Institut EURECOM
2229, route des Crêtes
B.P. 193
06904 Sophia Antipolis
FRANCE

Research Report N° RR-99-051

A Visual Analysis/Synthesis Feedback Loop for Unconstrained Face Tracking

Stéphane Valente & Jean–Luc Dugelay

November 1999

Telephone: +33 (0)4 93 00 26 27
+33 (0)4 93 00 26 41
Fax: +33 (0)4 93 00 26 27

E-mail: valente@eurecom.fr
dugelay@eurecom.fr

Abstract

We propose a novel approach for face tracking, resulting in a visual feedback loop: instead of trying to adapt a more or less realistic artificial face model to an individual, we construct from precise range data a specific texture and wire-frame face model, whose realism allows the analysis and synthesis modules to visually cooperate in the image plane, by directly using $2D$ patterns synthesized by the face model. Unlike other feedback loops found in the literature, we do not explicitely handle the $3D$ complex geometric data of the face model, to make real–time manipulations possible.

Our main contribution is a complete face tracking and pose estimation framework, with few assumptions about the face rigid motion (allowing large rotations out of the image plane), and without marks or makeup on the user's face. Our framework feeds the feature–tracking procedure with synthesized facial patterns, controlled by an extended Kalman filter. Within this framework, we present original and efficient geometric and photometric modelling techniques, and a reformulation of a block–matching algorithm to make it match synthesized patterns with real images, and avoid background areas during the matching. We also offer some numerical evaluations, assessing the validity of our algorithms, and new developments in the context of facial animation.

Our face tracking algorithm may be used to recover the $3D$ position and orientation of a real face and generate a MPEG 4 animation stream to reproduce the rigid motion of the face with a synthetic face model. It may also serve as a pre–processing step for further facial expression analysis algorithms, since it locates the position of the facial features in the image plane, and gives 3D information to take into account the possible coupling between pose and expressions of the analysed facial images.

Keywords

Analysis/Synthesis Cooperation; Face Tracking; Face Modeling; Kalman Filter; Face Cloning; Virtual Teleconferencing; MPEG 4.

Contents

1 Context

1.1 Facial Animation in MPEG 4

Facial animation is currently a hot topic for the MPEG 4 standard [1], which created the *Synthetic/Natural Hybrid Coding* (SNHC) working group [2] to mix real and computer–generated objects for the reproduction of real–world scenes. MPEG 4 proposes the use of a synthetic face model in a wide range of applications, from virtual actors, video games, human–machine communication, story tellers on demand, multimedia interactive environments, video–telephony and, as in the context of this work, virtual conferencing [3, 4, 5, 6, 7]. Facial animations are controlled by *face animation parameters* (FAP), which manipulate the displacements and angles of the face's features.

MPEG 4 defines a stream syntax and a decoder for the synthetic face model, but does not provide or impose any technique for obtaining the stream of facial animations, which is where many contributions are still possible. For example, it allows the use of text–to–speech systems for automated talking heads, or hand–defined animations. However, animating a face model given the performance of a real actor or user (*face cloning*), although it is an unsolved problem, remains a major issue, and could ensure the acceptance and success of future services based on this standard.

The main contribution of this paper is a rigid motion estimation algorithm of a face in a video sequence, that achieves near real–time performance, without any constraints on the user (no markers or makeup, unknown lighting and no specific background), allowing large rotations of the head. Offline, our framework can be used to generate precise orientation parameters for a MPEG–4 animation stream (FAP 48 to 50), regardless of the method used to obtain the other FAPs, be it from speech, text and hand. It can be used, for instance, in conjunction with other algorithms extracting facial expressions from images, or the animation rules of automated talking heads to obtain head motions with life–like timings (timings that happen to be difficult to obtain with key–frame animation systems by hand, even for animation experts). Online, apart from MPEG–4 systems, it can also be integrated in a virtual teleconferencing system, recovering the head pose during the session.

1.2 Face Cloning for Virtual Teleconferencing

Face cloning techniques are particularly useful for *Virtual Teleconferencing* systems, which is the context of this work [8]: the key idea is to provide the participants with a common meeting space as if they were all sitting in the same physical room, and to offer them individual points of view, in accordance with the virtual position they occupy in the meeting space. In addition to more flexible visualization possibilities, such as the scene restitution from different points of view [9], the model–based aspect of face cloning allows the meeting to be run over low bit–rate networks, such as the Internet, or mobile networks. However, due to the $3D$ and interactive nature of the system, several constraints must be added to the face cloning algorithm:

- the face analysis and synthesis frame–rates, and the image processing delays, should be as low as possible;
- the clones can be rendered from any point of view, and a full $3D$ model is needed (not only the frontal part of the face);
- the face cloning system should operate without colored marks taped on the performer's face;
- it should deal with unknown lighting conditions and background;
- the motions and rotations of the user should not be restricted in front of the camera;
- finally, the clones should be visually realistic.

Several efficient $2D$ face tracking algorithms exist in the literature, such as [10]. Nevertheless, when large motions are allowed by a face cloning system, tracking the user's face in the video sequence without recovering the exact position and orientation of the real face (i.e. its global motion) is pointless for such a system, firstly because facial expression analysis algorithms need to know the precise location of the facial features in the $2D$ image, and secondly because some coupling occurs between the pose and the facial expressions: for instance, when the performer looks downward with a neutral expression, his mouth shape will change in the image plane, and the system is likely to misinterpret it as a smile. Being aware of the subject's pose may help the system solve this coupling. Therefore, $2D$ face tracking techniques, like color segmentation, hidden markov models, deformable templates, FFT... are more suitable for face recognition or lipreading applications [11].

Assuming that a face model is available (be it generic or person–dependent), we will now concentrate on the different image–processing approaches that allow the $3D$ rigid motion of the user to be estimated, without any markers, voluntarily omitting $2D$–tracking algorithms.

1.3 Face Tracking and Global Motion Estimation

One method to estimate the rigid motion of an object is to use some kind of motion information, like the optical flow, and interpret it using some shape model, even a simple one. Azarbayejani *et al* [12] use feature point tracking, and project the $2D$ points onto an ellipsoid to incorporate some $3D$ information for the rigid motion estimation. Basu *et al* [13] regularize the velocity field of the user's face with a $3D$ ellipsoidal model to determine the best fit between the position and orientation of the model and the motion of the image pixels. DeCarlo and Metaxas [14] initialize a polygonal head model on the user's face, and also regularize the optical flow in terms of model displacements. Because their model has a shape closer to the real face than the ellipsoid, their motion estimation is finer. The main problem with these approaches is that they are "blind", in the sense that they have no way to ground the model to the real face, and errors in the estimation accumulate, resulting in a lack of accuracy or tracking failures.

To prevent cumulative estimation errors, an analysis–by–synthesis feedback loop can be introduced at the encoder, adding some texture information to the shape of the face model. Koch [15] extracts some texture from the first video frame to be mapped onto a face model. Images are then synthesized from his parametric scene description, and compared with the original input images of the camera to solve the model's motion. Li *et al* [16] use the CANDIDE model and the texture from the first image to iterate the motion estimation process until the synthesis error becomes satisfactory. The strength of their algorithm is that no costly optical flow is computed, instead, they have a direct computation approach, based on an aggregation of data, which is more robust to noise. Eisert and Girod [17] perform the motion estimation on the optical flow between the synthesized image of a realistic textured face model and the real image. In [18], they add an illumination model to prevent breaking the brightness constancy assumption of their optical flow computation. Although these algorithms lead to simple formulations and fewer cumulative errors, interpreting the pixel differences or the pixel velocity field can be costly, because of the linearization of the model $3D$ displacements in the $2D$ image frame for each frame. As a result, they do not claim to be real–time. Moreover, in principle, such algorithms can deal with self–occlusions due to large rigid motion, and the coupling between the pose and expression of the real face, but none of them report their ability to track large head rotations.

Another approach is taken by Schödl *et al* [19]. The first image of the face is projected on a generic polygonal face model, and tracking is achieved by synthesizing the face model, and computing the error with the real images. The derivative of the error with respect to the motion parameters is mapped to the intensity gradients in the image, and a steepest descent is iterated until a local minimum of error is reached. This is also a costly procedure, because the face model must be rendered at each iteration, and changes of the face lighting, combined with the lack of precision of the shape of their model, may lead to an incorrect registration of the synthetic image.

1.4 A Novel Approach Based on Visual Realism

We noticed that in the literature, none of the face cloning or face tracking approaches takes advantage of the *visual* realism of their face model to track and/or analyze facial deformations in a more direct manner, as they all rely on the explicit use of all the geometric information contained in the face model to perform the motion regularization, even at the penalizing cost implied by the linearization of the motion of the face vertices in the $2D$ image plane. We believe that a better alternative to reach near real–time performance, is to use a realistic face model, let computer graphics hardware translate the information of the $3D$ shape (including self–occlusions and changes of lighting) into the $2D$ image plane, and work only at the image level, using little $3D$ information from the model. Confirming the idea that a dense motion estimation is not necessary, Jebara and Pentland [20] track a few facial features, which are extracted from the first video frame, with correlation–based trackers, and a Kalman filter estimates the $3D$ motion and structure of the facial feature points. Their system works in real–time, and allows large head rotations. Because they do not have a face model, their algorithm cannot model self–occlusions or changes of

lighting due to the face pose, resulting sometimes in incorrect registrations. To overcome these limitations, we propose to update the tracked patterns with synthesized images of the facial feature.

