Real-time Hand Tracking With Variable-Length Markov Models of Behaviour

Nikolay Stefanov, Aphrodite Galata, Roger Hubbold
Advanced Interfaces Group, School of Computer Science
University of Manchester, Oxford Road, Manchester M13 9PL, UK
{nstefanov, agalata}@cs.man.ac.uk, roger.hubbold@manchester.ac.uk

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.

1 Introduction

Recently, there has been increased research interest in unencumbered virtual reality interaction. Traditional VR equipment, such as gloves and 3D mice can be detrimental to users' immersion in the VR environment. A visual tracking and registration system is a possible replacement option, especially for the task of manipulating virtual objects in a natural manner, using hand movements and gestures. However, hand tracking poses several difficult challenges. Because of problems caused by the high-dimensional search space, direct evaluation of the hand posture is often not possible. Moreover, methods that rely on local optimisation fail because of discontinuous changes in appearance. These problems are further exacerbated by certain postures which introduce self-occlusion.

In order to robustly solve the problem of hand tracking, we make use of the fact that human-computer interaction is mainly composed of structured behaviour; in other words, there is a high-level structure governing the hand's movement and the temporal ordering of different gestures.

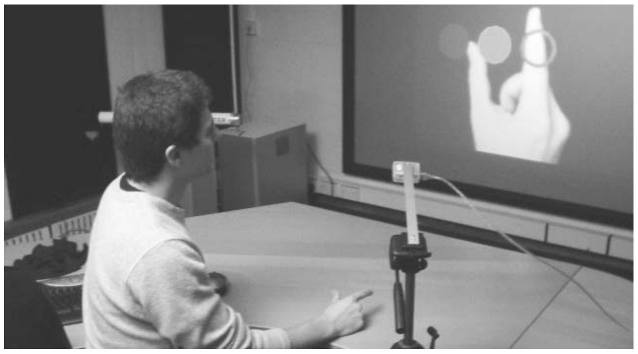


Figure 1: Manipulation of simple virtual objects with our tracker.

Thus, a model of human behaviour can be applied so as to help the tracker by reducing the 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.

In this paper, we present a robust hand tracking algorithm applied to the problem of human-computer interaction (Figure 1). Our approach is based on a variable-length Markov model which learns structured behaviour in an off-line training stage. The model is extended to handle continuous problems via annealed particle filtering. Combined with a fast image feature detector we are able to achieve 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.

In order to explore the applicability of our algorithm as an input system for VR, we evaluated its usability with a Fitts' law drag-and-drop task. Fitts' law is a widely studied model of the act of pointing, and can be applied both to pointing with hands and with input devices such as the mouse. In this paper, we compare user performance in three different scenarios – with visual tracking, with mouse input when a delay that matched the one from the camera was introduced, and finally with non-delayed mouse input.

Our goal is to develop a system that allows manipulation of virtual objects via a relatively limited vocabulary of gestures that does not require extensive user training. Consequently, we place a particular emphasis on the real-time aspect of the tracker, and at the same time are interested in the robustness and accuracy of the posture estimates. The underlying framework implementation fully supports a 3D representation of the hand posture and image observations. However, this paper presents results from a two-dimensional Fitts' law task with a single camera view.

The paper is structured as follows: in Section 2 we provide a brief description of related research. Section 3 explains the hand model representation that is used in our system. Section 4 discusses particle filtering as applied to the problem of visual tracking and registration. Section 5 introduces variable-length Markov models and describes a general approach to combine them with a particle filtering framework. Section 6 describes the model evaluation and feature detection parts of our system. Section 7 contains test results that show the robustness and accuracy of our proposed algorithm. Section 8 presents the results from user evaluation and latency analysis. Lastly, in Section 9, we comment on the results and discuss future work.

2 Related work

Early research in real-time hand tracking [18, 13] identified several problems, namely the high-dimensional search space, and the inefficiency of approaches that do not use some form of model-based constraints. More recent research uses better shape detectors [14], or incorporates various Bayesian techniques, like Kalman filtering [17], tree-based filtering [23] or particle filtering [20].

