Recognition of Deictic Gestures for Wearable Computing

Moeslund, Thomas B.; Nørgaard, Lau

Published in:
Gesture in Human-Computer Interaction and Simulation

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Moeslund, T. B., & Nørgaard, L. (2006). Recognition of Deictic Gestures for Wearable Computing. In S. Gibet, N. Courty, & J. F. Kamp (Eds.), *Gesture in Human-Computer Interaction and Simulation* (pp. 112-123). IEEE Computer Society Press.

Recognition of Deictic Gestures for Wearable Computing

Thomas B. Moeslund and Lau Nørgaard

Laboratory of Computer Vision and Media Technology,
Aalborg University, Denmark
tbm@cvmt.dk

Abstract. In modern society there is an increasing demand to access, record and manipulate large amounts of information. This has inspired a new approach to thinking about and designing personal computers, where the ultimate goal is to produce a truly wearable computer. In this work we present a non-invasive hand-gesture recognition system aimed at deictic gestures. Our system is based on the powerful Sequential Monte Carlo framework which is enhanced with respect to increased robustness. This is achieved by using ratios in the likelihood function together with two image cues: edges and skin color. The system proves to be fast, robust towards noise, and quick to lock on to the object (hand). All of which is achieved without the use of special lighting or special markers on the hands, hence our system is a non-invasive solution.

1 Introduction

In modern society there is an increasing demand to access, record and manipulate large amounts of information involved in many aspects of professional and private daily life. This has inspired a new approach to thinking about and designing personal computers, where the ultimate goal is to produce a truly wearable computer. Wearable in the sense of being a natural extension of the body like clothes, shoes or glasses. A brief historic overview of wearable computing is listed below, see [11] for further details.

1268 Earliest recorded mention of eyeglasses
1665 Robert Hooke calls for augmented senses
1762 John Harrison invents the pocket watch
1907 Aviator Alberto Santos-Dumont commissions the creation of the first wristwatch
1960 Heilig patents a head-mounted stereophonic TV display
1960 Manfred Clynes coins the word "Cyborg"
1961 Edward O. Thorp and Claude Shannon (MIT) builds the first wearable computer, which is used to predict roulette wheels
1966 Ivan Sutherland creates the first computer-based head-mounted display (HMD)
1977 Hewlett-Packard releases the HP 01 algebraic calculator watch
1979 Sony introduces the Walkman
1981 Steve Mann designs backpack-mounted computer to control photographic equipment

S. Gibet, N. Courty, and J.-F. Kamp (Eds.): GW 2005, LNAI 3881, pp. 112–123, 2006.

1985 (Wearable) Device for prediction or card counting in casino games were outlawed in the state of Nevada
1989 Private Eye's HMD sold by Reflection Technology
1991 Doug Platt debuts his 286-based "Hip-PC"
1991 Carnegie Mellon University (CMU) team develops VuMan 1 for viewing and browsing blueprint data
1993 Thad Starner starts constantly wearing his computer, based on Doug Platt's design
1993 Feiner, MacIntyre, and Seligmann develop the KARMA augmented reality system
1994 Lamming and Flynn develop "Forget-Me-Not" system, a continuous personal recording system
1996 Boeing hosts "Wearables Conference" in Seattle
1997 CMU, MIT, and Georgia Tech co-host the first IEEE International Symposium on Wearable Computers
1997 First Wearable computer fashion show at MIT

1.1 Related Work

A number of different devices have been developed or adopted to the special user interface requirements in wearable computing [9]. Depending on the context the requirements differ. However, one common issue in most wearable computing interfaces is the need for a pointing device, similar to the computer mouse used in standard WIMP interfaces. Without it, precise deictic interaction is either not possible or very cumbersome.

The most common way of achieving this is by the use of a data glove, see e.g., [10]. Glove-based methods are by nature intrusive and besides often too expensive and bulky for widespread use [13]. Other intrusive devices which have been used to provide deictic input are bend sensors [13], ultrasonic devices [4], and accelerometers [13]. Less intrusive methods are based on head-mounted cameras segmenting the hand(s) in the image. Compared to some of the more intrusive devices cameras in general produce poor signal-to-noise ratios and therefore either infrared light and cameras, see e.g., [14] and [12], or markers on the hands/fingers, see e.g., [10], are used. For further information on state-of-the-art see [8].

