A CANONICAL CORRELATION ANALYSIS BASED MOTION MODEL FOR PROBABILISTIC VISUAL TRACKING

Tom Heyman[1], *Vincent Spruyt*[1,2], *Sebastian Grünwedel*[1], *Alessandro Ledda*[2], *Wilfried Philips*[1]

[1]IBBT, IPI, Ghent University
St. Pietersnieuwstraat 41, 9000 Ghent, Belgium

[2]Dept. of Applied Engineering, Artesis University College
Paardenmarkt 92, 2000 Antwerp, Belgium

ABSTRACT

Particle filters are often used for tracking objects within a scene. As the prediction model of a particle filter is often implemented using basic movement predictions such as random walk, constant velocity or acceleration, these models will usually be incorrect. Therefore, this paper proposes a new approach, based on a Canonical Correlation Analysis (CCA) tracking method which provides an object specific motion model. This model is used to construct a proposal distribution of the prediction model which predicts new states, increasing the robustness of the particle filter. Results confirm an increase in accuracy compared to state-of-the-art methods.

Index Terms— Object tracking, Canonical Correlation Analysis, prediction model, particle filter

1. INTRODUCTION

Predicting the movement of an object over time is difficult when prior knowledge about its motion is lacking. While deterministic tracking approaches provide fixed predictions based on the current and past information, probabilistic tracking is better capable of modeling uncertainties by estimating a proposal distribution rather than a single prediction. Using this distribution, the scenario is narrowed down to possible states which have to be compared to measurements to find out which will most likely be the object's true state.

1.1. Deterministic tracking

One example of deterministic tracking is tracking by detection, where the object is detected in each frame using a certain deterministic technique. Another method uses a flock of features [1], where a dense area of features within a frame provides a position estimation. Mean-shift tracking [2] finds a local maximum on a given confidence map of the new frame.

This work was financially supported by IWT through a Ph.D. grant, and by IBBT through the iCocoon project co-funded by IBBT, a research institute founded by the Flemish Government.

As the former methods usually rely on a simple appearance model such as color or edges, they are better suited for tracking shape changing objects compared to methods relying on higher level features. However, for tracking rigid objects more accurate results can be achieved by taking the rigidity into account. Linear regression methods such as Active Appearance Models (AAM) [3] or Active Shape Models (ASM) [4] allow to include spatial information of the object. While ASM is based only on the shape of an object, AAM also includes appearance based features. Similar to AAM and ASM, a Canonical Correlation Analysis (CCA) [5] based tracker has been proposed by J. Zepeda et al. [6]. This method also finds a linear regression between object positions and its respective appearances. However, CCA is a fast discriminative method whereas AAM or ASM are slower generative methods.

1.2. Probabilistic tracking

In any tracker some assumptions are made about the expected motion, yet this model will usually be imperfect. When assuming the deviations from the motion model behave as (Gaussian) noise, probabilistic models are able to deal with this noise. A general probabilistic approach would be the Bayes filter as explained in [7]. This algorithm calculates the probabilities of possible states and provides an estimated state of the object being tracked. A prior distribution contains possible starting states and corresponding probabilities. The Bayes filter consists of two phases for every new frame: A prediction phase which provides a proposal distribution of the next state, and an observation phase which measures the likelihoods of the estimates states. Finally, states are updated by combining the likelihood with the predicted new state in order to obtain a posterior distribution and a posterior estimate.

Kalman filters and particle filters are the most common examples of a Bayesian filter. A Kalman filter assumes a linear motion, a linear observation and Gaussian distributions for approximating the actual distribution. A particle filter on the other hand, uses a discrete approximation by means of

Monte Carlo samples for approximating the actual distribution, which makes it suitable for tracking motion in a real-life scene as this motion will likely have non-linear behavior.

1.3. Our approach

Because common prediction methods in probabilistic trackers (random walk, constant velocity, constant acceleration) often fail to predict real-life (non-linear) motion, yet are still frequently used (for example [8] and [9]) we propose a more accurate method by introducing an advanced deterministic tracker within these probabilistic trackers. As we intend to include spatial information, CCA was chosen for its computational simplicity. To the best of our knowledge, this deterministic tracker has never been integrated within a probabilistic framework. As shown in Figure 1, the CCA based tracker consists of an initialization, training and tracking phase. A particle filter is provided with an already mentioned prediction and observation model. The former will incorporate the tracking phase of a CCA based tracker.