Bayesian simulation is a useful way to improve the robustness of various simulations. The Kalman filter [25] is an optimal solution if the process being estimated can be represented by a Gaussian. Monte Carlo particle filters [8, 15] estimate the required densities with a set of discrete weighted samples. They have been applied successfully in the context of computer vision [11], and expanded to handle mixed-state particles [12], to avoid local maxima via annealing [4], and to track over discontinuous changes in the observations [10]. A good overview of this family of algorithms is given by Arulampalam et al. [2].

The work presented here expands on our previous efforts in the area of real-time tracking [22]. It combines the mixed-state particle filtering approach with annealing; with the help of a feature detector, we are able to reach real-time

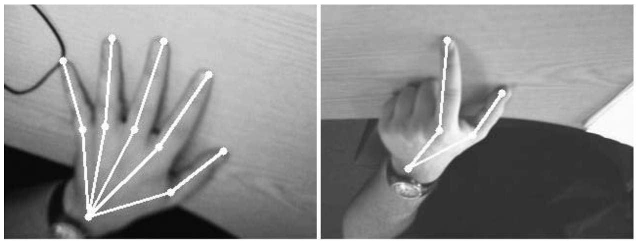


Figure 2: Two examples of hand models.

performance without any significant optimisation effort. We handle discontinuous changes in the observations by learning the transitions with a variable-length Markov model (VLMM) [19, 9]. As shown by [7], VLMMs encode high-order temporal dependencies (such as those observed in human-computer interaction) better than first-order Markov models. We provide a comparison between the two methods in Section 5.

Card, English and Burr [3] 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 can be compared across different experiments. We use MacKenzie's [16] definition of Fitts' law, who reverted to 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.

3 Hand representation

A hand model consists of a hierarchical set of bones $\{B_0, ..., B_n\}$, connected by joints $\{J_0, ..., J_n\}$. One can think of it as representing a skeletal structure, where the joints correspond to fingertips, phalange links and palm (see Figure 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_{J_i}, which is a 3D space transformation relative to the parent joint. P_{J_i} is composed of a rotational part R and a translational part T; 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. A constraint is specified in terms of an axis and minimum and maximum angles, according to which the rotation angle must be clamped. In order to increase flexibility, the translational part T of the transformation are also varied, thus effectively allowing us to increase the length of the bones so as to compensate for scale.

4 Particle filtering

Suppose we have a sequence of image frames and we wish to track a hand through time. Let X be the hand's model configuration vector with dimension d, and z_k be the observations for frame k. A particle filter [2] approximates the posterior density $p\left(X|z_{1:k}\right)$ with a set of weighted particles $\left\{\left(x^0w^0\right),...,\left(x^Nw^N\right)\right\}$. The particle weights $w^i \propto p\left(z_k|X=x^i\right)$ are normalised so that $\sum w^i = 1$. The estimated state χ_k can be obtained either by the weighted average of the particles $\chi_k = \sum x^iw^i$ or an approximation of the mode $\chi_k = x^j, w^j = max\left(w^i\right)$.

Similar to Sampling Importance Resampling (SIR) [8], in our algorithm the particles are drawn from the importance density $p\left(X_k|X_{k-1}\right)$. Additionally, after evaluating w^i, the particles are resampled with replacement from the set. We use the Systematic Resampling [2] algorithm which operates in $O\left(N\right)$ time.

In order to maintain a fair representation of the density, often a large number of particles, N, is required. However, in such a case a major bottleneck is drawing from $p\left(z_k|X=x^i\right)$, or in other words obtaining the weights for the particles. In practice, using a particle filter to solve a problem where X 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.

5 Variable Length Markov Models

Much of human-computer interaction consists of structured behaviour, or in other words sequences of primitive movements and gestures that exhibit some high-level structure. A particular behaviour can be described as a sequence of model configuration vectors $X^0, X^1, ...X^n$, where X^i correspond to different hand postures and have varying dimensions. This representation can be 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. A suitable way to obtain such knowledge are variable-length Markov models (VLMM). In contrast with n-th order Markov models for which the memory length required for prediction is fixed, VLMMs are more flexible and efficient; they are able to locally optimise memory length within the model and thus capture both short and long-term temporal dependencies [19, 7, 6].

