Available online at www.sciencedirect.com

ScienceDirect

Computer Vision and Image Understanding xxx (2007) xxx–xxx

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

A real-time hand tracker using variable-length Markov models of behaviour

Nikolay Stefanov, Aphrodite Galata *, Roger Hubbold

Advanced Interfaces Group, School of Computer Science, University of Manchester, Manchester M13 9PL, UK

Received 20 September 2005; accepted 13 October 2006

Communicated by Mathias Kolsch

Abstract

We present a novel approach for visual tracking of structured behaviour as observed in human–computer interaction. An automatically acquired variable-length Markov model is used to represent the high-level structure and temporal ordering of gestures. Continuous estimation of hand posture is handled by combining the model with annealed particle filtering. The stochastic simulation updates and automatically switches between different model representations of hand posture that correspond to distinct gestures. The implementation executes in real time and demonstrates significant improvement in robustness over comparable methods. We provide a measurement of user performance when our method is applied to a Fitts' law drag-and-drop task, and an analysis of the effects of latency that it introduces.

Keywords: Real-time; Hand tracking; Behaviour modelling; Variable length Markov models; Particle filtering; Human–computer interaction; Gesture recognition; Latency; Fitts' law

1. Introduction

Over the last decade, there has been an increased research interest in unencumbered human–computer interaction. Traditional virtual reality (VR) equipment, such as gloves and 3D mice can be detrimental to users' interaction and immersion in a VR environment. Visual-based hand tracking has been seen as a cost-effective, non-invasive alternative.

However, as early research demonstrated [28,19,27], hand tracking poses several difficult challenges due to the high dimensionality of the search space, fast movements and frequent self-occlusions. Recent research has introduced multiple cameras and better shape detectors [20], or has taken various Bayesian approaches like Kalman filtering [26], tree-based filtering [35] or particle filtering [30] to improve robustness.

The Kalman filter [37] is an optimal solution if the process being estimated can be represented by a Gaussian. Monte Carlo particle filters [12,21] estimate the required densities with a set of discrete weighted samples. They have been applied successfully in the context of computer vision [17,38], and expanded to handle mixed-state particles [18], to avoid local maxima via annealing [8], and to track over discontinuous changes in the observations [14]. An overview of this family of algorithms is given by Arulampalam et al. [2].

To tackle the problem of robustness of hand tracking in the context of human–computer interaction, we exploit the fact that interaction is mainly composed of structured behaviour; in other words, there is a high-level structure governing the hand's movement and temporal ordering of different gestures. Thus, a learnt model of human behaviour can be utilised to help the tracker reduce the

* Corresponding author.
E-mail address: a.galata@cs.man.ac.uk (A. Galata).

1077-3142/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2006.10.017

dimensionality of the search space and provide predictions for discontinuous appearance changes. Since we are concerned with recovering both semantic and manipulative gestures, we require a method that can encode both high- and low-order temporal dependencies. A powerful and efficient method to achieve this is through the use of a variable length Markov model [11].

We present a novel hand tracking algorithm applied to human–computer interaction (Fig. 1) by combining particle filtering with annealing and variable length Markov models (VLMMs). Discontinuous changes in the observations are handled by the learned behaviour model which, together with an efficient scheme for particle evaluation, provides real-time performance and robust tracking of hand postures and several different gestures. The underlying behaviour model also works reliably when tracking unstructured motion; in the case of previously observed events, the algorithm falls back to a first-order Markov model, while for unseen events the tracker reverts to regular particle filtering.

The closest related work to our Bayesian tracking framework is the work by Isard and Blake [18] and Heap and Hogg [14], but there are some important differences. We use a high-order temporal model (which we learn automatically) for propagating particles and handling discontinuous changes in hand appearances. The previous work uses only a first order dynamic model which can be seen as a special case of a VLMM (see Section 6 for a comparison). A further difference is our proposed hand representation which makes the evaluation of particles more efficient. Our method also allows for automatic recovery and re-initialisation after tracking failure.

In this work, we have also undertaken an evaluation of our visual tracking system as an input device for a VR system and the effects of latency in user performance. This has received little attention in previous video-based hand tracking research. We evaluate the tracker's usability with a Fitts' law drag-and-drop task. Fitts' law is a widely studied model of hand positioning for target acquisition, and can be applied both to pointing with hands and with input

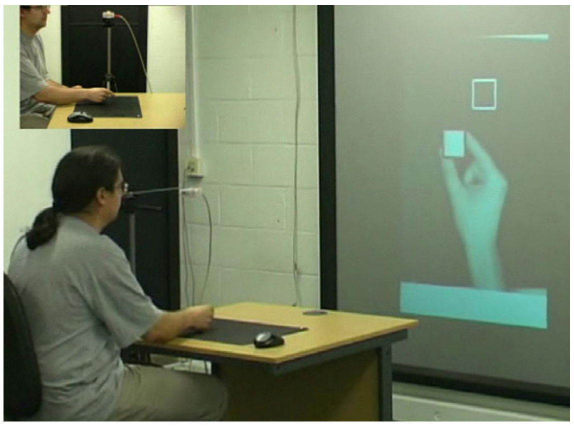
Fig. 1. Manipulation of simple virtual objects with our tracker.

devices such as a mouse. In this paper, we compare user performance in three different scenarios: visual tracking, mouse input with a delay matching the camera latency, and finally non-delayed mouse input.

Section 2 describes the hand model representation and hand feature detection that is used in our system, whilst Section 3 discusses particle filtering as applied to the problem of video-based hand tracking. Section 4 introduces variable-length Markov models, and describes the general approach of combining them with a particle filtering framework, and Section 5 describes the model evaluation part of our system. Section 6 contains test results that show the robustness and accuracy of our proposed algorithm. Section 7 presents the results from user evaluation and latency analysis. Lastly, in Section 8, we comment on the results and discuss future work.

2. Hand representation

2.1. Hand model

Our hand model consists of a hierarchical set of bones $\{B_0,\ldots,B_n\}$, connected by joints $\{J_0,\ldots,J_n\}$. One can think of it as representing a skeletal kinematic structure, where the joints correspond to fingertips, phalange links and palm (see Fig. 2).

A joint J_i has a parent joint J_p, except for the root joint J_0 which is at the top of the skeletal hierarchy. J_i is defined by its joint position P_i, which is a 3D space transformation relative to the parent joint. P_i is composed of a rotational part R_i and a translational part T_i; we use a quaternion and vector representation, respectively. Each joint of the model is assigned a set of predefined rotational constraints, which allow natural postures only. In order to increase flexibility, the translational part T_i of the transformation is also varied, thus effectively allowing us to increase the length of the bones to compensate for scale.

In our system, we represent a hand model with a configuration vector X_k which consists of the ordered set of joints transformation P_i:

$$X_k = \{R_0, R_1, \ldots, R_n, T_0, T_1, \ldots, T_n\} \quad (1)$$

where configuration vectors X_k can be of different dimension depending on the gesture observed in a particular frame (see Fig. 2).

Rotational constraints over the hand joints are specified in terms of an axis of rotation A_i and minimum and maximum angles $\theta_{i,\min}$, $\theta_{i,\max}$. In [23] and [22], the constraints are found by performing PCA over test joint angles gathered with a data glove. In our system, we enforce such constraints by using a quaternion representation of the joint rotations following the approach of Baltman et al. [3]. Additionally, distance constraints, specified as minimal and maximal distances between two joint positions $\delta_{i,\min}$, $\delta_{i,\max}$, are enforced in a similar manner.

The problem of solving several concurrent constraints arises. With a complex kinematic model such as the hand,

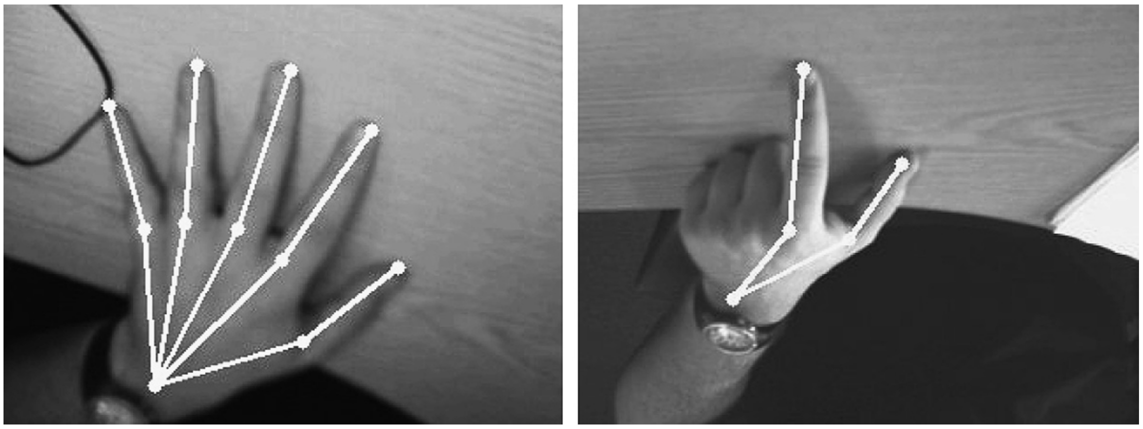

Fig. 2. Two examples of hand models.

it is not possible to find a closed form solution for all constraints at once. The simplest and most effective approach is to use constraint relaxation [3], which amounts to solving one constraint at a time over several iterations. By satisfying the different constraints in sequence, the model converges towards a global configuration that satisfies all constraints at the same time. The number of iterations controls how closely the model satisfies the constraints. A larger number of iterations results in stronger constraint satisfaction, while a smaller one gives a looser fit, but the configuration still tends to stabilise over time.

2.2. Feature detection

In our tracking framework, images are captured using a commodity IEEE 1394 camera, at 320×240 pixels resolution, and frames are transformed from YUV into HSV colour space to make our method less sensitive to lighting conditions. HSV is one of the most commonly used colour spaces for performing skin colour segmentation because it limits the overlap between skin colour and background colour distributions and is relatively invariant to minor illuminant changes [34,31].

Background subtraction is performed using an adaptive Gaussian mixtures algorithm described by Stauffer and Grimson [33]. Hands are detected by a simple thresholding operation on the pixels' hue value; as skin colour is characterised by low hue, thresholding tends to eliminate the effects of moving shadows that cannot be reliably detected by the background subtraction algorithm (see Figs. 3b and c).

Palms and fingertips are detected using a circular Hough transform. Canny edge detection is first applied to the images, followed by three separate Hough transforms. Two of the transforms detect fingertips and have small radiuses (typically 10 and 12 pixels for the experiments described in this paper). The other transform detects the palm centre, with a large radius (typically 64 pixels); as the palm does not always generate a circular shape, the result from this last transform is averaged with the centroid of the skin coloured blob in order to make detection of the palm centre more robust (see Fig. 4a). This approach is robust to changes in scale (most commonly observed when the hand moves perpendicularly to the image plane). Experimental results that demonstrate operation at different scales are presented in Section 6.