Therefore, we introduce a novel approach for a visual feedback loop, making the analysis and synthesis modules cooperate only at the image level, without explicitely handling the $3D$ geometric components of the face model within the loop. Our point in this paper is to provide key elements on the advances permitted by such a feedback, in particular to show how a realistic and precise face model, along with appropriate photometric modelling techniques, can be used to robustly track a real face, and accurately recover its rigid motion, with an uncalibrated camera, no markers, unknown lighting conditions, and a non–uniform background.

Section 2 outlines our rigid motion estimation algorithm, as shown in Figure 1. The next sections describe the technical issues that allow our analysis/synthesis cooperation in the image plane. In section 3, we present a reconstruction technique that creates realistic face geometric models suitable for real–time synthesis. Section 4 introduces a novel and efficient $3D$ illumination compensation algorithm while section 5 details the extended Kalman filter that drives the feedback loop. Section 6 describes the reformulation of the block–matching algorithm to track synthetic patterns in real images, including the robustness to the speaker's background. We discuss in section 7 the performance of our face tracking algorithm on a synthetic sequence. Finally, the local animation possibilities of our models are given as perspectives in section 8, as we are working on the extraction of facial expression parameters from images, still using the visual appearance of our face model.

2 Outline of the Tracking Algorithm

We have written a face tracking and pose determination system which relies heavily on the cooperation between analysis and synthesis techniques in the image plane. The algorithm can be outlined as follows (see Figure 1, where the next numbered steps are reported):

Initialization:

(a) the user aligns his/her head with the head model, or modifies the initial pose parameters to align the head model with his head. This first alignment obviously requires some user intervention, but it has to be performed only once per session, and we do not consider it as a major issue at the moment. Indeed, some publications have proposed interesting automatic solutions, such as frame–fitting [21, 22], or eigenfeatures [20];

(b) when completed, a $3D$ illumination compensation algorithm is run, to estimate the lighting parameters that will reduce the photometric differences between the synthetic face model and the real face in the user's environment;

Main Loop:

(i) a Kalman filter predicts the head $3D$ position and orientation for time t;

(ii) the synthetic face model is used to generate an approximation of the way the real face will appear in the video frame at time t; this approximation includes both geometric distortions, scales and shaded lighting due to the speaker's pose; the black background of this rendering is used by the encoder as visual clues about the location of the unkown environment with respect to the face facial features, and it will not be used for the $3D$ parameters restitution;

(iii) patterns are extracted from the synthesized image, representing contrasted facial features (like the eyes, eyebrows, mouth corners, nostrils);

(iv) a differential block–matching algorithm matches these patterns with the user's facial features in the real video frame;

(v) the $2D$ coordinates of the found positions are given to the Kalman filter, which estimates the current head $3D$ position and orientation;

(vi) finally, the filtered parameters are quantified, compressed and streamed to the visualization entities, and the rigid motion of the face is recomposed by a decoder, possibly from a different point of view and a new background, or even using a totally different face model.

The strength of the visual feedback loop is that it implicitly takes into account the changes of scale, geometry, lighting and background with almost no overload for the feature–matching algorithm: due to the synthesis module that performs a $3D$ illumination compensation scheme, the synthesized patterns will predict the geometric deformations, the lighting and the background of the user's facial features, making the differential block–matching pass more robust. This enhanced analysis/synthesis cooperation results in a stable face tracking framework without artificial marks highlighting the facial features, supports very large rotations out of the image plane (see Figure 11), and even copes with low–contrasting lightings (see the video demo [23]).

It is clear that such a feedback loop is successful because it combines a realistic face model, efficient lighting modelling techniques, a carefully designed Kalman filter to make up for the uncalibrated camera, and a block–matching algorithm that handles the background interferences during large head rotations. The next sections will deal with these points in details.

3 Geometric Modeling

We are currently using range data obtained from cylindrical geometry Cyberware range finders [24] to build person–dependent realistic face models. This part is largely independent from the global motion tracking algorithm, and in fact, any realistic face model could be integrated in the analysis/synthesis feedback loop, as long as it is compatible with real–time visualization rates. We should mention here that other techniques are available in the literature to obtain a textured wireframe, using structured lighting for example [25]. In the next paragraphs, we will describe a face model construction algorithm from range data, which provides a high degree of realism while at the same time limitting the number of $3D$ primitives due to an interactive refinement procedure.

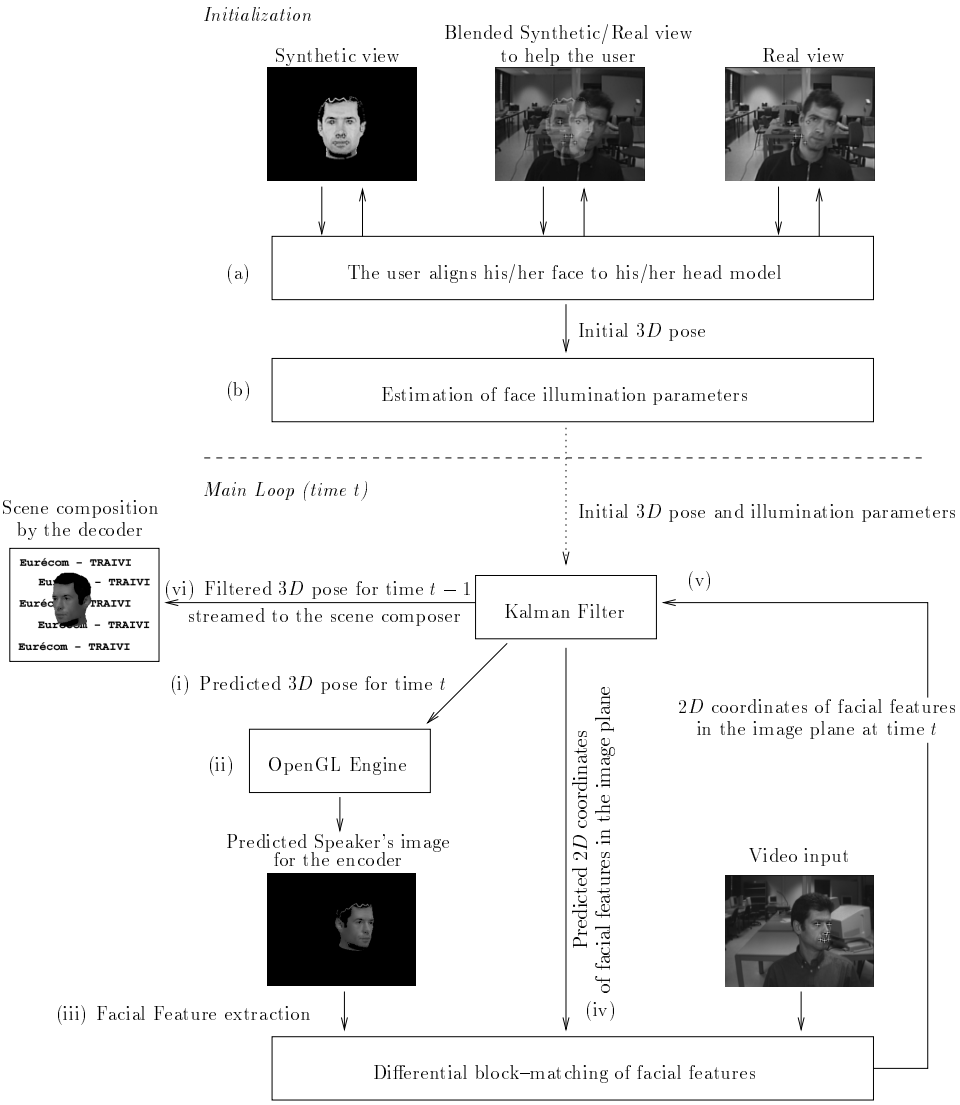


Figure 1: System overview: initialization, and face tracking loop — the numbered items (a), (b) and (i) to (vi) are detailed in section 2.

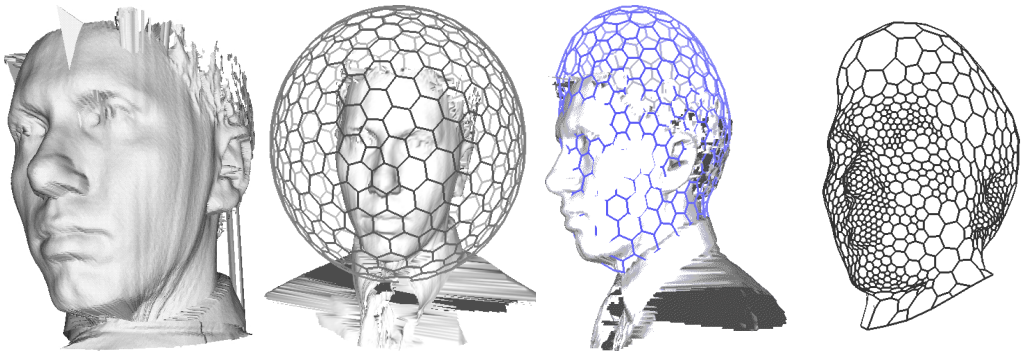
3.1 Initial Data

Cyberware scanners produce a dense range image with its corresponding color texture (see Figure 2(a) and (b)). This dataset is a highly realistic representation of the speaker's face. However, it cannot be used directly in a face cloning system for several reasons. First, this dataset is very dense (an average of 1.4 million vertices) and therefore is not well suited for real–time computations. Furthermore, due to the limitation of the acquisition technology, the dataset is often incomplete and sometimes includes some outliers (as in Figure 2(a)). Finally, it is not suitable for local deformation computations: it is just a $3D$ surface, with no anatomical knowledge or built–in animations for the facial features.

