FACE TRACKING USING CANONICAL CORRELATION ANALYSIS

José Alonso Ybáñez Zepeda[1], Franck Davoine[2] and Maurice Charbit[1]
[1]*Ecole Nationale Supérieure des Télécommunications, Rue Barrault, Paris, France*

[2]*Compiegne University of Technology, Heudiasyc lab., CNRS, Compiègne, France*

Keywords: Canonical Correlation Analysis (CCA), Kernel Canonical Correlation Analysis (KCCA), Head Tracking.

Abstract: This paper presents an approach that incorporates canonical correlation analysis for monocular 3D face tracking as a rigid object. It also provides the comparison between the linear and the non linear version (kernel) of the CCA. The 3D pose of the face is estimated from observed raw brightness shape-free 2D image patches. A parameterized geometric face model is adopted to crop out and to normalize the shape of patches of interest from video frames. Starting from a face model fitted to an observed human face, the relation between a set of perturbed pose parameters of the face model and the associated image patches is learned using CCA or KCCA. This knowledge is then used to estimate the correction to be added to the pose of the face from an observed patch in the current frame. Experimental results on tracking faces in long video sequences show the effectiveness of the two proposed methods.

1 INTRODUCTION

Nowadays video object tracking presents a challenging problem for video applications such as face-based biometric person authentication, human computer interaction, video games, teleconferencing, surveillance, etc. To deal with this problem, vision research groups have proposed several approaches that can be classified in two main groups: model-based and learning-based approaches. In the first category, tracking algorithms rely on a parametric model of the object to be tracked. In the second category, algorithms assume the availability of a training set of object examples, and use pattern recognition/classification techniques. When working with video images, we can obtain different features, such as edges, interest points, grey level intensities or color histograms, etc. The choice of these features will depend on the tracked object's characteristics. The tracker is traditionally composed of two components: a representation component to cope with changes in the target appearance, (caused by a variation in illumination, an occlusion, a change in orientation or scale, a facial expression, etc.), and a filtering component (it adds temporal continuity constraints across frames and deals with dynamics of the tracked object).

In this paper, we propose a deterministic approach based on Canonical Correlation Analysis (CCA). CCA has already been considered for the task of estimating an object's pose from rawbrightness still images, for example in (Melzer et al., 2003).In our case, using the work realized in (Davoine and Dornaika, 2005; La Cascia et al., 2000),we combine CCA with a 3D generic face model to track people's 3D head pose as a rigid object. This document is structured as follows: Section 2 introduces the 3D face model and how we use it to compute shape-free 2D image patches. Then, in section 3 we give a description of Canonical Correlation Analysis (CCA) algorithm in the linear and non linear form. After that, we present how we link these concepts to track a face in a video sequence. Section 4 presents the experimental results obtained from long video sequences. Finally we give our conclusions in section 5.

2 FACE REPRESENTATION

The use of a 3D geometric generic model for tracking purpose has been widely explored in the computer

Alonso Ybáñez Zepeda J., Davoine F. and Charbit M. (2007).
FACE TRACKING USING CANONICAL CORRELATION ANALYSIS.
In *Proceedings of the Second International Conference on Computer Vision Theory and Applications - IU/MTSV*, pages 396-402

vision community. It allows to acquire the 3D face characteristics of a given person and the corresponding texture map to this person's face. In this section we present the 3D geometric model used and the way we employed it to obtain a normalized raw brightness facial patch.

2.1 3d Geometric Model

In our work, we use the *Candide-3* wireframe model, proposed by (Ahlberg, 2001). It consists of a group of n 3D interconnected vertices to describe a face by means of a small number of triangles, enough to reach an acceptable realism to represent a face both statically and dynamically by means of shape and animation units.

