PARTICLE FILTER IN VISION TRACKING

FILTRO DE PARTÍCULAS PARA EL SEGUIMIENTO VISUAL

Erik Cuevas J.[1, 2], Daniel Zaldivar N.[1, 2] and Raul Rojas[1]
cuevas@inf.fu-berlin.de / zaldivar@inf.fu-berlin.de / rojas@inf.fu-berlin.de

Recibido: julio 13, 2006 / Aceptado: enero 17, 2007 / Publicado: enero 25, 2007

ABSTRACT. The extended Kalman filter (EKF) has been used as the standard technique for performing recursive nonlinear estimation in vision tracking. In this work, we present an alternative filter with performance superior to that of the EKF. This algorithm, referred to as the Particle filter. Particle filtering was originally developed to track objects in clutter (multi-modal distribution). We present as results the filter behavior, in a case, when there are objects with similar characteristic to the object to track.

KEYWORDS. Artificial vision, robotics.

RESUMEN. El filtro de Kalman ha sido usado exitosamente en diferentes aplicaciones de predicción o determinación del estado de un sistema. Un campo importante en la visión por computadora es el seguimiento de objetos. Diferentes tipos de movimiento así como el ocultamiento de objetos a seguir pueden obstaculizar la labor de seguimiento. En este trabajo, se presenta el uso del filtro de Kalman para el seguimiento de objetos. Además se considera tanto la capacidad del filtro para tolerar pequeños ocultamientos del objeto así como el uso del filtro de Kalman extendido para el modelaje de movimientos complejos.

PALABRAS CLAVE. Visión artificial, robótica.

Introduction

The extended Kalman filter (EKF) has been used as the standard technique for performing recursive nonlinear estimation in vision tracking [1]. The EKF algorithm, however, provides only an approximation to optimal nonlinear estimation. In this work, we present an alternative filter with performance superior to that of the EKF. This algorithm is referred as the Particle filter. The basic difference between the EKF and Particle filter stems from the manner in which random variables are represented for propagating through system dynamics. In the EKF, the state distribution is approximated by gaussian random variable which is then propagated analytically through the first-order linearization of the nonlinear system. This can introduce large errors in the true posterior mean and covariance of the transformed gaussian random variables which may lead to suboptimal performance and sometimes divergence of the filter. The state distribution is approximated by random variable (not necessarily gaussian), but is now represented using minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the random variable and, when propagated through the true nonlinear system, captures the posterior mean and covariance accurately to second order (Taylor series expansion) for any nonlinearity. The EKF, in contrast,

[1] Freie Universität Berlin, Institut für Informatik, Berlin, Germany. – http://www.inf.fu-berlin.de/inst/ag-ki/ger/index.html

[2] División de Electrónica y Computación del Centro Universitario de Ciencias Exactas e Ingenierías de la Universidad de Guadalajara. Blvd. Marcelino García Barragán No. 1451, Guadalajara, Jalisco, 44430, México - http://depel.cucei.udg.mx/

only achieves first-order accuracy. No explicit Jacobian or Hessian calculations are necessary for the Particle filter. Remarkably, the computational complexity of the particle filter is the same order as that of the EKF.

Optimal Recursive Estimation

Given observations $\mathbf{y}_k$, the goal is to estimate the state $\mathbf{x}_k$. We make no assumptions about the nature of the system dynamics at this point. The optimal estimate in the minimum mean-squared error (MMSE) sense is given by the conditional mean:

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k | \mathbf{Y}_0^k] \qquad (1)$$

where $\mathbf{Y}_0^k$ is the sequence of observations up to time *k*. Evaluation of this expectation requires knowledge of the *a posteriori* density $p(\mathbf{x}_k | \mathbf{Y}_0^k)$. Given this density, we can determine not only the MMSE estimator, but any "best" estimator under a specified performance criterion. The problem of determining the *a posteriori* density is in general referred to as the Bayesian approach, and can be evaluated recursively according to the following relations:

$$p(\mathbf{x}_k | \mathbf{Y}_0^k) = \frac{p(\mathbf{x}_k | \mathbf{Y}_0^{k-1}) p(\mathbf{y}_k | \mathbf{x}_k)}{p(\mathbf{y}_k | \mathbf{Y}_0^{k-1})} \qquad (2)$$