Simple heuristics are used to detect the wrist and knuckles. The contour of the hand (obtained after background subtraction and skin colour thresholding) is simplified into a single polygon (typically of approximately 20 segments), using the OpenCV software library [16] for contour detection and simplification. We find the wrist by first searching

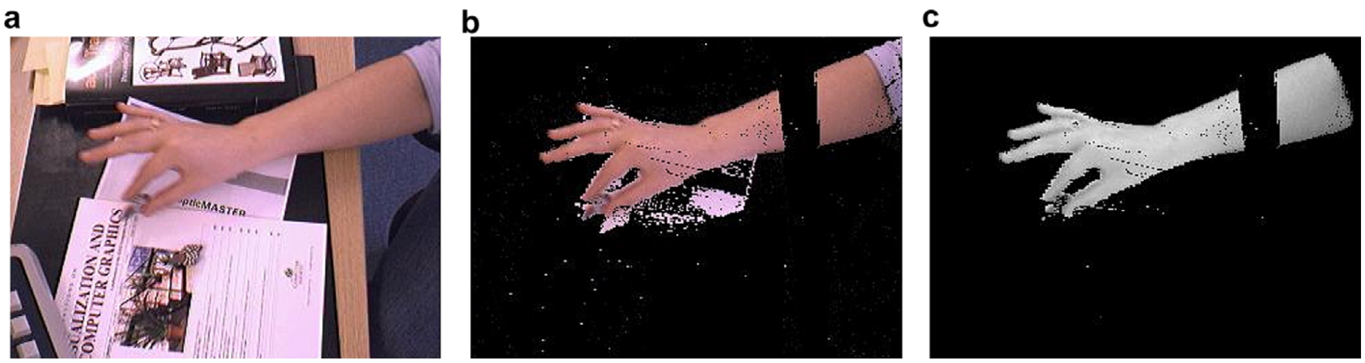

Fig. 3. Image processing: (a) Original image, (b) background subtraction and (c) skin detection.

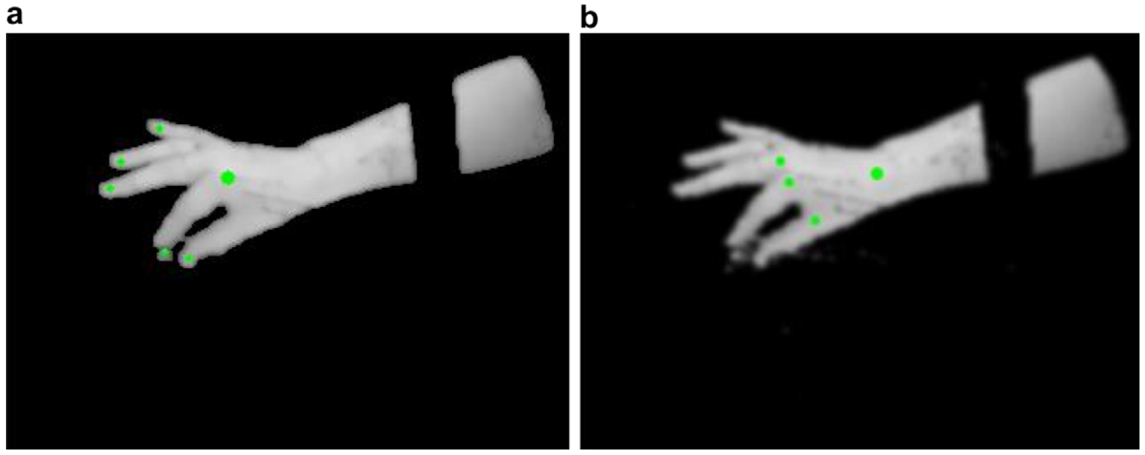

Fig. 4. Feature detection: (a) Hough transform, (b) detecting hand wrist and knuckles.

for the longest parallel segments of the contour and then finding the point which lies midway between the segmented points of shortest distance. The knuckles are found by first identifying the contour segments that correspond to fingertips (using the results from the Hough transform), and then searching for the corresponding finger base, by comparing segment lengths. All the lengths are measured as percentages of the total contour length; this allows hand contours of arbitrary scale to be processed.

Examples of detected features are shown in Fig. 4; as can be seen, features are not always reliably detected—in this case, only three of the knuckles have been found. From our experiments, we have found that this does not seriously affect the reliability and robustness of the tracker since the high-level tracking, which will be described in later sections, is robust to false positives or missing data that might result from this stage of feature detection. However, more experiments are needed to quantify this.

3. Particle filtering

Suppose we have a sequence of image frames and we wish to track a hand through time. Let X_k be a hand's model configuration vector with dimension d, and z_k be the observations for frame k. Configuration vectors of different dimension d can be used in the same sequence, as their length depends on the gesture observed in a particular frame.

A particle filter [2] approximates the posterior density $P(X_k|z_{1:k})$ with a set of weighted particles $\{(x^1w^1), \ldots, (x^Nw^N)\}$. The particle weights $w^i \propto P(z_k|X_k = x^i)$ are normalised so that $\sum w^i = 1$. The estimated state χ_k can be obtained either by the weighted average of the particles $\chi_k = \sum x^i w^i$ or an approximation of the mode $\chi_k = x^j, j = \underset{i=1\ldots N}{\operatorname{argmax}}\, w^i$.

Our approach to particle filtering is partially based on the sampling importance resampling (SIR) algorithm [12], where the particles are drawn from the importance density $P(x_k^i|x_{k-1}^i)$. After evaluating $w^i \propto P(z_k|x_k^i)$, the particles are resampled with replacement from the set with the systematic resampling algorithm [2] which operates in $O(N)$ time.

Often, in order to maintain a fair representation of the density, a large number of particles, N, is required. However, in such a case a major bottleneck is evaluating $p(z_k|x^i)$, or in other words obtaining the weights for the particles. This is due to the fact that for each particle, a non-trivial comparison between the state predicted by the particle and the state inferred by the observations has to be performed. In practice, using a particle filter to solve a problem where x_k has a large dimension can easily become intractable without placing constraints on the object and providing additional enhancements to the basic algorithm, such as observation labelling or hierarchical search. This is caused by the need for a large N, as many particles are needed to cover a reasonable part of the search space.

In this paper, we address two of the pitfalls commonly associated with particle filters: the high computational cost and the lack of recovery process. We also show that robustness and accuracy benefit from the predictions of a learnt high order dynamic model.

4. Tracking in a particle filtering framework using variable length Markov models

Frequently, human–computer interaction conforms to a highly structured behaviour; that is, sequences of primitive movements and gestures that exhibit some high-level structure. We capitalise on this by describing a particular behaviour as a sequence of model configuration vectors $X_0, X_1, \ldots X_n$, where X_i correspond to different hand postures and have varying dimensions. This representation is further simplified by clustering the configuration vectors, and using the descriptions of the corresponding clusters (behavioural prototypes) in place of the individual X_i.

By incorporating probabilistic knowledge of the underlying behavioural structure in the way we sample our particles, we can propagate particles only in plausible directions, and also provide automatic transitions between the different model configurations. Accurate prediction can greatly reduce the number of required particles. A suitable way to obtain such knowledge is a variable-length Markov model (VLMM). The choice of a VLMM as a mathematical framework for modelling complex, non-linear hand motion is based on its ability to capture dependencies at variable temporal scales in a simple and efficient manner.

4.1. Variable length Markov models

Variable length Markov models (VLMMs) deal with a class of random processes in which the memory length varies, in contrast to nth order Markov models. They have been previously used in the data compression [6] and language modelling domains [29,13]. More recently, they have been successfully introduced in the computer vision domain for learning stochastic models of human activities with applications to behaviour recognition and behaviour synthesis [11,10].

In this paper, we extend our previous work and utilise the generative capabilities of variable length Markov models for the purpose of improving the robustness and efficiency of object tracking systems. In particular, we integrate annealed particle filtering with a VLMM in such a way that both continuous hand transformations and a discrete representation of structured behaviour are jointly represented. The VLMM in the particle filter effectively constraints the space of plausible hand postures and hand posture transitions for specific gestures, and also allows for switching between different hand posture models that correspond to different gestures.

The advantage of a VLMM over a fixed memory Markov model is its ability to locally optimise the length of memory required for prediction. This results in a more flexible and efficient representation which is particularly attractive in cases where we need to capture higher-order temporal dependencies in some parts of the behaviour and lower-order dependencies elsewhere.

Assume w is a string of tokens used as a memory to predict the next token a' according to an estimate $\hat{\mathrm{P}}(a'|w)$ of $\mathrm{P}(a'|w)$. The main idea behind the variable length modelling method is that if the output probability $\hat{\mathrm{P}}(a'|aw)$ that predicts the next token a' is significantly different from $\hat{\mathrm{P}}(a'|w)$, then the longer memory aw may be a better predictor than w. A weighted Kullback–Leibler divergence [29] is used to measure the additional information that is gained by using the longer memory aw for prediction instead of the shorter memory w. Since there are cases where the statistical difference is large yet the probability of observing aw itself is small, Ron et al. [29] proposed weighing the KL divergence by the prior probability of observing aw:

$$\Delta H(aw,w) = \hat{\mathrm{P}}(aw)\sum_{a'}\hat{\mathrm{P}}(a'|aw)\log\frac{\hat{\mathrm{P}}(a'|aw)}{\hat{\mathrm{P}}(a'|w)}. \tag{2}$$

If $\Delta H(aw,w)$ exceeds a given threshold ε, then the longer memory aw is used, otherwise the shorter memory w is considered sufficient for prediction.

The transition probabilities and priors are derived from estimates of $\mathrm{P}(a_n|a_1a_2\ldots a_{n-1})$ and $\mathrm{P}(a_1a_2\ldots a_n)$ calculated for various values of n ($n = 1,2,\ldots,N$). The estimates are given by:

$$\hat{P}(a_n|a_1a_2\ldots a_{n-1}) = \frac{v(a_1a_2\ldots a_{n-1}a_n)}{v(a_1a_2\ldots a_{n-1})}, \tag{3}$$

and

$$\hat{P}(a_1a_2\ldots a_n) = \frac{v(a_1a_2\ldots a_n)}{v_0}, \tag{4}$$

where $v(a_1a_2,\ldots,a_n)$ is the number of times the string of tokens $a_1a_2,\ldots,a_n$ appears in the training data and v_0 is the total length of the training sequences.

The training algorithm involves building a prefix tree [13] where each node corresponds to a string up to a predetermined length N. The transition frequencies are counted by traversing the tree structure repeatedly with strings of length N where the strings are generated by sliding a window of fixed length N along a training sequence of tokens. Transition probabilities are computed using Eq. 4 and a pruning procedure is then applied, while the prefix tree is converted to a prediction suffix tree [29]. For each node n_p in the prefix tree, a corresponding node n_s in the suffix tree is created if and only if the Kullback–Leibler divergence between the probability distribution at n_p and the probability distribution at its ancestor node in the prefix tree is larger than threshold ε. Finally, the suffix tree is converted to an automaton representing the trained VLMM. A more detailed description on building and training variable length Markov models is given by Ron et al. [29].

A VLMM can be represented by a probabilistic finite state automaton (PFSA) $\mathcal{M} = (Q,\Sigma,\tau,\gamma,s)$. Σ is a set of tokens that represent the finite alphabet of the VLMM, and Q is a finite set of model states. Each state corresponds to a string in Σ of length at most $N_{\mathcal{M}}$ ($N_{\mathcal{M}} \geqslant 0$), representing the memory for a conditional transition of the VLMM. The transition function τ, the output probability function γ for a particular state, and the probability distribution s over the start states are defined as:

$$\tau : Q\times\Sigma \to Q$$

$$\gamma : Q\times\Sigma \to [0,1], \quad q\in Q, \sum_{a\in\Sigma}\gamma(q,a) = 1$$

$$s : Q \to [0,1], \qquad \sum_{q\in Q} s(q) = 1$$

The VLMM is a generative probabilistic model; by traversing the model's automaton $\mathcal{M}$ we can generate sequences of the tokens in Σ. By using a set of behavioural prototypes as the alphabet, we can capture the temporal ordering and space constraints associated with the primitive movements and gestures. Consequently, traversing $\mathcal{M}$ will generate statistically plausible examples of the behaviour.

4.2. Training the VLMM

Recall that a particular behaviour is represented by a sequence of hand model configuration vectors $X_0, X_1, \ldots X_n$, where X_i corresponds to different hand postures and have varying dimensions. To train the VLMM, we first capture multiple training sequences of a particular behaviour and identify the model configuration X_k observed in each frame k. Thus, we obtain training data sets of model configurations $\{X_k^d\}$, where d is the dimensionality of hand posture X_k; note that these data configurations may belong to different training sequences.