The $3n$-vector $\mathbf{c}$ consists of the concatenation of all the vertices, and can be written in terms of the modifications that it can be subject to, as:

$$\mathbf{c} = \mathbf{c}_s + \mathbf{A}\boldsymbol{\tau}_a \tag{1}$$

where $\mathbf{A}\boldsymbol{\tau}_a$ stands for the dynamic part of the model, being the columns of $\mathbf{A}$ the Animation Units and $\boldsymbol{\tau}_a$ the animation control vector. The vector $\mathbf{c}_s = \bar{\mathbf{c}} + \mathbf{S}\boldsymbol{\tau}_s$ corresponds to the static characteristics of a given person (like the nose size, eye separation distance, etc.), being $\bar{\mathbf{c}}$ the standard shape of the *Candide* model, the columns of $\mathbf{S}$ are the Shape Units, and $\boldsymbol{\tau}_s$ is the shape control vector (Ahlberg, 2001). The vectors $\boldsymbol{\tau}_s$ and $\boldsymbol{\tau}_a$ are initialized manually, by fitting the *Candide* shape over the face in the first video frame. Figure 1 shows the initialization as well as the 3D model with the texture warped on it in a frontal view and in three rotated views. These views are useful to adjust the z-component parameters of $\boldsymbol{\tau}_s$ and $\boldsymbol{\tau}_a$ during the model initialization step. The vector $\boldsymbol{\tau}_a$ is supposed to be constant: the face is seen as a rigid object during the tracking process, with a fixed expression.

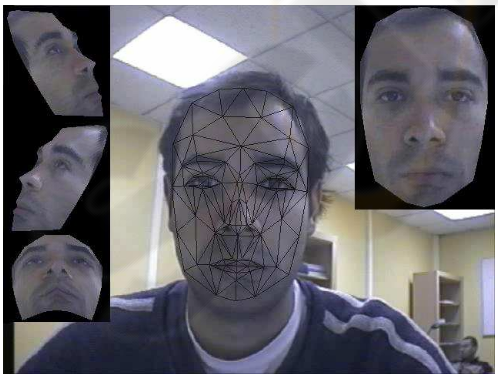

Figure 1: *Candide.* model placed over the target face in the first video frame with frontal and three rotated views.

When placing the *Candide* model over the first frame, we obtain the 3D pose parameters, that we put in our state vector $\mathbf{b}$, given by:

$$\mathbf{b} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z] \tag{2}$$

where the θ elements stand for the rotations and the t elements stand for the translations that we want to track. When we place the model over the first video frame, we crop the texture that lies under it and warp it to the 3D model surface for tracking purposes, as described below.

2.2 Normalization of the Raw Brightness Facial Patch

When tracking a moving face in a 3D environment, we face the problem that its size and geometry are not constant, making difficult the direct construction of a facial appearance model. To cope with this problem, we build a stabilized 2D shape free image patch (a texture map) to code the facial appearance of the person facing the camera. This patch is obtained by warping the rawbrightness image vector lying under the initialized model $\mathbf{c}(b)$ at time $t = 0$ into a fixed size (in our case $d = 96 \times 72$ pixels) 2D projection of the reference Candide model without any expression, i.e. $\boldsymbol{\tau}_a$ set to zero. The patch is finally augmented with the two semi-profile views of the face, as shown in Figure 2.

We can express this mathematically as it was done in (Davoine and Dornaika, 2005) as a transformation $\mathcal{W}$ of the texture observed at each frame $\mathbf{Y}_t$. For a given state vector $\mathbf{b}$, the observation vector $\mathbf{x}_t = \mathcal{W}(\mathbf{b}, \mathbf{Y}_t)$ consists of the columns of the stabilized face image stacked one after the other and normalized such that $\sqrt{\sum_{i=1}^{d} x_i^2} = 1$.

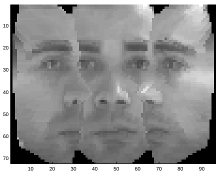

Figure 2: Expression-free patch, used to represent the target face.

3 TRACKING PROPOSITION

The main idea of our algorithm is to estimate the relation that exists between the variation of the state vector and the difference between the current observation vector and the reference vector. To perform this, we consider two approaches to learn a relation between a set of perturbed state parameters and the associated image patches. One is based on Canonical Correlation Analysis, and the other one on kernel CCA to search for the relationships in the higher dimensional space. In this section we will briefly introduce CCA and KCCA, as well as the way we use them to find the model that establishes the relation we are looking for.

3.1 Canonical Correlation Analysis

Canonical correlation analysis is a way of identifying and quantifying the linear relationship between two data sets of random variables. CCA can be seen as the problem of finding basis vectors for two sets of variables, one for $\mathbf{Q}_1$ and the other for $\mathbf{Q}_2$, such that the correlation between the projections of the variables onto these basis vectors are mutually maximized.

Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be the centered data sets corresponding to the $\mathbf{Q}_1$ data matrix of dimension $m \times n$ and $\mathbf{Q}_2$ the data matrix of dimension $m \times p$ respectively. The maximum number of correlations that can be found is equal to the minimum of the data sets' column dimension $min(n, p)$. If we map our data to the directions $\mathbf{w}_1$ and $\mathbf{w}_2$ we obtain, for each pair of directions, two new vectors defined as:

$$\mathbf{z}_1 = \mathbf{A}_1\mathbf{w}_1 \quad \text{and} \quad \mathbf{z}_2 = \mathbf{A}_2\mathbf{w}_2 \tag{3}$$

These vectors are called the *scores* (Borga et al., 1997; Dehon et al., 2000; Weenink, 2003), and we are interested in finding the correlation between them, which is defined as:

$$\rho = \frac{\mathbf{z}_2^T\mathbf{z}_1}{\sqrt{\mathbf{z}_2^T\mathbf{z}_2}\sqrt{\mathbf{z}_1^T\mathbf{z}_1}} \tag{4}$$

Our problem is to find the vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ that maximize (4) subject to the constraints $\mathbf{z}_1^T\mathbf{z}_1 = 1$ and $\mathbf{z}_2^T\mathbf{z}_2 = 1$. In order to do that, we formulate our problem in a Lagrangian form.

As we have the data matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ we can use the method proposed in (Weenink, 2003), to reduce the number of matrix operations, where they perform the singular value decomposition of the data matrices $\mathbf{A}_1 = \mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T$ and $\mathbf{A}_2 = \mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T$, then they introduce the singular value decomposition: $\mathbf{U}_1^T\mathbf{U}_2 = \mathbf{U}\mathbf{D}\mathbf{V}^T$, to finally get:

$$\mathbf{W}_1 = \mathbf{V}_1\mathbf{D}_1^{-1}\mathbf{U} \quad \text{and} \quad \mathbf{W}_2 = \mathbf{V}_2\mathbf{D}_2^{-1}\mathbf{V} \tag{5}$$

where we denote $\mathbf{W}_1$ and $\mathbf{W}_2$ as the matrices containing respectively the *canonical correlation basis vectors* $\mathbf{w}_1$ and $\mathbf{w}_2$.

The main advantage of this procedure is that it avoids the estimation of the covariance matrices and the calculation of the corresponding inverses, and instead, we need to perform three singular value decompositions, which are numerically more robust.

3.2 Tracking Implementation Using the Cca

Once we have shown the CCA, we can proceed to describe our tracking algorithm. It consists of three steps: Initialization, Training process, and Tracking process.

Initialization. During the initialization we place the *Candide* model over the first video frame $\mathbf{Y}_0$, and adjust it to the person's characteristics as previously described, obtaining the state vector $\mathbf{b}_0$, and the reference vector at time $t = 0$ as:

$$\mathbf{x}_0^{(ref)} = \mathcal{W}\left(\mathbf{b}_0, \mathbf{Y}_0\right) \tag{6}$$

Training process. It consists of obtaining a linear model containing the relation between the variation in the state vector $\Delta\mathbf{b}_t$ and the vector resulting from the difference of the observation vector and the reference vector $\mathbf{x}_t - \mathbf{x}_t^{(ref)}$.

To obtain this relationship, we need to define the data matrices $\mathbf{A}_1$ and $\mathbf{A}_2$. The matrix $\mathbf{A}_1$ contains the difference between the m training observation vectors $\mathbf{x}_{Training} = \mathcal{W}\left(\mathbf{b}_{Training}, \mathbf{Y}_0\right)$ and the reference $\mathbf{x}_0^{(ref)}$, and the matrix $\mathbf{A}_2$ contains the corresponding variation in the state vector obtained from the relation $\Delta\mathbf{b}_{Training} = \mathbf{b}_{Training} - \mathbf{b}_0$. The m training points were chosen empirically from a non-regular grid around the vector state obtained at initialization. From these matrices we can obtain the *canonical correlation basis vectors* as described before. Once we have obtained this basis, the general solution consist in doing a linear regression between $\mathbf{z}_1$ and $\mathbf{z}_2$. However, if we develop 4 for each pair of directions with the assumptions made above, we arrive at $\|\mathbf{A}_1\mathbf{w}_1 - \mathbf{A}_2\mathbf{w}_2\|^2 = 2(1-\rho)$ similarly as in (Hardoon et al., 2004). In our case, we have $\rho \approx 1$ (from experimental data), so, we have the following relation:

$$\mathbf{A}_1\mathbf{w}_1 = \mathbf{A}_2\mathbf{w}_2 \tag{7}$$

that simplifies the obtention of our model by avoiding the linear regression needed when $\rho \neq 1$.