A VLMM can be thought of as a probabilistic finite state automaton (PFSA) $\mathcal{M} = \left(Q, \Sigma, \tau, \gamma, s\right)$. 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 $N_{\mathcal{M}}$ ($N_{\mathcal{M}} \geq 0$), 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:

$$\begin{aligned}
\tau :& Q \times \Sigma \rightarrow Q \\
\gamma :& Q \times \Sigma \rightarrow [0,1]\,, \quad q \in Q,\ \sum_{a\in\Sigma} \gamma\left(q,a\right) = 1 \\
s :& Q \rightarrow [0,1]\,, \qquad\qquad q \in Q,\ \sum_{q\in Q} s\left(q\right) = 1
\end{aligned}$$

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.

5.1 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.

We can describe behaviour with a set of clusters, or prototypes, in model configuration space $\mathcal{C} = \{c_1, c_2, ..., c_{N_c}\}$, where c_i is a statistically plausible example of X; note that the dimensions of the clusters may vary. Thus, a behaviour can be represented by a sequence $c^0...c^i \in \mathcal{C}$, which we can obtain by finding the closest cluster in $\mathcal{C}$ for the state X_k in frame k.

In order to use the VLMM in a particle filter framework, we augment the state of each particle x^i so that it also contains a PFSA state q_{x^i}. When propagating the particles, we also advance their VLMM state, first randomly choosing the next token $a_{x^i} \in \Sigma$ with probability $\gamma\left(q_{x^i}, a_{x^i}\right)$ and then setting $q_{k+1} = \tau\left(q_{x^i}, a_{x^i}\right)$.

Assume a particle's PFSA state at frame k corresponds to a sequence of behaviour prototypes $c^0 ... c^k$. At frame $k+1$, we choose the next VLMM state, thus adding a prototype c^{k+1} to our sentence. If $c^k = c^{k+1}$, then the particle's associated model configuration x^i is propagated as in the SIR algorithm; however, we also ensure that X_{x^i} remains a plausible example of c^{k+1}. If $c^k \neq c^{k+1}$, or in other words the VLMM indicates a transition between different configuration clusters, x^i is sampled from the neighbourhood of c^{k+1}.

This scheme can be easily extended to an annealing particle filter [4]. With annealing, the particle filter runs multiple iterations over a single frame. Due to weight biasing, the particles gradually migrate toward the global maximum without being distracted by local ones. One annealing layer per iteration is used; the particles' VLMM states are updated only at the first iteration, otherwise they are simply perturbed with a random Gaussian variable $x^i = x^i + v_k$.

Effectively, this process generates a sequence in Σ for each particle, which is in fact a statistically plausible sequence of behaviour prototypes $c^i \in \mathcal{C}$. 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 reintroduce 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 [7]. If we are presented with a prototype a such that $\gamma\left(q_{x^i}, a\right) = 0$ for all particles' VLMM states q_{x^i}, 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 $w^j = max\left(w^i\right)$ with some predefined threshold, depending on the application.

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

5.2 Training the VLMM

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. The model configuration is augmented with its first derivative $\dot{X}_k$ in order to resolve spatial and temporal ambiguities. Additionally, depending on the training data, $\dot{X}_k$ is scaled in order to further emphasise its influence. Thus, we obtain a set of prototypes P^d for each different model configuration size d; note that these prototypes may belong to different sequences.

In order to obtain the prototypes $\mathcal{C}^d$, we cluster each P^d. After obtaining all $\mathcal{C}^d$, we set $\mathcal{C} = \bigcup \mathcal{C}^d$. 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 $\mathcal{C}$. An image sequence is now represented by a sequence in the alphabet of the prototypes $\mathcal{C}$. These sequences are then used for training the VLMM. The memory size $w_{\mathcal{M}}$ and threshold ϵ parameters (see [7, 6] for details) can be used to control the actual VLMM construction, depending on the nature of the training data.

6 Tracker description

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 x^i. Commonly, the particles x^i are evaluated by projecting their corresponding model configuration into image space and computing the difference between generated and observed contours. Clearly, this would rapidly become unfeasible if we wished to maintain a large number of particles N at real-time frame rates.

In our system, each particle's state for a frame k, x^i consists of a model configuration as described in Section 2, and represents an estimate of the real hand posture. The expectation is that since the length of the configuration vector X is not large, and sampling from $p\left(z_k|X = x^i\right)$ is inexpensive, using approaches like annealing will substantially improve the accuracy and decrease the jitter of the tracking estimates.