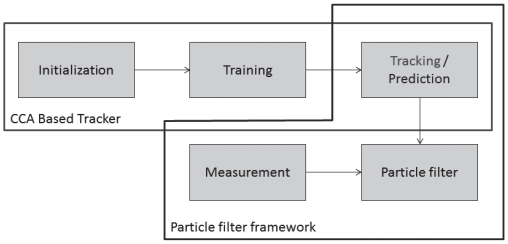


Fig. 1. Overview of method integration.

Although possible, not all degrees of freedom will be tracked by this method. A restriction has been made to include only 2D translations (t_x,t_y), scaling (t_z) and rotations around the z-axis (θ_z). Adding more degrees of freedom or allowing non-rigid objects is possible, yet lowers the reliability of the CCA tracker or requires a longer training duration. Although any object can be tracked, this paper will provide a rigid hand tracking approach as an example. In the following sections, this approach will be elaborated. First, our CCA based tracker will be explained in Section 2. Next, the particle filter and integration of the new prediction model will be covered during Section 3. Finally, results and a conclusion will be given in the final sections.

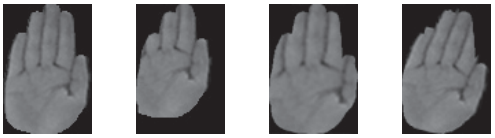
2. CCA BASED TRACKER

2.1. Initialization phase

After manually selecting the user's hand within the first frame of a video sequence, a reference image patch is created from a selected region of interest. After applying a hand mask, the image is converted and rescaled to a 65×95 grayscale image similar to Figure 2(a). This image patch is called the reference image patch, as it will be a reference during both training and tracking phases. The resolution is empirically chosen to still contain enough information for training different states, without providing too much overhead during tracking.

2.2. Training phase

After obtaining the reference image patch, a dataset Q_a is populated from generated hand position image patches based on this reference image patch. By applying translation, rotation and scaling operations on the underlying image, we are able to create image patches which represent movement. Figure 2 provides image patch examples of a 2D translation (b), a scaling (c) and a rotation (d), created from the reference image patch (a). During training we want to find a correlation between these patches and their respective states as explained in section 2.3.



(a) (b) (c) (d)

Fig. 2. Reference patch and possible trained positions.

Before Q_a is populated, the pixels of image patches are stored in image patch vectors $\mathbf{x}_i$. Each image patch vector is normalized by the norm of all pixel values within the corresponding patch in order to allow for changes in illumination. Finally, the reference image patch vector is subtracted from each image patch vector as shown in Equation (1). The resulting vectors $\mathbf{a}_i$ are stored in Q_a.

$$\mathbf{a}_i = \mathbf{x}_i^{norm} - \mathbf{x}_{ref}^{norm} \quad (1)$$

A different dataset Q_s will contain state vectors $\mathbf{s}$ as shown in Equation (2), which correspond to the generated image patches stored in Q_a.

$$\mathbf{s} = [t_x, t_y, t_z, \theta_z] \quad (2)$$

Although it is possible to find a correlation based on these two datasets, they are separated into three groups to avoid the curse of dimensionality; 2D translations, scaling and rotations around the z-axis. By introducing this state separation and assuming independence amongst these clusters, the training sets require less data to find a reliable linear correlation and states can be tracked hierarchically within a particle filter.

As different users and scenarios provide different reference image patches, no optimal training set can be provided for every scenario. Hence, time needed to find an accurate linear correlation should be minimized as training needs to be performed before every tracking session. The training positions for each state are obtained by random sampling a normal distribution as illustrated in Figure 3, and contain 200

2D translations, 80 scaled images and 80 rotations. Because

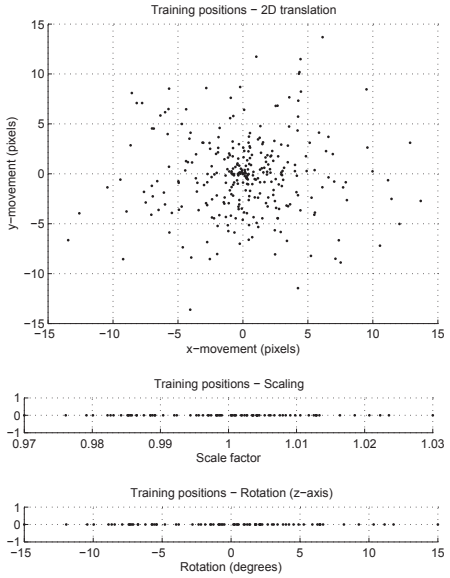


Fig. 3. Trained positions for all state groups.

of the separation of states, the three pairs of datasets will be trained separately. Three motion model matrices will be generated corresponding to their respective state group. However, in the next section we will assume a general state vector $\mathbf{s}$ and only one pair of datasets to avoid confusion as we explain how to retrieve the mentioned motion model matrices.

