large widths ($\alpha = 10$) approach a linear estimator, while smaller ones specialise around each sample.

ters $\boldsymbol{\mu}_j$ can be sampled at random either uniformly or taking in account some prior expectation of the motion.

The estimation of $\boldsymbol{\mu}_k$ can now be formulated as

$$\tilde{\boldsymbol{\mu}}_k = \frac{\sum_{j=1}^{m} \boldsymbol{\mu}_j d_{jk}}{\sum_{j=1}^{m} d_{jk}}, \tag{9}$$

where the weighting of $\boldsymbol{\mu}_j$ is profiled as a Gaussian density

$$d_{jk} = \exp\left(-\sum_{i=1}^{n} \frac{D(c_{ki}, c_{ji})}{\alpha \sigma_i^2}\right) \tag{10}$$

where $D(c_{ki}, c_{ji})$ is the Euclidean distance between elements i of current sample $\mathbf{c}_k$ and learned example $\mathbf{c}_j$. Eq. 9 is therefore a normalised weighted sum whose output has a larger contribution from motion parameters $\boldsymbol{\mu}_j$ if the difference between the current image sample $\mathbf{c}_k$ and learned example $\mathbf{c}_j$ is small.

The vector $\boldsymbol{\sigma}$ of standard deviations is computed per column in the matrix containing all samples $\mathbf{c}$. The product of scaling factor α with the variance σ^2 controls the smoothness of the regression — smaller values allowing better modelling of non-linearities and larger ones smoothing out noise. The value of α can be obtained either by prior experiment or by robust techniques such as the hold out method [10] that consists of taking out each one of the samples to test the quality of the regression given a proposed value for $\boldsymbol{\sigma}$. However, for a video-rate implementation, α must be fixed in advance.

Planar template tracking with general regression

Stage 1: Training (once)
1: Capture image $\mathtt{I}_0$ and select template $\mathtt{T}_0 = [\mathbf{x}_0, \mathbf{y}_0, \mathbf{c}_0]$
2: **for** example $j = 1$ to m **do**
3: obtain random sample of expected motion parameters $\boldsymbol{\mu}_j$
4: compute $\mathbf{c}_j$ at warped positions $\boldsymbol{\Phi}^{-1}(\boldsymbol{\mu}_j)[\mathbf{x}_0, \mathbf{y}_0]$
5: **end for**
6: compute standard deviation $\boldsymbol{\sigma}$ per column over all the $\mathbf{c}_j$

Stage 2: Tracking
1: Initialize $[\mathbf{x}_k, \mathbf{y}_k]$=$[\mathbf{x}_0, \mathbf{y}_0]$
2: **while** tracking **do**
3: sample latest image $\mathtt{I}_k$ at $[\mathbf{x}_k, \mathbf{y}_k]$ to obtain colours $\mathbf{c}_k$
4: estimate $\boldsymbol{\mu}_k$ from $\mathbf{c}_k$ using Equation 9
5: warp $[\mathbf{x}_k, \mathbf{y}_k]$ with $\boldsymbol{\Phi}(\tilde{\boldsymbol{\mu}}_k)$ to obtain new $[\mathbf{x}_k, \mathbf{y}_k]$
6: **end while**

Fig. 2 Steps involved for the use of general regression estimation for planar template tracking.

4 Implementation details

Fig. 2 summarises the steps involved in using general regression for planar template tracking. The algorithm has been implemented in C++ to run at video rate from live or captured greyscale imagery, and also in Matlab for off-line analysis.

During capture, the camera's white balance is fixed to prevent additional image fluctuations and the shutter speed kept sufficiently fast to avoid motion blur.

4.1 Training

A region to be tracked is manually selected and a number, typically $m = 1000$, of training examples generated from it using random samples $\boldsymbol{\mu}_j$ of the 5-parameter distortion model of Jurie and Dhôme [5], given earlier in Eq. 7, with standard deviations of $\pm 15\%$ in scale, $\pm 15°$ in rotation, and ± 15 pixels in translation. For a camera with a horizontal field of view of 42°, and capturing at 30 Hz, this corresponds to anticipated maximum angular speed in the horizontal plane of $60°\mathrm{s}^{-1}$ and about the optic axis of $450°\mathrm{s}^{-1}$.