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. Such a labelling of the observations z_k will help to reduce the degrees of freedom required in tracking. Both fingertips and palm centre are detected by finding circles in a frame k with different radii, with an approach based on the classical Hough transform [24]. We perform a transformation of the input image into circular Hough space with different values for the circle radius. These values are chosen according to our particular camera setup and are typically around 10 and 64 pixels respectively for the fingertips and the palm, and are adjusted to accommodate different users. Since the Bayesian simulation takes care of incomplete data and false positives, we do not rely on the feature detection to return accurate results in each individual frame.

The advantage of such a scheme over related contour-based work is that instead of finding target contours in each image and then having to compute contour differences for each particle in the filter set, we only have to compare with a limited number of feature points. The image data is accessed only once per frame, and the amount of work that we have to perform per frame is small compared to contour-based methods. We found that the accuracy provided by this simple algorithm was sufficient for a useful vocabulary of motion and gestures.

7 Evaluation

In this section, we describe the tests that we performed in order to evaluate the robustness and accuracy of our proposed algorithm. 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 sec. in length; we used a window size $w_{\mathcal{M}} = 4$ and threshold $\epsilon = 0.00001$. Each sequence consisted of a simple structured behaviour; we annotated the frames manually to obtain the training model prototypes. These were were then clustered using 5 clusters for the grab, measure and point gestures, and 2 clusters for the simpler move model. The number of clusters was determined using the algorithm described by Singer and Warmuth [21]; the actual clustering was done with a simpler agglomerative scheme. In order to ensure that particle models stay within the cluster bounds, we also compute an oriented bounding box for each cluster. At runtime, the weight of a particle is decreased if the particle falls outside the bounds of the cluster it was sampled from.

Our first test was designed to evaluate the robustness of the algorithm when detecting which gesture is observed in a particular frame. We used a particle filter with 800 particles and 3 annealing layers over a longer (1m30s) sequence of structured behaviour. Our results showed that the tracker failed to identify the correct gesture in 5 out of 625 frames. Figure 3 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 Figure 5. In order to detect tracking failure, we tested whether the pixel error (summed over all model joints) of the mode $\chi_k = x^j, w^j = max\left(w^i\right)$ exceeded the value of 10 pixels. As can be seen, the tracker successfully recovered each time the particles' memories were deleted.

We also evaluated the algorithm when using a first-order Markov model (FOMM) (as used in [10]) rather than a VLMM. Our results showed that the tracker failed to identify the correct gesture in 164 out of 625 frames. Figure 4 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. We found that in order to successfully learn such dependencies in the training data, memory lengths larger than 3 were needed. Using a memory length above 6 caused overfitting of the VLMM and poor performance when encountering unseen events.

To evaluate the accuracy of our proposed algorithm,

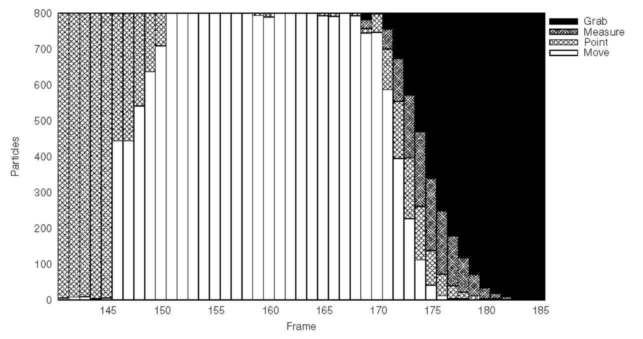


Figure 3: Number of particles tracking different gestures in a short example interval of one test sequence, using a VLMM.

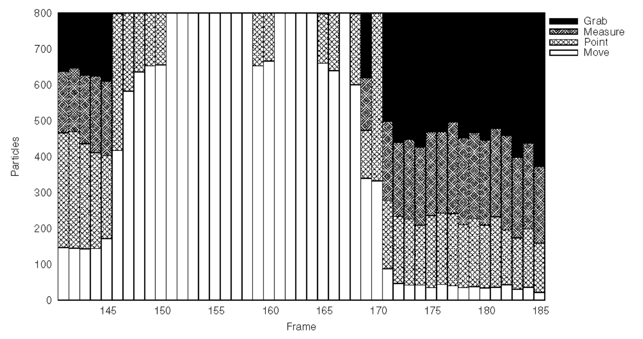


Figure 4: Number of particles tracking different gestures in a short example interval of one test sequence, using a FOMM.