where

$$p(\mathbf{x}_k | \mathbf{Y}_0^{k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{y}_k | \mathbf{x}_k) d\mathbf{x}_{k-1} \qquad (3)$$

and the normalizing constant $p(\mathbf{y}_k | \mathbf{Y}_0^{k-1})$ is given by

$$p(\mathbf{y}_k | \mathbf{Y}_0^{k-1}) = \int p(\mathbf{x}_k | \mathbf{Y}_0^{k-1}) p(\mathbf{y}_k | \mathbf{x}_k) d\mathbf{x}_k \qquad (4)$$

This recursion specifies the current state density as a function of the previous density and the most recent measurement data. The state-space model comes into play by specifying the state transition probability $p(\mathbf{x}_k | \mathbf{x}_{k-1})$ and measurement probability or likelihood, $p(\mathbf{y}_k | \mathbf{x}_k)$. Specifically, $p(\mathbf{x}_k | \mathbf{x}_{k-1})$ is determined by the process noise density $p(\mathbf{w}_k)$ with the state-update equation

$$\mathbf{x}_{k+1} = \mathbf{f}(k, \mathbf{x}_k) + \mathbf{w}_k \qquad (5)$$

For example, given an additive noise model with Gaussian density $p(\mathbf{w}_k) = \mathrm{N}(0, \mathbf{R}^{\mathrm{v}})$, then $p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \mathrm{N}(\mathbf{F}(\mathbf{x}_{k-1}), \mathbf{R}^{\mathrm{v}})$. Similarly, $p(\mathbf{y}_k | \mathbf{x}_k)$ is determined by the observation noise density $p(\mathbf{v}_k)$ and the measurement equation

$$\mathbf{y}_k = \mathbf{h}(k, \mathbf{x}_k) + \mathbf{v}_k \qquad (6)$$

In principle, knowledge of these densities and the initial condition $p(\mathbf{x}_0|\mathbf{y}_0) = \frac{p(\mathbf{y}_0|\mathbf{x}_0)p(\mathbf{x}_0)}{p(\mathbf{y}_0)}$ determines $p(\mathbf{x}_k|\mathbf{Y}_0^k)$ for all *k*. unfortunately, the multidimensional integration indicated by Eqs. (2)-(4) makes a closed-form solution intractable for most systems. The only general approach is to apply Monte Carlo sampling techniques that essentially convert integrals to finite sums, which converge to the true solution in the limit.

Particle filtering [2-4] was originally developed to track objects in clutter or a variable of interest as it evolves over time, typically with a non-Gaussian and potentially multi-modal probability density function (pdf). The basis of the method is to construct a sample-based representation of the entire pdf (equation 2). A series of actions are taken, each one modifying the state of the variable of interest according to some model (equation 5). Moreover at certain times an observation arrives that constrains the state of the variable of interest at that time.

Multiple hypothetical state (particles) of the variable of interest $\mathbf{x}_k$ are used, each one associated with a weight that signifies the quality of that specific particle. An estimate of the variable of interest is obtained by the weighted sum of all the particles. The particle filter algorithm is recursive in nature and operates in two phases: *prediction* and *update*. After each action, each particle is modified according to the existing model $\mathbf{f}(k, \mathbf{x}_k)$ *(prediction* stage), including the addition of random noise $\mathbf{w}_k$ in order to simulate the effect of noise on the variable of interest. Then, each particle's weight is re-evaluated based on the latest measurements available *(update* stage). At times the particles with small weights are eliminated, with a process called *resampling*. More formally, the variable of interest (in this case the object position $\mathbf{x}_k = [x_k \quad y_k]$) at time *k is* represented as a set of *M* samples (the "particles") ($S_k^i = [\mathbf{x}_k^i \quad b_k^i] : i = 1, 2, ..., M$), where the index *i* denotes the particle number, each particle consisting of a hypothetical value of the variable of interest $\mathbf{x}_k$ and a weight *b* that defines the contribution of this particle to the overall estimate of the variable, where $\sum_{i=1}^{M} b_k^i = 1$. The figure 1 shown the process carried out by the particle filter.