1.2 The Content of This Paper

The aim of this paper is to develop a head mounted camera-based gesture interface for wearable computing that neither requires special lighting (infrared) nor markers attached to the hands/fingers. Our approach is to adopt an advanced tracking framework: the Sequential Monte Carlo (SMC) method [3], which is often used in the computer vision research field, see e.g., [5][1][2], and tailor it to the needs originating when the camera is head mounted.

The paper is structured as follows. In section 2 the gestures are defined and a representation is derived. In section 3 the tracking framework for the gesture recognition is presented. In section 4 and section 5 the recognition of the pointing gesture is described. In section 6 the system is tested and in section 7 a conclusion is given.

2 Modeling the Pointing Gesture

We require two gestures to be recognizable, pointing and clicking. The former is defined as an outstretched index finger and thumb with all other fingers bend. The latter is defined in the same way except that the thumb is now held against the index finger, see figure 1. As only the movement of the thumb is modeled explicitly and the other fingers are kept still or hidden, the range of internal motion is limited to the three joints of the thumb, see figure 1. As the DIP (Distal Interphalangeal) joint is always outstretched when doing the click gesture, and since the MCP (Meta CarpoPhalangeal) joint is difficult to bend independently of the CMC (Carpal MetaCarpal) joint, the two gestures can be represented and distinguished by one only one rotational DoF (degree of freedom).

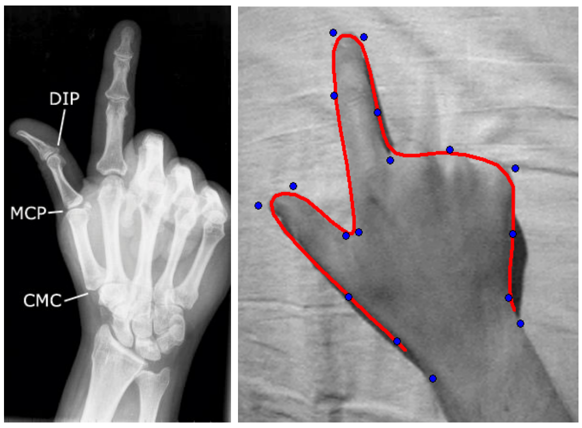

Fig. 1. A hand showing the point gesture. Left: X-ray image. Right: B-spline approximation of the contour. Control points are marked with circles.

We represent the appearance of the two gestures by the contour of the hand, see figure 1. The contour is manually created using B-splines defined by a set of control points. Applying a linear transformation to these points is the same as applying the transformation to the B-splines.

To create the set of two basis splines required to model the two gestures, a spline was first fitted to an image of the pointing gesture and saved to an ASCII file. Then an image with the hand in the same position but showing a clicking gesture was loaded, and the control points moved to make the spline follow the thumb. These two splines were then loaded into the tracker and combined using linear interpolation. So one parameter, ϕ, controls the internal DoF in our hand model.

Regarding the external DoF for the hand model, we assumed weak perspective transformation and applied an affine transformation. So the final external transformation of the B-spline contour is given as:

$$\mathbf{r}(s) = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} s_x\, cos(\theta) & a\, s_y\, cos(\theta) - s_y\, sin(\theta) \\ s_x\, sin(\theta) & a\, s_y\, sin(\theta) + s_y\, cos(\theta) \end{bmatrix} \mathbf{r}_0(s) \quad (1)$$

where $\mathbf{r}_0(s)$ is the set of untransformed control points representing the contour of the hand, $\mathbf{r}(s)$ is the set of affine transformed control points, d_x and d_y define the translation in the image plane, s_x and s_y define the scaling in the image plane, a is the shear, and θ is the rotation in the image plane.

In total we ended up with 7 DoF, one internal and six externals. That is, our state-space is seven-dimensional and one configuration of the hand is defined by the state vector, $\mathbf{x}$, as:

$$\mathbf{x} = (d_x, d_y, \theta, s_x, s_y, a, \phi) \tag{2}$$

where ϕ is the angle between the index finger and the thumb.