2.3. Finding correlation using CCA

After populating two groups of data Q_a and Q_s, a correlation can be found between the canonical variates of these sets. After centering all vectors in Q_a and Q_s around zero, resulting in A_a and A_s respectively, we can assume $\mathbf{w}_a$ and $\mathbf{w}_s$ to be the unknown direction vectors of optimal correlations for A_a and A_s. By projecting the datasets onto these vectors, the canonical variates $\mathbf{z}_a$ (3) and $\mathbf{z}_s$ (4) are obtained.

$$\mathbf{z}_a = A_a \mathbf{w}_a \quad (3)$$

$$\mathbf{z}_s = A_s \mathbf{w}_s \quad (4)$$

The cross-correlation ρ as defined in Equation (5) should now be maximized ($\rho \approx 1$) to obtain the maximum correlation between the datasets.

$$\rho = \frac{\mathbf{z}_a^T \mathbf{z}_s}{\sqrt{\mathbf{z}_a^T \mathbf{z}_a}\sqrt{\mathbf{z}_s^T \mathbf{z}_s}} \quad (5)$$

As ρ is not affected by rescaling the canonical variates, we can introduce the following constraints:

$$\mathbf{z}_a^T \mathbf{z}_a = \mathbf{w}_a^T A_a^T A_a \mathbf{w}_a = 1 \quad (6)$$

$$\mathbf{z}_s^T \mathbf{z}_s = \mathbf{w}_s^T A_s^T A_s \mathbf{w}_s = 1 \quad (7)$$

Where $A_a^T A_a = \Sigma_{aa}$ and $A_s^T A_s = \Sigma_{ss}$ are covariance matrices. We now use the constraints above and solve the maximization problem in Lagrange form as shown in [5]. The following equations govern CCA:

$$(\Sigma_{as}^T \Sigma_{aa}^{-1} \Sigma_{as} - \rho^2 \Sigma_{ss})\mathbf{w}_s = 0 \quad (8)$$

$$(\Sigma_{as} \Sigma_{ss}^{-1} \Sigma_{as}^T - \rho^2 \Sigma_{aa})\mathbf{w}_a = 0 \quad (9)$$

Where $A_a^T A_s = \Sigma_{as}$ is again, a covariance matrix. These generalized eigenvalue problems are solved using the method elaborated in [5]. By computing singular value decompositions of $A_a = U_a D_a V_a^T$ and $A_s = U_s D_s V_s^T$ and finally $U_a^T U_s = U D V^T$, we find sets of canonical variates W_a (10) and W_s (11):

$$W_a = V_a D_a^{-1} U \quad (10)$$

$$W_s = V_s D_s^{-1} V \quad (11)$$

As we want the correlation to be maximized ($\rho \approx 1$), $W_a A_a \approx W_s A_s$ becomes valid. This equation also holds for new data instead of the datasets A_a and A_s, for example a new image patch vector $\mathbf{a}$ and a state vector $\mathbf{s}$. Substituting (10) and (11) in this formula, we can retrieve the state vector $\mathbf{s}$ by (12).

$$\mathbf{s} = V_a D_a^{-1} U V^T D_s V_s^T \mathbf{a} \quad (12)$$

Where $G = V_a D_a^{-1} U V^T D_s V_s^T$ is called the motion model matrix which will be used during tracking. Keep in mind that there will actually be three matrices $G_{t_x t_y}$, G_{t_z} and G_{θ_z}, as training states were separated.

2.4. Tracking phase

When training is complete and motion models are generated, the tracker will start at an initial position $\mathbf{s}_0$. Each frame, an image patch vector $\mathbf{a}_t$ is created from this frame using the last known position $\mathbf{s}_{t-1}$. The image patch vector is then used in (13) to find the object's estimated current position $\mathbf{s}_t$.

$$\mathbf{s}_t = \mathbf{s}_{t-1} + G\mathbf{a}_t \quad (13)$$

However, we actually have three state groups and three motion model matrices $G_{t_x t_y}$, G_{t_z} and G_{θ_z}. This tracking method will be applied to all particles, and provides a proposal distribution within our particle filter.

