Linear tracking of pose and facial features

José Alonso Ybáñez Zepeda
Télécom Paris,
37/39, rue Dareau,
75014 Paris, France
ybanez@tsi.enst.fr

Franck Davoine
CNRS, UTC, BP20529,
60205 Compiègne cedex,
France
Franck.Davoine@hds.utc.fr

Maurice Charbit
Télécom Paris,
37/39, rue Dareau,
75014 Paris, France
maurice.charbit@enst.fr

Abstract

We present an approach for simultaneous monocular 3D face pose and facial animation tracking. The pose and facial features are estimated from observed raw brightness shape-free 2D image patches. A parameterized 3D face model is adopted to crop out and to normalize the shape of patches from video frames. Starting from the face model aligned on an observed human face, we learn the relation between a set of perturbed parameters of the face model and the associated image patches using a Canonical Correlation Analysis. This knowledge, obtained from an observed patch in the current frame, is used to estimate the correction to be added to the pose of the face and to the animation parameters controlling the lips, eyebrows and eyes. Ground truth data is used to evaluate both the pose and facial animation tracking efficiency in long real video sequences.

1 Introduction

This paper addresses the problem of tracking in a monocular video sequence the global pose of a face as well as the local motion of its main inner features, due to expressions or other facial behaviors. Face tracking poses challenging problems because of the variability of facial appearance within a video sequence, most notably due to changes in head pose, expressions, lighting or occlusions. Many popular learning-based or model-based approaches have been proposed. The first ones, also called view-based approaches, can be formulated as a classification problem based on labeled training examples of 2D face appearances. The second ones generally use a 3D model that is projected into the image and matched to the face to track. Most approaches rely on image cues like key points, curves, optical flow, appearance or skin color, and make use of linear/nonlinear generative or discriminative statistical models to work with 2D facial shape or global appearance manifolds (AAMs, etc.). Others consider part-based statistical representations of the face. A recent work that addresses pose estimation from images is [**?**]. The method is based on the use of kernel Canonical Correlation Analysis to learn the dependencies between the pose and the appearance of an object.
The idea proposed in this paper consists in combining a 3D parameterized geometric face model, used to crop out 2D image patches from incoming video frames, and a linear CCA. The 3D model state (pose and internal geometry) at a given time is efficiently estimated from the observed image patches in the current frame using CCA. This approach provides an elegant and simple way to estimate both the 3D pose of the face and the internal facial features. CCA corresponds to a low-rank approximation of multiple linear regression: this makes the model fitting more robust to noise, if compared to linear regression based approaches. The training is done on synthetic images with known parameters, and needs fewer samples, if compared to other strategies based on Active Appearance Models. In the paper, we compare both the pose and the facial animation estimation with publicly available ground truth data to access the method effectiveness.

2 Face representation

In this work we use the so-called *Candide-3* [**?**] 3D generic face model to acquire the 3D geometry of a person's face and the associated texture map for tracking purposes.

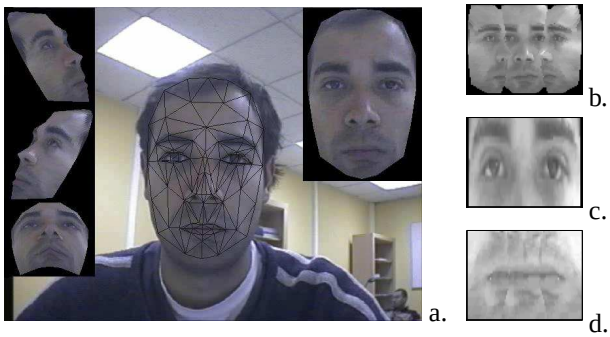

Figure 1: (a) 3D *Candide* model aligned on the target face in the first video frame with the 2D image patch mapped onto its surface (upper right corner) and three other semi-profile synthesized views (left side). Three stabilized face images used for tracking (b) the pose: `SFI_1`, (c) the eyebrows and the eyes: `SFI_2`, and (d) the mouth: `SFI_3`.