If at time *k* we know the pdf of the system at the previous instant *k* -1 then we model the movement effect with $\mathbf{f}(k, \mathbf{x}_k)$ to obtain a prior of the pdf at time *k* (prediction). In other words, the *prediction* phase uses a model in order to simulate the effect that a movement has on the set of particles with the appropriate noise added $\mathbf{w}_k$. The *update* phase uses the information obtained from the measurements to update the particle weights in order to accurately describe the moving object's pdf. Algorithm 1 presents a formal description of the particle filter algorithm.

Particle filter in Vision Tracking

Robust real-time tracking of non-rigid objects in computer vision is a challenging task. Particle filtering has proven very successful for non-linear and non-Gaussian estimation problems. Particle filtering was originally developed to track objects in clutter, that is to say, one of its main characteristics represents the possibility to track objects although exists the presence of other objects that have similar characteristic.

We want to apply a particle filter in a color-based context. Color distributions are used as target models as they achieve robustness against non-rigidity, rotation and partial occlusion. Suppose that the distributions are discretized into *m*-bins. The histograms are produced with the function $h(\mathbf{x}^i)$, that assigns the color at location $\mathbf{x}^i$ to the corresponding bin (considering $\mathbf{x}^i$ the pixel coordinates (x, y)). In our experiments, the histograms are typically calculated in the RGB space using 8x8x8 bins. To make the algorithm less sensitive to lighting conditions, the HSV color space could be used instead with less sensitivity to V (e.g. 8x8x4 bins).

We determine the color distribution inside an upright circular region centered in $\mathbf{x}^i$ with radius r. To increase the reliability of the color distribution when boundary pixels belong to the background or get occluded, smaller weights are assigned to the pixels that are further away from the region center by employing a weighting function.