Similarly to previous work (e.g. Toyama and Blake [36]), we use prototypes for visual tracking. In order to get a discrete representation of behaviour, we cluster each $\{X_k^d\}$ to obtain the prototypes $\mathscr{C}^d$. Instead of manually choosing the number of clusters, we used an algorithm by Singer and Warmuth [32] to find a lower estimate of their optimal number; the actual clustering was done with a simple agglomerative scheme.

After obtaining all $\mathscr{C}^d$, we set $\mathscr{C} = \bigcup \mathscr{C}^d$. Next, for each frame k in a particular training sequence, we identify which prototype c_i^d the observed model configuration X_k is closest to. We do so by finding the nearest neighbour of X_k (in Euclidian space) from $\mathscr{C}$. An image sequence is now represented by a sequence in the alphabet of the prototypes $\mathscr{C}$.

These sequences are then used for training the VLMM. The memory size $w_{\mathscr{M}}$ and threshold ε parameters are used to control the actual VLMM construction, depending on the nature of the training data. The choice of values for these parameters can be based on a measure of how well the learned model describes the training data. One such measure, traditionally used in text compression [4] and language modelling [13,15], is the model cross-entropy rate (or model entropy) [7,15] $\hat{\mathrm{H}}_M$.

In our case, given a sequence $S = \mathbf{c}_{r_1} \ldots \mathbf{c}_{r_n}$ of prototype labels of length n, an estimate of model entropy for the learnt VLMM model is given by [10]:

$$\hat{H}_M = -\frac{1}{n}\sum_{i=1}^{n} \log P(\mathbf{c}_{r_i}|q_{r_i}). \quad (5)$$

where $P(\mathbf{c}_{r_i}|q_{r_i}) = \gamma(q_{r_i}, c_{r_i})$.

A measure of how well the learned VLMM describes the training data is given by calculating the model entropy Eq. (5) over the training data. A good approximation of observed behaviour is indicated by a low model entropy value. In the general case, an increase in the value of ε will result in an increase of the model's entropy whereas an increase on the model's maximum memory $w_{\mathscr{M}}$ will decrease the model entropy (however, increasing $w_{\mathscr{M}}$ does not always guarantee lower model entropy, see [13,15] and [10] for details.)

The performance of the model can then be measured by the model entropy over the test data. A model that has achieved a good generalisation of the observed behaviour will have very close model entropy values over both the training and test data. Otherwise, significantly different values indicate a model that is overfitted to the training data and thus more training is required.

4.3. Using the VLMM in a particle filtering framework

VLMMs as described in the previous section operate with discrete tokens, while tracking human hand posture is concerned with estimating a continuous state. Particle filtering offers a way to represent a continuous function with multiple discrete samples. In this section, we describe how a VLMM can handle estimation of a continuous state by combining both approaches. A concise description is provided in Algorithm 1 (see Fig. 5).

We augment the state of each particle x^i so that it contains both the model configuration vector X^i and a PFSA state q^i. For a particular frame k, let X_k and q_k correspond to a particle x_k (in the following, we drop the index i for convenience). We choose the following importance density to update a particle x_k:

$$X_k \sim P(X_k|X_{k-1}, q_{k-1}) \quad (6)$$

$$P(X_k|X_{k-1}, q_{k-1}) \propto P(X_k \mid X_{k-1})\ P(q_{k-1}|X_k) \quad (7)$$

Algorithm 1 VLMM Particle Filtering

```
for $i = 1 : N$ do
    Choose next cluster $c_k^i$ with probability $\gamma\left(q_{k-1}^i, c_k^i\right)$
    Update the VLMM state $q_k^i = \tau\left(q_{k-1}^i, c_k^i\right)$.
end for
for $j = 1 : J$ do
    for $i = 1 : N$ do
        Draw $X_{j,k}^i \sim P\left(X_{j,k}^i|X_{j-1,k}^i, q_{k-1}^i\right)$
        Assign weight for annealing iteration $j$: $w_{j,k}^i \propto p(z_k|X_{j,k}^i)^{\beta_j}$
    end for
    Normalise weights $w_k^i$
    Resample particles using Systematic Resampling [2]
end for
Compute weights $w_k^i \propto p(z_k|X_k^i)$
Compute maximum particle weight $w_{max,k} = max\,(w_k^i)$
if $w_{max,k} < W_T$ then
    for $i = 0 : N$ do
        Reset VLMM state $q_k^i = q_0$
    end for
end if
```

Fig. 5. VLMM Particle Filtering.

The particle's weight w_k is computed as:

$$w_k \propto P(z_k|X_k) \tag{8}$$

Since we apply resampling at each time index k as in the SIR algorithm, a particle's weight at $k-1$ is simply the constant $w_{k-1} = 1/N$ and therefore can be omitted. $P(z_k|X_k)$ is therefore approximated by the set of normalised weights w_k; details on how we evaluate an individual particle's weight are given in Section 5.

Sampling from the importance density is done as follows. We first determine the new VLMM state q_k, first by randomly choosing the next hand model cluster $c_k \in \mathscr{C}$ with probability $\gamma(q_{k-1},c_k)$ and then setting $q_k = \tau(q_{k-1},c_k)$. Note that since the VLMM is trained without removing repeated clusters, it is possible that $c_k \equiv c_{k-1}$. When this is the case, drawing from $P(X_k|X_{k-1})$ is done by updating X_k through the process model. In our system we set $X_k = X_{k-1} + v_k$ where v_k is a random Gaussian variable; we also ensure that X_k remains a plausible example of c_k, by satisfying the constraints imposed by the cluster. When $c_k \neq c_{k-1}$, or in other words a transition between two different configuration clusters occurs, we sample around the new cluster's centre, or in other words $X_k = c_k + v_k$.

This approach was found to work well, as long as the behaviour prototypes are organised in sufficiently fine-grained clusters; therefore, a requirement is that model convergence can happen over a single run of the particle filter. In other words, it must be ensured that no cluster size is so big that a model sampled from the cluster centre cannot converge on a configuration sampled from its boundary over a single run. This is reflected in the algorithm which generates c_k, which requires that the cluster sizes fall below a certain threshold before stopping.

This scheme is trivially extended to an annealing particle filter; we use the same approach to annealing as in Deutscher et al. [8]. With annealing, the particle filter runs multiple iterations over a single frame. A weighting function W^j is used at each iteration j; we implement these by biasing the particle weights $w_i = w_i^{\beta_j}$. The bias factors β_j are ordered in such a way that they cause the particles to gradually migrate toward the global maximum avoiding the problem of local minima. Deutscher et al. provide a more thorough and formal description of the procedure which is beyond the scope of this paper. For a particle at frame k, x_k, q_k is updated only at the first annealing iteration; we then update the hand models as $X^i_{j,k} \sim P(X^i_{j,k} \mid X^i_{j-1,k}, q^i_{k-1})$, where j indicates the annealing iteration and $X^i_{0,k} = X^i_{k-1}$.

The particle filtering process described here effectively generates a statistically plausible sequence of behaviour prototypes $c_i \in \mathscr{C}$. Transitions and temporal ordering between different model configurations and behavioural prototypes are thus automatically provided by the VLMM. Even if a particular model configuration is completely discarded during the resampling stage, the expectation is that in some future frame, the VLMM will re-introduce instances of it in the set. Furthermore, such newly introduced particles correspond only to prototypes that have been observed in the training sequences.

We handle the problem of unseen events with a backing-off scheme as in [11]. If we are presented with a prototype c_k such that $\gamma(q_{k-1},c_k) = 0$ for all particles, then the particles are returned to the initial model state and their memory is deleted. We detect such situations by comparing the maximum particle weight $w_{\max,k} = \max(w^i_k)$ with a predefined threshold W_T. This threshold is determined from a maximum error between a model configuration and the observations for a particular frame, which can be specified before each the start of the tracking process. Details on how the weights w^i_k are determined are presented in the next section.

The augmented particles $x^i_k = (X^i_k, w^i_k, q^i_k)$ form a discrete representation of the belief that a certain prototype c is observed in the current frame k. The most probable prototype $c_{\max}$ has the maximum number of particles $q^i_k \equiv c_{\max}$ in the set. Furthermore, since different particles x^i_k can correspond to the same prototype, we can use the filter to estimate problems in the continuous domain.

5. Observation model

The choice of an appropriate model configuration is governed by two considerations: first, we want to reduce the dimensionality of the configuration vectors, and second, we seek to minimise the time spent evaluating the weights of the particles.

Ideally, we want to process the image data only once per frame, in order to have fast weighting of the particles x^i. Conventionally, the particles x^i are evaluated by projecting their state estimate into image space and computing the difference between contours generated from the projection, and contours observed in the image (e.g. [17]). Clearly, this rapidly becomes unfeasible if we are to maintain a large number of particles N at real-time frame rates.

Our solution to this problem is as follows. In the feature detection stage, we identify fingertips, palms and other parts of the hand, in order to reduce the complexity of evaluating the likelihood of a particular model configuration. Evaluating an estimate of the error of a particular particle is then trivial. Let $\{F_0,\ldots,F_n\}$ be the set of identified feature points, and $\{P_0,\ldots,P_m\}$ be the set of the joint positions of a model configuration X^i projected on the image plane. Both F_i and P_i consist of a 2D point in image space p and a semantic label l, which corresponds to the type of feature that the point represents (fingertip, palm base, etc.). We then compute the weight w^i as

$$1/\sum(p_{P_i} - p_{F_j})^2 \tag{9}$$

where p_{P_i} is the 2D point corresponding to projected joint position P_i and p_{F_j} is the closest 2D point for a detected hand feature F_j, where the semantic labels match $(l_{P_i} = l_{F_j})$. In other words, the estimate of the error of a particular feature is computed by finding the Euclidean distance between the model joint and the closest of the detected features. This operation is computationally inexpensive compared to projecting the model into image space and finding the edge distances. Furthermore, the image data is accessed only once per frame, rather than for every particle. We found that the accuracy provided by this simple algorithm was sufficient for a useful vocabulary of motion and gestures.

6. Tracking results

In this section, we present the results of our tracker's performance and compare them with other similar methods. Cases of both structured and unstructured gestures are considered, and a comparison of tracking without the use of a behaviour model is provided.

For the experiments presented in this paper, the VLMMs were trained using a maximum memory length $w_{\mathscr{M}} = 4$ and threshold $\varepsilon = 0.00001$. A lower estimate for the number of prototypes used for training the behaviour model was determined using the algorithm described by Singer and Warmuth [32] (see Section 4.2). In order to ensure that particle models stay within the cluster bounds, an oriented bounding box for each cluster was also computed. At runtime, the weight of a particle is decreased if the particle falls outside the bounds of the cluster it was sampled from.

For all experiments, the estimate of the tracker was computed as follows. For each frame in a test sequence, the hand posture was obtained by calculating the weighted sum of each of the models in the particle filter's sample set. For each particle, the predicted feature points were computed and then scaled with the particle's weight w_i; the sum of all scaled feature points yielded the estimate of the filter. The error of the filter was computed as the sum of the distances between the predicted model and the ground truth, divided by the number of bones in the model. Effectively, the tests measured the error as the Euclidian distance for each joint of the model that represents the tracker's estimate.

Ground truth data was obtained by manually annotating the input images and assigning labels to the different hand features that are detected by the tracker. For each experiment, the mean and standard deviation of the error is reported. The results are averaged over 50 runs of the particle filter.