This 3D parameterized face model is controlled by Animation Units (AUs). The wireframe consists of a group of 3D interconnected vertices to describe a face with a set of triangles. The vector $\mathbf{g}$ consists of the concatenation of all

the vertices, and can be written as $\mathbf{g} = \mathbf{g}_s + \mathbf{A}\boldsymbol{\tau}_a$, where the columns of $\mathbf{A}$ code 69 face Animation Units and the vector $\boldsymbol{\tau}_a$ is used to control facial mimics so that different expressions can be obtained. $\mathbf{g}_s$ corresponds to the static geometry of a given person's face. $\mathbf{g}_s$ and $\boldsymbol{\tau}_a$ are initialized manually, by fitting the *Candide* shape to the face shape facing the camera in the first video frame (see Figure 1.a). The facial 3D pose and animation state vector $\mathbf{b}$ is then given by:

$$\mathbf{b} = \left[\theta_x, \theta_y, \theta_z, t_x, t_y, t_z, \boldsymbol{\tau}_a^T\right]^T, \quad (1)$$

where $\theta_.$ and $t_.$ components stand respectively for the model rotation around three axes and translation. In this work, we limit the dimension of $\boldsymbol{\tau}_a$ to 9, in order to only track eyebrows, eyes and lips ($\mathbf{b} \in \mathbb{R}^{15}$).
The geometric model $\mathbf{g}(\mathbf{b})$ is used to crop out underlying image patches from the video frames and to transform faces into a normalized facial shape for tracking purposes. We consider here a stabilized 2D shape free image patch to represent the facial appearance of the person facing the camera and to represent observations from the incoming video frame $\mathbf{Y}$. The patch is build by warping the rawbrightness image vector lying under the model $\mathbf{g}(\mathbf{b})$ into a fixed size 2D projection of the standard *Candide* model without any expression ($\boldsymbol{\tau}_a = 0$). Depending on the face's region of interest, we use one of the patches depicted in Figure 1.b, 1.c, and 1.d. These patches can be written as $\mathbf{x} = \mathcal{W}(\mathbf{g}(\mathbf{b}), \mathbf{Y})$, where $\mathcal{W}$ is a piecewise affine warping operator.

3 Integrated tracking framework

Our algorithm for face and facial animation tracking is composed of three steps: initialization, learning and tracking. These three steps are more precisely described in the following sub-sections.

3.1 Initialization

The *Candide* model is placed manually over the first video frame $\mathbf{Y}_0$ at time $t = 0$ and reshaped to the person's face. The correct alignment is obtained considering the 3D model with the corresponding 2D image patch mapped onto its surface, combined with three other semi-profile synthesized views (Figure 1) used mainly, if necessary, to refine the adaptation of the face's depth. Once the model is aligned, we get the state vector $\mathbf{b}_0$, and the reference stabilized face image:

$$\mathbf{x}_0^{(ref)} = \mathcal{W}(\mathbf{g}(\mathbf{b}_0), \mathbf{Y}_0). \quad (2)$$

3.2 Training

Due to the high dimensionality that arises when working with images, the use of a linear mapping to extract some linear features is common in the computer vision domain. In our case, we are interested in identifying and quantifying the linear relationship between two data sets: the change in state of the *Candide* model and the associated facial appearance variations. We propose to use a *Canonical Correlation Analysis* (CCA) to find linear relations between two sets of random vectors [**?**, **?**]. CCA finds pairs of directions or basis vectors (also called canonical factors) for two sets of m vectors, $\mathbf{Q}_1 \in \mathbb{R}^{m\times n}$ and $\mathbf{Q}_2 \in \mathbb{R}^{m\times p}$, such that the correlations between the projections of the vectors onto the directions are mutually maximized.
Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be the centered versions of $\mathbf{Q}_1$ and $\mathbf{Q}_2$, respectively. The maximum number of basis vectors that can be found is $min(n, p)$. If we map our data to the directions $\mathbf{w}_1$ and $\mathbf{w}_2$ we obtain two new vectors defined as:

$$\mathbf{z}_1 = \mathbf{A}_1\mathbf{w}_1 \quad \text{and} \quad \mathbf{z}_2 = \mathbf{A}_2\mathbf{w}_2. \quad (3)$$

and we are interested in maximizing the correlation $\rho = \frac{\mathbf{z}_2^T\mathbf{z}_1}{\sqrt{\mathbf{z}_2^T\mathbf{z}_2}\sqrt{\mathbf{z}_1^T\mathbf{z}_1}}$. The solution consists in finding vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ that maximize $\mathbf{z}_2^T\mathbf{z}_1$ subject to the constraints $\mathbf{z}_1^T\mathbf{z}_1 = 1$ and $\mathbf{z}_2^T\mathbf{z}_2 = 1$.