3.2 Mesh Construction

To achieve both visual realism and real–time computation, we need a geometric model with a limited number of vertices but with enough details in order to distinguish facial features such as the lips or the eyebrows. We use a reconstruction system based on deformable simplex meshes [26] to build such models. Unlike classic approaches, those deformable models are handled as discrete meshes, not relying on any parameterization. Because they are topologically equivalent to triangulations, they can be easily converted as a set of triangles for display purposes or standard $3D$ file formats like VRML [27]. Finally, they can represent geometric models independently of their topology and they lead to rapid computations.

The different stages of construction from a Cyberware dataset, where the hair information is missing and with some outliers, are shown in Figure 2. The deformable model is initialized as a sphere (Figure 2(b)) and then deformed to roughly approximate the face geometry (Figure 2(c)). The last stage consists in refining the mesh model based on the distance between the range image and the surface curvature (Figure 2(d)), by selecting the areas where more modelling precision is desired.



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

Figure 2: Reconstruction of a geometric model from a Cyberware dataset: (a) initial range data; (b) initialization; (c) main deformation; (d) mesh refinement — We have interactively selected the areas of interest (chin, ears, nose, lips) where the refinements are performed. The resulting mesh has 2084 vertices.

The face model is then texture–mapped by computing for each vertex of the simplex mesh, its (u, v) coordinates in the range texture image. Since the deformation of each vertex requires the computation of its closest point in the range image, we simply use the texture coordinates of the closest point. Where no range data is available (at the hair level for instance), we project the vertex onto the image plane through the cylindrical transformation of the Cyberware acquisition. This algorithm therefore produces an accurate geometric and texture face model, and has to be run only once for a given head model.

Although some authors reported that adapting a generic face model (CANDIDE) is quite feasible [28, 29], we believe that their model could not suit our analysis/synthesis feedback loop due to a low number of primitives, mainly because it

would not provide enough lighting normals, which are crucial for the illumination compensation procedure, discussed in the next section.

4 Photometric Modeling

4.1 Motivation

The goal of photometric modelling is to reduce the photometric discrepancies between the speaker's face in the real world environment and his synthetic model directly at the $3D$ level within the visual feedback loop — otherwise, the block–matching algorithm would constantly fail to match the tracked facial features, because it assumes that the brightness distribution and the shading of synthetic patterns are close to the real facial features, which is far from being true by default (see Figures 4(a) and 4(b)). This modelling can be seen as an alternative and elegant technique to other $2D$ view–based techniques, such as histogram fitting [20]. In [18], Eisert and Girod propose an algorithm to recover the $3D$ position and intensity of a single infinite light source from a static view assuming an initial guess of the position prior to the motion estimation. Bozdağı *et al.* [30] have a more complex approach that determines the mean illumination direction and surface albedo to be included in their Optical Flow equation for motion estimation. Both approaches are based on a Lambertian illumination model (i.e. composed of ambient and diffuse lighting) without specular reflections and cast shadows. However, in the real world, cast shadows, and specular highlights (if the user does not have make–up), are likely to occur on a face, and will be difficult to compensate using only a single light as in the previous algorithms.

In [31], Belhumeur and Kriegmann derive that the set of images of a convex Lambertian object under all possible lighting conditions is a cone, which can be constructed from three properly chosen images, and empirically show that cast shadows and specular reflections generally do not damage the conic aspect of the set.

Motivated by the reconstruction possibility of an arbitrary illuminated view from several object images, we propose to recover the face illumination from a single speaker's view by using a set of light sources at different infinite positions. The main advantage of our algorithm is that it can rely on graphics hardware acceleration and compensate unknown light sources with ambient, diffuse and specular components at the $3D$ level in real–time. A similar idea, applied to interior design, can be found in [32], where the scene's global lighting is computed from the illumination of some objects painted by hand by the scene designer. In our algorithm, the synthetic scene lighting is adjusted by observing the illumination of the facial features in their real environment.

4.2 Proposed Algorithm

We propose to adopt the following general lighting model, including ambient, diffuse and specular reflections induced by N independent infinite light sources for a $3D$ textured primitive, with an additional degree of freedom (a luminance offset L_{offset},

which will be justified in section 4.5)

$$L_{\text{real}} = L_{\text{offset}} + L_{\text{texture}} \times (A + \sum_{i=0}^{i=N-1} [(\max\{\boldsymbol{l}_i.\boldsymbol{n}, 0\}) \times D_i + (\max\{\boldsymbol{s}_i.\boldsymbol{n}, 0\})^{\text{shininess}} \times S_i]) \quad (1)$$

where L_{real} denotes the luminance of a pixel in the real image, L_{texture} the corresponding texture luminance of the face model, A the global ambient light intensity, D_i and S_i the diffuse and specular intensity for the i^{th} light, $\boldsymbol{n}$ and $\boldsymbol{l}_i$ the object normal and the i^{th} light source direction, $\boldsymbol{s}_i$ the normalized bisector between the i^{th} light source direction and the viewing direction, and finally "shininess", the specular exponent controlling the size and brightness of specular highlights.

One can readily verify that the rendered image pixels values in equation (1) are linear with respect to the components of the light sources. All the unknowns (the light source intensities, and the luminance offset if needed) can be estimated by a simple least mean square inversion for all the face pixels. Our algorithm thus consists of the following steps:

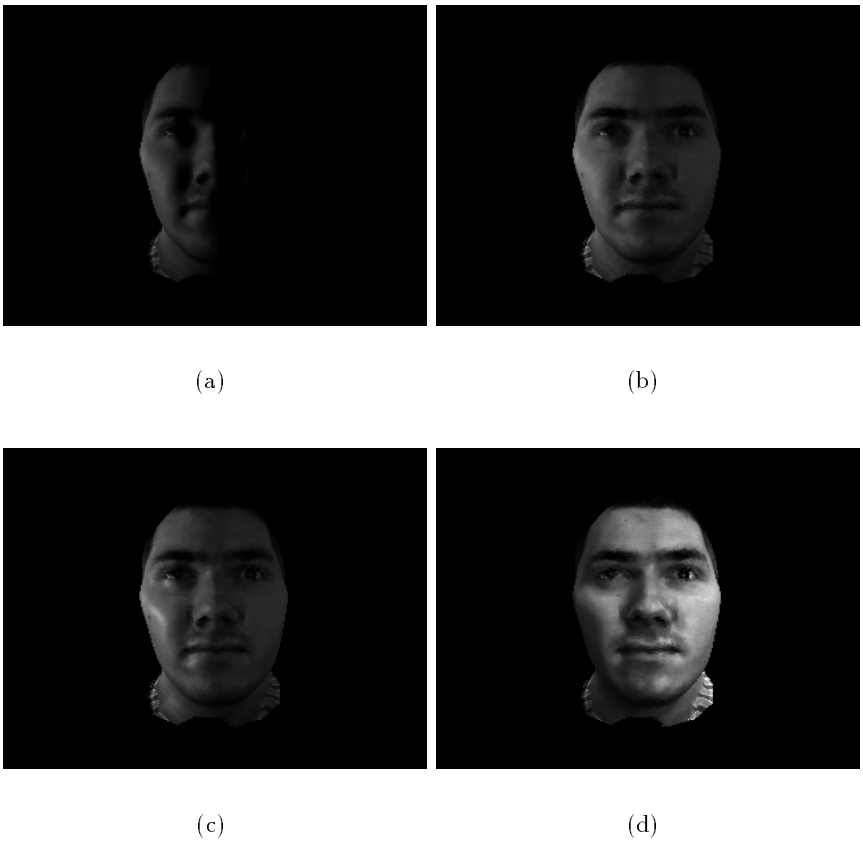
- manually align the synthetic model with the speaker's image;
- extract, from the real speaker's image, pixel luminance values around the facial features of interest. Over–bright pixels are discarded to avoid areas where the camera sensor might have saturated (the luminance of such pixels would not depend linearly on the contributions of the light sources);
- extract, from the synthetic image, the corresponding texture luminance values L_{texture} and object lighting normals $\boldsymbol{n}$;
- the light sources intensities A, D_i, S_i and the offset L_{offset} are finally estimated by solving equation (1) in the least mean square sense.

4.3 Numerical Evaluation on Synthetic Images

To validate the assumption that unknown light sources can be compensated by a set of lights at predefined positions, we conducted experiments on synthetic images in order to avoid problems of misalignment between a face model and an unknown image. To that extent, four images of the same model were created respectively with a left diffuse illumination (3(a)), an ambient and left diffuse lighting (3(b)), ambient, left diffuse and specular components (3(c)), and ambient, diffuse and specular illuminations from two different light sources (3(d)).