3 Tracking Framework

Due to the low signal-to-noise ratio mentioned in section 1.1 the hand can not always be segmented perfectly from the background. Hence, the recognition in a particular frame will not always be unique, or in statically terms, the conditional probability of a gesture given the image measurements will in general be multi modal. This calls for a Sequential Monte Carlo (SMC) method which can handle such situations [3]. The SMC algorithm operates as most other tracking frameworks, by using a predict-match-update structure.

The SMC is defined in terms of Bayes' rule and by using the first order Markov assumption. That is, the posterior PDF (probability density function) is proportional to the observation PDF multiplied by the prior PDF, where the prior PDF is the predicted posterior PDF from time $t-1$:

$$p(\mathbf{x}_t|\mathbf{u}_t) \propto p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{u}_{t-1}) \tag{3}$$

where $\mathbf{x}$ is the state and $\mathbf{u}$ contains the image measurements. The predicted posterior PDF is defined as

$$p(\mathbf{x}_t|\mathbf{u}_{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{u}_{t-1})\,\mathrm{d}\mathbf{x}_{t-1} \tag{4}$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the motion model governing the dynamics of the tracked object, i.e., the prediction, and $p(\mathbf{x}_{t-1}|\mathbf{u}_{t-1})$ is the posterior PDF from the previous frame.

This formula is exactly what we are after, as it imposes no constraints on the posterior PDF, as for example the Kalman filter does. However, even with a coarse resolution for the different parameters in the state vector (both internal and external DoF), too many combinations exist and it is not computationally feasible to evaluate the integral in equation 4. Therefore, the SMC algorithm approximates the posterior by only sampling the N most appropriate combinations. In praxis this is done by estimating $p(\mathbf{x}_t|\mathbf{u}_t)$ by selecting a number, N, of (hopefully) representative states (particles) from $p(\mathbf{x}_{t-1}|\mathbf{u}_{t-1})$, predicting these using $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and finally giving each particle a weight in accordance with the observation PDF, $p(\mathbf{u}_t|\mathbf{x}_t)$. For the next time step N new particles are drawn from the existing set with probabilities proportional to the calculated weights.

This approach will concentrate most particles around the likely hypotheses, but since the prediction contains both a deterministic and a stochastic element, some particles will also spread out randomly making the SMC algorithm able to cope with unpredicted events.

4 Motion Model

In order to apply the SMC method we need to be able to predict a state vector over time. In this section the motion model, which implements the prediction, is defined.

We assume the individual dimensions in the state space are independent, and can be modeled by first order auto-regressive (AR) processes:

$$x_t - \overline{x} = a(x_{t-1} - \overline{x}) + bw_t \Leftrightarrow x_t = \overline{x} + a(x_{t-1} - \overline{x}) + bw_t \tag{5}$$

where x_t is the value at time t, $\overline{x}$ is the mean or expected value and w_t is a random variable with distribution $\mathcal{N}(0,1)$.

The only exception is x and y translation, d_x and d_y, which are modeled by second order AR processes with constant velocity:

$$x_t = x_{t-1} + (x_{t-1} - x_{t-2}) + bw_t \tag{6}$$

Consequently there are 12 motion constants to be determined during training. In addition, the mean and standard deviation for each dimension are also calculated in order to determine the *a priori* distributions used in the initialization of the SMC tracker [9].

The motion model and initialization assume the value and step size along each dimension of the state space to be normally distributed except d_x and d_y which are assumed uniform. Test justify these assumptions except for the value of the thumb angle, ϕ, which is not normally distributed [9]. Its histogram has two distinct modes at positions corresponding to the point and click gestures. We handle this by modeling the two modes by two first order AR processes, as in equation 5, and extend the state vector with a variable indicating the current mode (gesture) [6]:

$$\mathbf{x}' = (\mathbf{x}, \gamma)\ , \gamma \in \{1, 2\} \tag{7}$$

The motion model for the thumb angle is modified to include the modes:

$$p(\phi_t|\phi_{t-1}) = p(\phi_t|\gamma_t, \phi_{t-1})p(\gamma_t|\phi_{t-1}) \tag{8}$$

where $p(\phi_t|\gamma_t, \phi_{t-1})$ is one of the two AR processes and $p(\gamma_t|\phi_{t-1})$ is the probability for a gesture given the angle value in the last frame x_{t-1}. This last probability represents the knowledge of when the hand changes from one gesture to the other. This model will accurately follow the movement of the thumb through fast clicks with a reasonable amount of particles. Note that the representation in equation 7 means that the gesture recognition problem becomes an explicit part of the tracking process, i.e., when the tracking is correct so is the recognized gesture.

5 Likelihood Function

When the contour description of the tracked object has been created and the changes in state from frame to frame can be predicted, comparing the predicted contours with the actual images is the next step. In the SMC tracker this is represented by the observation PDF described in this section.

This comparison is accomplished by locating edges in the image and examining how well each predicted contour corresponds to these edges. The edges are located by searching along a number of normals to the contour. Edges detected on the normals are referred to as *features*. The function that measures the correspondence is called a likelihood function and defined via a generative model.

In this section we describe a generative model inspired by the order statistic likelihood in [7]. However, where the order statistic likelihood only counts the total number of features found along the different normals, the likelihood proposed in this section will utilize individual counts of interior and exterior features, see figure 2, and we hereby obtain an IEOS (interior-exterior order statistic) likelihood function.

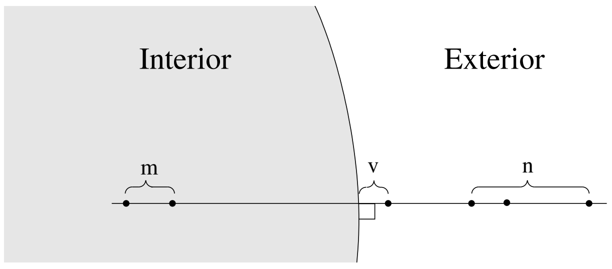

Fig. 2. Illustration of the different types of features. m is the number of features inside the hand, n is the number of features in the background, and v is the position of the feature nearest to the contour.

Both exterior and interior features are assumed to be generated by two individual Poisson processes with density parameters λ and μ, respectively. As no *a priori* knowledge exist regarding local properties of the background λ is considered constant over the entire image. μ may vary between normals to model the presence of known interior features. The density for the individual normals μ_i can be learned from training data. The probability $b_L(n)$ of finding n features on a piece of a measurement line of length L placed on the background is:

$$b_L(n) = e^{(-\lambda L)} \frac{(\lambda L)^n}{n!} \tag{9}$$

Similarly the probability of detecting m features on a piece of a measurement line of length L lying entirely inside the object at the position of normal number i is:

$$f_{L,i}(m) = e^{(-\mu_i L)} \frac{(\mu_i L)^m}{m!} \tag{10}$$

The positions of the n exterior features $\{b_1, b_2, \ldots b_n\}$ or the m interior features $\{c_1, c_2, \ldots c_m\}$ are considered uniformly distributed.

The edge of the hand is assumed to produce a single feature with a position normally distributed around the center of the normal. There is a fixed probability q_0 of this feature not being detected.

The generative model for the i'th normal with length L can be defined as:

1. Draw a from the truncated Gaussian

$$\mathcal{G}(a) = \begin{cases} ce^{(-\frac{a^2}{2\sigma^2})} & \text{, for } a \in [-L/2; L/2] \\ 0 & \text{, otherwise} \end{cases} \tag{11}$$