In this work, we use the numerically robust method proposed in [**?**]. We compute singular value decompositions of the data matrices $\mathbf{A}_1 = \mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T$ and $\mathbf{A}_2 = \mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T$, and then, the following the singular value decomposition: $\mathbf{U}_1^T\mathbf{U}_2 = \mathbf{U}\mathbf{D}\mathbf{V}^T$, to finally get:

$$\mathbf{W}_1 = \mathbf{V}_1\mathbf{D}_1^{-1}\mathbf{U} \quad \text{and} \quad \mathbf{W}_2 = \mathbf{V}_2\mathbf{D}_2^{-1}\mathbf{V}, \quad (4)$$

where matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ contain respectively the full set of *canonical correlation basis vectors*. In our case, the matrix $\mathbf{A}_1$ contains the difference between the training observation vectors $\mathbf{x}_{Training} = \mathcal{W}(\mathbf{g}(\mathbf{b}_{Training}), \mathbf{Y}_0)$ and the reference $\mathbf{x}_0^{(ref)}$, and the matrix $\mathbf{A}_2$ contains the variation in the state vector $\Delta\mathbf{b}_{Training}$ given by $\mathbf{b}_{Training} = \mathbf{b}_0 + \Delta\mathbf{b}_{Training}$. The m training points were chosen empirically from a non-regular grid around the vector state obtained at initialization.

Once we have obtained all the canonical correlation basis vectors, the general solution consist in performing a linear regression between $\mathbf{z}_1$ and $\mathbf{z}_2$. However, if we develop the correlation for each pair of directions with the assumptions made above, we get $\|\mathbf{A}_1\mathbf{w}_1 - \mathbf{A}_2\mathbf{w}_2\|^2 = 2(1-\rho)$ similarly as in [**?**]. Based on our experiments, we observe that $\rho \approx 1$, and so, we use the relation $\mathbf{A}_1\mathbf{w}_1 \approx \mathbf{A}_2\mathbf{w}_2$. If we substitute matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ by $\Delta\mathbf{b}_t$ and $(\mathbf{x}_t - \mathbf{x}_t^{(ref)})$ in the last relation, we come that $\Delta\mathbf{b}_t\mathbf{w}_2 = (\mathbf{x}_t - \mathbf{x}_t^{(ref)})\mathbf{w}_1$. This is true for all the *canonical variates*, so we use equations (4) to get a result for all the directions:

$$\Delta\mathbf{b}_t = (\mathbf{x}_t - \mathbf{x}_t^{(ref)})\mathbf{G}, \quad (5)$$

where $\mathbf{G} = \mathbf{V}_1\mathbf{D}_1^{-1}\mathbf{U}\mathbf{V}^T\mathbf{D}_2\mathbf{V}_2^T$, encodes the linear model used by our tracker, which is explained in the following section.

3.3 Tracking

The tracking process consists in estimating the state vector $\Delta\mathbf{b}_t$ when a new video frame $\mathbf{Y}_t$ is available. In order to do that, we need, first, to obtain the stabilized face image, from the incoming frame by means of the state at the preceding time, as:

$$\mathbf{x}_t = \mathcal{W}(\mathbf{g}(\mathbf{b}_{t-1}), \mathbf{Y}_t), \quad (6)$$

and then make the difference between this image and the reference stabilized face image $\mathbf{x}_t^{(ref)}$. This gives an error vector from which we estimate the changes in state with (5). Then we can write the state vector update equation as:

$$\hat{\mathbf{b}}_t = \mathbf{b}_{t-1} + (\mathbf{x}_t - \mathbf{x}_t^{(ref)})\mathbf{G}. \quad (7)$$

We iterate a fixed number of times (5, in practice) and estimate another $\hat{\mathbf{b}}_t$ according to (7) and update the state vector. Once the iterations are done, we update $\mathbf{x}_{t+1}^{(ref)} = \alpha\mathbf{x}_t^{(ref)} + (1-\alpha)\hat{\mathbf{x}}_t$, with $\alpha = 0.99$ obtained from experimental results.

4 Implementation

The algorithm has been implemented on a PC with a 3.0 GHz P4 processor and a NVIDIA Quadro NVS 285 graphic card. Our non optimized code uses OpenGL for texture mapping and OpenCV for video capture. We use a standard desktop Winnov analog video camera to generate the sequences for tests. We retain the following nine animation parameters, for facial gesture tracking:

(1) upper lip raiser
(2) jaw drop
(3) mouth stretch
(4) lip corner depressor
(5) eyebrow lowerer
(6) outer eyebrow raiser
(7) eyes closed
(8) yaw left eyeball
(9) yaw right eyeball

Based on the algorithm described in section 3, we have implemented a tracker that uses three stabilized face images (see Figure 1) sequentially: one to track the head pose (`SFI_1`), one to track the lower face animation parameters (`SFI_3`), and a last one (`SFI_2`) to track the upper face animation parameters. `SFI_1`, `SFI_2` and `SFI_3` are respectively composed of 96×72, 86×28, and 88×42 pixels. For training, we use 317 state vectors with the corresponding appearance variations for the pose, 240 for the upper face region and 200 for the mouth region. We chose these points empirically, from a symmetric grid centered on the initial state vector. The sampling is dense close to the origin and coarse when getting far from it. Due to the high dimensionality of our state vectors, we did not use all the combinations between the selected points.