With these images, we performed two different experiments, A and B:

(A) we compensated the face illumination with lights located *at the same positions* as the sources used to synthesize the images, enabling and disabling the luminance degree of freedom of equation (1). Such experiments can point out numerical differences between the lighting model of equation (1), implemented by floating point computations in our software, and the OpenGL lighting operations, implemented by dedicated hardware;



(a) (b)

(c) (d)

Figure 3: The synthetic test images used in the validation experiments A and B. The reconstructed images are not displayed since no significant differences are visible — (a) left diffuse illumination; (b) ambient and left diffuse lighting; (c) ambient, left diffuse and specular lighting; (d) ambient, diffuse and specular components from two light sources.

(B) we then tried to measure the quality of the compensation using light sources that are *at completely different locations* from the original ones, to evaluate how well unknown light sources can be simulated by lights at arbitrary positions: we compensated for the face illumination with all the light sources on except the ones used for the image creation (our software has seven predefined lights, namely top, bottom, left, right, and three lights around the camera).

Table 1 presents the mean error and the root of the mean square error around the model facial features. The conclusion is threefold:

- with synthetic lights at the right positions, experiments A prove that numerical errors are marginal (but exist) in the algorithm, and that the equation (1) can be correctly implemented by OpenGL;
- experiments B suggest that it is fairly reasonable to expect to compensate for ambient, diffuse and specular reflections of unknown intensities and unknown

Table 1: Illumination compensation errors (root of mean square error and mean error in brackets) on synthetic images, for experiments A (compensation using the same lighting directions as the original image) and B (compensation using all except the same lighting directions as the original image).

Synthetic Images	Fig. 3(a)	Fig. 3(b)	Fig. 3(c)	Fig. 3(d)
Without compensation	77.66 [74.25]	87.31 [79.92]	86.79 [78.84]	44.27 [36.21]
Exp. A: lights at known positions	0.24 [0.06]	0.15 [−0.02]	0.15 [−0.02]	0.36 [0.00]
Exp. B: lights at unknown positions	0.30 [0.02]	4.12 [−0.20]	3.58 [−0.43]	3.56 [−0.14]

directions by a limited set of lights at predefined locations, without trying to recover the lighting directions.

4.4 Experimental Results on Real Images

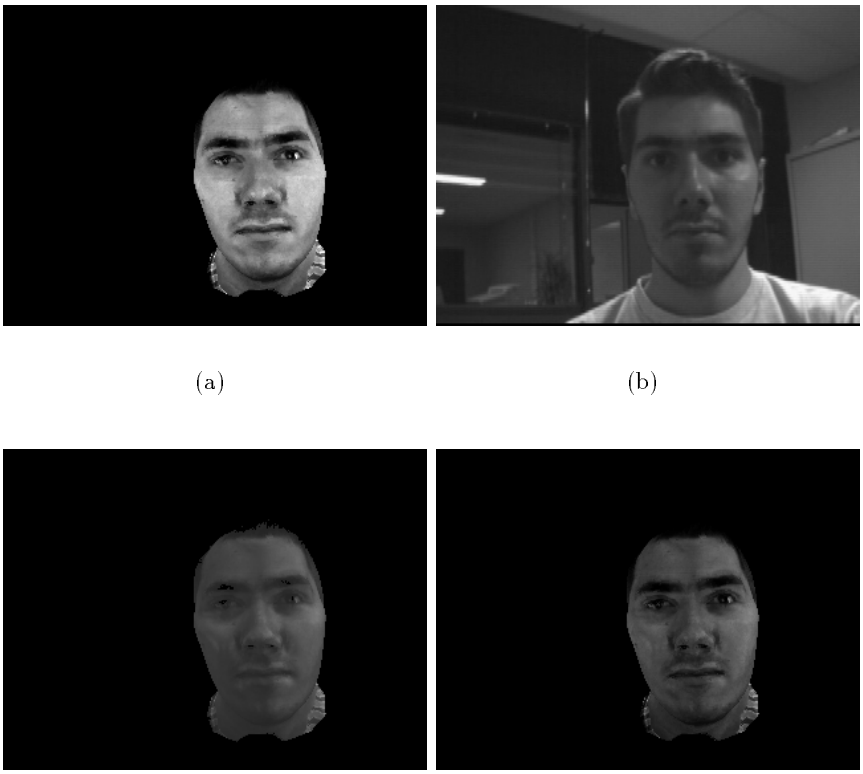
Figure 4 shows the illumination compensation for *a real world case*. It is clear that without the compensation (Figure 4(a)), it would become difficult for a block–matching algorithm to match the synthetic facial features with the real ones (Figure 4(b)), not only because of the low brightness of the real image, but also because of the shading of the face.

Table 2 displays the numerical errors, before and after the compensation. Our algorithm greatly minimizes the gap between the synthetic and real images (4(c) and 4(b)) by dividing the root of the mean square error by 6.7, although the enhancements are not as good as for the experiments on synthetic images only. We believe that this is mainly due to the misalignment between the face and the synthetic model, which cannot be perfectly matched "by hand", to the uncalibrated acquisition camera, which does not realize the same perfect perspective projection as the one implemented in our synthesis module, and finally to the texture map of the face model: this texture map should actually correspond to the user's face viewed in ambient lighting, but the scanning device has built–in bright light sources which create parasitic diffuse and specular reflections on the user's face during the texture acquisition. These unpredictible differences are the reason why the brightness offset L_{offset} was introduced in equation 1.

4.5 Justification of L_{offset}

We removed the term L_{offset} from the lighting model, and reran the compensation algorithm. In Table 3, we see that removing the offset does not bring any significant degradation of the error between synthetic images, which means that equation (1) without L_{offset} corresponds to the lighting operations made by OpenGL to render the synthetic test images.

However, for the compensation on a real image, removing L_{offset} introduces a higher error (see Table 4), which can be seen in Figure 4(d), leading to a face model



(a) (b)

(c) (d)

Figure 4: Illumination compensation on a real face: (a) the speaker's head model with no directional light source; (b) the speaker in a real environment; and the same model with illumination compensation ((c) with and (d) without the illumination offset L_{offset}).

Table 2: Illumination compensation error (root of mean square error and mean error in brackets) between a real and synthetic image.

Real Image	(Fig. 4(b))	
Without compensation	(Fig. 4(a))	62.42 [51.40]
With compensation	(Fig. 4(c))	9.31 [−0.02]

Table 3: Illumination compensation errors (root of mean square error and mean error in brackets) on synthetic images, without using the illumination offset. These results should be compared to Table 1.

Synthetic Images	Fig. 3(a)	Fig. 3(b)	Fig. 3(c)	Fig. 3(d)
No compensation	77.66 [74.25]	87.31 [79.92]	86.79 [78.84]	44.27 [36.21]
Exp. A: known positions, without L_{offset}	0.11 [−0.01]	0.09 [−0.01]	0.09 [−0.01]	0.36 [−0.04]
Exp. B: unknown positions, without L_{offset}	0.39 [0.15]	4.12 [0.04]	3.56 [−0.02]	3.55 [−0.02]

more constrasted than the real face. In this case, the root of the mean square error is only reduced by a factor of 4.1. L_{offset} is therefore an appropriate degree of freedom to make up for the non–ambient texture of the face model, and imprecisions due to the face model's misalignment and the uncalibrated camera.

Table 4: Illumination compensation error (root of mean square error and mean error in brackets) between a real and synthetic image, without the illumination offset. These results should be compared to Table 2.

Real Image	(Fig. 4(b))	
Without compensation	(Fig. 4(a))	62.42 [51.40]
With compensation, no L_{offset}	(Fig. 4(d))	15.12 [−3.98]

4.6 Concluding Remarks About the Illumination Compensation

We do not claim that the proposed algorithm recovers the exact illumination of the scene, but it helps smooth out the photometric discrepancies around the facial features (not the whole face) between the feedback loop and the video input. Figure 4 shows that the distributions of dark and bright areas on the real and synthetic views

are coherent. The main advantage of our algorithm is that the estimated lighting parameters can be directly injected at the $3D$ level within the face synthesis module, which is based on the OpenGL library, taking advantage of hardware accelerations that are more and more common on entry-level graphics boards. Furthermore, the lighting parameters estimation is straightforward and not CPU–demanding, since it does not need to be constrained to output positive light intensities[1].

To set up the light positions in practice, we found a convenient solution by defining an interface where the user can switch on and off various light sources at predefined positions like the ceiling, the camera, on the left and right hand–side... according to his/her environment. As a conclusion, even if it requires some user intervention, we do not think it damages the face tracking system usability, since it has to be done only once at the beginning of the session. Furthermore, it contributes to the realism of the model (with respect to the speaker's real view), and therefore makes the tracking of real facial features from synthetic patterns by differential block–matching possible, as described in the next sections.

5 Kalman Filter

Kalman filters [33] are often used in head tracking systems for two different purposes: the first one is to temporally smooth out the estimated head global parameters, as in [34], the second one is to convert the $2D$ facial features positions observations into $3D$ estimates and predictions of the head position and orientation [12]. In our application, the Kalman filter is the central node of our face tracking system, since it has three different objectives: it recovers the head global position and orientation, it predicts the $2D$ positions of the feature points for the block–matching algorithm, and — this point is new — it makes the synthesized model have the same scale, position and orientation as the speaker's face in the real view, despite the acquisition by an uncalibrated camera.