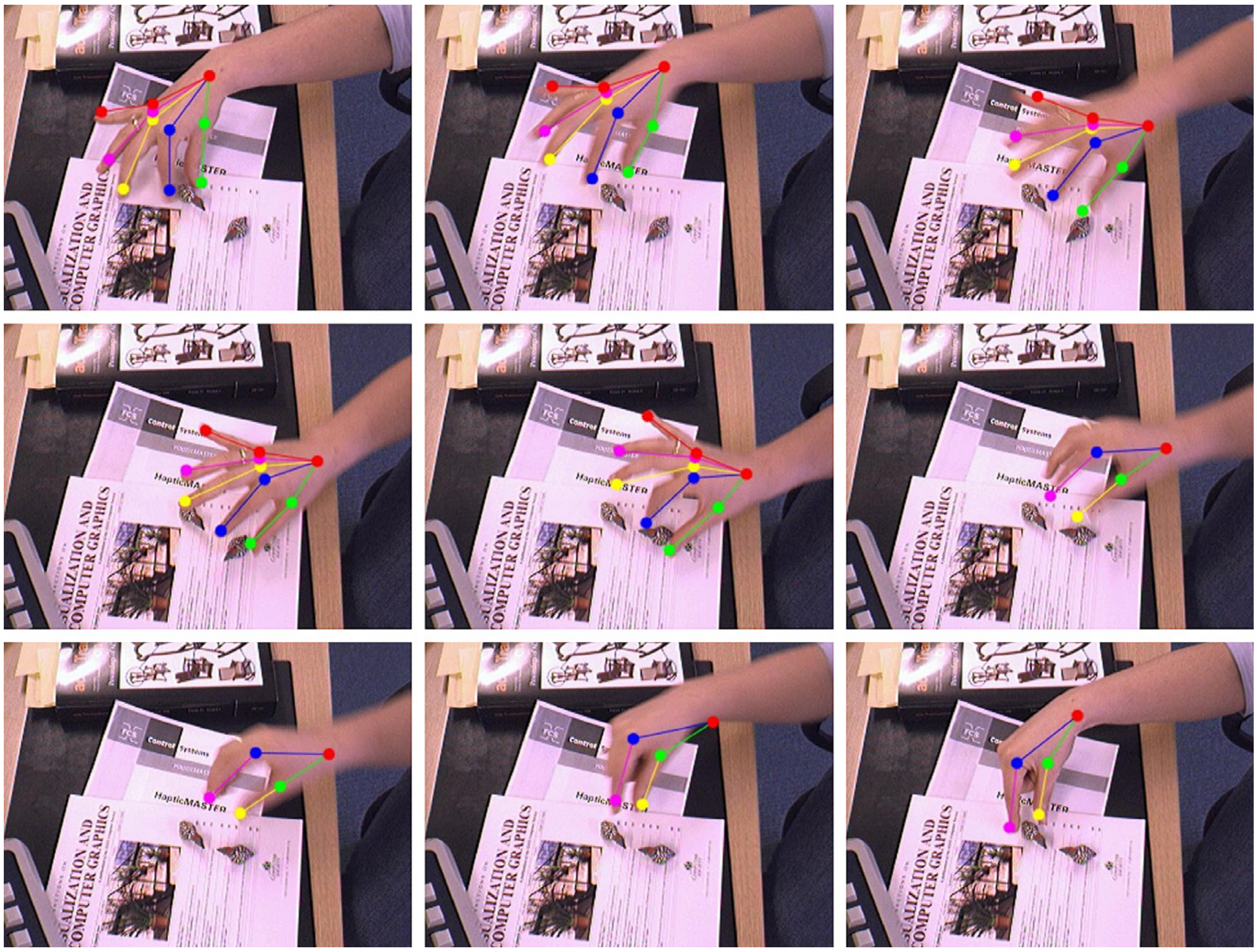

Fig. 6. Tracking results demonstrating the handling of fast motion and frequent gesture changes. Even though the video sequence exhibits a lot of fast motion, tracking is successful due to the model's knowledge of the underlying behaviour.

The error in the ground truth data may be as much as 6 pixels in the worst case (approximately one-third of the width of a finger), giving a maximum error in the reported results in the order of 8 or 9 pixels, or about 7.5 mm. It should be noted, however, that in interactive applications (such as the Fitts' law experiment) users automatically compensate for such errors by moving their hand—the set-up is a closed loop feedback, so that small errors are not noticeable at all to users, provided that the jitter is low.

6.1. Qualitative and quantitative results

This section presents qualitative and quantitative results of tracker performance. In particular, we assess tracker performance in handling fast motion, discontinuous changes in shape appearance, and arbitrary rotation and scale of the hand.

Fig. 6 demonstrates tracking using a VLMM, given a video sequence that exhibits fast motion and frequent changes of gestures. The test video sequence consists of 988 frames; training was done on five separate, smaller video sequences, using the same person. Thirty-two clusters were used for training the VLMM. There were four different models in total: an open hand (comprised of 11 model bones), grabbing gesture (six bones), pointing (three bones) and a closed hand (only the centre of palm was tracked).

The performance of the tracker was evaluated by varying the particle set size (10 different particle set sizes, between 100 and 1000). Fig. 7 shows the average error values for the whole sequence. The reported results are averaged over 50 runs of the particle filter. As we can see, the error steadily decreases as more particles are added.

To investigate the accuracy and robustness of the tracker given a video sequence that involves arbitrary rotation and scaling, a test video sequence of 950 frames exhibiting 3D rotation and significant scale variation of the hand was used (see Fig. 8). Five smaller sequences of 100 frames each were manually annotated to obtain training data for the behaviour model. Automatic clustering of the training data generated 31 separate hand model clusters. As in the previous experiment, there were four different types of hand models: open, closed, grasping and pointing hand.

Fig. 9 shows the average error per model joint and its standard deviation for the entire sequence, given different numbers of particles. The reported results are averaged over 50 runs of the particle filter. As can be seen from the data, the proposed tracker performed robustly, maintaining a low error variation throughout the test sequence. The accuracy, although worse compared to that exhibited in the previous test, is still quite low at approximately 2.5 pixels per bone. Again, the change in accuracy starts to decrease more gradually with particle set sizes of above 300.

The feature detection algorithm also managed to provide accurate observations in over 80% of the frames where the scale of the hand in the images was significantly greater than at the beginning of the sequence. In the rest of the frames, the high-level tracking managed to continue without sufficient evidence in terms of detected feature points, because the search was guided by the VLMM.

As can be seen from this experiment, the proposed algorithm can successfully cope with sudden changes in appearance and variable scale. This is due to the combination of flexible feature detection and robust behaviour modeling, which learns discontinuous model transitions and compensates in the cases where no reliable evidence can be obtained. Fig. 8 demonstrates handling of 3D rotation and Fig. 9 shows the corresponding errors.

6.2. Tracking with and without a behaviour model

To evaluate the effect of the learnt behaviour model when tracking a single gesture, a short section from the previous test sequence was used, exhibiting an example of structured motion. The sequence was first tracked without using a behaviour model (Fig. 10). As we can see, the per-joint error is quite high, averaging nearly three pixels. Increasing the particles beyond 500 provided little improvement to accuracy.

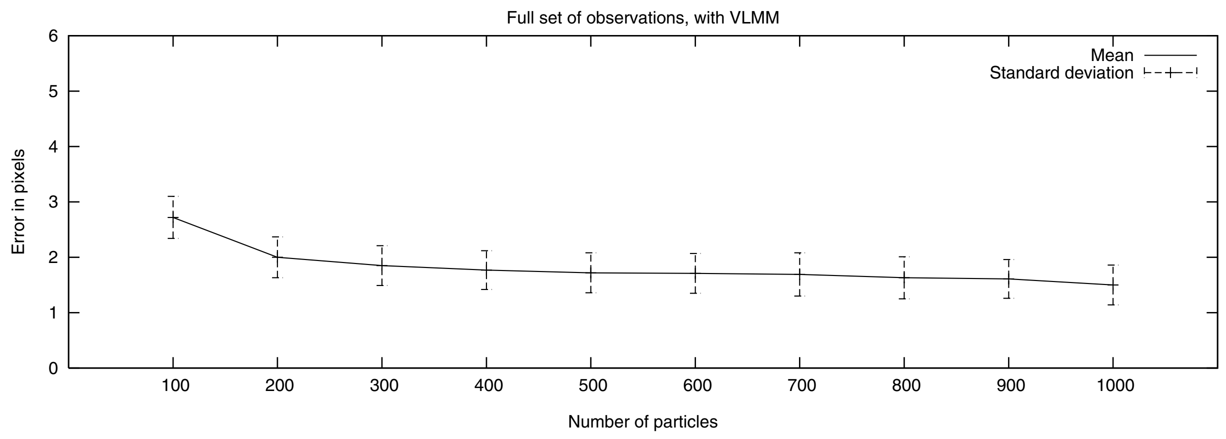

Fig. 7. Mean error and standard deviation, with VLMM. The tracker successfully estimates a complex motion with 4 different gestures.

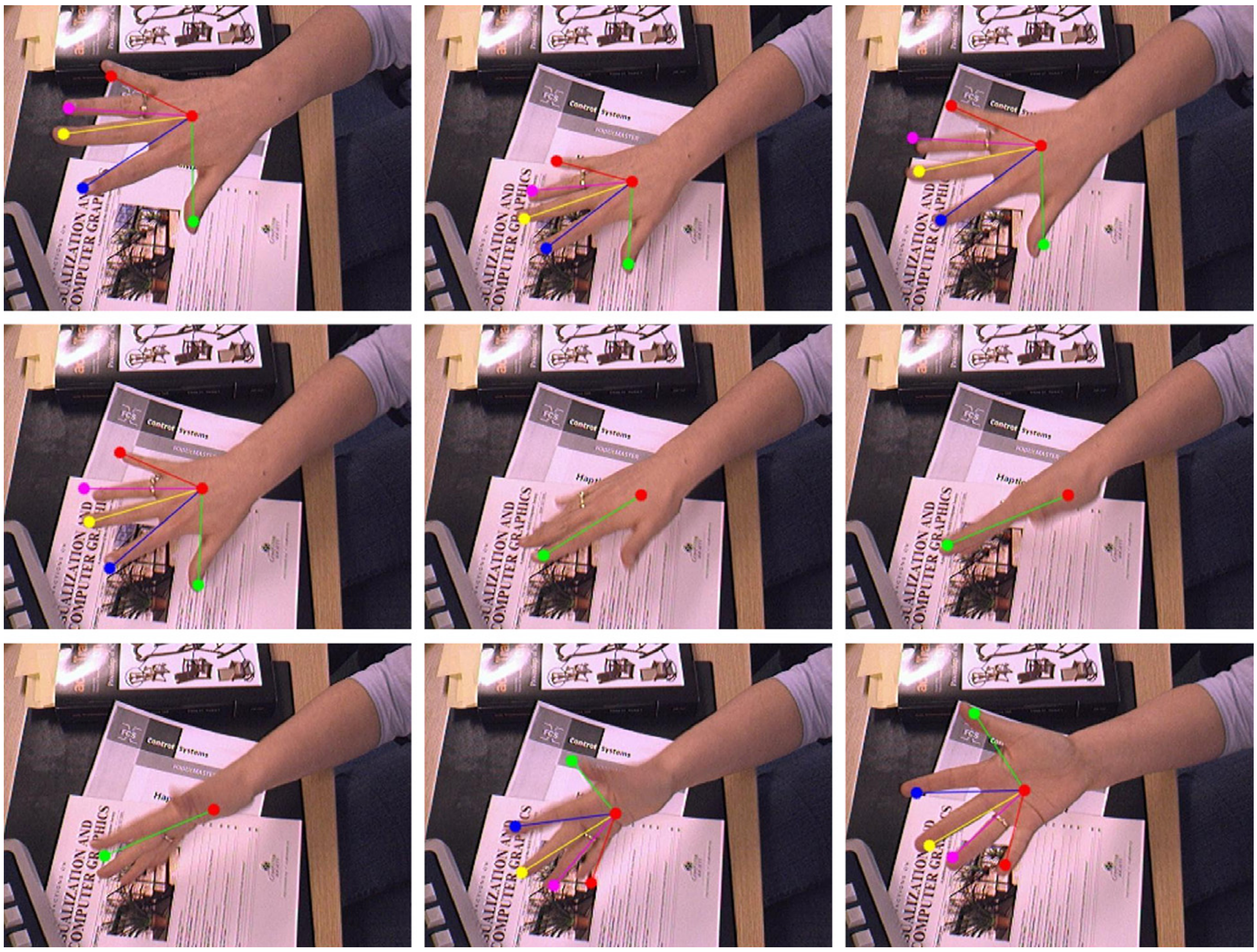

Fig. 8. Tracking results demonstrating the handling of 3D rotation and change of scale. We used 300 particles and three iterations for annealing.