   where a is the distance from the correct edge (originating from the hand) to the center of the normal, σ is found during training, and c is set so that the CDF (cumulative density function) of $\mathcal{G}(a)$ integrates to one.
2. Draw d randomly from $\{\texttt{True}, \texttt{False}\}$ with probabilities $1 - q_0$ and q_0, respectively. d represents whether the edge of the object was detected or not.
3. Draw the number of interior features m randomly from $f_{L/2+a,i}(m)$, and draw their positions $\{c_1, c_2, \dots c_m\}$ from $\text{Rect}[-L/2, a]$.
4. Draw the number of exterior features n randomly from $b_{L/2-a}(n)$, and draw their positions $\{b_1, b_2, \dots b_n\}$ from $\text{Rect}[a, L/2]$.
5. If d is `True`:
   (a) Set v to the position of the most central feature in the set $\{c_1, c_2, \dots c_m, a, b_1, b_2, \dots b_n\}$.
   If d is `False`:
   (a) Set v to the position of the most central feature in the set $\{c_1, c_2, \dots c_m, b_1, b_2, \dots b_n\}$.
   (b) If $v \in \{c_1, c_2, \dots c_m\}$: Set m = m -1. Otherwise: Set n = n - 1
6. Report $\{v, m, n\}$.

The derivation of the likelihood function is divided into the two cases. One where the object edge is detected ($d = \texttt{True}$) and one where it is not ($d = \texttt{False}$).

5.1 Edge Not Found ($d = \texttt{False}$)

As v is the distance to the center of the most central of the $m + n + 1$ features found, all other features must have a distance greater than or equal to v. The PDF for the position of the most central feature can not be found directly. It will be established by determining the corresponding CDF and then differentiating.

The probability of the distance from the center to a single feature being greater than or equal to y is[1]:

$$P(|v| \geq y) = (1 - y2/L) \tag{12}$$

As the positions of the features are assumed independent, the combined probability, that k features all lie at distances from the center greater than or equal to y, can be calculated as a product of the k individual probabilities:

$$P(|v| \geq y) = (1 - y2/L)^k \tag{13}$$

[1] For the following four equations: $y \in [0; L/2]$.

The CDF $F(y)$ for the position of the most central of k features from a uniform distribution can be found from equation 13:

$$F(y) = P(|v| \leq y) = 1 - (1 - y2/L)^k \tag{14}$$

Differentiating 14 with regard to y yields:

$$\frac{d}{dy}F(y) = \frac{d}{dy}(1 - (1 - y2/L)^k) = \frac{2k}{L}(1 - y2/L)^{k-1} \tag{15}$$

As $|v|$ will always be in the interval $[0; L/2]$, the resulting PDF is:

$$p(v|d = \texttt{False}) = \frac{2k}{L}(1 - |v|2/L)^{k-1} \tag{16}$$

The probability of getting m interior- and n exterior features on the i'th normal, if it is centered on the border of the object, can be calculated as:

$$p_i(m, n|d = \texttt{False}) = f_{L/2,i}(m)b_{L/2}(n) \tag{17}$$

The distance from the center of the normal from the most central feature v is not used, as this feature is known to be either an interior or exterior feature and not from the edge of the object. However it is not accounted for in m or n, and it will have to be added to the right category. If the feature at v lies on the interior part of the normal, it should count as an interior feature or as an exterior feature if it lies on the outside part. That is if $v < 0$ then set $m = m + 1$ otherwise set $n = n + 1$. Adding this to equation 17 yields:

$$p_i(m, n|d = \texttt{False}, v) = \begin{cases} f_{L/2,i}(m+1)b_{L/2}(n) & \text{, if } v < 0 \\ f_{L/2,i}(m)b_{L/2}(n+1) & \text{, if } v \geq 0 \end{cases} \tag{18}$$

Equations 16 and 18 can be combined to form the likelihood that the i'th normal, centered on the border of the object, will produce $m + n + 1$ features where the most central is at position v:

$$\begin{aligned} p_i(v, m, n|d = \texttt{False}) &= p_i(m, n|d = \texttt{False}, v)p(v|d = \texttt{False}) = \\ &\begin{cases} f_{L/2,i}(m+1)b_{L/2}(n)\frac{2(m+n+1)}{L}\cdot \\ \quad (1 - |v|2/L)^{m+n} & \text{, if } v < 0 \\ f_{L/2,i}(m)b_{L/2}(n+1)\frac{2(m+n+1)}{L}\cdot \\ \quad (1 - |v|2/L)^{m+n} & \text{, if } v \geq 0 \end{cases} \end{aligned} \tag{19}$$