5.1 Extended Kalman Filtering Theory

Let us first define a few notations. What we want to estimate is the state vector Ψ_t of a system at time t. Ψ_t cannot be measured directly. Instead, we have access to measurements s_t that depend non–linearly on the state vector by the relation $s_t = h(\Psi_t)$, and we have a general idea about the evolution of the system by $\Psi_{t+1} = a(\Psi_t)$. To reflect the uncertainty of the observations and the dynamic evolution of the system, two Gaussian white noises are introduced, denoted v_t and w_t, of covariance matrices R and Q, and we have the system observation and evolution equations:

$$\left\{ \begin{array}{rcl} s_t & = & h(\Psi_t) + v_t \\ \Psi_{t+1} & = & a(\Psi_t) + w_t \end{array} \right. \tag{2}$$

The measurement and evolution functions are linearized respectively around the

[1] OpenGL can perfectly deal with negative lights.

a priori estimate $\Psi_{t/t-1}$ and the *a posteriori* estimate $\Psi_{t/t}$ with:

$$\begin{array}{rrcl} & h(\Psi_t) & \approx & h(\Psi_{t/t-1}) + H_t(\Psi_t - \Psi_{t/t-1}) \\ \text{and} & a(\Psi_t) & \approx & a(\Psi_{t/t}) + A_t(\Psi_t - \Psi_{t/t}) \end{array} \tag{3}$$

by denoting $H_t = \frac{\partial h}{\partial \Psi}\big|_{\Psi_t=\Psi_{t/t-1}}$ and $A_t = \frac{\partial a}{\partial \Psi}\big|_{\Psi_t=\Psi_{t/t}}$, the Jacobians of the measurement model and of the dynamic model. After some matrix manipulations, the *a posteriori* estimates and their error covariance matrices $P_{t/t}$ are given by the filtering equations:

$$\left\{ \begin{array}{rcl} K_t & = & P_{t/t-1}H_t^T \left(R + H_t P_{t/t-1} H_t^T\right)^{-1} \\ \Psi_{t/t} & = & \Psi_{t/t-1} + K_t(s_t - h(\Psi_{t/t-1})) \\ P_{t/t} & = & (I - K_t H_t) P_{t/t-1} \end{array} \right. \tag{4}$$

and the *a priori* estimates with their error covariance matrices $P_{t+1/t}$ by the prediction equations:

$$\left\{ \begin{array}{rcl} \Psi_{t+1/t} & = & a(\Psi_{t/t}) \\ P_{t+1/t} & = & A_t P_{t/t} A_t^T + Q \end{array} \right. \tag{5}$$

5.2 Interpretation of the Equations

By considering the equations in (4), the filter produces $\Psi_{t/t}$ by rectifying the predicted estimate $\Psi_{t/t-1}$ by the correction term $K_t(s_t - h(\Psi_{t/t-1}))$, based on the difference between the real observations s_t and the predicted ones $h(\Psi_{t/t-1})$. An interesting interpretation of the Kalman filter behavior is that it iterates to make the system state vector $\Psi_{t/t}$ *fit* both the observations s_t and the dynamic evolution model represented by the equations in (5).

This "differential fitting" interpretation helps to understand why, with a carefully chosen measurement model, the Kalman filter is able to align the synthesized images with the speaker's image even though the video camera is *not* calibrated: the idea is to derive the observation function from the synthesis operations, so that $s_t = h(\Psi_t)$ is the vector of the $2D$ positions of the synthetic facial feature points in the image plane. In this case, Ψ_t includes the 6 degrees of freedom of the synthetic face in the synthetic world, and the filter will modify Ψ_t to output the $2D$ positions of the tracked facial features, thus aligning the speaker's face and the synthetic model.

5.2.1 Dynamic Model

The 6 parameters needed to render the synthetic head are

$$\boldsymbol{\mathcal{S}} = (t_X, t_Y, t_Z, \alpha, \beta, \gamma)^T$$

representing the 3 translations and the 3 rotations with respect to the X, Y and Z axes, and they are included with their first– and second–order time–derivatives in the filter state vector $\Psi = (\boldsymbol{\mathcal{S}}^T, \dot{\boldsymbol{\mathcal{S}}}^T, \dddot{\boldsymbol{\mathcal{S}}}^T)^T$ for the dynamic model of the system evolution, simply based on Newtonian physics under a constant acceleration assumption:

$$\boldsymbol{\mathcal{S}}_{t+dt} = \boldsymbol{\mathcal{S}}_t + \dot{\boldsymbol{\mathcal{S}}}_t dt + \frac{1}{2}\dddot{\boldsymbol{\mathcal{S}}}_t dt^2 \tag{6}$$

5.2.2 Measurement Model

Although the viewing camera is not calibrated, the observation model of the Kalman filter mimics a perspective projection of focal length $\mathcal{F}$ (see Figure 5). The actual value of $\mathcal{F}$ is not sensitive for the tracking algorithm, as the Kalman filter will adapt its state vector computation to fit the observations, as if they had been viewed by an ideal camera of this focal length.

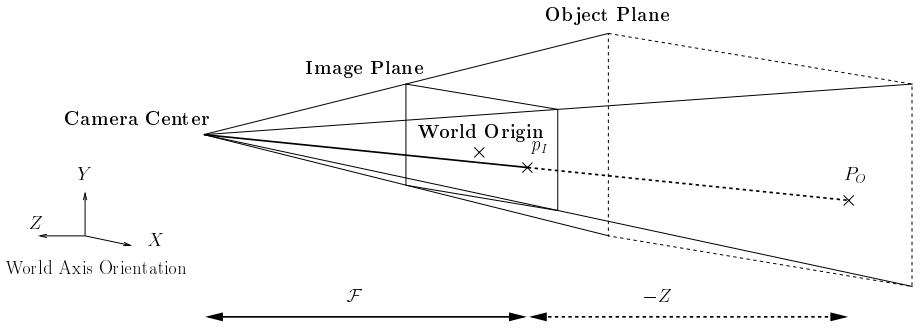


Figure 5: Camera analogy for the view model, with $P_O = (X, Y, Z, 1)^T$, and $p_I = (x_I, y_I)^T = \left(\frac{X\mathcal{F}}{\mathcal{F}-Z}, \frac{Y\mathcal{F}}{\mathcal{F}-Z}\right)^T$ (only the negative part of the Z axis is seen by the camera).

Considering the rigid transformation applied by the OpenGL library on the $3D$ facial feature localization $(x, y, z)^T$ in head mesh coordinates, the final $2D$ measurement $(x_I, y_I)^T$ in the image plane given the parameters $\boldsymbol{\mathcal{S}} = (t_X, t_Y, t_Z, \alpha, \beta, \gamma)^T$ and $(x, y, z)^T$ is

$$\begin{pmatrix} x_I \\ y_I \end{pmatrix} = \begin{pmatrix} \frac{\mathcal{F}}{\mathcal{N}} \left(c_\beta c_\gamma x - c_\beta s_\gamma y + s_\beta z + t_X\right) \\ \frac{\mathcal{F}}{\mathcal{N}} \left(\left(s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma\right) x + \left(-s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma\right) y - s_\alpha c_\beta z + t_Y\right) \end{pmatrix} \quad (7)$$

with $\mathcal{N} = \left(-c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma\right) x + \left(c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma\right) y + c_\alpha c_\beta z + t_Z - \mathcal{F}$ (for clarity reasons, c_α, s_α, c_β, s_β, c_γ and s_γ respectively stand for $\cos(\alpha)$, $\sin(\alpha)$, $\cos(\beta)$, $\sin(\beta)$, $\cos(\gamma)$ and $\sin(\gamma)$)

6 Tracking

6.1 Differential Block–Matching

The main complications encountered by a block–matching (or *pattern correlation*) algorithm are, on the one hand, the local geometric distortions, scales of facial features and changes of lighting due to the speaker's $3D$ motions, and on the other hand, the large rotations out of the image plane: in this case, the facial features might not fit in rectangular blocks, and the reference patterns are matched against portions of other objects in the background. Both issues are addressed by tracking synthetic facial features in the real speaker's view, as long as the block–matching procedure is able to allow some photometric differences between the synthetic and real patterns, and to be aware of the background pixels that are likely to interfere in the correlation score for extreme head poses. Extending the theory presented in [35],

we propose that a classic differential block–matching formulation be adapted to be robust to the photometric model failures on synthesized features in section 6.1.1, and support non–rectangular patterns and extreme orientations in section 6.1.2.