Typically $n = 256$ grey level values c_{ij} are sampled. However, rather than warp the image $\mathtt{I}_0$ and sample at the original positions, the image is left unwarped and the sampling positions warped from template according to

$$[x_{ji}, y_{ji}, 1]^\top = \boldsymbol{\Phi}^{-1}(\boldsymbol{\mu}_j)\,[x_{0i}, y_{0i}, 1]^\top, \tag{11}$$

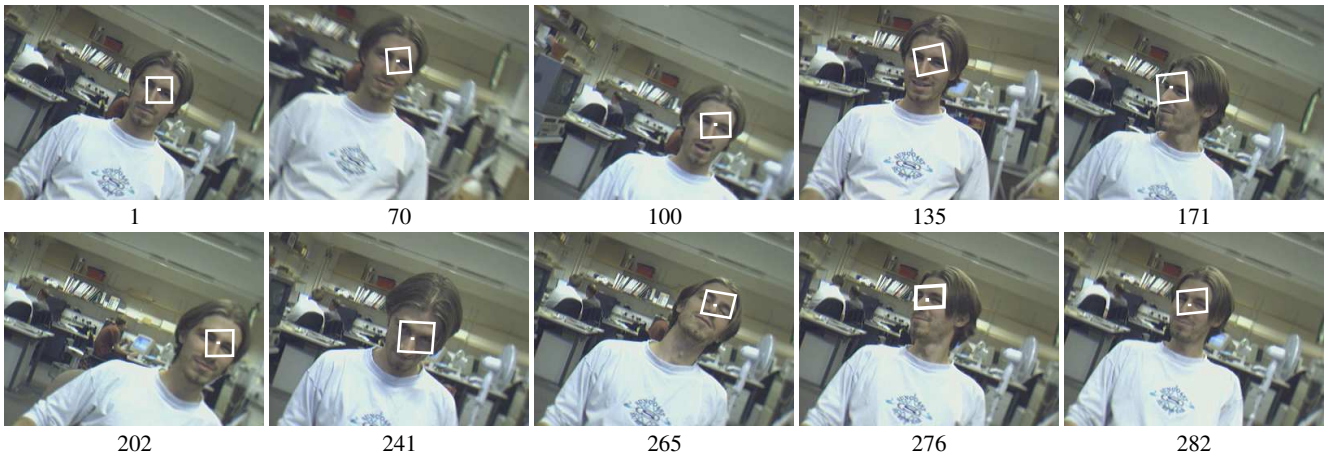


Fig. 3 Tracking an independently moving object from a moving camera. (A video is available from the authors' websites)