3. PARTICLE FILTER

We use a SIR particle filter [10], consisting of a prediction and an observation model which are explained below. As mentioned before, a prior distribution model is known which contains possible starting states and corresponding probabilities of the object having that state in the frame.

3.1. Prediction model

The prediction model provides a proposal distribution of the states for the next time frame. This distribution will be sampled and the resulting particles will be compared to the observations during the observation phase using an importance sampling technique in order to obtain a posterior distribution. Our proposed method will be compared to a random walk, constant velocity and constant acceleration model shown in Equations (14)(15)(16), where n is a zero-mean Gaussian stochastic component, $\mathbf{v}_t$ and $\mathbf{a}_t$ represent velocity and acceleration at time t respectively, and $\mathbf{s}_t$ is the proposed state of each particle.

$$\mathbf{s}_t = \mathbf{s}_{t-1} + n \quad (14)$$

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{v}_{t-1} + n \quad (15)$$

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{v}_{t-1} + \frac{\mathbf{a}_{t-1}}{2} + n \quad (16)$$

By incorporating the latest observations when creating a proposal distribution, we can provide a superior accuracy over these widely used models. An integrated CCA tracking phase proposes new states $\mathbf{s}_t$ as shown in Equation (17) for each state group. To the best of our knowledge, this integration has never been performed.

$$\mathbf{s}_t = \mathbf{s}_{t-1} + G\mathbf{a}_t + n \quad (17)$$

3.2. Observation model

Our observation phase, which measures the likelihoods of these estimated states, depends on color information detected in each frame. Pixel values pix contain hue and saturation, while brightness has been ignored to provide more robustness during illumination changes. As proposed in [11], hue pixels which have very low or very high corresponding brightness values are also ignored as these values tend to become unstable at low brightness or take on flesh hue at high brightness. A maximum likelihood probability distribution image as shown in Figure 4 is generated from the current frame by assuming $P(obj) = 0.5$ and estimating $P(obj|pix)$ (18) from two histograms created during initialization: An object color histogram $P(pix|obj)$ from the hand region at initialization and a background color histogram $P(pix|\neg obj)$, where obj represents the estimate of containing the hand.

$$P(obj|pix) = \frac{P(pix|obj)}{P(pix|obj) + P(pix|\neg obj)} \quad (18)$$

This generated image is then used for measuring the probabilities of estimated states. The probability $P(obj|s)$ of each particle at state $\mathbf{s}$ is shown in Equation (19), where only pixels within the hand mask of the corresponding image patch region are relevant.

$$P(obj|s) = E[P(obj|pix_{mask})] \quad (19)$$

(a) Video frame

(b) PD image

Fig. 4. Example of probability distribution image.

However, when a smaller scaling state than the actual size is estimated such that the hand mask is positioned within the object, $P(obj|s)$ could provide similar probabilities and the scaling state could be wrongly estimated. To overcome this problem, we introduce a scale correctness probability $P(corr_{t_z})$ with $corr_{t_z}$ being the estimate of scale correctness as shown in Equation (20).

$$P(corr_{t_z}) = \frac{E[P(obj|pix_{mask})]}{E[P(obj|pix_{mask})] + E[P(obj|pix_{\neg mask})]} \quad (20)$$

By including the probability of background pixel values outside the hand mask, $P(corr_{t_z})$ will be maximized when no skin pixels are found outside the hand mask. The particle weights are finally found by multiplying its corresponding probability $P(obj|s)$ with its scale correctness $P(corr_{t_z})$. The posterior state is found as a weighted average of all particles.

3.3. Partitioned sampling

Because the CCA tracking phase will be integrated into the prediction model of our particle filter, a partitioned sampling method [12] is an ideal approach as the states of our CCA based tracker are already separated into groups. During partitioned sampling, 2D translations (t_x,t_y) are first predicted and updated by the observation model. Once this position is known, the rotation parameter (θ_z) is found based on the updated position and thus a new image patch vector from the same frame. Finally, the scaling factor (t_z) is found using the same principle.

4. RESULTS

4.1. Comparison

To our knowledge, no suitable hand tracking dataset exists. Therefore, we created six different video sequences of 20 seconds. These unbiased video sequences were acquired in dark, normal and bright illuminated locations, while performing normal and fast hand movements. Every 250 milliseconds, ground truth was manually selected and linearly interpolated between data points. A mean error measurement of 2D translations, scaling and rotation states during an all-state tracker

is shown in Figure 5, while the number of particles vary between 1 and 200. We have optimized the zero-mean Gaussian stochastic component n to achieve the best results for each prediction model.