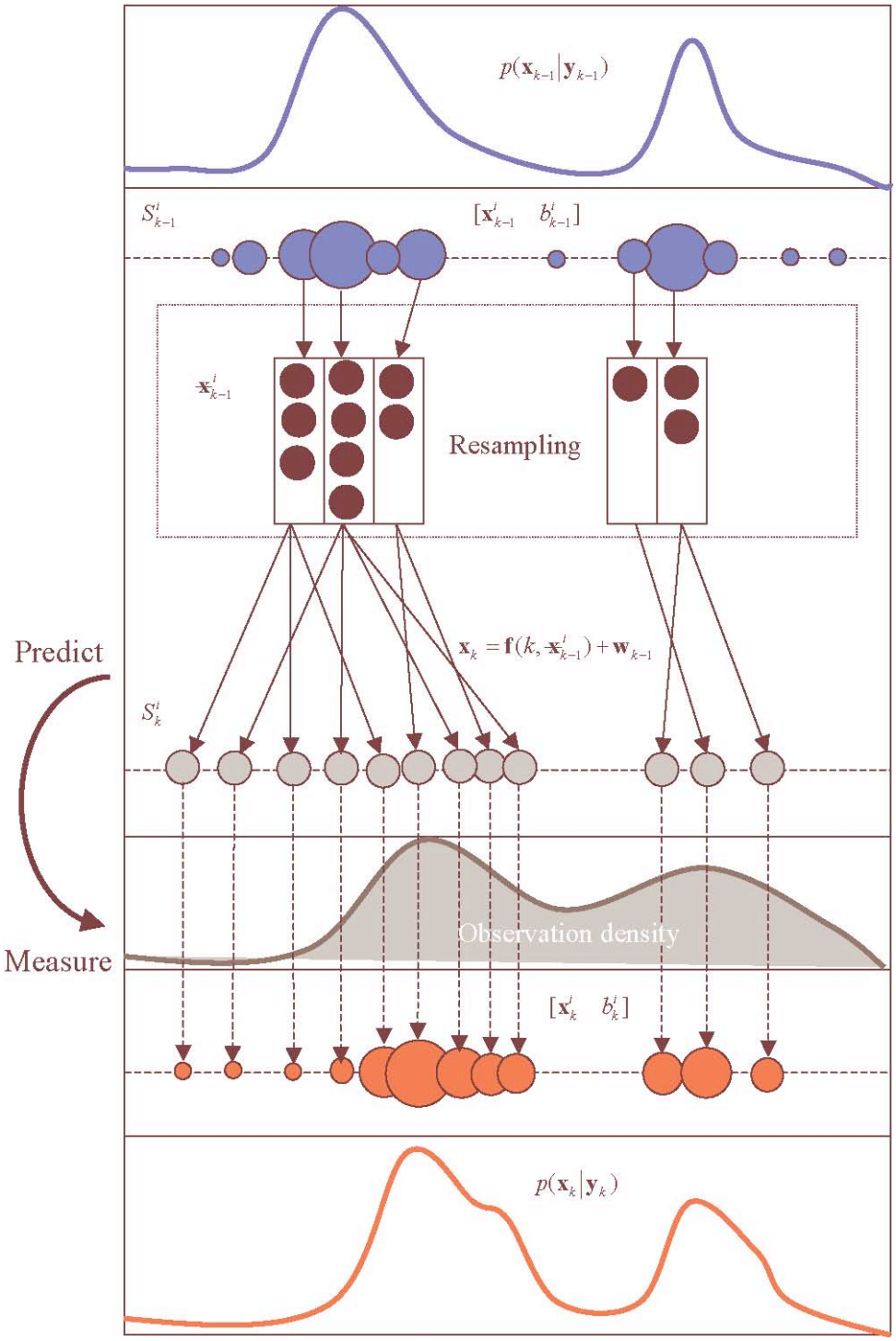


Figure 1. Process carried out by the particle filter

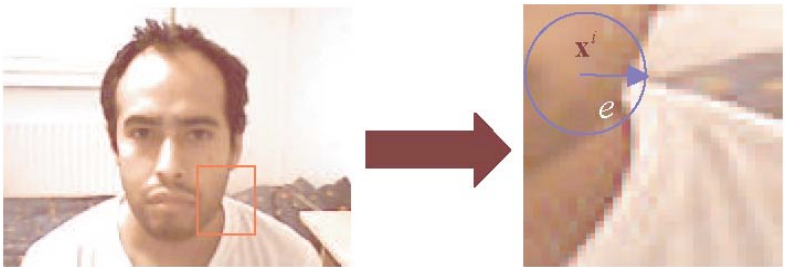


Figure 2. Configuration of the density of the particles, centered in x^i dependent of the distance e.

$$k(e) = \begin{cases} 1 - e^2 : e < 1 \\ 0 : \text{otherwise} \end{cases} \qquad (7)$$

where e is the distance from the region center. Thus, we increase the reliability of the color distribution when these boundary pixels belong to the background or get occluded. The figure 2 shows the advantage of using the distance e to improve the reliability of the measurement.

The color distribution $p_y = \{p_y^{(u)}\}_{u=1,2,3,...m.}$ at location y is calculated as

$$p_{\mathbf{y}}^{(u)} = f \sum_{i=1}^{I} k\left(\frac{\left\|\mathbf{y} - \mathbf{x}^i\right\|}{r}\right) \delta[h(\mathbf{x}^i) - u] \qquad (8)$$

where I is the number of pixels in the circular region, δ is the Kronecker delta function and the normalization factor

$$f = \frac{1}{\sum_{i=1}^{I} k\left(\frac{\left\|\mathbf{y} - \mathbf{x}^i\right\|}{r}\right)} \qquad (9)$$

ensures that $\sum_{u=1}^{m} p_{\mathbf{y}}^{(u)} = 1$.

In a tracking approach, the estimated state is updated at each time step by incorporating the new observations. Therefore, we need a similarity measure which is based on color distributions. A popular measure between two distributions $p(u)$ and $q(u)$ is the Bhattacharyya coefficient [5, 6].

$$\rho[p,q] = \int \sqrt{p(u)q(u)}\,du \qquad (10)$$

Considering discrete densities such as our color histograms $p = \{p^{(u)}\}_{u=1,2,3,...m.}$ and $q = \{q^{(u)}\}_{u=1,2,3,...m.}$ the coefficient is defined as:

$$\rho[p,q] = \sum_{u=1}^{m} \sqrt{p^{(u)} q^{(u)}} \qquad (11)$$

The larger ρ is, the more similar the distributions are. For two identical normalized histograms we obtain $\rho = 1$, indicating a perfect match. As distance between two distributions we define the measure:

$$d = \sqrt{1 - \rho[p,q]} \qquad (12)$$

which is called the Bhattacharyya distance.