5 Experimental results

For validation purposes, we use the video sequences described in [**?**] for pose tracking, and the talking face video made available from the *Face and Gesture Recognition Working Group*, for both pose and facial animation tracking. These sequences are supplied with ground truth data. In this section, we show and analyze quantitatively the performance of the tracker over the two types of video sequences.

3D pose tracking. Video sequences provided in [**?**][1] are 200 frames long, with a resolution of 320×240, 30 *fps.*, taken under uniform illumination, where the subjects perform free head motion including translations and both in-plane and out-of-plane rotations. Ground truth has been collected via a "Flock of Birds" 3D magnetic tracker. Figure 2 shows the estimated pose compared with the ground data. Temporal shifts can be explained because the center of the coordinate systems used in [**?**] and ours are slightly different. In our case, the three axes cross close to the nose, due to the *Candide* model specification, and in the ground truth data, the 3D magnetic tracker is attached on the subject's head. We check experimentally on all the provided video sequences the stability and precision of the tracker and do not observe divergences of the tracker.

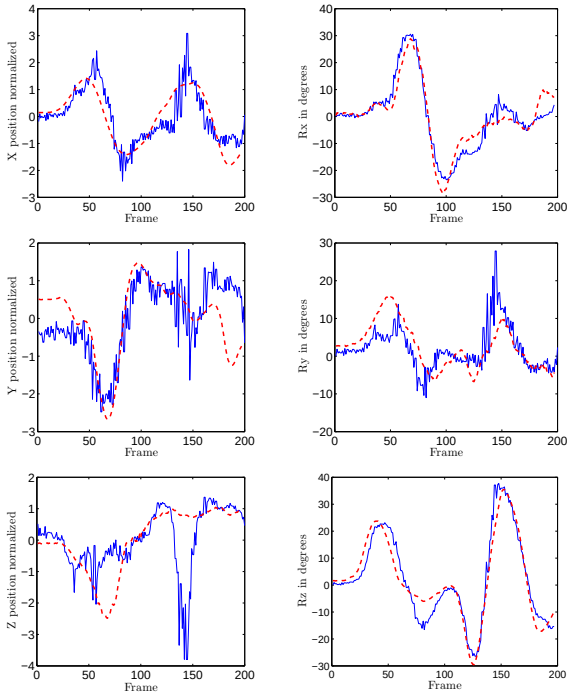

Figure 2: 3D pose tracking: the graphs show the estimated 3D pose parameters during tracking (dashed lines) compared to ground truth (solid lines).

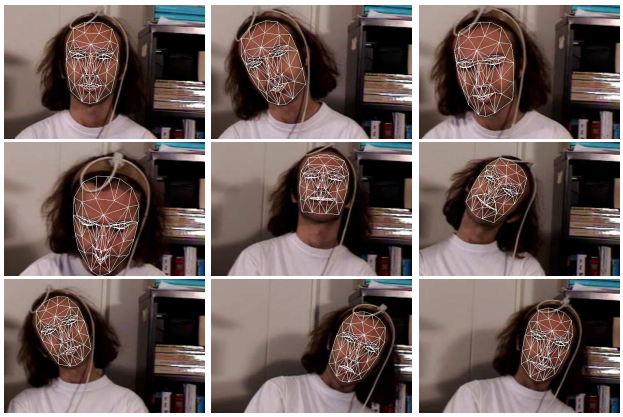

Figure 3: Sample frames at times 16, 40, 54, 67, 106, 128, 146, 185 and 197.

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

Simultaneous pose and facial animation tracking. The talking face video sequence consists of 5000 frames (about 200 seconds of recording)[2], with a resolution of 720×576, taken from a video of a person engaged in conversation. For practical reasons (to display varying parameter values on readable graphs) we used 1720 frames of the video sequence, where the ground truth consists of characteristic 2D facial points annotated semi-automatically. From 68 annotated points per frame, we select 52 points that are closer to the corresponding *Candide* model points. In order to evaluate the behavior of our algorithm we calculated for each point the standard deviation of the distances between the ground truth and the estimated coordinates.