We suppose that the learned relations from the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are valid for the current $\Delta\mathbf{b}_t$ and $(\mathbf{x}_t - \mathbf{x}_t^{(ref)})$ in (7), then we can then write:

$$\Delta\mathbf{b}_t\mathbf{w}_2 = (\mathbf{x}_t - \mathbf{x}_t^{(ref)})\mathbf{w}_1 \qquad (8)$$

If we write the result for all the pair of directions in a matrix form, we can replace equations (5) in the last equation, and after some mathematical manipulation, we arrive at:

$$\Delta\mathbf{b}_t = (\mathbf{x}_t - \mathbf{x}_t^{(ref)})\mathbf{G} \qquad (9)$$

where the $\mathbf{G}$ matrix, that encodes the linear model used by our tracker, is given by:

$$\mathbf{G} = \mathbf{V}_1\mathbf{D}_1^{-1}\mathbf{U}\mathbf{V}^T\mathbf{D}_2\mathbf{V}_2^T \qquad (10)$$

It is important to notice that the matrix $\mathbf{G}$ is only valid for a certain rotation interval and it is very dependent on the training points used and on the initialization of the *Candide* model.

Tracking process. The tracking process estimates the state vector variation $\Delta\mathbf{b}_t$ from the difference between the current observation vector and the reference vector by means of the $\mathbf{G}$ matrix built during the training process. To perform this estimation, we obtain the current observation vector, which depends on the current video frame $\mathbf{Y}_t$ and the state vector at the preceding instant $\mathbf{b}_{t-1}$ as:

$$\mathbf{x}_t = \mathcal{W}(\mathbf{b}_{t-1}, \mathbf{Y}_t) \qquad (11)$$

and then, we obtain the difference between the observation vector and the the reference vector $\mathbf{x}_t^{(ref)}$. Then we use this difference vector and the $\mathbf{G}$ matrix to update the state vector as:

$$\hat{\mathbf{b}}_t = \mathbf{b}_{t-1} + (\mathbf{x}_t - \mathbf{x}_t^{(ref)})\mathbf{G} \qquad (12)$$

With $\hat{\mathbf{b}}_t$, we get a new observation vector $\hat{\mathbf{x}}_t = \mathcal{W}(\hat{\mathbf{b}}_t, \mathbf{Y}_t)$ and we compare it with the reference $\mathbf{x}_t^{(ref)}$ to compute the error measure:

$$e(\mathbf{b}_t) = \sum_{i=1}^{d}\left(\frac{\hat{x}_{i,t} - x_{i,t}^{(ref)}}{\sigma_{i,t}^{(ref)}}\right)^2 \qquad (13)$$

where σ_t^2 is the variance of the reference vector. If $e(\mathbf{b}_t)$ is bigger than a certain threshold, we use the state vector estimated $\hat{\mathbf{b}}_t$ in (12) and obtain again an error measure. We iterate a fixed number of times (5, in practice). Once the iterations are done, and in order to increase the robustness of the tracker to weak illumination changes, we proceed to update the reference vector and its variance according to:

$$\mathbf{x}_{t+1}^{(ref)} = \alpha\mathbf{x}_t^{(ref)} + (1-\alpha)\hat{\mathbf{x}}_t \qquad (14)$$

$$\sigma_{t+1}^2 = \alpha\sigma_t^2 + (1-\alpha)(\hat{\mathbf{x}}_t - \mathbf{x}_t^{(ref)})^2 \qquad (15)$$

being $\alpha = 0.99$ obtained empirically.