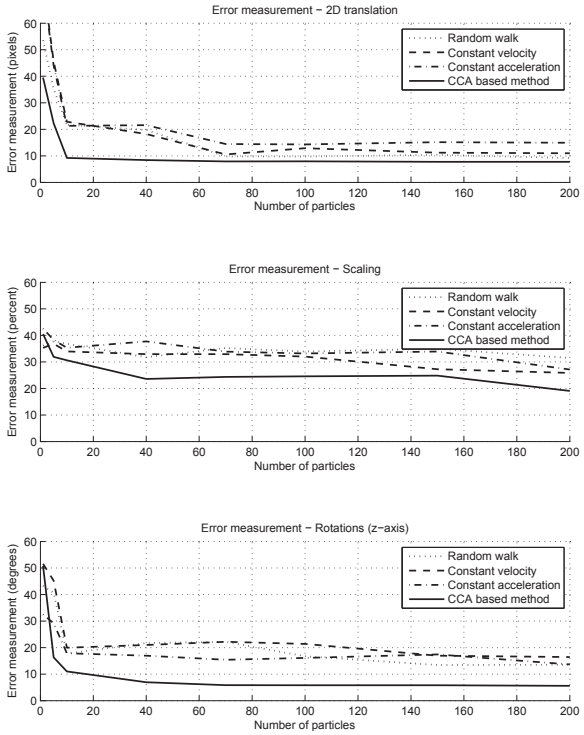


Fig. 5. Error measurement for all tracking states compared to ground truth.

Standard deviations of these values are shown in Table 1. As a very small dataset has been used to evaluate our method, these values are rather high. In the future, a more extensive dataset should provide a more accurate result.

Number of particles	1	5	10	40	70	100	150	200
2D translation (pixels)	30.56	18.45	5.12	4.51	4.31	4.30	4.19	4.13
Scaling (percent)	0.14	0.14	0.14	0.11	0.11	0.11	0.10	0.09
Rotation (degrees)	23.99	8.83	6.37	4.78	3.81	3.74	3.82	3.65

Table 1. Standard deviation values for CCA based prediction

It is obvious that by using more particles, accuracy will increase for all mentioned methods. Moreover, we notice the CCA based prediction model provides a more accurate tracking across all states. During 2D translation tracking, the importance lies with CCA's steep error measurement decrease when just a few particles are provided. Although we need to ignore the first data points due to high standard deviations, the processing time can still be lowered by using a CCA based prediction method instead of the common approaches, assuming a certain maximum error measurement is wanted. The ideal error measurement (0) for any state will never be reached, as these prediction methods have a one-frame delay and video sequences with fast movement have been included.

For scale tracking, rather high overall error measurements are found. These are caused by both the simplicity of our observation model and the fact that the scaling state is the last state to be updated within the partitioned sampling approach. When any of the former updated states are inaccurate, scaling robustness will decrease. A CCA approach however, is trained to recover from any inaccurate position, and thus clearly outperforms the other methods as can be seen in Figure 5. During rotation tracking we again see a steeper increase in accuracy by our CCA approach because of this reason. While the common methods also provide a reasonably good result using a small amount of particles, they approach the ideal error measurement much slower than our method.

Finally, a few challenging frames are illustrated in Figure 6, where all methods are compared while using 70 particles.

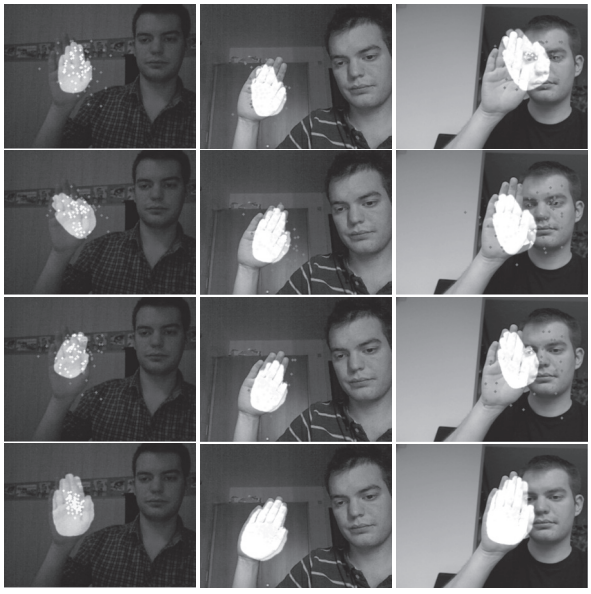


Fig. 6. Webcam tracking examples. (top to bottom; random walk, constant velocity, constant acceleration, CCA based)