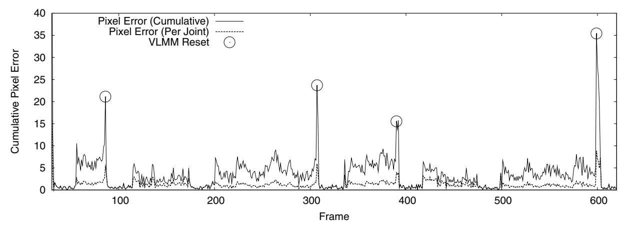


Figure 5: Cumulative pixel error (summed over all model joints) for one whole test sequence of structured behaviour using a VLMM.

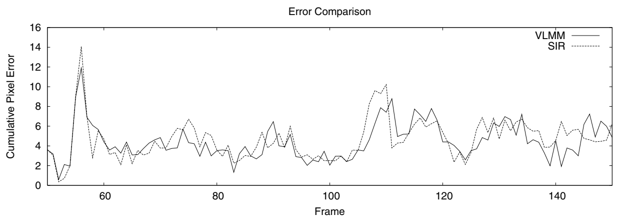


Figure 6: Comparison of cumulative pixel errors using VLMM and SIR for a test sequence of unstructured behaviour with a single gesture; note similar performance.

we compared the model estimates for each frame with the ground truth data gathered by labelling. Figure 5 shows the pixel error summed over all model joints for the whole test sequence. The frames where the particles' VLMM memories were reset as described in Section 4.1 are indicated with circles. The pixel error over a single joint averaged 1 pixel, which is sufficient for most human-computer interaction applications.

Lastly, we used a separate sequence of unstructured behaviour (i.e. random movements not observed during training) to investigate what impact using a VLMM has 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 configuration with the SIR scheme. The results are presented in Figure 6. The pixel errors are similar for both approaches, which shows that using a VLMM does not have a negative impact when tracking unstructured behaviour.

8 User evaluation

To test the viability of visual tracking with our algorithm as an input system for virtual reality, we used a standard Fitts' law task. There were 10 participants in total, 3 female and 7 male, all right-handed and experienced computer users. The participants in the experiment started at a fixed position, acquired a red square object and moved it to a green square target of the same size (see Figure 7). 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, a mouse was used for dragging and dropping the object in a standard fashion. The sensitivity of the mouse was adjusted so that the sizes of the user workspace in all cases were identical. We measured separately the times taken in the acquisition and dragging sub-tasks.

The independent variables were the dragging distance $D = \langle 8, 16, 24 \rangle$ units (1 unit = 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 $AA = \langle 0^o, 90^o, 180^o, 270^o \rangle$; the distance from the starting position to the object depended on the approach angle. Since the targets were square, we used the MacKenzie formulation of Fitts' law to compute the ID, rather than some of the more general 2D models. The 24 possible cases were randomised, and the participants repeated the experiments under three different conditions - first with visual tracking, next with mouse input delayed by 40ms and frame rate constrained to 25 frames per second, and finally with normal mouse input. The 40ms lag was introduced so as to account for the latency of the input images (we send the captured frames over the network in order to support multiple camera input in our system). Similarly, the constraint of 25 FPS was designed to match the frame rate which is sustained over the network.

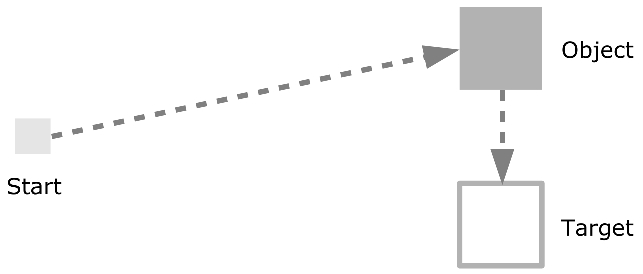


Figure 7: An example test case from the Fitts' law task used for evaluating user performance.