6.1.1 Photometric Robustness

The differential block–matching algorithm is derived by considering that a reference pattern, denoted $\boldsymbol{I}(\boldsymbol{\mu} = \mathbf{0}, \tau = 0) = (p_1, \cdots, p_m)^T$ (all pixels are placed in a column vector), undergoes some perturbations $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_n)^T$ (most often displacements in the image, with sub–pixel precision). Writing a Taylor series expansion for small perturbations between two consecutive frames, we have

$$\boldsymbol{I}(\boldsymbol{\mu}, \tau) \approx \boldsymbol{I}(\mathbf{0}, 0) + \boldsymbol{M}\boldsymbol{\mu} + \boldsymbol{I}_t\tau \tag{8}$$

with $\boldsymbol{M} = \left(\frac{\partial \boldsymbol{I}}{\partial \mu_1}(\mathbf{0}, 0) | \cdots | \frac{\partial \boldsymbol{I}}{\partial \mu_n}(\mathbf{0}, 0)\right)$ and $\boldsymbol{I}_t = \frac{\partial \boldsymbol{I}}{\partial t}(\mathbf{0}, 0)$. Solving for $\boldsymbol{\mu}$ in the least mean square fashion yields

$$\boldsymbol{\mu} = -(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\boldsymbol{I}_t \tag{9}$$

which may be rewritten by identifying the lines of $-(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T$ as

$$\begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_1 \\ \hline \vdots \\ \hline \boldsymbol{r}_n \end{pmatrix} \boldsymbol{I}_t \tag{10}$$

In equation (8), the $\boldsymbol{\mu}$ perturbations are general enough to represent a local pattern rotation or scaling [35], and to integrate a luminance scale and offset perturbation in the case of a photometric model failure for a synthesized feature ($\frac{\partial \boldsymbol{I}}{\partial \text{lum. scale}}(\mathbf{0}, 0) = \boldsymbol{I}(\mathbf{0}, 0)$ and $\frac{\partial \boldsymbol{I}}{\partial \text{lum. offset}}(\mathbf{0}, 0) = (1, \cdots, 1)^T$). The luminance offset degree of freedom defined in the photometric model (equation (1)) is taken into account, and need not be explicitly recovered during the matching procedure, since only the translation parameters $\mu_1 = \boldsymbol{r}_1\boldsymbol{I}_t$ and $\mu_2 = \boldsymbol{r}_2\boldsymbol{I}_t$ are computed from (9) to be given to the Kalman filter: the only overhead implied by additional degrees of freedom takes place during the computation of $-(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T$, but they do not mean more computations during the iterative estimation of μ_1 and μ_2.

6.1.2 Background Robustness

Used together, the Kalman filter and the synthetic patterns can predict the background position near the participant's facial features due to the $3D$ pose: in Figure 6, the synthesized model is rendered on a black background, which appears in the extracted rectangular nose pattern for this particular head orientation. The pattern pixels are classified into 2 subsets, $\boldsymbol{I}|_F$ and $\boldsymbol{I}|_B$, whether they belong to the face or to the background area[2]. We restrict the computation of the temporal gradient $\boldsymbol{I}_t$

[2]In practice, the background is drawn in black at the encoder level within the feedback loop, to be easily segmented from the face regions.

in equation (10) to $\boldsymbol{I}|_F$, so that the potential background objects in the real view have no impact on the correlation.

Hence, the synthetic patterns are turned into visual clues to match only the potential feature areas in the real speaker's image, and our tracking algorithm finds the correct best match despite other objects in the features neighborhood (see also Figure 11 for extreme head rotations).

Figure 6: For this typical rotation, the synthetic background used by the encoder appears in the tracked nose area. As its pixels would lead to a wrong match in the real image, they are discarded by the tracker for the computation of the temporal gradient $\boldsymbol{I}_t$. As a result, the tracker restricts itself only to the potential face pixels in the video frame, and is able to find the right match despite the unknown background in the real world.

Restricting $\boldsymbol{I}_t$ in equation (10) to the $\boldsymbol{I}|_F$ pattern subset, we get our modified block–matching algorithm giving the $2D$ displacement $(d_x, d_y)^T$ of each feature area:

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_x \\ \hline \boldsymbol{r}_y \end{pmatrix} \begin{pmatrix} g_1 \\ \vdots \\ g_m \end{pmatrix} \text{ with } \begin{cases} g_i = p_i^{\text{real}} - p_i^{\text{synthetic}} & \text{if } p_i^{\text{synthetic}} \in \boldsymbol{I}|_F \\ g_i = 0 & \text{if } p_i^{\text{synthetic}} \in \boldsymbol{I}|_B \end{cases} \quad (11)$$

7 Numerical Evaluation of the Global Motion Tracking

To assess the accuracy of the estimation of the global motion, we ran our algorithm on a synthetic video sequence, where the parameters to be recovered are known, using 12 tracked areas surrounding the main facial features, shown in Figure 7. The face in this sequence oscillates around its initial position, undergoing respectively 3 translations along the X, Y and Z axes, 3 rotations around the same axes, and a combination of all degrees of freedom. This sequence is can be viewed on the web [36].

This sequence lasts 20 seconds, at 30 frames per second, for a total amount of 600 frames, and its resolution is 320×242 pixels. To make this evaluation more significant, a background image has been inserted behind the face model to mimick the real world case shown by Figure 8. The images were rendered using an ideal perspective camera, with a focal length of 2. The global motion parameters used to synthesize the face model were obtained using a polynomial interpolator of third

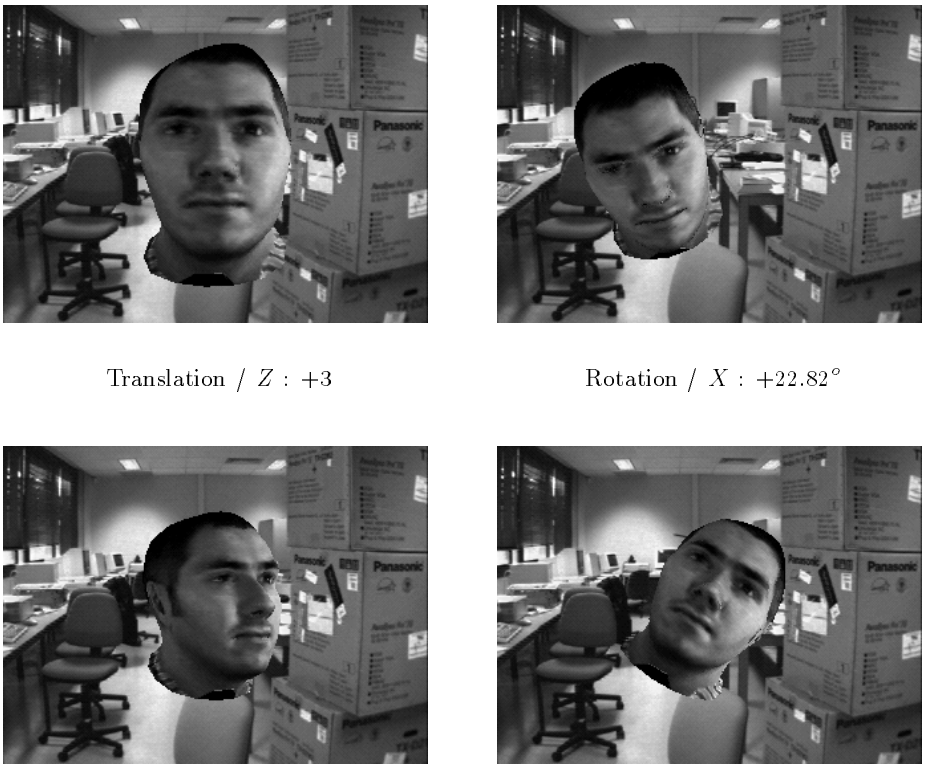
(a) (b)

Figure 7: The 12 facial features used to evaluate the tracking algorithm: (a) feature positions; (b) sizes of the tracked areas — Each eye represents 2 facial features, once with the eyebrows, and once without them.

order between 23 keyframes, without enforcing the constant acceleration assumption of the Kalman filter. The maximum rotations are depicted in Figure 9, and are large enough to verify the behavior of the tracking algorithm when the background is likely to interfere with the face patterns. Rotations are expressed in degrees, and translations in the face model coordinates: as the face vertices are scaled in the range $[-1, 1]$, a translation of $+3$ means a translation of about 1.5 times the face size.

Figure 8: The real world sequence, which inspired the synthetic sequence of Figure 9 made to assess the tracking accuracy.

We plotted the recovered parameters against the true ones for the 600 frames in Figure 10. It can be seen that the algorithm successfully recovers the rigid motion of the face in the video sequence, but not surprisingly, the most difficult parameters to be recovered are the rotations around the X and Y axes, since they lead to non–affine diplacements in the $2D$ image plane, and are solved by the linearization of the observation model computed by the Kalman filter at each frame. One may also notice that the recovered parameters are noisy, especially when the true parameters do not change. This is due to the Kalman filter, which does not sufficiently smooth out the *a posteriori* estimations. We could have obtained smoother results if we had incremented the noise level of the observations in the filter. However, there is always



Translation / Z : $+3$ | Rotation / X : $+22.82^o$

Rotation / Y : $+35^o$ | Rotation / Z : $+33^o$

Figure 9: Maximum amplitude of the parameters used to evaluate the tracking algorithm, with respect to the initial position. The X, Y and Z axes of rotation are local to the head model (i.e. they are not related to the camera).

a compromise to be made between the smoothing capability of a Kalman filter (by decreasing the uncertainty of the predictions), and its reactivity to new situations (by decreasing the uncertainty of the incoming observations): for instance, with the noise settings used in this evaluation, it can be seen on the plots that the filter already tends to overestimate the parameters after a rapid transition.