The proposed tracker employs the Bhattacharyya distance to update the a priori distribution calculated by the particle filter. Each sample of the distribution represents a circular region and is given as:

$$\mathbf{x}_k^i = [x_k^i \quad y_k^i \quad \Delta x_k^i \quad \Delta y_k^i] \qquad (13)$$

where *x, y* specify the location of the circular region (center), Δx and Δy the motion. As we consider a whole sample set, the tracker handles multiple hypotheses simultaneously.

The sample set is propagated through the application of a dynamic model

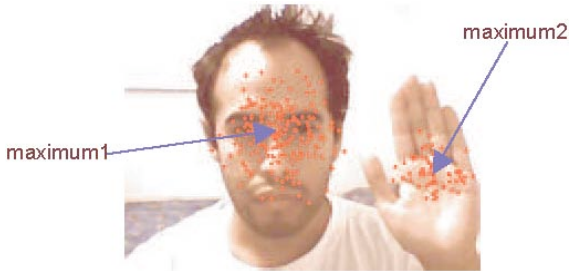
$$\mathbf{x}_{k+1}^i = \mathbf{f}(k, \mathbf{x}_k^i) + \mathbf{w}_k^i \qquad (14)$$

where $\mathbf{f}(k, \mathbf{x}_k^i)$ defines the deterministic component of the model and $\mathbf{w}_k^i$ is a multivariate Gaussian random variable. In this thesis we currently use an unconstrained Brownian model for describing the region movement with velocity Δx, Δy and radius *r*.

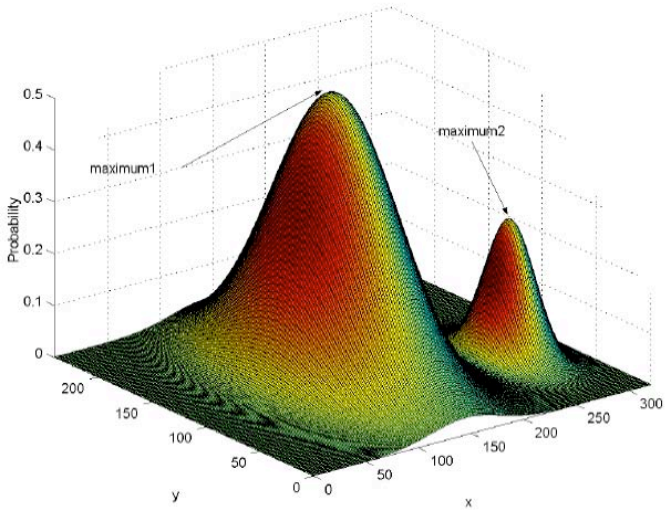
$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \Delta x_{k+1} \\ \Delta y_{k+1} \end{bmatrix} = \begin{bmatrix} \exp\left(-\frac{1}{4}(x_k + 1.5\Delta x_k)\right) \\ \exp\left(-\frac{1}{4}(y_k + 1.5\Delta y_k)\right) \\ \exp\left(-\frac{1}{4}\Delta x_k\right) \\ \exp\left(-\frac{1}{4}\Delta y_k\right) \end{bmatrix} + \mathbf{w}_k \qquad (15)$$

To weight the sample set, the Bhattacharyya coefficient has to be computed between the target histogram and the histogram of the hypotheses. Each hypothetical region is specified by its state vector S_k^i. Both the target histogram q and the candidate histogram $p_{S_k^i}$ are calculated from Eq. 8 where the target is centered at the origin of the circular region.

As we want to favor samples whose color distributions are similar to the target model, small Bhattacharyya distances correspond to large weights. The following equation shows the form in which the weights are calculated for the particles.



(a)



(b)

Figure 3. a) Distribution of the sample set and b) generated multi-modal probability density function.

$$b^i = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(1-\rho[p_{S_k^i}, q])}{2\sigma} \right) \qquad (16)$$

Of the equation we can observe that small distances correspond to big weight values, while big distances (non correspondence) generated small weight values. That is specified by a Gaussian with variance σ. During filtering, samples with a high weight may be chosen several times, leading to identical copies, while others with relatively low weights may not be chosen at all. The programming details for one iteration step are given in Algorithm 1.