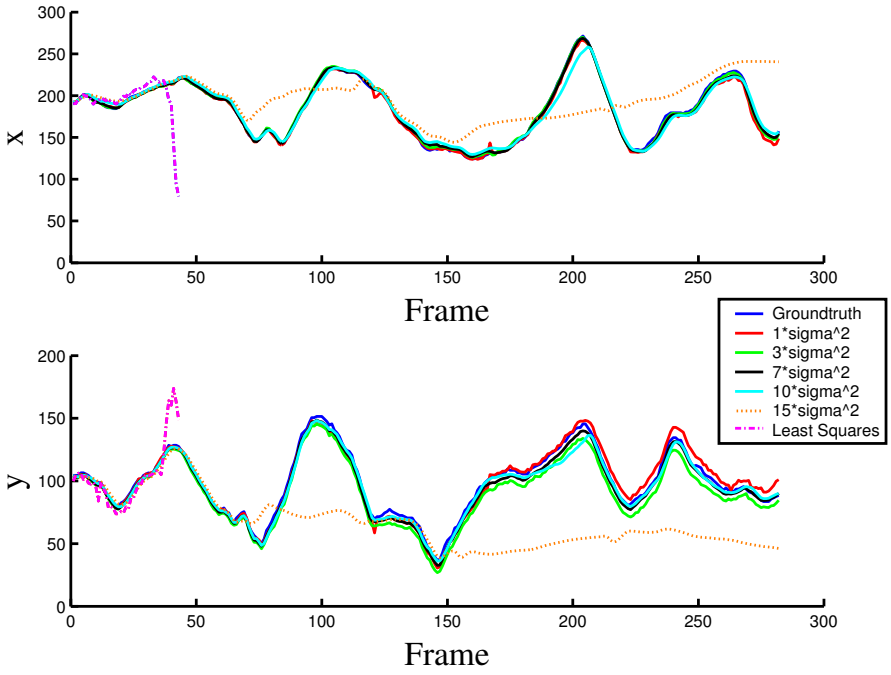


Fig. 4 The x and y coordinates of the centre of the tracked template in Fig. 3, computed for five different values of the scaling factor α, and compared with ground truth. All but the highest scaling value lie close together. The dashed curve, ending at around frame 45 in each graph, records the performance of the least squares approach of Section 2. It breaks soon after the face starts to rotate out of the fronto-parallel plane.

where

$$\Phi(\boldsymbol{\mu}_j) = \begin{bmatrix} \mathtt{R}(\theta_j) & \mathbf{t}_j \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathtt{S}_j$$

and $\mathtt{R}(\theta)$ is a rotation matrix, $\mathbf{t} = [t_x, t_y]^\top$ a translation, and $\mathtt{S} = \mathrm{diag}(s_x, s_y, 1)$ an anisotropic scaling.

The intensity value c_{ji} to be stored is then generated by bi-linear interpolation

$$\begin{aligned} c_{ji} = {} & \mathtt{I}_0(\beta+1, \gamma+1)ab + \mathtt{I}_0(\beta+1, \gamma)(1-b)a \\ & + \mathtt{I}_0(\beta, \gamma+1)b(1-a) + \mathtt{I}_0(\beta, \gamma)(1-a)(1-b)\ , \end{aligned} \tag{12}$$

where

$$\begin{aligned} \beta &= \mathrm{floor}(x_{ji}) \ , \quad a = (x_{ji} - \beta) \\ \gamma &= \mathrm{floor}(y_{ji}) \ , \quad b = (y_{ji} - \gamma)\ . \end{aligned}$$

Finally, to provide a level of compensation against illumination changes, every element of $\mathbf{c}_j$ is normalised according to

$$c_{ji} \leftarrow \frac{c_{ji} - I_{\min}}{I_{\max} - I_{\min}}, \tag{13}$$

where $I_{\max}$ and $I_{\min}$ are the maximum and minimum intensity values in an area around the template's origin in image

Width scaling factor α	Rms error in x (pixel)	Rms error in y (pixel)
1	2.4	3.8
3	1.5	7.0
7	1.8	3.2
10	4.5	3.0
15	23.7	33.2

Table 1 The root mean squared errors in the x and y of the tracked position from ground truth for the experiment of Fig. 4 for different values of scale factor α.

$\mathtt{I}_0$. As explained later in Section 5.1, the value of the scale factor α was determined empirically, and then fixed for all further experiments.

4.2 Tracking

The motion is recovered incrementally between successive pairs of frames. In order to use the regression recovered during training, the positions of the samples must be progressively warped, just as in the linear case [5]. For frames k and $k+1$ grey level samples are interpolated at positions $[\mathbf{x}_k, \mathbf{y}_k]$ and, after estimating the motion $\tilde{\boldsymbol{\mu}}$, the sampling positions are warped linearly ready for the next pair of frames

$$[x, y, 1]^\top_{k+1} = \Phi(\tilde{\boldsymbol{\mu}}_k)[x, y, 1]^\top_k \ .$$

Note that the update Φ is therefore found as an increment, but that increment is computed from changes occuring in the original template.

To increase the robustness against occlusion, two filters are applied in tandem. In the first, if the largest d_{jk} is smaller than a threshold τ, it is assumed that learnt data cannot explain the current sample. In this case, the parameters $\boldsymbol{\mu}$ computed in the previous frame remain unchanged. This has proved to be effective in providing immunity against occluding transient objects like hands and arms moving across the field of view, and allows the tracker to recover when the transient object has passed.