Table 5 contains some statistics about the errors, and the signal to noise ratios (SNR) of the estimated parameters. They show that the global motion parameters are recovered quite well. The mean error is about 0.5% of the face scale for the translation parameters, and about 1 or 2 degrees for the rotation parameters, which makes the results quite fair. The largest errors are obtained during or after rapid transitions, indicating that the prediction model of the Kalman filter, assuming a constant acceleration, may not be the most appropriate.

8 Conclusions About the Face Tracking Algorithm

In the previous sections, we proposed a face tracking framework which has the possibility to feed the feature–tracking procedure with synthesized facial patterns,

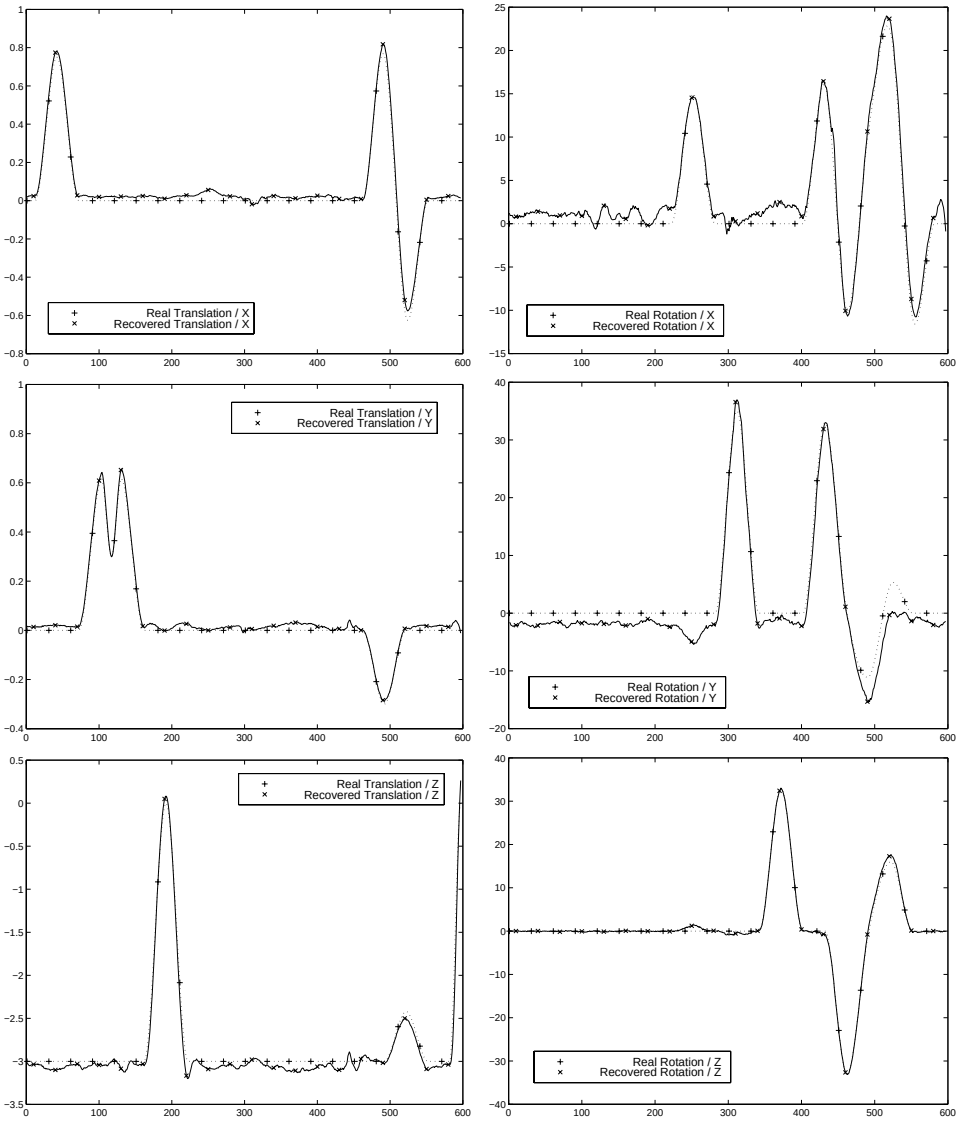


Figure 10: Real and recovered XYZ positions and orientations for the 600 frames.

implementing a *visual feedback loop*, thus solving the problems of lighting, scale and geometric deformations of the face implied by its rigid motion, with no explicit manipulation of the $3D$ geometry of the face model, as in other optical flow–based techniques. Our face cloning module is designed to work in real–world lighting, with few assumptions about the speaker's motion (allowing large rotations out of the image plane), and without marks or makeup on the user's face (see Figure 11). To that purpose, we presented geometric and photometric modelling techniques to make the synthesized patterns closer to their real counterparts. We also reformulated a block–matching algorithm to make it work with synthetic input data, and showed how to handle the presence of the speaker's background in extreme positions. It is important to note that such an analysis/synthesis cooperation is successful because of the realism of our modelling techniques, and the design of the Kalman filter to drive the synthetic images. Our system is independent of any facial expression estimation

Table 5: Statistics about the tracking accuracy on a synthetic test sequence.

	Max. Abs. Error	Mean Error	Variance	$10 \log_{10}$ SNR
Translation				
/ X	0.0681 (frame 510)	0.0231	2.0372×10^{-4}	18.49
/ Y	0.0421 (frame 444)	0.0147	8.1550×10^{-5}	19.70
/ Z	0.4235 (frame 589)	-0.0567	3.9956×10^{-3}	30.51
Rotation				
/ X	3.4551 (frame 445)	0.9130	0.5732	15.43
/ Y	5.4403 (frame 531)	-1.9609	1.5645	12.44
/ Z	1.7035 (frame 529)	0.0606	0.2428	25.87

algorithm, and can be used to generate a global animation stream for MPEG–4 encoders with life–like timings, leaving the task of facial expression animation to other methods.

The result of the tracking algorithm on a real face can be seen in a video sequence available on the WWW [23]. The original analyzed sequence lasted 30 seconds, and was captured in a 320×242 resolution at 10 frames per second. The analysis/synthesis speed mainly depends on the workstation graphics hardware acceleration and its video acquisition speed. On a O^2 SGI workstation, the analysis frame rate using 12 facial feature areas is:

- 1 image per second, when synthesizing patterns, computing the product $-(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T$, and updating the Kalman filter for every frame;
- 10 frames per second, when disabling synthetic pattern calculation for every frame, but still enabling the Kalman filter — in this case, large face rotations might cause the system to lose the user's head;
- full frame rate, when disabling both pattern synthesis and the Kalman filter — the system just tracks the facial features in $2D$, without recovering the head $3D$ pose, and becomes very sensitive to rotations.

One of the questions that might be raised about the robustness of the tracking is what happens when the user closes his eyes, or has a facial expression that differs from the static expression of any feature area. In such a situation, some of the individual trackers may lose their facial feature, but there are enough tracked features to allow the outliers to be minimized by the Kalman filter. The prediction of the positions of the features in the next frame brings the trackers back in the right area until they produce correct matches. It is clear that the system will be even more robust if it can take into account the user's facial expressions, be they obtained by image analysis, or other means (like processings on the audio signal to infer the animation of the mouth).

Figure 11: Head rotations supported by our face tracking system, without markers and uniform lighting.

Perspectives: Synthesis of Facial Expressions

Now that we have a robust algorithm to track a human face and estimate its pose, our current work focuses on the synthesis of facial expressions (local animations), and their analysis by image processing only, still without any markers.

The face models constructed in section 3 are unanimated. To efficiently generate facial expressions, we implemented several original or well–known animation techniques, operating on the $3D$ vertices, the texture coordinates attaching the vertices to the texture image, or the texture image itself:

mesh deformations: by applying mesh morphing, it is possible to deform the face shape, to close the eyelids or squeeze the mouth of the face model (see Figure 12(b));

displacements of texture coordinates: extending the concept of mesh morphing to texture coordinates morphing, we can make the texture slide over the face model, without deforming the shape of the $3D$ wireframe. This technique is used to animate the eyebrows (Figure 12(c));

texture sliding: instead of adding new primitives for each eye–ball, we created holes in the texture image (via the transparency channel), and two separate

textures behind the main one can be displaced to alter the model's gaze direction (Figure 12(d));

texture blending: beside moving some texture portions, it is possible to blend several textures together to produce a new one, for example to fade expression wrinkles into the model texture at low–cost in terms of real–time animation, instead of hard–coding them in heavy spline–based meshes (Figure 12(e)).

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

Figure 12: Examples of facial animations: (a) neutral (initial) face model; (b) mesh deformations; (c) displacements of texture coordinates; (d) texture sliding; (e) texture blending.

We are currently porting our displacements of texture coordinates and texture animations into an existing MPEG 4 viewer [37], to benefit from the streaming and networking capabilities of the standard (defined by the Digital Media Integration Framework, DMIF) while respecting the FAP syntax [1]. All these techniques, of course, can be combined together to produce more complex facial expressions (see the video sequence [38]).