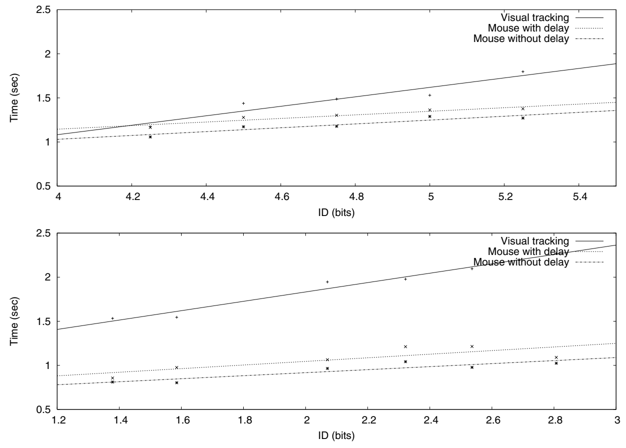


Figure 8: Time taken to complete the acquire (top) and drag (bottom) tasks as a function of the index of difficulty.

Figure 8 shows the regression lines for the average time taken to complete the tasks as a function of the index of difficulty ID. The overall trend 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 when using

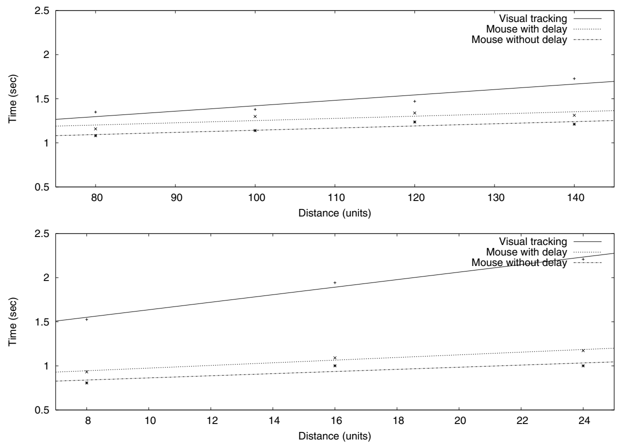


Figure 9: Time taken to complete the acquire (top) and drag (bottom) tasks as a function of the distance. (1 unit = 8 pixels)

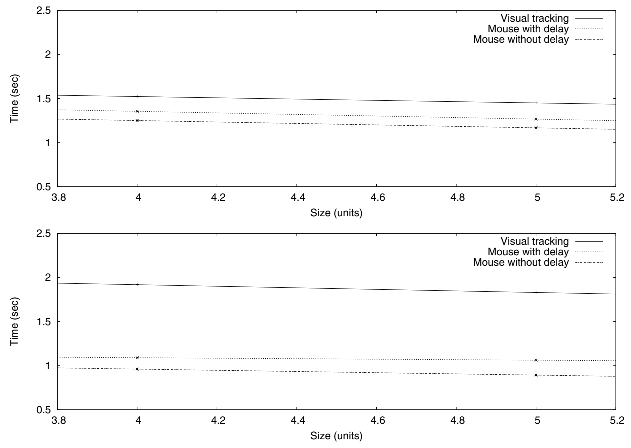


Figure 10: Time taken to complete the acquire (top) and drag (bottom) tasks as a function of the size. (1 unit = 8 pixels)

the mouse. However, for larger distances, the time is longer by over 0.5 sec. This is accounted for by the fact that the particles had to travel a longer path, and consequently a longer time was required for the initial estimate to obtain a large enough support map in the particle set. For the dragging task, we observe that completion time with visual tracking is at most 1.5 sec. longer than with the other input schemes. This is due to the lack of tactile feedback when moving the object, which some users complained of. Two participants reported hand fatigue at the end of the visual tracking test.

Figures 9 and 10 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 previous research [5, 1] which shows that user performance decreases as input delay increases.

Finally, Figure 11 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. As might be expected, the lowest number of such errors were made with the mouse without any delay. The number of errors increases by four when the latency from the camera is introduced. With visual tracking, the number of errors doubles over that observed with delayed mouse input.

To explain the performance of the participants, we analysed the latency introduced by the various stages of the visual tracking pipeline. The results of our analysis are shown in Figure 12. The time for the current frame to be compressed and sent over the network is approximately 40ms.