In the second filter, parameters $\boldsymbol{\mu}$ from the previous iteration are also kept if the error between image intensities $\mathbf{c}_0$ at $\mathtt{I}_0$ and proposed sampled points $\mathbf{c}_k$ at the current warp is bigger than the error obtained in the previous cycle. This effectively adds a memory to the otherwise exclusively forward-looking estimation, and has the effect of preventing early degeneration of the solution.

On a 2.8 GHz Pentium IV cpu the algorithm takes 0.5 s to train from 1000 examples at start up, and around 2 ms to estimate $\boldsymbol{\mu}$ for each frame, easily allowing tracking at the camera's 30 Hz framerate. Indeed it would be possible to carry out multiple iterations of Equation 9 (or lines 3-5 of the tracking phase in Fig. 2) for each frame, although in practice we observe the improvement beyond the first iteration to be marginal.

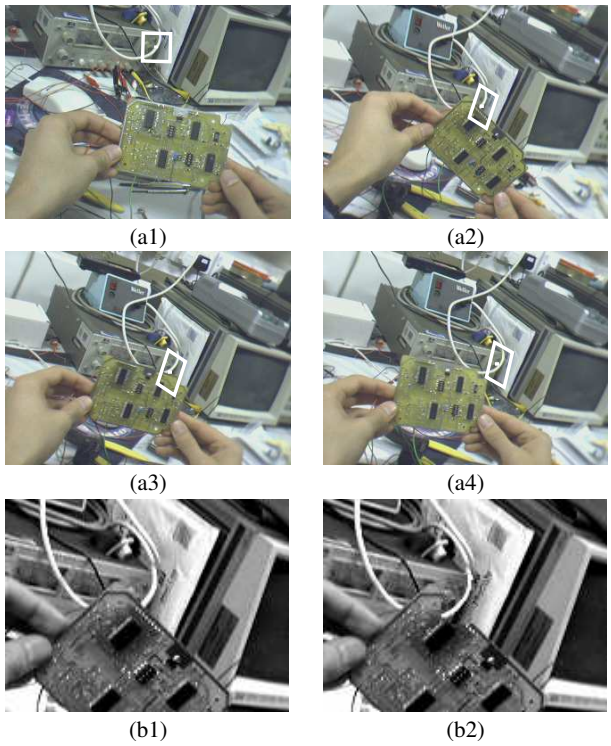


Fig. 5 (a) Tracking where part of the template is occluded in frames (2) and (3). (b1) An enlargement of the tracking region in (a3). (b2) shows the original template warped by $\tilde{\boldsymbol{\mu}}$ and overpainted.

5 Experiments and results

5.1 Tracking a natural object

Fig. 3 illustrates the tracking performance when the scene underlying the template is not strictly planar. The tracked head rotates both within and out of the fronto-parallel plane, and translates laterally and in depth, while the camera, which is been worn by an observer, undergoes modest rotation and translation. Fig. 4 shows the evolution of the image coordinates of the tracked region for different factors α applied to the kernel variance (Eq. 10), and compares them with ground truth obtained by inspection. Also shown on the graphs is the performance of the least squares method of Section 2, Eqns. 4 to 6. Least squares tracking starts to break as the face rotates out of the fronto-parallel plane and background enters the region of interest, and does not recover. These graphs also show that performance is not over-sensitive to the value of α. The r.m.s. pixel errors in the x- and y-directions for the same range of scale factor are shown in Table 1: the value of $\alpha = 7$ is used in all later experiments.