To illustrate the distribution of the sample set, Figure 3 shows the samples distribution considering the flesh color as target histogram *q*. The samples are located around the maximum of the Bhattacharyya coefficient which represents the best match to the target model.

Given a particle distribution S_k^i, we need to find the state which defines with accuracy the object position. Three different methods of evaluation have been used in order to obtain an estimate of the position. First, the weighted mean ($\hat{\mathbf{x}}_k \approx \sum_{i=1}^{M} b_k^i \mathbf{x}_k^i$) be used; second, the best particle (the $\mathbf{x}_k^j$ such that $b_k^j = \max(b_k^i) : i = 1,2,...M$) and, third, the weighted mean in a small window around the best particle (also called robust mean) can be used. Each method has its advantages and disadvantages: the weighted mean fails when faced with multi-modal distributions, while the best particle introduces a discretization error. The best method is the robust mean but it is also the most computationally expensive. In cases where the object to track is surrounded of objects whose characteristics are similar the best method is to use as state that defines the object position "the best particle".

Algorithm 1 Particle Filter Algorithm

From the particles at time-step *k*-1 $S_{k-1}^i = \{\mathbf{x}_{k-1}^i, b_{k-1}^i \,|\, i = 1,2,...,M\}$.

1. For each particle we calculate the cumulative probability c_{k-1}^i as

$$\left.\begin{aligned} c_{k-1}^0 &= 0 \\ c_{k-1}^i &= c_{k-1}^{i-1} + b_{k-1}^i \end{aligned}\right| i = 1,2,...,M$$

we have in this way $S_{k-1}^i = \{\mathbf{x}_{k-1}^i, b_{k-1}^i, c_{k-1}^i \,|\, i = 1,2,...,M\}$

2. We select *M* states (they can repeat) starting from S_{k-1}^i (resampling), carrying out the following procedure
   - We generate a random number $r \in [0,1]$, uniformly distributed.
   - We find, the smallest *j* for which $c_{k-1}^j \geq r$.
   - The elected state is $\mathbf{x}_{k-1}^i = \mathbf{x}_{k-1}^j$.

3. We spread the states $\{\mathbf{x}_{k-1}^i \,|\, i = 1,2,3...,M\}$ using the model

$$\mathbf{x}_k^i = \mathbf{f}(k, \mathbf{x}_{k-1}^i) + \mathbf{w}_k .$$

4. For each new state $\mathbf{x}_k^i$ we find their corresponding *b* starting from the measurement $p(\mathbf{y}|\mathbf{x})$ obtained for each hypothesis.

5. We carry out the normalization $\sum_{i=1}^{M} b_k^i = 1$ and build the particles

$$S_k^i = \{\mathbf{x}_k^i, b_k^i \,|\, i = 1, 2, ..., M\} .$$

6. Once the *M* samples have been constructed: estimate, if desired, moments of the tracked position at time *k* as

$$E[S_k^i] = \sum_{i=1}^{M} b_k^i \mathbf{x}_k^i$$

Results and conclusions

To illustrate the adaptive color-based particle filter performance and its behavior, we applied the proposed method to a *face* sequence and show the tracking results. The experiments have been processed with an 800 MHz Pentium PC under windows, using the RGB color space with 8x8x8 bins and images of size 352x288 pixels. The goal of the experiments has been to track a manually initialized object region during the sequence until it has disappeared.

In Figure 4, the *face* sequence of 700 frames is shown. The tracked face is affected by changing illumination conditions and facial expressions as well as a large scale changes. In frame 500, the tracked position is not exact as the model does not match by changing illumination intensity very well. Nevertheless, the person can still be tracked and the position improves rapidly.

The proposed tracking method is based on color distributions to particle filtering. The color-based tracker can efficiently and successfully handle non-rigid and fast moving objects under different appearance changes. Moreover, as multiple hypotheses are processed, objects can be well tracked in cases of occlusion or clutter. As a limitation of the proposed approach the tracker might loose an object when it changes its appearance quickly, for example in occlusion, followed by a rapid movement at the same time.