4.2. Processing time

All results of our C++ implementation were obtained on an Intel Core i7-2620M CPU using a frame resolution of 320×240. OpenCV 2.0 and Boost 1.4 libraries were included for image processing. The training phase of our CCA based tracker takes 4.68 seconds for training 2D translations (t_x, t_y), and 1.84 seconds each for training rotation (θ_z) and scaling states (t_z). Processing speed during tracking depends on both number of particles and which states are tracked. However, as we've used a Logitech Webcam C300, the actual frame rate will be no more than the webcam's rate of 30fps. As can be

seen in Figure 7, the achieved frame rate drops exponentially when the number of particles increase. Keep in mind that our implementation is a non-optimized prototype of the proposed method. Only a constant velocity model is shown as random

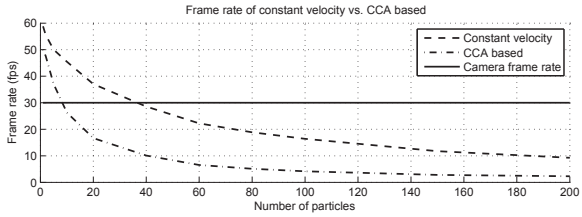


Fig. 7. CCA and constant velocity frame rate comparison.

walk or constant acceleration are very similar in processing speed. By introducing a CCA based prediction model, higher delays are observed compared to these common approaches. Although this problem exists, the CCA based technique provides a higher accuracy and thus fewer particles are needed to provide the same tracking robustness.

5. CONCLUSION

The proposed prediction method provides a superior accuracy over other widely used techniques. Although a steeper linear increase in processing time exists when raising the number of particles, a CCA based tracking solution is still able to provide a more robust prediction than other commonly used models with an equal frame rate. The proposed method can be used to improve current state-of-the-art particle filters which apply these prediction models, provided a training phase can be included.

6. FUTURE WORK

A first limitation of the proposed method is the use of a limited number of states (translations and z-rotation). Provided the right training and image patch adjustments are performed, it becomes possible to track more degrees of freedom. For example, J. Zepeda et al. [6] proposed a CCA based tracker capable of following semi-3D movement.

Currently, normally distributed models are provided which represent the positions that are trained. Obviously, these models do not include prior knowledge of object's characteristic movements. A more accurate approach, for hand tracking for example, would be to firstly observe common hand movements and adjust the model to provide more training positions along these movements. Another future improvement could provide different sets of trained positions for each state group. For example, 2D translations could be trained using a more dense and a more sparsely generated normally distributed model, resulting in two motion models for slow and fast movements respectively.

7. REFERENCES

[1] M. Kolsch and M. Turk, "Fast 2d hand tracking with flocks of features and multi-cue integration," in *In IEEE Workshop on Real-Time Vision for Human-Computer Interaction at CVPR*, 2004, p. 158.

[2] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 2000, vol. 2, pp. 142–149 vol.2.

[3] X. Gao, Y. Su, X. Li, and D. Tao, "A review of active appearance models," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 145–158, march 2010.

[4] C. Vogler, Zhiguo L., A. Kanaujia, S. Goldenstein, and D. Metaxas, "The best of both worlds: Combining 3d deformable models with active shape models," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, oct. 2007, pp. 1–7.

[5] D. Weenink, "Canonical correlation analysis," in *Proceedings 25, Institute of Phonetic Sciences, University of Amsterdam*, 2003, pp. 81–99.

[6] J.A.Y. Zepeda, F. Davoine, and M. Charbit, "A linear estimation method for 3d pose and facial animation tracking," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, june 2007, pp. 1–7.

[7] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, feb 2002.

[8] V. Spruyt, A. Ledda, and S. Geerts, "Real-time multi-colourspace hand segmentation," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 3117 –3120.

[9] X.F. Wang, J.F. Chen, Z.G. Shi, and K.S. Chen, "Fuzzy-control-based particle filter for maneuvering target tracking," 2011, vol. 118, pp. 1–15.

[10] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, apr 1993.

[11] G.R. Bradski, "Computer vision face tracking for use in a perceptual user interface," 1998.

[12] J. Maccormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *In ECCV*, 2000, pp. 3–19.