The procedure for the case where the edge is found on the contour $(d = \texttt{True})$ follows a similar pattern [9] and results in:

$$\begin{aligned} p_i(v, m, n|d = \texttt{True}) =& \left(\frac{2(m+n)}{L}(1 - |v|2/L)^{m+n-1}\left(1 - \int_{-|v|}^{|v|}\mathcal{G}(a)da\right)\right. \\ &\left. +2(1 - |v|2/L)^{m+n}\mathcal{G}(|v|)\right)f_{L/2,i}(m)b_{L/2}(n) \end{aligned} \tag{20}$$

where v is the distance from the center of the normal to the most central feature. Combining equations 19 and 20 we obtain the IEOS likelihood function for the $i'th$ normal:

$$p_{ieos_i}(\mathbf{x}) = p_i(v, m, n) = \\ q_0 p_i(v, m, n | d = \texttt{False}) + (1 - q_0) p_i(v, m, n | d = \texttt{True}) \tag{21}$$

Assuming independence of the normals, the likelihood of the entire contour corresponding to the state vector $\mathbf{x}'$ is:

$$\mathcal{P}_{ieos}(\mathbf{x}') = \prod_{i=1}^{M} p_{ieos_i}(\mathbf{x}') \tag{22}$$

where M is the total number of normals on the contour investigated for one predicted particle (state). Equation 22 is too long to be stated in its entirety, and consequently seems to be a very costly expression to evaluate. However, this is not the case as most terms can be reduced to lookup tables [9].

5.2 Creating a Likelihood Ratio

The generative models described in the previous section form the basis for the contour likelihood functions. They can, however, also be used to derive background likelihood functions, that is, functions expressing the likelihood that a given set of features was produced by the background. Based on the generative model for the IEOS likelihood function the background likelihood for a single normal will be:

$$p_0 = b_L(f) \frac{2(f)}{L} (1 - |v| 2/L)^{m+n} \tag{23}$$

where $f = m + n + 1$. The corresponding likelihood for all M normals on the entire contour is:

$$\mathcal{B}_{ieos}(\mathbf{x}') = \prod_{i=1}^{M} b_L(f_i) \frac{2(f_i)}{L} (1 - |v_i| 2/L)^{m_i+n_i} \tag{24}$$

where $f_i = m_i + n_i + 1$. The likelihood function can now be expressed as a ratio which is more robust to noise:

$$\mathcal{R}_{ieos}(\mathbf{x}') = \frac{\mathcal{P}_{ieos}(\mathbf{x}')}{\mathcal{B}_{ieos}(\mathbf{x}')} \tag{25}$$

5.3 Adding Color Information

In order to improve the likelihood function we learn the hue and saturation values of hands and model the colors by a Gaussian distribution. The distribution in the background is assumed to be uniform over both hue and saturation. Given these assumptions we can derive a color based ratio between the likelihood of a contour matching and the likelihood of the contour being located on a random background [9]. This color based likelihood ratio is denoted $\mathcal{R}_{color}(\mathbf{x}')$ and together with equation 25 it forms the final likelihood function used in this work:

$$\mathcal{B}_{ieosc}(\mathbf{x}') = \mathcal{R}_{ieos}(\mathbf{x}') \mathcal{R}_{color}(\mathbf{x}') \tag{26}$$

6 Results

The HW used to test our system was the Sony Glasstron PLM-S700E HMD, a Philips ToUcam Pro USB mounted on the HMD, and a Windows PC with AMD Athlon 2100+ and 512 MB RAM. With this HW we recorded different test sequences each having different characteristics among the following: translation and rotation of the hand [slow, moderate, fast, very fast], head movement [none, slow, fast], illumination [indoor, outdoor, mixed], and background [uniform wall, wooden table, very cluttered desk].

In general the tracking of the hand and the recognition of the current gesture works well and especially the index finger is tracked reliably. In figure 3 successful tracking is illustrated for a challenging sequence with very cluttered background, fast hand movements, and additional hands entering the scene. The very fast motion of the hand and head combined with low lighting conditions cause a high level of motion blur resulting in weak edges. For a few frames with excessive motion blur, the tracker reports an erroneous position of the hand. However, track of the hand is not lost due to the stochastic nature of the SMC framework, and precise lock is regained only two frames later.