This approach is useful when we work with data that has a linear behavior. In our case this linear approach showed a good performance, but for comparison reason, we studied also the use of the kernel CCA, to find out if an approach that can cope with non linear relations, can outperform the linear approach previously described. In the following part, we will present the Kernel Canonical Correlation Analysis and the algorithm used for tracking purposes.

3.3 Kernel Canonical Correlation Analysis

The main idea behind kernel methods is that we can still apply a linear method to analyse a given data set, but first we need to map this data into a high dimensional feature space. Thus, using kernel-functions we can formulate our problem as a non-linear version of the original one with the advantage that the complexity of the transformed problem is not linked to the feature space dimension, but to the training data set dimension, which means that we can use kernel transformations to feature spaces of high dimensionality.

If we define $\mathbf{A}_1$ and $\mathbf{A}_2$ as the centred data sets of dimension $m \times n$ and of dimension $m \times p$ respectively, we can apply the CCA to the vectors $\phi(\mathbf{A}_1) = (\phi(\mathbf{a}_1^{(1)})...\phi(\mathbf{a}_m^{(1)}))$ and $\theta(\mathbf{A}_2) = (\theta(\mathbf{a}_1^{(2)})...\theta(\mathbf{a}_m^{(2)}))$, according to the kernel trick (see (Melzer et al., 2003)), which are two non linear mappings.

Then we define the kernel matrices $\mathbf{K}, \mathbf{L}$ by $\mathbf{K}_{ij} = \phi(\mathbf{a}_i^{(1)})\phi(\mathbf{a}_j^{(1)})^T$ and $\mathbf{L}_{ij} = \theta(\mathbf{a}_i^{(2)})\theta(\mathbf{a}_j^{(2)})^T$, with $\mathbf{f}_\phi$ and $\mathbf{g}_\theta$ which can be seen as the coefficients of the linear expansion of the principal vectors $\mathbf{w}_\phi$ and $\mathbf{w}_\theta$ in terms of the transformed data, i.e., $\mathbf{w}_\phi = \phi(\mathbf{A}_1)^T\mathbf{f}_\phi$ and $\mathbf{w}_\theta = \theta(\mathbf{A}_2)^T\mathbf{g}_\theta$.

We can then define as for the CCA the correlation between the transformed data as:

$$\rho = \frac{\mathbf{g}_\theta^T\mathbf{L}^T\mathbf{K}\mathbf{f}_\phi}{\sqrt{\mathbf{g}_\theta^T\mathbf{L}^2\mathbf{g}_\theta}\sqrt{\mathbf{f}_\phi^T\mathbf{K}^2\mathbf{f}_\phi}} \qquad (16)$$

which we maximize in the same way as for the CCA.

The problem that arises from KCCA is that we work with a finite number of points in a high dimensional feature space, which can lead us to useless results. To force useful solutions we introduce a penalizing factor in the norms of the associated weights which leads us to the eigenvalue equations (for more details see (Hardoon et al., 2004)):

$$(\mathbf{K}+\kappa\mathbf{I})^{-1}\mathbf{L}(\mathbf{L}+\kappa\mathbf{I})^{-1}\mathbf{K}\mathbf{f}_\phi = \rho^2\mathbf{f}_\phi \qquad (17)$$

$$(\mathbf{L}+\kappa\mathbf{I})^{-1}\mathbf{K}(\mathbf{K}+\kappa\mathbf{I})^{-1}\mathbf{L}\mathbf{g}_\theta = \rho^2\mathbf{g}_\theta \qquad (18)$$

We can then obtain the vectors $\mathbf{f}_\phi$ and $\mathbf{g}_\theta$ as the eigenvectors of this equations.

3.4 Algorithm Implementation Using the Kcca

As for the CCA, we can divide the algorithm in three parts, the initialization being exactly the same as in the case of the CCA. For the training process, we need to obtain vectors $\mathbf{f}_\phi$ and $\mathbf{g}_\theta$ as described in the last section, using the same data matrices as the ones described for the CCA approach. It is important to point out that in the case of the variation vectors, we did not use any kernel. Once this is done, the starting point for the tracking is equation (7) that can be developed according to the KCCA as:

$$\mathbf{K}\mathbf{f}_\phi \approx \mathbf{A}_2\mathbf{A}_2^T\mathbf{g}_\theta \qquad (19)$$