5.2 Tracking under partial and complete occlusion

Fig. 5(a) demonstrates that without further modification, and without the special filters of Section 4, the regression tracker can handle occlusions of a substantial fraction of the tem-

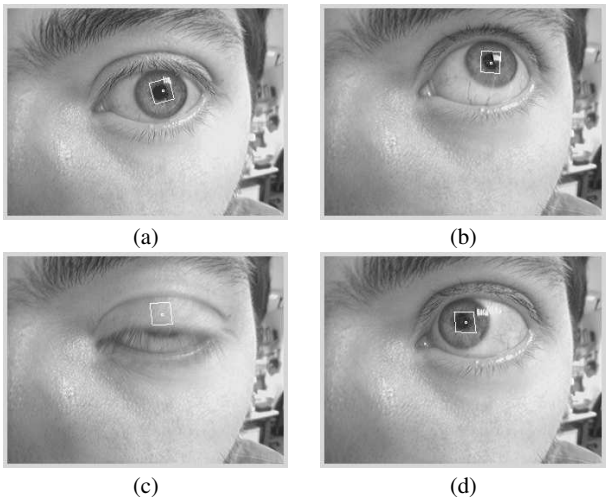


Fig. 6 Tracking under complete occlusion with object motion of about $40°\mathrm{s}^{-1}$. (A video is available from the authors' websites)

Fig. 7 Tracking under significant scale change as the object approaches the camera.

plate. After $\tilde{\boldsymbol{\mu}}_k$ is recovered, it is possible to reconstruct the occluded area of image $\mathtt{I}_k$ by warping the original template according to the current recovered motion. Fig. 5 (b) shows such an example.

An example of the tracker surviving a total occlusion of the target is shown in Fig. 6, where eye movements of some $40°\mathrm{s}^{-1}$ are followed. When the eyelid fully occludes the region of interest, the filter that monitors regression quality through the condition $d_{jk} < \tau$ fires, and the previous frame's motion parameters are preserved allowing successful resumption of tracking after the brief occlusion.

5.3 Tracking under scale change

The tracker's performance under considerable scale change is shown in Fig. 7. The linear dimension of the region increases twofold over a period of 1.5 s.

5.4 Tracking using a highly distorting lens

The algorithm has been found to perform satisfactorily without modification or resetting parameters when moving to cameras of substantially different focal lengths and where the lenses exhibit large distortion. Fig. 8 shows such a case, where a region is tracked through a fish-eye lens mounted on a bipedal robot. The robot walks parallel to the target first to the right and then to the left. As well as handling the extreme distortion at the edge of the images, it is noticeable from the video sequence that the tracker is undisturbed by vibration transmitted through the stiff robot structure.

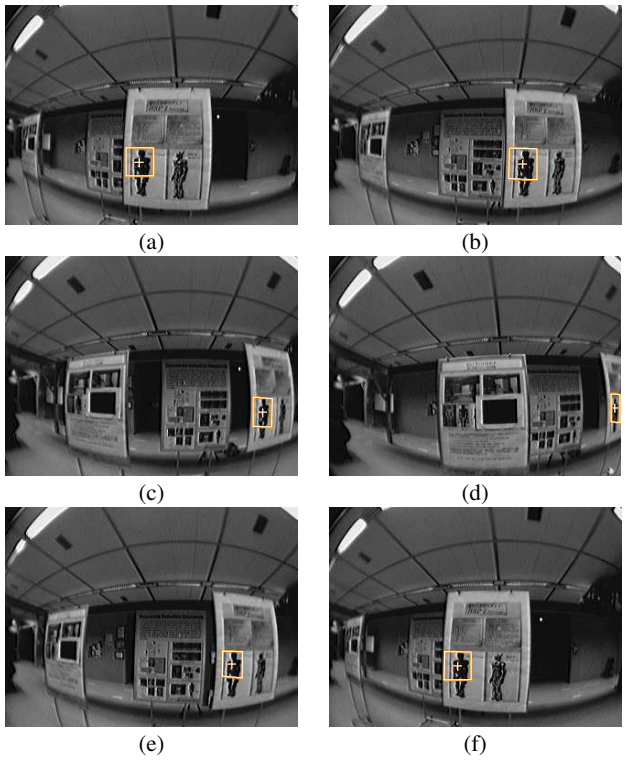


Fig. 8 Tracking with a different focal-length and lens does not require "retuning" of the algorithm. Note both that the tracker handles the extreme radial distortion at the edge of the fish-eye lens and that the region recovers its original shape and size at the end of the sequence. (A video is available from the authors' websites)

6 Comparison with a RVM approach

A superficially similar approach to tracking was taken by Avidan, who combines a support vector machine with an optic flow tracker [12]. However the emphasis of that work was to determine an image warping (actually a 2D translation) that maximizes the SVM classification of a region as being the object of interest, rather than minimizing image differences. Hence the training is on static images of a particular class of objects, in [12], the rears of vehicles. Support vector tracking does however provide an informed method of initializing tracking and, using a pyramid, the ability to handle discontinuous motions, both of which the present method does not.