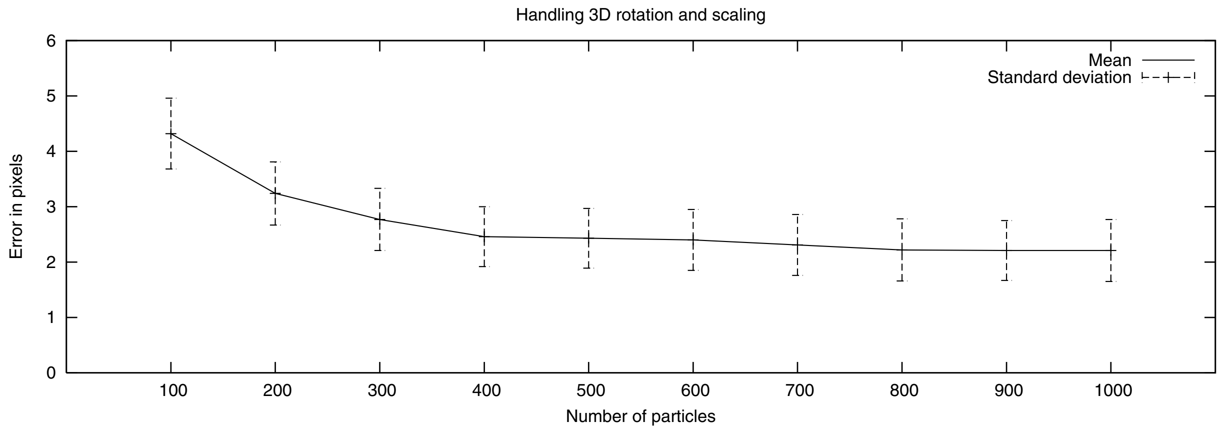

Fig. 9. Mean error and standard deviation of the tracker estimates for the sequence with 3D rotation and change of scale.

Tracking the same sequence with a VLMM (Fig. 11) provided better accuracy and most importantly, reduced jitter. This was due to the fact that the VLMM spread the particles in ways consistent with the training set, thus reducing the random effects of the Monte Carlo particle filter algorithm. The results demonstrate that even in the case when only a single gesture has to be tracked, employing a behaviour model has an advantage over a purely stochastic method.

We also used a separate test sequence of unstructured behaviour (i.e. random movements not observed during training) to investigate what impact using a VLMM has

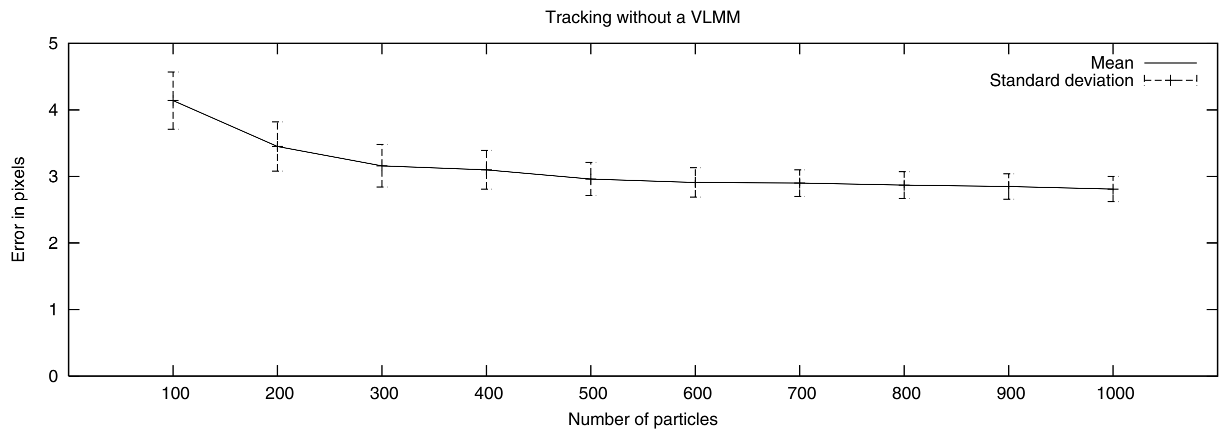

Fig. 10. Mean error and standard deviation of the estimates when tracking structured motion without a VLMM.

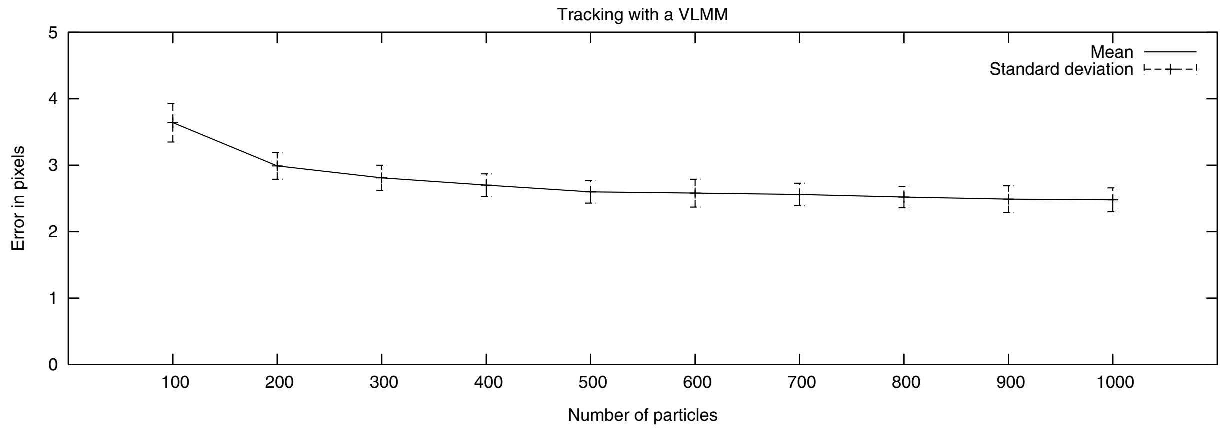

Fig. 11. Mean error and standard deviation of the estimates when tracking structured motion with a VLMM.

over accuracy and robustness. We compared the performance of our algorithm with that of a simple SIR particle filter. There was only one gesture present in the sequence, as it is not possible to have automatic switching between different model configurations with the SIR scheme. The results are presented in Fig. 12. The pixel errors are similar for both approaches, which shows that using a VLMM does not have a negative impact when tracking unstructured behaviour.

This is due to the fact that the weights of the particles are adjusted according to two criteria: first, how closely they match the cluster they are taken from and second, how closely they match the actual observations. Even in the case when the particles are outside the bounding box of the cluster they belong to, they are bound to be in a consistent posture because of the model constraints that have been imposed. A particle will receive a higher weight if it matches the observations well. The stochastic nature of the updating of the particles' states ensures that particles that fall outside the cluster bound will not be resampled if they correspond well to the observations. One can therefore see that the tracking with a behaviour model falls back to the simpler SIR case when there is unstructured behaviour that was not observed during training.

6.3. Tracking with a VLMM compared to a first-order Markov model

Finally, in this section we evaluate the robustness of the algorithm in detecting which gesture is observed in a particular frame and compare its performance when using a first-order Markov model (FOMM) instead of a VLMM.

We used four model configurations representing different gestures: grab, measure, point and move. The current gesture was determined by identifying which model configuration had the most particles representing it in the set. We trained the VLMM using 17 sequences, each approximately 10 seconds long. The training data were clustered into five clusters for the grab, measure and point gestures, and two clusters for the simpler move model.

For testing, a particle filter with 800 particles and three annealing layers was used, over a longer (1 m, 30 s) sequence of structured behaviour comprising the above gestures. Our results showed that the tracker failed to identify the correct gesture in 5 out of 625 frames. Fig. 13 shows the distribution of particles for a short interval of the test sequence. The frames where the particles' VLMM states were reset can be seen in Fig. 15. In order to detect tracking failure, we tested whether the pixel error (summed over all

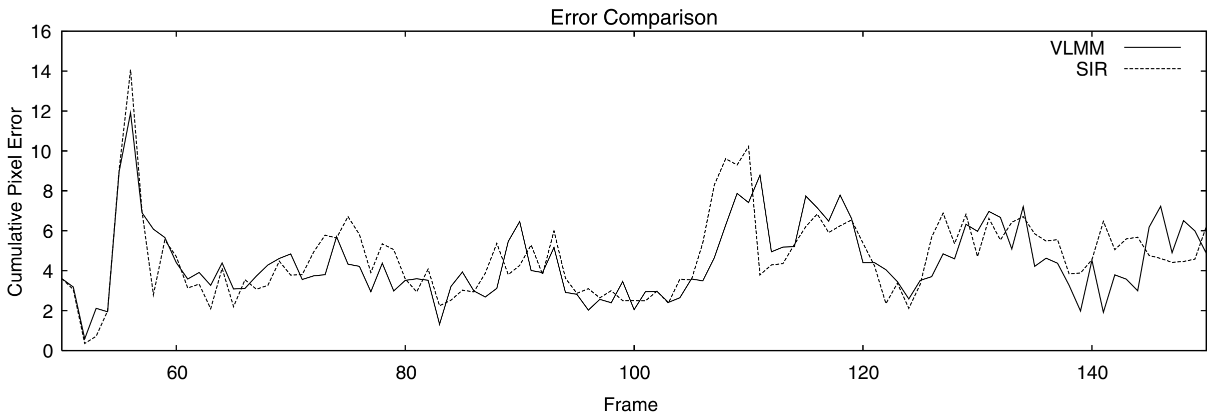

Fig. 12. Comparison of cumulative pixel errors using VLMM and SIR for a test sequence of unstructured behaviour with a single gesture; note similar performance.