If we develop as above, we can assume that the tracking data satisfies also this equation and after some mathematical manipulation we arrive at:

$$\Delta\mathbf{b}_t = \mathbf{K}_t\mathbf{f}_\phi(\mathbf{A}_2^T\mathbf{g}_\theta)^{-1} \qquad (20)$$

Here we can see that the $\mathbf{K}_t$ corresponds to the kernel matrix at time t obtained between the training vectors $\mathbf{x}_{-1}^{(i)} = \mathcal{W}(\mathbf{b}_{-1}^{(i)}, \mathbf{Y}_0)$, $i = 1 \ldots m$ and the actual patch $\mathbf{x}_t = \mathcal{W}(\mathbf{b}_{t-1}, \mathbf{Y}_t)$. However, for the algorithm implementation, we compute a linear regression between the result of the product of the matrix kernel $\mathbf{K}\mathbf{f}_\phi$ and the variation vector $\mathbf{Q}_2$, so that we have the actualization equation:

$$\Delta\mathbf{b}_t = \mathbf{K}_t\mathbf{f}_\phi\mathbf{G} \qquad (21)$$

being the matrix $\mathbf{G}$ obtained from the training kernel matrix $\mathbf{K}$. We use in our case the Gaussian RBF kernel function:

$$\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x_i}-\mathbf{x}_j\|^2}{2\sigma^2}\right) \qquad (22)$$

In our experiments, the two parameters σ and κ are set respectively to 0.009 and 0.003. They were obtained empirically from simulations using training sequences, and based on the method used in (Melzer et al., 2003) to estimate a starting point.

4 RESULTS

In order to evaluate our proposals, we implemented our algorithm on a PC with a 3.0 GHz Intel Pentium IV processor and a NVIDIA Quadro NVS 285 graphics card. The non optimized implementation uses OpenGL for the texture mapping and OpenCV for the video capture. We have used the video sequences used by (La Cascia et al., 2000) [1], the annotated talking face video [2], as well as some video sequences made with a Winnov analog video camera XC77B/320.

Figure 3 shows the results obtained from the CCA algorithm. It depicts the state vector estimated compared to the ground truth provided by (La Cascia et al., 2000). We see that the performance of our tracker is close to the ground truth. However, the coordinate system used by our system is not the same as that of the data given, specially for the translationand hence compare them we had normalized both data. There is also a issue with the rotation axes, which are not located at the same point in our system (where the three axes cross close to the nose, due to the *Candide* model's specification), and in the ground truth data provided (where the 3D magnetic tracker was attached on the subjects head). This discrepancy caused principally the difference in the translation parameters, because what in one system represents a rotation, in the other system represents a rotation and a translation.

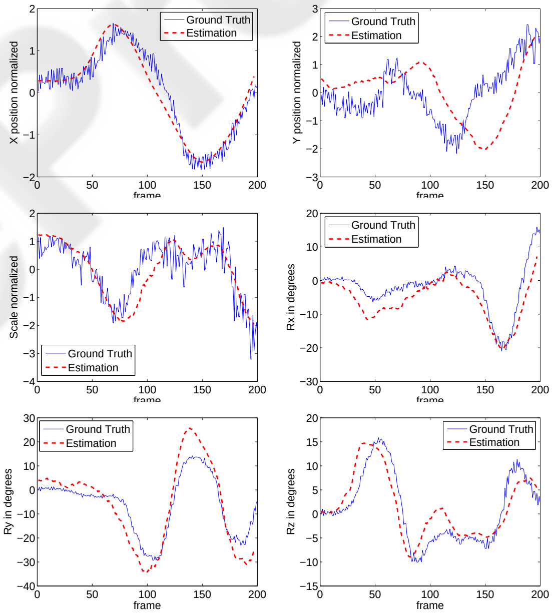

Figure 3: Ground truth compared with the CCA algorithm's results.

Figure 4 depicts some video frames taken from two tracked videos from (La Cascia et al., 2000) and

[1] *http://www.cs.bu.edu/groups/ivc/HeadTracking/*

[2] *http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html*

a webcam video. The CCA and the KCCA algorithm performance did not present any problem when tracking the head in these video sequences, especially because there are not facial gesture involved. In the case of the webcam video, we succeeded in tracking correctly rotations in the y plane going as far as $\pm 35^\circ$. However, when trying to go further, the algorithm could not estimate the correct variation of the angle and got lost.