A more similar learning based approach is that of Williams et al. [13] who employed a relevance vector machine (RVM) [14] to reduce by optimisation the size of the set of kernels required. Fig. 9 compares the performance of the general regression tracker and an RVM tracker using the sequence in Figs. 3 and 4. For this test we used a Matlab version of

the general regression tracker and publicly available Matlab code for the RVM [15], and both recover translation only. Some 200 training examples were generated at random with a standard deviation of ± 15 pixels. The performance of both is similar, but we consistently observe that the RVM's output is noisier than that of the general regression tracker. This characteristic is also reported in [13], and arises because the RVM removes samples after training which is inevitably incomplete.

The noise could be reduced by using a larger and richer training set, but this reveals a limitation of the RVM tracker when considering real-time performance. Whereas the training time for one-shot learning in a general regression tracker increases linearly with the number of samples, selecting relevant vectors uses an iterative optimisation with higher order complexity. This is evident in Fig. 10 which compares the times taken to train the GR and RVM trackers for various sizes of training set.

While the RVM approach could be important when there are memory constraints, its increased training burden appears unacceptable for initialization — particular so if the template needs to be reset on the fly, an essential feature for template-based tracking over longer periods of time [16].

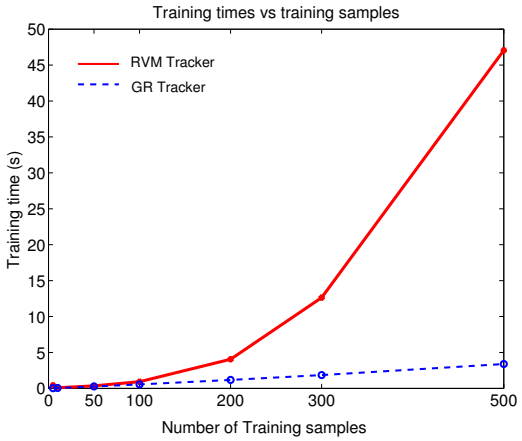


Fig. 10 A comparison of training times for different number of samples for the RVM and general regression trackers. (Note that Matlab implementations were used in both cases, and so the GR time is longer than that for the optimised C++ implementation reported earlier in the paper.)

7 Discussion and conclusions

This paper has posed tracking planar and near-planar objects as a problem of general or non-parametric regression, a well established statistical method to estimate functions from sparse samples. Unlike some other techniques that require training, there is no bias on the form of the regression: it is general and will approach the actual form of the underlying space as the number of examples increases.

The method is straightforward and modular, and would allow a variety of motion models, kernels and metrics to be used, perhaps to incorporate robustness against particular forms of noise or motion. Here, we estimate a specialized affine transform, use Gaussian kernels, and use a simple image evaluation, sampling the image directly according to the previously estimated motion parameters. Experiment has shown that the method runs comfortably at video rate, does not depend critically on the chosen kernel width, and is immediately transferrable to different cameras and lens combinations.

For video rate applications, the time bottleneck in learning based trackers lies in the initial training stage, not in frame by frame computations. For low-dimensional tracking applications using "immediate" pixel templates, our results suggest that spending time eliminating irrelevant samples with an RVM is an investment which is both wasteful and unnecessary.