Analysis of Facial Expressions

We are also investigating an original view–based approach to relate the analysis and the synthesis of facial expressions: using our realistic face model, we sample the visual space of facial expressions across various poses, via a set of $\boldsymbol{\mu}$ vectors containing facial animation parameters. Image patches for the facial features of interest (like the eyes, eyebrows, and the mouth) are extracted, to produce distinct datasets of training examples. Then, a principal component analysis is performed over those training datasets, to extract a limited number of images optimally spanning the training space. These images (called *eigenfeatures*) allow us to characterize the facial expression of the user's face via a simple correlation mechanism, yielding a compact $\boldsymbol{\lambda}$ vector. And finally, a linear estimator will be designed to map the analysis scores $\boldsymbol{\lambda}$ to the face animation parameters $\boldsymbol{\mu}$. Such a system will not be limited by the amount of available key–frames (as noted in the introduction for traditional view–based techniques), since all degrees of freedom permitted by the synthetic face can be precisely, automatically and systematically exploited by the training strategy.

Figure 13 shows some preliminary results concerning the analysis of synthetic facial images using a linear estimator trained over the responses of the eigenfeatures, which are reported in [39]. We are currently extending this strategy to the analysis of real facial images.

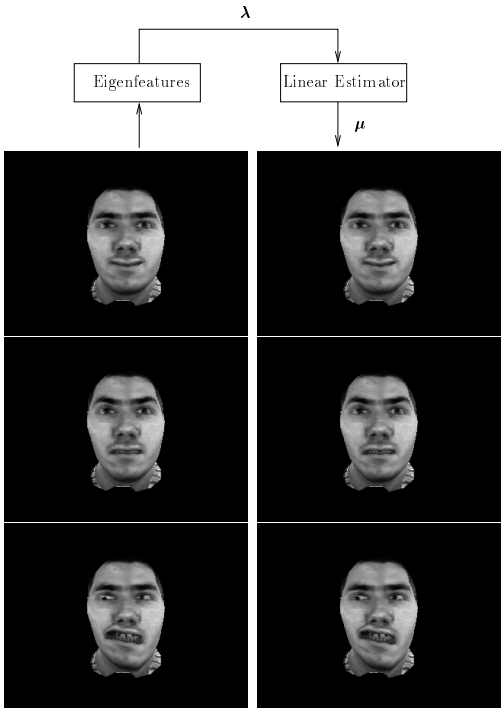


Figure 13: Some analyses of facial expressions: each image of the left column was quantified by some eigenfeatures, giving a $\boldsymbol{\lambda}$ vector. A linear estimator mapped $\boldsymbol{\lambda}$ to the animation parameters $\boldsymbol{\mu}$, which were rendered into the images of the right column.

Acknowledgments

Eurécom's research is partially supported by its industrial members: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

The authors would like to thank Hervé Delingette (INRIA Sophia–Antipolis) for his contribution in the model mesh construction; the University of Erlangen, Germany, and the L.U.A.P. (Université Paris–*VII*) for the original Cyberware scans; Renaud Cazoulat (CNET France–Telecom) for the MPEG–4 issue; and Katia Fintzel (Espri Concept/Institut Eurécom) for her special appearance in section 8.

References

[1] Overview of the MPEG–4 standard. Technical Report ISO/IEC JTC1/SC29/WG11 N2725, International Organisation for Standardisation,

Seoul, South Korea, March 1999.

[2] MPEG–4 synthetic/natural hybrid coding. URL http://www.es.com/mpeg4-snhc/ .

[3] S. Valente and J.-L. Dugelay. A multi–site teleconferencing system using VR paradigms. In *Ecmast*, Milano, Italy, 1997.

[4] J. Ohya, Y. Kitamura, F. Kishino, and N. Terashima. Virtual space teleconferencing: Real-time reproduction of tridimensional human images. *Journal of Visual Communication and Image Representation*, 6(1):1–25, March 1995.

[5] VIDAS — VIDeo ASsisted audio coding and representation. URL http://miralabwww.unige.ch/Vidas-TO.html.

[6] Institut National de l'Audiovisuel. Televirtuality project: Cloning and real-time animation system. URL http://www.ina.fr/INA/Recherche/TV.

[7] T. Kanade, P. J. Narayanan, and P. W. Rander. Virtual*ized* reality: Concepts and early results. In *IEEE Workshop on Representation of Visual Scenes*, Cambridge, Massachusetts, June 1995. In conjonction with ICCV'95.

[8] J.-L. Dugelay, K. Fintzel, and S. Valente. Synthetic natural hybrid video processings for virtual teleconferencing systems. In *Picture Coding Symposium*, Portland, Oregon, April, 21-23 1999.

[9] K. Fintzel and J.-L. Dugelay. Visual spatialization of a meeting room from 2D uncalibrated views. In H. Niemann, H.-P. Seidel, and B. Girod, editors, *Image and Multidimensional Digital Signal Processing'98*, pages 323–326, Alpbach, Austria, 1998.

[10] A. Eleftheriadis and A. Jacquin. Automatic face location detection and tracking for model–assisted coding of video teleconferencing sequences at very low bit–rates. *Signal Processing: Image Communication*, 47:231–248, June 1995.

[11] *Proceedings of the Second International Conference on Audio– and Video–based Biometric Person Authentification (AVBPA'99)*, Washington, D.C., March 22-23 1999.

[12] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):602–605, June 1993.

[13] S. Basu, I. Essa, and A. Pentland. Motion regularization for model–based head tracking. Technical Report 362, MIT Media Lab., 1996.

[14] D. DeCarlo and D. Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 231–238, San Francisco, California, June 18-20 1996.

[15] R. Koch. Dynamic 3–D scene analysis through synthesis feedback control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):556–568, June 1993.

[16] H. Li, P. Roivainen, and R. Forchheimer. 3–D motion estimation in model–based facial image coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):545–555, June 1993.

[17] P. Eisert and B. Girod. Analysing facial expressions for virtual conferencing. *IEEE Computer Graphics and Applications*, pages 70–78, September 1998.

[18] P. Eisert and B. Girod. Model–based 3D–motion estimation with illumination compensation. In *6^{th} International Conference on Image Processing and its Applications (IPA 97)*, pages 194–198, Dublin, Ireland, July 1997.

[19] A. Schödl, A. Haro, and I. A. Essa. Head tracking using a textured polygonal model. In *Workshop on Perceptual User Interfaces*, San Francisco, California, November, 4-6 1998.

[20] T. S. Jebara and A. Pentland. Parametrized structure from motion for 3D adaptive feedback tracking of faces. In *IEEE Conference on Computer Vision and Pattern Recognition*, November 1996.

[21] P. M. Antoszczyszyn, J. M. Hannah, and P. M. Grant. Accurate automatic frame fitting for semantic–based moving image coding using a facial code–book. In *IEEE International Conference on Image Processing*, Lausanne, Switzerland, September 1996.

[22] P. M. Antoszczyszyn, J. M. Hannah, and P. M. Grant. Automatic frame fitting for semantic–based moving image coding using a facial code–book. In *Eusipco–96*, Trieste, Italy, September 1996.

[23] MPEG demo of the face tracking system. URL http://www.eurecom.fr/∼image/TRAIVI/valente-8points.mpg . (1,782,100 bytes).

[24] CYBERWARE Home Page. URL http://www.cyberware.com.

[25] M. Proesmans and L. Van Gool. One–shot 3D–shape and texture acquisition of facial data. In *Audio– and Video–based Biometric Person Authentification*, pages 411–418, Crans–Montana, Switzerland, March 1997.

[26] H. Delingette. General object reconstruction based on simplex meshes. Technical Report 3111, INRIA, February 1997.

[27] VRML. URL http://vrml.sgi.com.

[28] L. Tang and T. S. Huang. Automatic construction of 3D human face models based on 2D images. In *IEEE International Conference on Image Processing*, Lausanne, Switzerland, September 1996.

[29] M.J.T. Reinders, P.L.J. van Beek, B. Sankur, and J.C.A. van der Lubbe. Facial feature localization and adaptation of a generic face model for model–based coding. *Signal Processing: Image Communication*, 7:57–74, 1995.

[30] G. Bozdağı, M. Tekalp, and L. Onural. 3–D motion estimation and wireframe adaptation including photometric effects for model–based coding of facial image sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 246–256, June 1994.

[31] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions? In *IEEE Conference on Computer Vision and Pattern Recognition*, November 1996.

[32] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with light. In *SIGGRAPH 93*, pages 143–146, Anaheim, California, August 1-6 1993.

[33] S. M. Kay. *Fundamentals of Statistical Signal Processing Estimation Theory*, chapter 13, pages 419–477. PTR Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[34] A. Saulnier, M.-L. Viaud, and D. Geldreich. Real–time facial analysis and synthesis chain. In *International Workshop on Automatic Face— and Gesture— Recognition*, pages 86–91, Zurich, Switzerland, 1995.

[35] G. Hager and P. Belhumeur. Real–time tracking of image regions with changes in geometry and illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*, November 1996.

[36] MPEG sequence used to evaluate the tracking algorithm. URL http://www.eurecom.fr/∼image/TRAIVI/accuracy-test.mpg . (2,906,348 bytes).

[37] J. Signes and J. Close. Streaming VRML and mixed media with MPEG–4. In *SIGGRAPH 98*, Orlando, Florida, July 19-24 1998.

[38] MPEG demo of the animation system. URL http://www.eurecom.fr/∼image/TRAIVI/animation.mpg .

[39] S. Valente and J.-L. Dugelay. Face tracking and realistic animations for telecommunicant clones. In *IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, June 7-11 1999.