Another test has been performed using a part of the talking face video. This video presents slight head pose changes compared with the previous employed videos, but it presents more significant movements due to facial gesture. This video shows a person engaged in conversation in front of a camera. It comes with ground truth data that consist of characteristic face points annotated semi-automatically. From the 68 annotated points, we chose the 52 points that were closer to the corresponding *Candide* model's points. Because these points were not exactly the same as the ones given in the ground truth database, there existed an initial distance between the points. In order to measure the behavior of our algorithm, we calculated the standard deviation of this distance, as shown in Figure 5. We can see that the points that presented the higher variance were those in the head's contour.

In Figure 6 we can see the result of tracking the talking face over 1720 frames. The importance of this figure is that we can see the evolution of the error during the video. We have seen that the peaks appearing in this figure represent the moments when there was a facial gesture or a important rotation. However, as seen in the frames displayed, we can cosider that these peaks does not represent a significant error between the state vector estimated and the real head pose.

The time required per frame processing depends on the video size, as can be seen in the table 1. In that table we show also the comparison between the CCA and the KCCA implementation.

Table 1: Comparation of time per frame.

Video's size [pixels]	time per frame [ms]
CCA 640 × 480	147.6
CCA 720 × 576	179.5
KCCA 320 × 240	2486.7

5 CONCLUSIONS

We have seen that the pose tracking is well performed with the two trackers implemented. They managed to follow the head movements in long video sequences of more than 1700 frames. The main advantage of this algorithm is that it is simple and proved to be robust to facial gesture. However, we observed from simulations that the effectiveness of this kind of tracker is dependant on the mask initialization, i.e., the 3D mask must be correctly initialized, in pose and in facial features at the first frame, otherwise, the tracker can get lost because the model affects directly the texture extraction and consequently the state vector predictor.

The results obtained by means of the CCA and the KCCA did not present a significant difference. However, if we consider the computation time required for the KCCA algorithm, which was 10 times slower than the CCA algorithm, we can conclude that for the type of data we use, it is better to use the linear approach.

In our future work we will add the gesture tracking, based on the CCA approach, principally for tracking the mouth and eyebrows, and based on the work of (La Cascia et al., 2000), we will include a robust measure to the tracking algorithm.

REFERENCES

Ahlberg, J. (2001). Candide-3 – an updated parameterized face. Technical Report LiTH-ISY-R-2326, Linkoping University, Sweden.

Borga, M., Landelius, T., and Knutsson, H. (1997). A unified approach to PCA, PLS, MLR and CCA. Report LiTH-ISY-R-1992, ISY, SE-581 83 Linköping, Sweden.

Davoine, F. and Dornaika, F. (2005). *Real-Time Vision for Human Computer Interaction*, chapter Head and Facial Animation Tracking using Appearance-Adaptive Models and Particle Filters. Springer Verlag.

Dehon, C., Filzmoser, P., and Croux, C. (2000). Robust methods for canonical correlation analysis. In Kiers, H., Rasson, J., Groenen, P., and Schrader, M., editors, *Data Analysis, Classification, and Related Methods*, pages 321–326. Springer-Verlag.

Hardoon, D., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis; an overview with application to learning methods. *Neural Computation*, 16:2639–2664.

La Cascia, M., Sclaroff, S., and Athitsos, V. (2000). Fast, reliable head tracking under varying illumination: an approach based on registration of texture-mapped 3D models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):322–336.

Melzer, T., Reiter, M., and Bischof., H. (2003). Appearance models based on kernel canonical correlation analysis. *Pattern Recognition*, 36(9):1961–1973.

Weenink, D. (2003). Canonical correlation analysis. In *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam, Netherlands*, volume 25, pages 81–99.