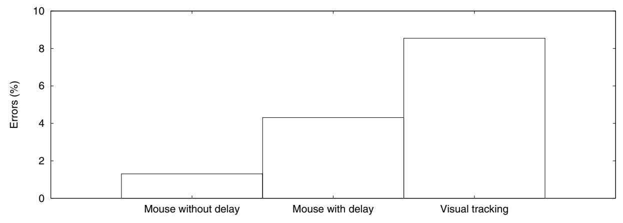
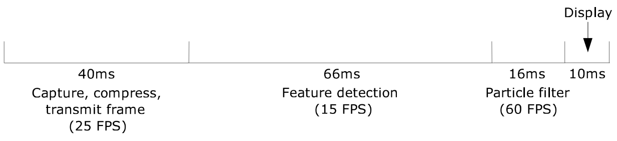


Figure 11: Percentage of errors for all experiments.



Display
40ms
Capture, compress, transmit frame (25 FPS)
66ms
Feature detection (15 FPS)
16ms
Particle filter (60 FPS)
10ms

Figure 12: Latency analysis.

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 actual drawing routines and the time it takes for the image to be displayed on the computer screen with a refresh rate of 80Hz. 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 as demonstrated in Section 6, the tracker is robust to failure and operates in real time, the fact that such high latency is introduced results in degraded user performance.

9 Discussion and future work

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 16ms.

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 acqusition, processing and display pipeline, has a detrimental effect on user task performance. The effects of latency are often ignored; some papers concentrate on framerate, 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 what seems natural may not, in fact, be the most effective design; this too warrants further exploration.

As well as exploring these aspects, we intend to investigate automating the acquisition of data for training the VLMM, by acquiring posture information from video, or from a data glove.

References

[1] R.S. Allison, L.R. Harris, M. Jenkin, U. Jasiobedzka, and J.E. Zacher. Tolerance of temporal delay in virtual environments. *Proc. IEEE Conf. Virtual Reality*, pages 247–254, 2001.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-Gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, pages 100–117, 2001.

[3] S. Card, W. English, and B. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys and text keys for text selection on a CRT. *Ergonomics*, 21:601–603, 1978.

[4] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2:126–133, 2000.

[5] S.R. Ellis, M. J. Young, B. D. Adelstein, and S. M. Ehrlich. Discrimination of changes in latency during voluntary hand movement of virtual objects. *Proc. Human Factors and Ergonomics Society*, pages 1182–1186, 1999.

[6] A. Galata, A. G. Cohn, D. Magee, and D. Hogg. Modeling interaction using learnt qualitative spatio-temporal relations and variable length Markov models. *Proc. European Conference on Artificial Intelligence*, pages 741–745, 2002.

[7] A. Galata, N. Johnson, and D. Hogg. Learning variable-length markov models of behavior. *Computer Vision and Image Understanding 81*, pages 398–413, 2001.

[8] N. Gordon, J. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F*, pages 107–113, 1993.

[9] I. Guyon and F. Pereira. Design of a linguistic postprocessor using variable length models. *Proc. Intl. Conf. on Document Analysis and Recognition*, pages 454–457, 1995.

[10] T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. *Proc. 6th Int. Conf. on Computer Vision*, pages 344–349, 1998.

[11] M. Isard and A. Blake. Visual tracking by stochastic propagation of conditional density. *Proc. 4th European Conf. Computer Vision*, pages 343–356, 1996.

[12] M. Isard and A. Blake. A mixed-state Condensation tracker with automatic model-switching. *Proc. 6th Int. Conf. on Computer Vision*, pages 107–113, 1998.

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

[14] K.Abe, H.Saito, and S.Ozawa. Virtual 3-d interface system via hand motion recognition from two cameras. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A, Vol.32, No.4:536–540, 2002.

[15] G. Kitagawa and W. Gersch. Smoothness priors analysis of time series. *Lecture Notes in Statistics*, 116, 1996.

[16] I. MacKenzie. A note on the information-theoretic basis for Fitts' law. *Journal of Motor Behavior*, 21:323–320, 1989.

[17] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, 2002.

[18] J. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. *Third European Conf. on Computer Vision*, pages 35–46, 1994.

[19] D. Ron, S. Singer, and N. Tishby. The power of amnesia. *Advances in Neural Information Processing Systems*, 6:176–183, 1994.

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

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

[22] N. Stefanov and R. Hubbold. A robust, self-initialising, real-time hand tracker. *Proc. IEE Intl. Conf. on Visual Information Engineering (to appear)*, 2005.

[23] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Learning a kinematic prior for tree-based filtering. *Proc. British Machine Vision Conference*, September 2003.

[24] D. Vernon. *Machine Vision*. Prentice-Hall, 1991. Chap. 6.

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