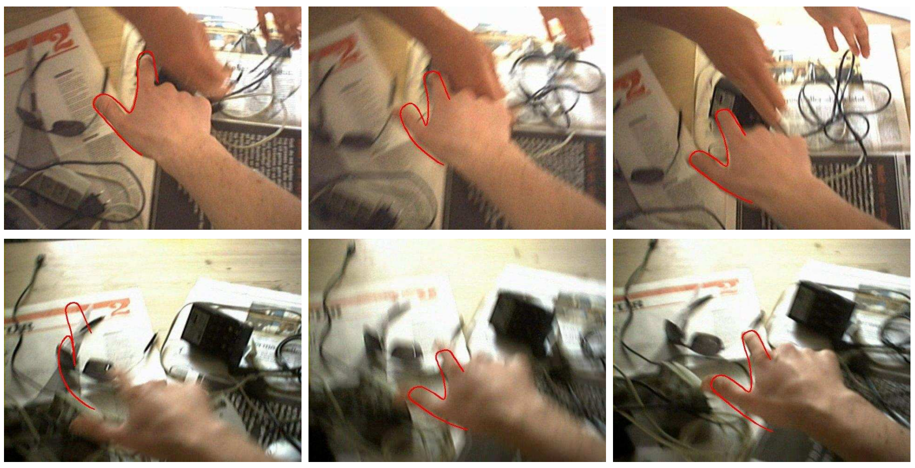

Fig. 3. Images form a test sequence. The state of the tracker at a particular frame is calculated as the weighted average of all particles and overlaid in the figure.

The speed of our system depends primarily on the number of particles required to get a robust tracking. This number in general varies in accordance with the complexity of the scene. For our test sequences the number of required particles is between 100 and 600. This corresponds to a frame rate of $100Hz$ to $12Hz$, which for this application can be considered real-time.

In order to get quantitative results we compared the estimated state (contour) with hand-segmented data and calculated the difference. In average the mean pixel error and standard deviation are around 8 and 3 for the point gesture, respectively, and around 5 and 3 for the click gesture, respectively. This is found to be usable for most interface purposes. To stress this point we made a qualitative test where test persons were asked to control the interaction with the game pieces while playing Tic-Tack-Toe against the computer. On a uniform background the game is definitely playable. A few erroneous clicks appeared when playing on a cluttered background, especially in combination with fast hand movements. These clicks were few and could for the most parts be eliminated by a temporal filter.

Fast head motion was in general not a problem. It was observed, that during interaction the hand was kept relatively steady wrt to the head. It seems not plausible to move the head independently of the hand while pointing at something shown on the HMD.

Another very interesting issue to test is the benefits of including color information into the likelihood function. As the tracker always produces an output, i.e., a best state of the object at a particular frame, we made a sequence where the hand moves in and out of the field of view. In figure 4 the likelihoods of the best state with (left) and without (right) color information are shown. The exact values of the likelihoods are difficult to interpret as they depend on the appearance of both the object and the background. But it is evident that when the hand is not in the image (indicated by the dots) the likelihood values drop significantly when color information is used. This seems reasonable as the background is not likely to contain any hand-colors in the shape of a hand. In other words, our likelihood function provides a better signal-to-noise ratio for the system, and hence a better tracking result.