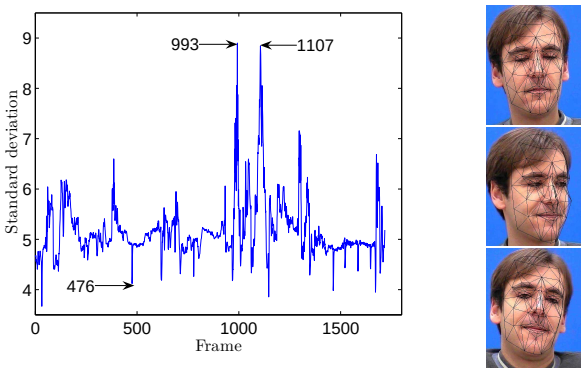

Figure 4: Left: mean standard deviation of the 52 facial points for each frame. Right: video frames from top to bottom: 476, 993, and 1107.

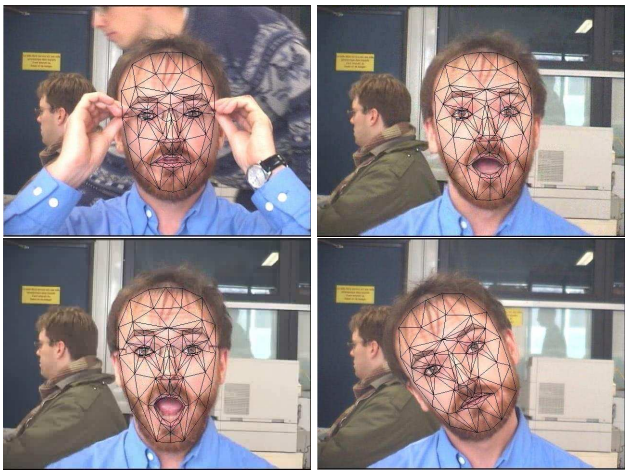

Figure 5: Tracking results using a webcam.

Figure **??** depicts the standard deviation over the whole video sequence for each point. We can see that the points with the greater standard deviation correspond to those on the contour of the face. The precision of these points is strongly related to the correctness of the estimated pose parameters. We see that the mean standard deviation of the 52 facial points stays approximately constant with some peaks. These peaks correspond to important facial movements. In case of frame 993 the rotation around the y axe corresponds to $36.62°$. In frame 1107, the rotations around on the x, y and z axes are respectively $-13.3°$, $18.9°$ and $-10.5°$. We observe on the whole video sequence that even if peak values are large, the tracker still performs correctly.

The average time for pose and facial animation tracking is about 46 ms per frame if we exclude the time for video read, decompression and write/display operations. The average time for training is 23.2 seconds.

Experiments have been conducted to evaluate the sensitivity of the facial animation tracker in case of unprecise 3D pose estimations. We added some random noise to the six pose estimated parameters before estimating the facial animation, within the following intervals: $\pm 10\%$ of the estimated head width added to the three translation parameters, and $\pm 3°$ added to the three rotation parameters. Such perturbations not not introduce visible effects on the tracking results. Figure **??** displays sample frames from other video sequences, to show the robustness of the tracker even in case of cluttered backgrounds.

[2] www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/

6 Conclusion

We have presented a method to track both 3D pose and facial animation parameters from persons in monocular video sequences. The approach is simple, from the training and tracking point of views, robust and accurate in case the out-of-plane face rotation angles stay in the interval $\pm 30°$. The method can still be improved. As regards immediate extensions, the method will be combined with a facial feature detection algorithm to re-synchronize the tracking in case of divergence.

References

[1] J. Ahlberg. Candide-3 – an updated parameterized face. Technical Report LiTH-ISY-R-2326, Linkoping University, Sweden, Jan 2001. 1

[2] M. Borga, T. Landelius, and H. Knutsson. A unified approach to PCA, PLS, MLR and CCA. Report LiTH-ISY-R-1992, Linköping University, Sweden, 1997. 2

[3] D.R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis; an overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004. 2

[4] M. La Cascia, S. Sclaroff, and V. Athitsos. 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, April 2000. 3

[5] T. Meltez, M. Reiter, and H. Bischof. Appearance models based on kernel canonical correlation analysis. *Pattern recognition*, 36:1961–1971, 2003. 1

[6] David Weenink. Canonical correlation analysis. In *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam, Netherlands*, volume 25, pages 81–99, 2003. 2