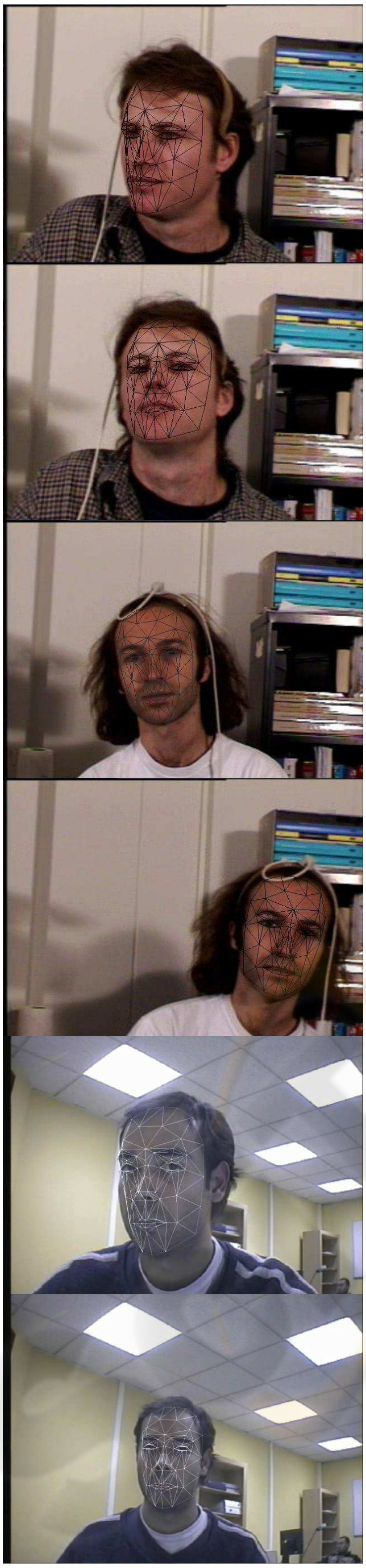

Figure 4: Results from tracking three video sequences using the CCA algorithm in the first five images, and the KCCA algorithm in the last one.

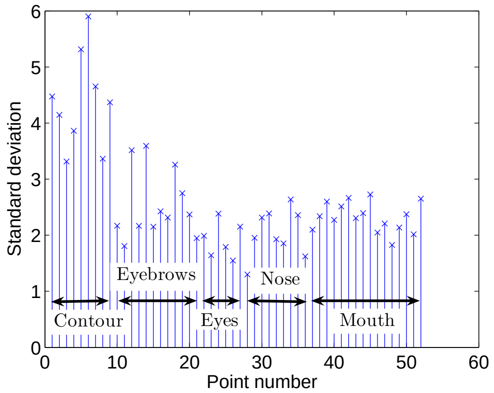

Figure 5: Standard deviation of the points provided for the talking face video sequence.

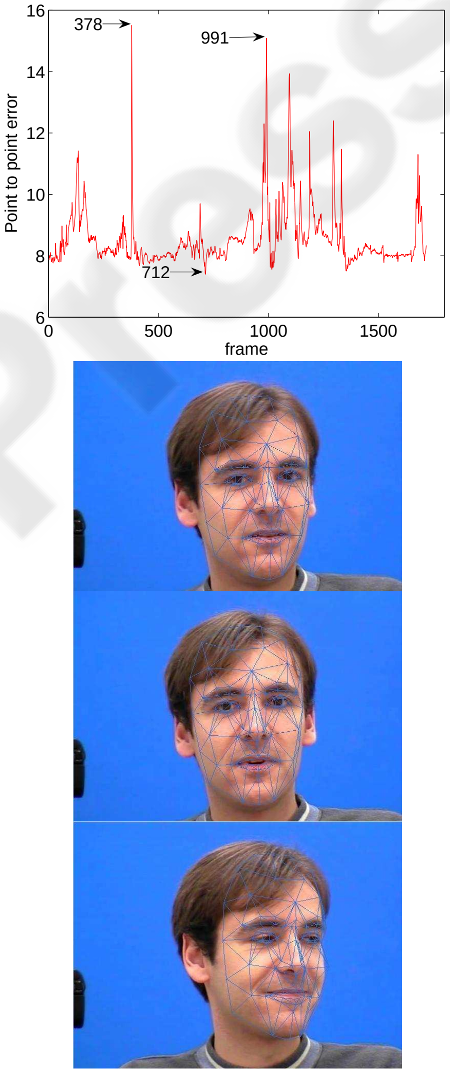

Figure 6: Mean point to point error, and three frames of the sequence corresponding, to frame 378 to frame, 712, and 991 respectively, using the CCA algorithm.