Once the object has already been stabilized (the tracking uncertainty is small) it is obtained by means of moments the relative values of the main axes that contain the head, this is shown in the figure 4 by means of the red lines.

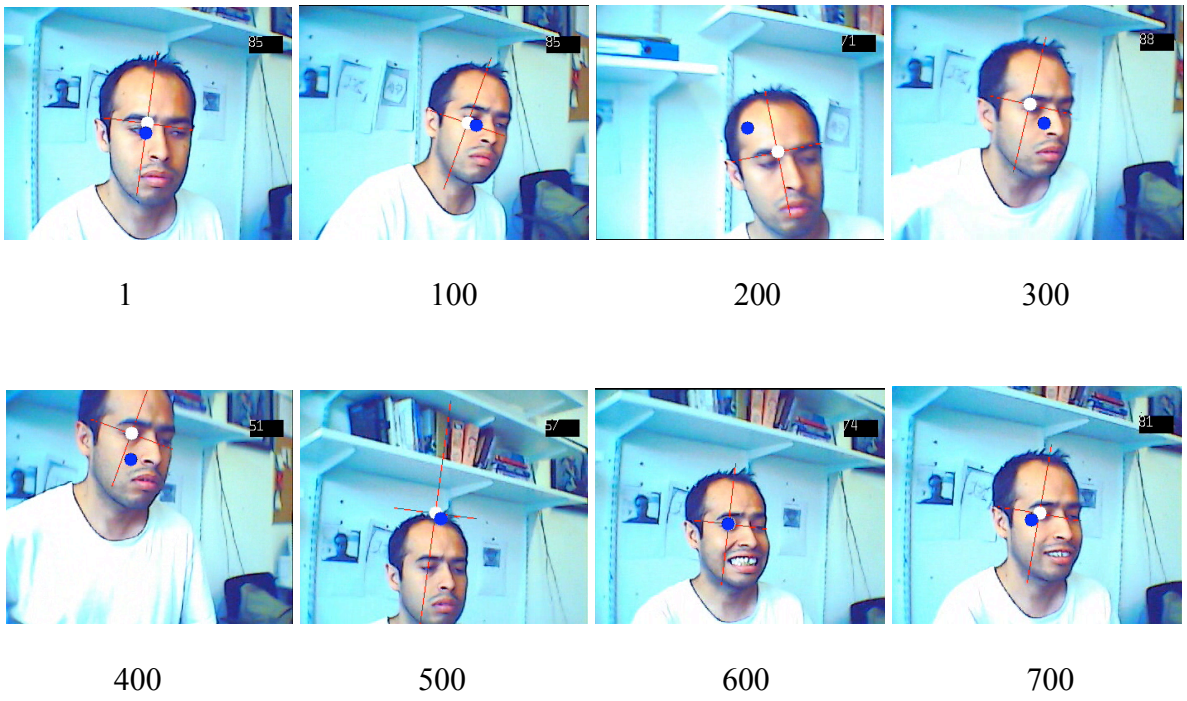


Figure 4. *face* sequence of 700 frames and the tracking perform achieved by our algorithm.

References

1. Cuevas E., Zaldivar D. and Rojas R. (2005). Kalman Filter for vision tracking, *Technical Report B 05-12*, Freie Universität Berlin, Fachbereich Mathematik und Informatik. Berlin, Germany.

2. Isard M., and Blake A. (1996). Contour Tracking by Stochastic Propagation of Conditional Density, *European Conference on Computer Vision*. 343-356.

3. Isard M., and Blake A. (1998). A Mixed-State Condensation Tracker with Automatic Model-Switching, *International Conference on Computer Vision*. 107-112.

4. Isard M., and Blake A. (1998). CONDENSATION (Conditional Density Propagation for Visual Tracking), *International Journal on Computer Vision* 1. (29): 5-28.

5. Aherne F., Thacker and N., and Rockett P. (1997). The Bhattacharyya Metric as an Absolute Similarity Measure for Frequency Coded Data, *Kybernetika*. 32(4): 1-7.

6. Kailath T. (1967). The Divergence and Bhattacharyya Distance Measures in Signal Selection, *IEEE Transactions on Communication Technology* COM-1 5(1): 52-60.