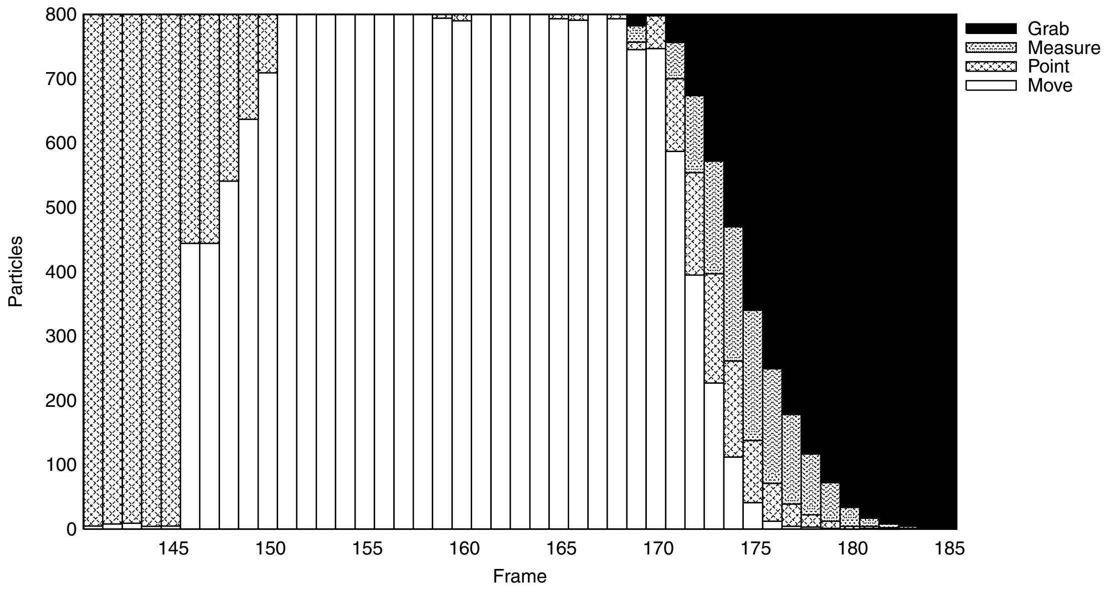

Fig. 13. Number of particles tracking different gestures in a short example interval of one test sequence, using a VLMM.

model joints) of the mode exceeded the value of 10 pixels. As can be seen, the tracker successfully recovered each time the particles' memories were deleted.

The VLMM managed to successfully learn the transitions between different model spaces and was able to predict the changes with a great degree of accuracy. It is worth noting that handling different models in this way would be hard to achieve with an ordinary particle filter; unless a separate filter were used for each model type.

We also evaluated the algorithm when using an FOMM (as used in [14]) rather than a VLMM. Our results showed that the tracker failed to identify the correct gesture in 164 out of 625 frames. Fig. 14 shows the distribution of particles for a short interval of the test sequence, where the FOMM failed to capture the spatial and temporal ordering of the behaviour. As this particular test sequence exhibits longer temporal dependencies, the short memory length of the FOMM becomes inadequate in some frames. The advantage of the VLMM is that it can automatically decide how to adjust the memory length for a particular example sequence, thus opting for a shorter memory when long temporal memory is not needed.

7. User evaluation

To test the viability of visual tracking with our algorithm as an input system for HCI, a user evaluation experiment was conducted that involved participants completing a Fitt's law task (see Fig. 16).

In this experiment, the primary focus was on testing our approach for a simple, but representative 'drag and drop' task. The training data used was provided by a single user, and the experiment was conducted using 10 different participants.

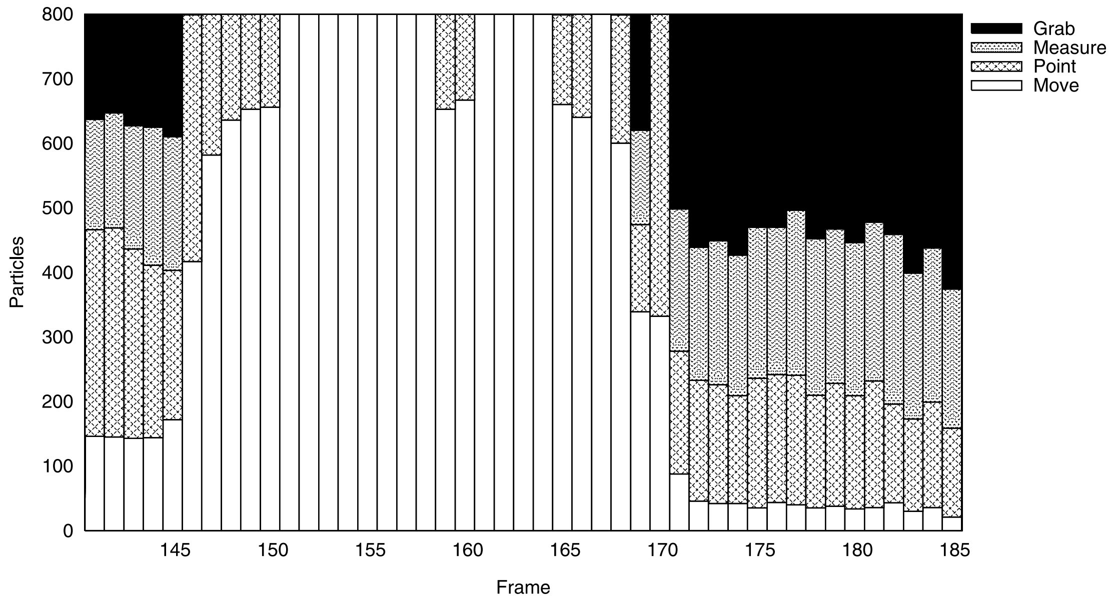

Fig. 14. Number of particles tracking different gestures in a short example interval of one test sequence, using a FOMM.

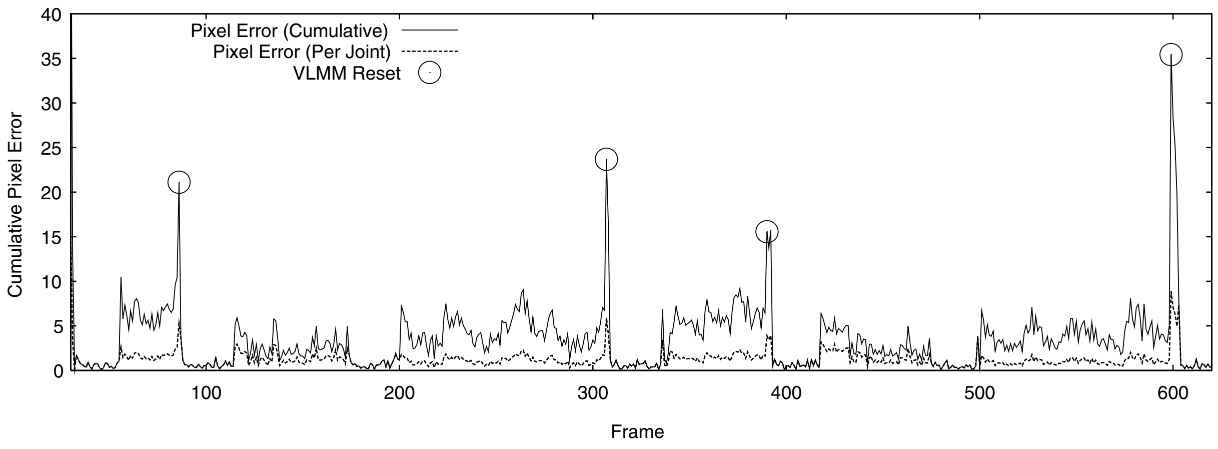

Fig. 15. Cumulative pixel error (summed over all model joints) for one whole test sequence of structured behaviour using a VLMM.

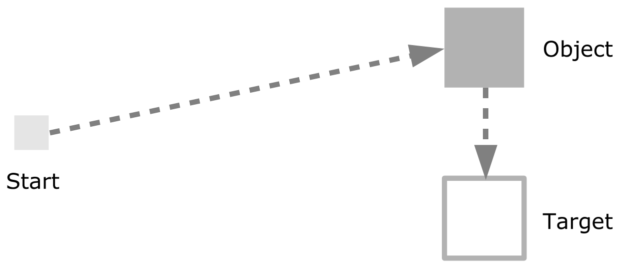

Fig. 16. An example test case from the Fitts' law task used for evaluating user performance.

7.1. Fitts' law

Card, English and Burr [5] were the first to introduce human performance models in studies of input devices. They applied Fitts' law to computer pointing tasks. Their intent was to standardise usability studies, using a single metric that could be compared across different experiments. Fitts' law relates the time taken to acquire a target—that is, to select it, for example by clicking on it with a mouse—with the distance that the hand needs to travel and the size of target. Intuitively, time increases as the distance increases and the size decreases, and vice versa. Fitts' law can be used to estimate the 'throughput' of an input device; that is, the number of target selections that can be performed in a certain time. The more ergonomically efficient a device is, the greater its throughput.

Fitts' law defines the logarithmic relationship between the time to select a target, the distance to that target and its size. Several variants of Fitts' law have been proposed that incorporate empirically derived constants in order to give the best possible linear relationship for the log function. We use MacKenzie's definition of Fitts' law [24], based on the original Shannon formula of information in a noisy channel, to avoid problems of negative values:

$$ID = \log_2\left(\frac{D}{W} + 1\right)$$
$$MT = a + b\ ID$$

where D and W are target distance and size respectively, ID is the index of difficulty, MT is movement time, and a and b are empirically determined constants, depending on the input system that is used.

Fitts' law can be tested experimentally by recording the times taken by human subjects in a controlled target acquisition task. From the average timing data across all users, one can construct a regression line which describes the efficiency of the device. This regression line can be compared with that of other devices.

7.2. Description of the experiment

The main objectives of our simple drag-and-drop experiment were: to demonstrate that Fitts' law holds when using our proposed tracker as an input device, to compare the tracker with an equivalent mouse-based interface, and to explore the effect of latency on task performance. As Fitts' law predicts motor movement, a correctly operating tracker should exhibit similar characteristics to other input devices, such as the mouse, although one might reasonably expect users to be more familiar with, and therefore faster in completing tasks with the latter.

Prior research has demonstrated that latency impacts negatively on user performance for these types of tasks. The threshold above which latency (or delay) becomes noticeable is around 50 ms—as has been demonstrated for both visual and haptic feedback [25]. The latency introduced by our tracker arises from several sources. First, images are acquired on a separate processor. The images are then compressed and sent over a fast network to the PC executing the tracker code. This preprocessing and data transmission consumes 40 ms. Next, background subtraction and feature detection are performed, followed by the tracking process, and finally the display of the projected view of the hand model.

We ran our drag and drop experiment with three different configurations of input device: our visual tracker, a standard mouse, and the same mouse with 40 ms of added latency. The 40 ms of mouse lag was introduced to account for the image acquisition and preprocessing, while the base case with no added latency is, in some sense, the gold standard. In the case of the mouse with added latency we further constrained the frame rate to be 25 FPS to match the image capture frame rate sustained over the network.

There were 10 participants—three female and seven male—all right-handed, experienced computer users. The participants in the experiment started at a fixed position, acquired a square red object and moved it to a square green target of the same size and then released it (see Fig. 16). In the case of visual tracking, the participants started each test case with their hands closed, then picked up the object with two fingers held out as if grasping it, and released it over the target by opening their hand. In the two other cases, the mouse was used for dragging and dropping the object in a standard fashion. The sensitivity of the mouse was adjusted so that the amount of hand movement—the distance to the target and its size—was identical for each case. We measured separately the times taken in the acquisition and dragging sub-tasks.

The independent variables were the dragging distance $D = \{8, 16, 24\}$ units (1 U = 8 pixels), the sizes of the object and the target $W = \langle 4,5 \rangle$ units, and the approach angle from the object to the target $AA = \langle 0°, 90°, 180°, 270° \rangle$; the distance from the starting position to the object depended on the approach angle. The 24 possible cases were randomised, and the participants repeated the experiments under the three different conditions: first with visual tracking, next with mouse input delayed by 40 ms and frame rate constrained to 25 frames per second, and finally with normal mouse input.