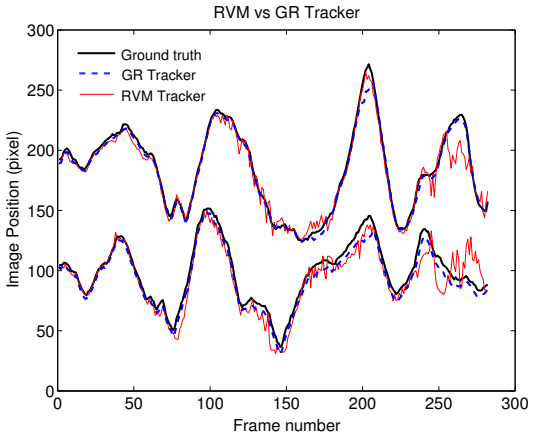


Fig. 9 The performance of the Matlab versions of the "translation-only" GR Tracker and RVM tracker compared with ground truth for the sequence in Fig. 3. The upper and lower curves are for x and y coordinates respectively.

Acknowledgements: The authors express their thanks to Dr N Kita and Dr H Hirukawa from AIST for the use of the HRP robot used in the sequence of Fig. 9, and to Dr B J Tordoff and Dr A J Davison for useful discussions.

References

1. Comaniciu D, Ramesh V, Meer P (2003) Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(5):564–577
2. Perez P, Vermaak J, Blake A (2004) Data fusion for visual tracking with particles. Proceedings of the IEEE 292(3):495–513
3. Yang C, Duraiswami R, Davis L (2005) Efficient mean-shift tracking via a new similarity measure. Proc IEEE Conf on Computer Vision and Pattern Recognition, San Diego CA, June 20-25, 2005 pp 176-183.
4. Gleicher M (1997) Projective registration with difference decomposition. Proc IEEE Conf on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, June 17-19, 1997 pp 331–337
5. Jurie F, Dhôme M (2002) Real time tracking of 3d objects: an efficient and robust approach. Pattern Recognition 35:317–328

6. Jurie F, Dhôme M (2002) Real time robust template matching. Proc British Machine Vision Conference, Cardiff, Sep 2-5, 2002 pp 123–132
7. Nadaraya EA (1964) On estimating regression. Theory of Probability and its Applications 9:141–142
8. Watson GS (1964) Smooth regression analysis. Sankhyā: The Indian Journal of Statistics Series A 26(4):359–372
9. Cootes TF, Edwards GJ, Taylor CJ (1998) Active Appearance Models Proc 5th European Conf on Computer Vision, Freiburg, Germany, June 2-6, 1998 Lecture Notes in Computer Science 1406:484-498, Springer
10. Specht DF (1991) A generalized regression neural network. IEEE Transactions on Neural Networks 2(5):568–576
11. Epanechnikov V (1969) Nonparametric estimates of a multivariate probability density. Theory of Probability and its Applications 14:153–158
12. Avidan S (2004) Support vector tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(8):1064–1072
13. Williams O, Blake A, Cipolla R (2003) A sparse probabilistic learning algorithm for real-time tracking. Proc Int Conf on Computer Vision, Nice, 2003 pp 353–360
14. Tipping ME (2000) The relevance vector machine. In: Solla SA, Leen TK, Müller K-R (eds) Advances in Neural Information Processing Systems 12:652–658
15. Tipping ME (2006) RVM Matlab code from www.miketipping.com
16. Matthews I, Ishikawa T, Baker S (2004) The template update problem. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(6):810–815

Walterio Mayol-Cuevas studied Computer Engineering at the National University of Mexico (UNAM) and graduated in 1999. There, he was founding member of the LINDA research group before moving to the University of Oxford (St Anne's College) to study his doctorate degree which he obtained in 2005. His DPhil thesis investigated wearable active vision. He was appointed Lecturer (Assistant Professor) at the Computer Science Department University of Bristol in September 2004 where he currently researches about his main interests that include active vision, wearable computing and robotics.

David Murray graduated with first class honours in physics from the University of Oxford in 1977 and continued there to complete a doctorate in low-energy nuclear physics in 1980. He was Research Fellow in physics at the California Institute of Technology before joining the General Electric Company's research laboratories in London, where he developed research interests in motion computation, structure from motion and active vision. He moved to the University of Oxford in 1989 as a University Lecturer in Engineering Science and Fellow of St Anne's College, and was made a Professor of Engineering Science in 1997. His research interests continue to centre on active approaches to visual sensing, with applications in surveillance, telerobotics and navigation.