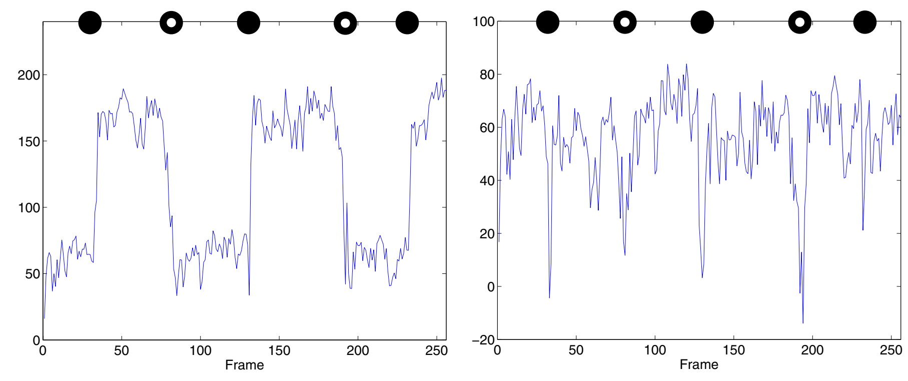

Fig. 4. The likelihood functions (left: equation 26, right: equation 22) as a function of the frame number. The black dots indicate when the hand enters the field of view while the black/white dots indicate when the hand leaves the field of view.

The clear difference in the values as a function of whether the tracker has locked on an object or not can also be used to decide when the tracker should re-initialize, i.e., use more particles or draw particles randomly from the a priori distribution. Furthermore, the steepness of the transitions in the left figure illustrates how fast the tracker regains lock when the hand reappears. In quantitative terms, the average number of frames required to regain lock is 4 for a uniform background and 7 for a cluttered background.

7 Conclusion

We have presented a non-invasive hand-gesture recognition algorithm for wearable computing. The recognized gestures are point and click gestures which are essential in virtually all interfaces where deictic information is required. Our algorithm is based on the powerful SMC tracker which can handle multiple hypotheses in the search space. In order to increase the robustness of the tracker we use ratios in the likelihood function and base it on two image cues which can complement each other: edges and skin color. The likelihood function proves to be very robust towards noise as illustrated in figure 3. Furthermore, as illustrated in figure 4 the algorithm locks on to the object very quickly and gives a clear indication of whether the hand is present in the frame or not.

The above mentioned characteristics combined with the speed of the algorithm and the user feedback allow us to conclude that we have developed a powerful deictic interface for wearable computing, and that is without requiring special lighting or markers on the hands or fingers, hence our system is a non-invasive solution.

References

1. T.J. Cham and J.M. Rehg. A Multiple Hypothesis Approach to Figure Tracking. In Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, 1999.
2. K. Choo and D.J. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In International Conference on Computer Vision, Vancouver, Canada, 2001.
3. A. Doucet, N. Freitas, and N. Gordon, editors. Sequential Monte Carlo Methods in Practice. Springer, 2001.
4. E. Foxlin and M. Harrington. Weartrack: A self-referenced head and hand tracker for wearable computers and portable vr. In International Symposium on Wearable Computing, Atlanta, Georgia, 2000.
5. M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. International Journal on Computer Vision, 29(1), 1998.
6. M. Isard and A. Blake. A mixed-state CONDENSATION tracker with automatic model-switching. In International Conference on Computer Vision, Bombay, India, 1998.
7. J. MacCormick. Stochastic Algorithms for Visual Tracking: Probabilistic Modelling and Stochastic Algorithms for Visual Localisation and Tracking. Springer, 2002.
8. T.B. Moeslund and L. Nrgaard. A Brief Overview of Hand Gestures used in Wearable Human Computer Interfaces. Technical Report CVMT 03-02, AAU, Denmark, 2003.
9. L. Nrgaard. Probabilistic hand tracking for wearable gesture interfaces. Masters thesis, Lab. of Computer Vision and Media Technology, Aalborg University, Denmark, 2003.
10. W. Piekarski and B.H. Thomas. The tinmith system: demonstrating new techniques for mobile augmented reality modelling. In Australasian conference on User interfaces, 2002.
11. B. Rhodes. A brief history of wearable computing. www.media.mit.edu/wearables/lizzy/timeline.html.
12. T. Starner, J. Auxier, D. Ashbrook, and M. Gandy. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In International Symposium on Wearable Computing, Atlanta, Georgia, 2000.
13. K. Tsukada and M. Yasumura. Ubi-Finger: Gesture Input Device for Mobile Use. In Asia Pacific Conference on Computer Human Interaction, Beijing, China, 2002.
14. N. Ukita, Y. Kono, and M. Kidode. Wearable vision interfaces: towards wearable information playing in daily life. In Workshop on Advanced Computing and Communicating Techniques for Wearable Information Playing, Nara, Japan, 2002.