7.3. Results of performance evaluation

Figs. 17 and 18 show the regression lines for the average time taken to complete the tasks as a function of the index

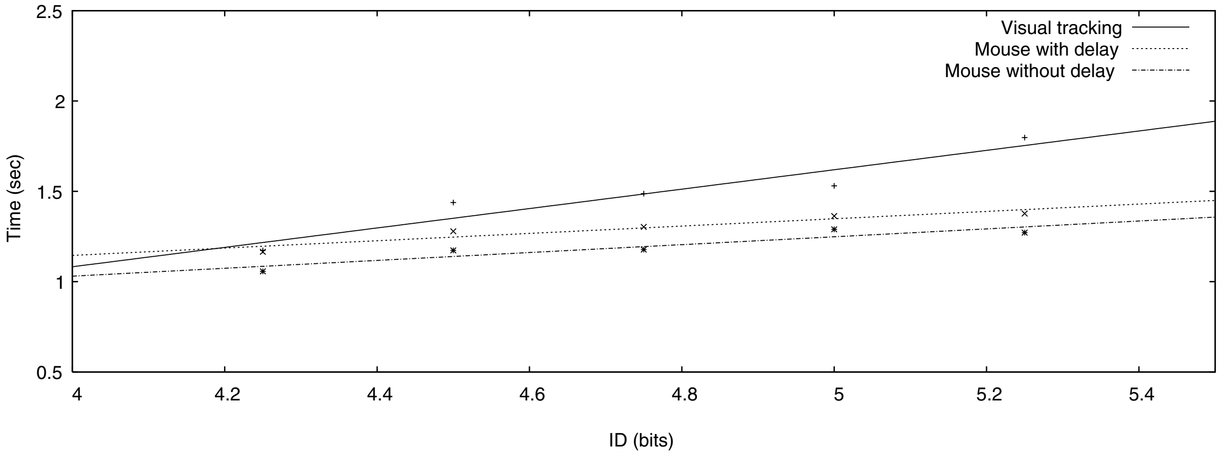

Fig. 17. Time taken to complete the acquisition task as a function of the index of difficulty.

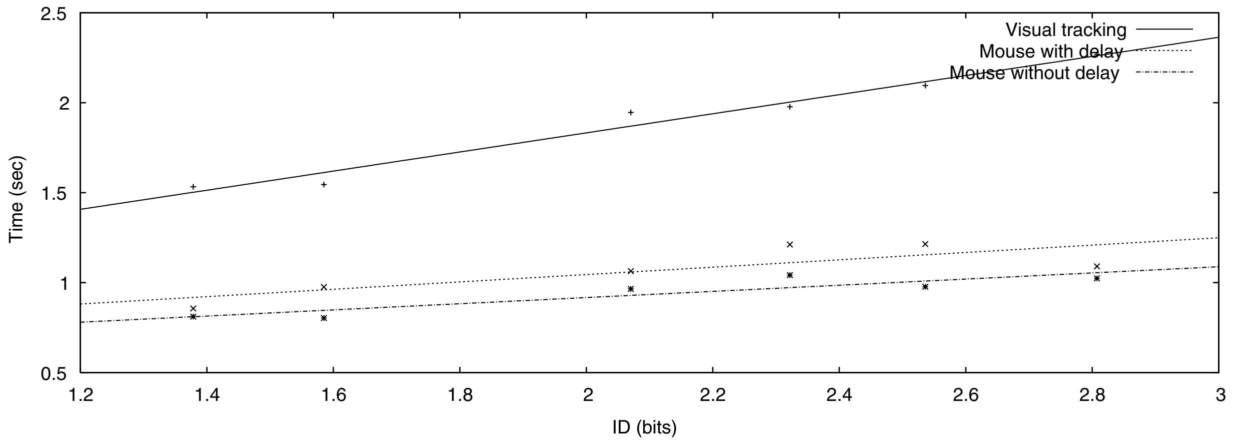

Fig. 18. Time taken to complete the drag task as a function of the index of difficulty.

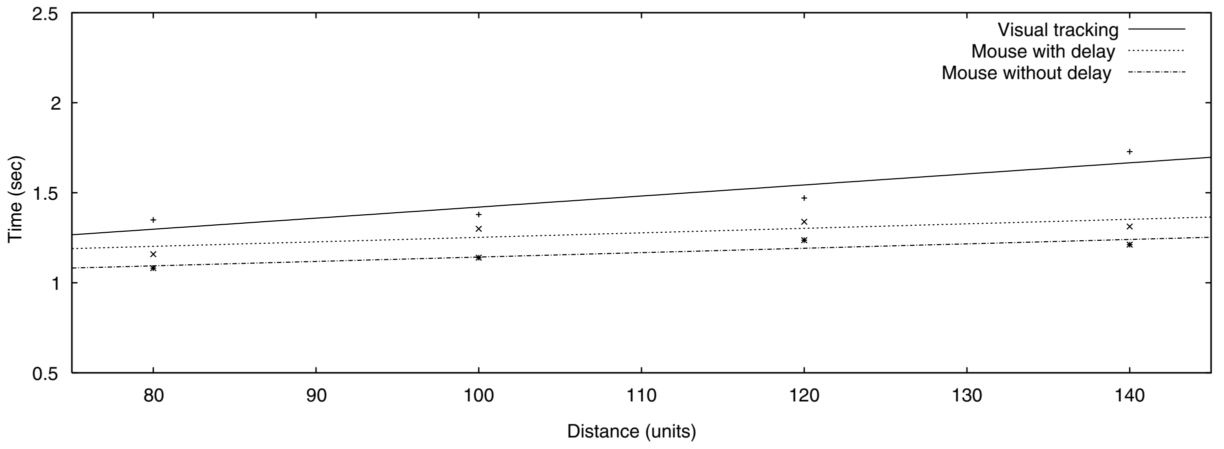

Fig. 19. Time taken to complete the acquisition task as a function of the distance (1 unit = 8 pixels).

of difficulty ID. As can be seen, the relationship is linear and this is consistent with Fitts' law. The results show that for the acquisition task over shorter distances (small IDs), the performance of the participants when using our algorithm was comparable to using the mouse. However, for larger distances, the time is longer by over 0.5 s. This is accounted for by the extra time needed by the particle filter to obtain an initial estimate with a sufficiently large support map in the particle set. For the dragging task, we observe that completion time with visual tracking is at most 1.5 s. longer than with the other input schemes. We ascribe this to the lack of tactile feedback when moving the object,

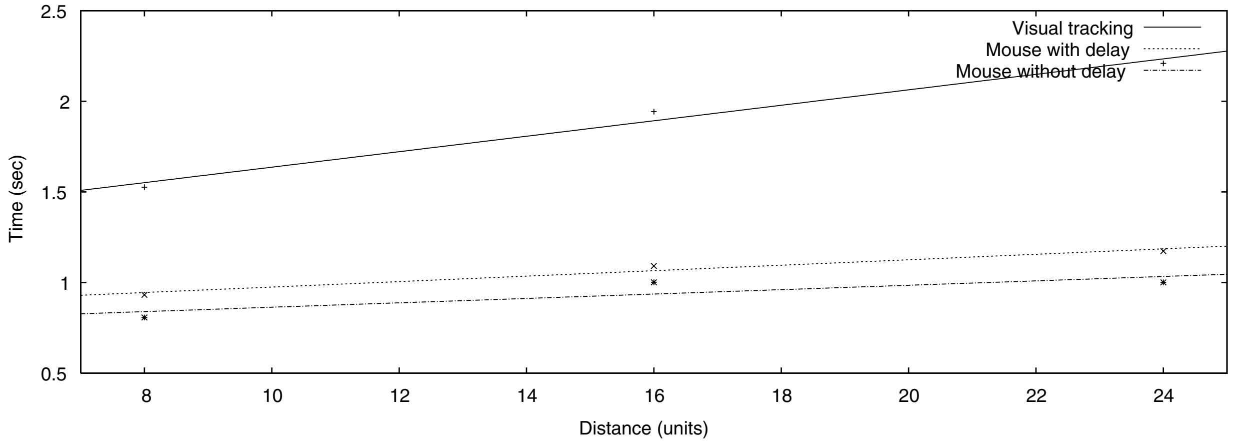

Fig. 20. Time taken to complete the drag task as a function of the distance (1 unit = 8 pixels).

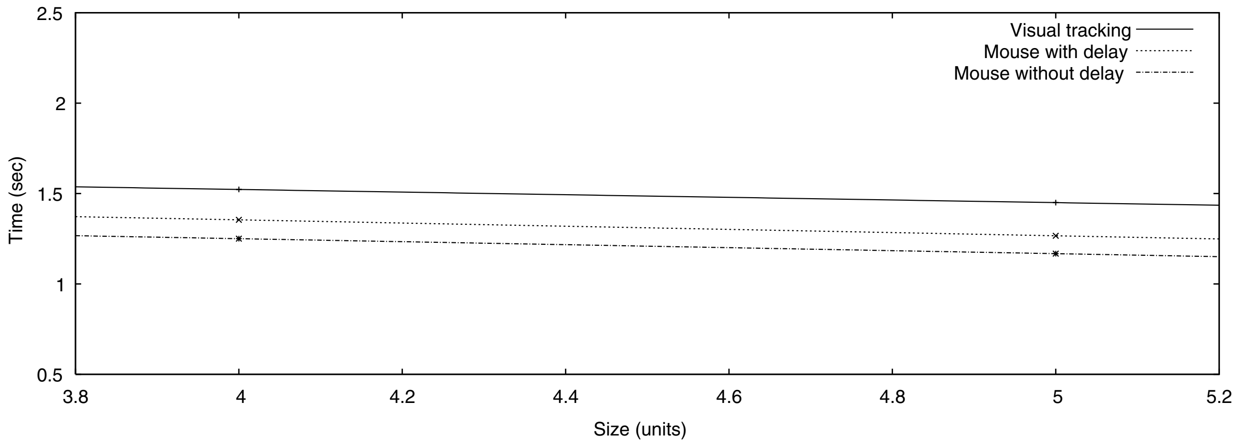

Fig. 21. Time taken to complete the acquisition task as a function of the size (1 unit = 8 pixels).

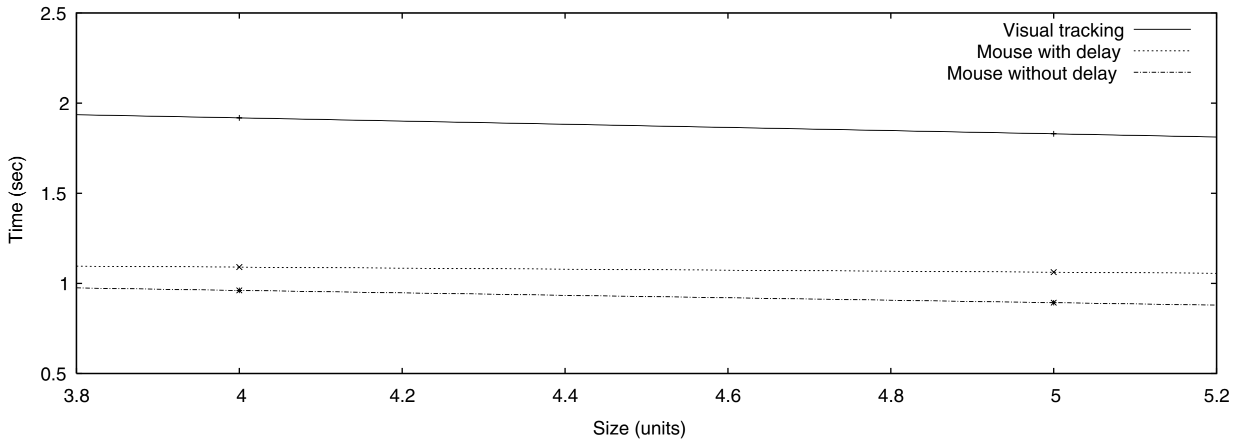

Fig. 22. Time taken to complete the task as a function of the size (1 unit = 8 pixels).

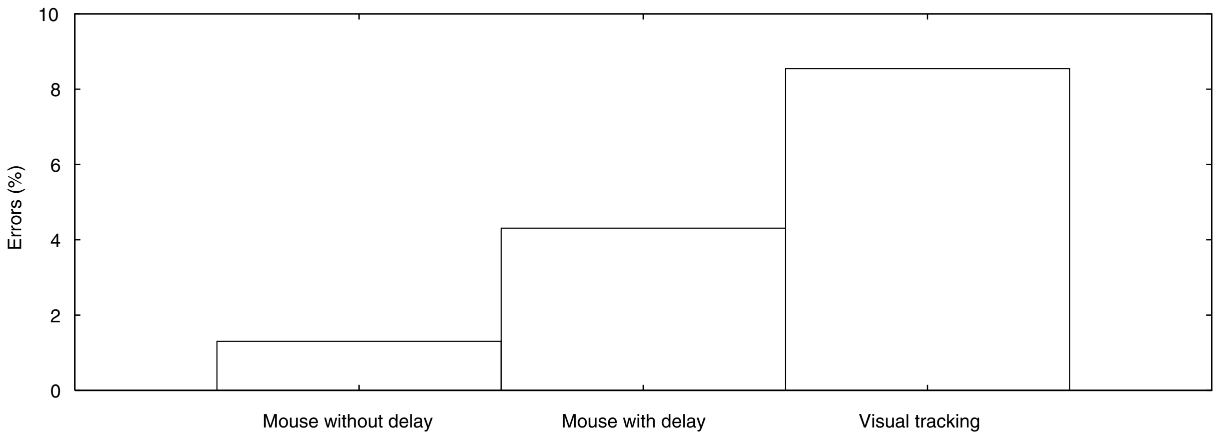

Fig. 23. Percentage of errors for all experiments.

which some users complained of. Two participants reported hand fatigue at the end of the visual tracking test.

Figs. 19–22 show the time taken to complete the task respectively as a function of the distance travelled, and the size of the object and target. The overall trend is similar in all cases and demonstrates that the completion time decreases as the size of the target becomes bigger and the distance that needs to be travelled becomes smaller. Again, the results are consistent with Fitts' law and with previous research [9,1], which shows that user performance decreases as input delay increases.

Finally, Fig. 23 shows the percentage of errors with the three different input methods. We count as an error every occurrence when the object is not dropped inside the target.

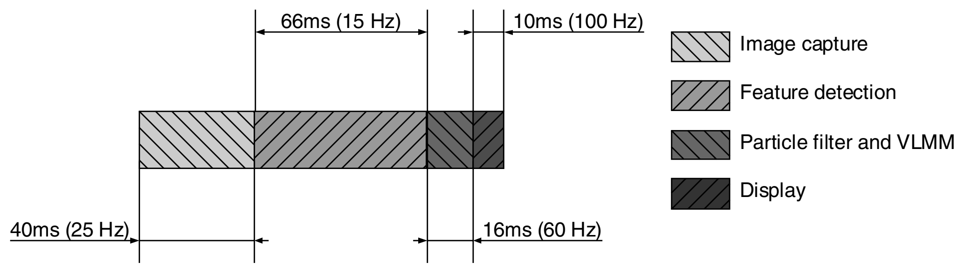

Fig. 24. Latency analysis. There is a 132 ms delay between capturing an image and the tracker estimates being displayed to the user.

As might be expected, the lowest number of errors was made with the undelayed mouse. The number of errors increased by four when the latency from the camera was introduced. With visual tracking, the number of errors doubles over that observed with delayed mouse input.

To explain the performance of the participants, we measured the latency introduced by the various stages of the visual tracking pipeline. The results are summarised in Fig. 24. As already stated, the time for the current frame to be compressed and sent over the network is approximately 40 ms. After the image is transmitted, background subtraction and feature detection consume most of the time (around 60 ms). The stochastic Bayesian simulation itself has a relatively low impact of 16 ms per frame. The rest of the delay is accounted for by the display routines and the time it takes for the image to be displayed on the computer screen with a refresh rate of 80 Hz. One can see that despite the fact the framerate sustained by the tracker is 15 FPS, the total latency for the whole pipeline is close to 132 ms. Although the tracker is robust to failure and operates in real time, the fact that such high latency is introduced results in degraded user performance.

8. Discussion

In this paper, we have demonstrated a system that combines behavioural knowledge with a stochastic simulation to achieve robust tracking of hands. Distinct gestures are handled by using different models; transitions between them are learned with a variable-length Markov model that captures both high- and low-level structure. An efficient way to evaluate likelihood allows us to achieve real-time performance, even when using computationally expensive techniques such as annealing. The cost of this robust tracking is quite modest—around 16 ms.

We also provided an evaluation of our proposed system with a Fitts' law task. The results from this highlighted two important issues. First, latency in the data acquisition, processing and display pipeline, has a detrimental effect on user task performance. The effects of latency are often ignored; some papers concentrate on frame rate, which is not the same thing. For interaction, latency cannot be ignored, and warrants further investigation. In this paper we have shown where this latency arises and this indicates where attention must be focused in future. Second, gestures must be carefully designed so that users can employ them comfortably and straightforwardly to signify different actions. In the experiment reported here we opted for a design where the users picked up an object by 'holding' it between their fingers. Some users had difficulty with this because they were 'grasping at thin air'. The results indicate that what seems natural may not, in fact, be the most effective design; this too warrants further exploration.

References

[1] R.S. Allison, L.R. Harris, M. Jenkin, U. Jasiobedzka, J.E. Zacher, Tolerance of temporal delay in virtual environments. Proc. IEEE Conf. Virt. Real, pp 247–254, 2001.
[2] S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for on-line non-linear/non-Gaussian bayesian tracking, IEEE Transa. Signal Process. (2001) 100–117.
[3] R. Baltman, R. Radeztsky Jr. Verlet integration and constraints in a six degree of freedom rigid body physics simulation. Proc. GDC, 2004.
[4] T. Bell, J. Cleary, I. Witten, Text Compression, Prentice Hall, 1990.
[5] S. Card, W. English, B. Burr, Evaluation of mouse, rate-controlled isometric joystick, step keys and text keys for text selection on a CRT, Ergonomics 21 (1978) 601–603.
[6] G. Cormack, R. Horspool, Data compression using dynamic Markov modelling, Comput. J. 30 (6) (1987) 541–550.
[7] T. Cover, J. Thomas, Elements of Information Theory, Wiley, 1991.
[8] J. Deutscher, A. Blake, I. Reid, Articulated body motion capture by annealed particle filtering. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. 2:126–133, 2000.
[9] S.R. Ellis, M.J. Young, B.D. Adelstein, S.M. Ehrlich, Discrimination of changes in latency during voluntary hand movement of virtual objects. Proc. Hum. Factors Ergonom. Soc. pp. 1182–1186, 1999.
[10] A. Galata, A.G. Cohn, D. Magee, D. Hogg, Modeling interaction using learnt qualitative spatio-temporal relations and variable length Markov models. Proc. Eur. Conf. Artif. Intell. pp. 741–745, 2002.
[11] A. Galata, N. Johnson, D. Hogg, Learning variable-length Markov models of behavior, Comput. Vis. Image Understand. 81 (2001) 398–413.
[12] N. Gordon, J. Salmond, A. Smith, Novel approach to non-linear/non-Gaussian Bayesian state estimation, IEE Proc. F (1993) 107–113.
[13] I. Guyon, F. Pereira. Design of a linguistic postprocessor using variable length models. Proc. Int. Conf. Document Anal. Recognit. 454–457, 1995.
[14] T. Heap, D. Hogg, in: Wormholes in shape space: Tracking through discontinuous changes in shape, in: Proc. Sixth Internat. Conf. on Computer Vision, pp. 344–349, 1998.
[15] J. Hu, W. Turin, M. Brown, Language modelling using stochastic automata with variable length contexts, Comput. Speech Language 11 (1) (1997) 1–16.
[16] Intel Research. Open Computer Vision Library. <http://www.intel.com/research/mrl/research/opencv/>.
[17] M. Isard, A. Blake, Visual tracking by stochastic propagation of conditional density, in: Proc. Fourth Eur. Conf. on Computer Vision, pp. 343–356, 1996.

[18] M. Isard, A. Blake, A mixed-state condensation tracker with automatic model-switching, in: Proc. Sixth Internat. Conf. on Computer Vision, pp. 107–113, 1998.

[19] Y. Iwai, Y. Yagi, M.Yachida. A system for 3D motion and position estimation of hand from monocular image sequence, in: Proc. Internat. Conf. on Human–Computer Interaction, 2:809–814, 1995.

[20] K. Abe, H. Saito, S. Ozawa, Virtual 3D interface system via hand motion recognition from two cameras, IEEE Trans. Syst. Man Cybernet. A 32 (4) (2002) 536–540.

[21] G. Kitagawa, W. Gersch, Smoothness priors analysis of time series, Lect. Notes Statist. 116 (1996).

[22] J. Lin, Y. Wu, T. Huang. Capturing human hand motion in image sequences, in: Proc. Workshop on Motion and Video Computing (MOTION'02), 99, 2002.

[23] J. Lin, Y. Wu, T.S. Huang. Modeling the constraints of human hand motion, in: IEEE Human Motion Workshop, 121–126, 2000.

[24] I. MacKenzie, A note on the information-theoretic basis for Fitts' law, J. Motor Behav. 21 (1989) 320–323.

[25] I. MacKenzie, C. Ware. Lag as determinant of human performance in interactive systems, in: Proc. SIGCHI Conf. on Human Factors in Computing Systems, pp. 488–493, 1993.

[26] K. Oka, Y. Sato, H. Koike, Real-time fingertip tracking and gesture recognition, IEEE Comput. Graph. Appl. 22 (6) (2002) 64–71.

[27] V.I. Pavlovic, R. Sharma, T.S. Huang, Visual interpretation of hand gestures for human–computer interaction: a review, PAMI 19 (7) (1997) 677–695.

[28] J. Rehg, T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking, in: Proc. Third Eur. Conf. on Computer Vision, pp. 35–46, 1994.

[29] D. Ron, S. Singer, N. Tishby, The power of amnesia, Adv. Neural Inform. Process. Syst. 6 (1994) 176–183.

[30] C. Shan, Y. Wei, T. Tan, F. Ojardias, Real time hand tracking by combining particle filtering and mean shift, in: Proc. Sixth IEEE Internat. Conf. on Automatic Face and Gesture Recognition, pp. 669–674, 2004.

[31] L. Sigal, S. Sclaroff, V. Anthitsos, Skin color-based video segmentation under time-varying illumination, IEEE Trans. Pattern Anal. Mach. Intell. 26 (7) (2004) 862–877.

[32] Y. Singer, M. Warmuth. Batch and on-line parameter estimation of Gaussian mixtures based on the joint entropy, in: Proc. Conf. on Advances in Neural Information Processing Systems II, pp. 578–584, 1999.

[33] C. Stauffer, W. Grimson. Adaptive background mixture models for real-time tracking. Proc. Compu. Vis. Pattern Recognit. (CVPR'99), 246–252, 1999.

[34] J. Terrillon, S. Akamatsu, Comparative performance of different chrominance spaces for color segmentation and detection of human faces in complex scene images. Proc. Vision Interf., 180–187, 1999.

[35] A. Thayananthan, B. Stenger, P. Torr, R. Cipolla, Learning a kinematic prior for tree-based filtering. Proc. Br. Mach. Vis. Conf., September 2003.

[36] K. Toyama, A. Blake, Probabilistic tracking in a metric space. Proc. Int. Conf. Comput. Vis. (ICCV), pp. 50–57, 2001.

[37] G. Welch, G. Bishop, An introduction to the Kalman filter. TR 95-041, 1995. University of North Carolina at Chapel Hill, Department of Computer Science.

[38] Y. Wu and T.S. Huang. Robust visual tracking by co-inference learning, in: Proc. Int. Conf. Comput. Vis. (ICCV), 26–33, 2001.