Real Time Head and Facial Features Tracking from Uncalibrated Monocular Views

Andrea Bottino
Dipartimento di Automatica ed Informatica, Politecnico di Torino
E-mail: bottino@polito.it

Abstract

In this paper we present a system to track head position and facial features in real time. First the head position is reconstructed by means of a model based head tracker, which matches a 3D generic textured head model and 2D image features extracted from the input sequence. Head tracking is achieved by minimizing an error function which describes the discrepancies between model and image features. Motion and texture information are considered in order to make tracking robust. After pose reconstruction, the input image is warped into the texture map of the model. The resulting stabilized view of the face is then processed to reconstruct significant face features, like eyes, brows and lips. The proposed method uses multi-state deformable templates which are fitted to the warped image exploiting motion, shape and color information. The final output of the system is a stream of head and face animation parameters. The overall performance of the head tracking algorithm is about 30 frame/s on a Pentium III 600, while face expression reconstruction process runs at about 45 frame/s. Data about the reconstruction accuracy of head position are also presented.

1. Introduction

Head and facial features tracking is important for several application in computer vision, like 3D animation systems, expression analysis, face identification and surveillance systems. Head motion can be used for recognition of simple gestures, like head shaking or nodding, or for capturing a person's focus of attention, providing a natural cue for human machine interfaces. Tracking face features plays an important role in several areas, like lip-reading, speech recognition systems, expression analysis and automatic face identification. Also for telecommunication and videoconferencing, encoding the head motion and the face expression of the speaker according to known standards, like MPEG4 compliant Facial Animation Primitives (FAPs), allows to produce very low bit rate data streams. Many of these applications calls for non intrusive and robust reconstruction techniques from monocular views.

Regarding the related works, one of the first effective studies about head tracking is [3], which presents an estimation process based on tracking facial features like eye and mouth corners. Similar methods have been presented in [6] and [7]. The most successful approaches are model-based techniques exploiting 2D or 3D models. Examples of 2D approaches can be found in [4] and [5]. However, 3D tracking has several advantages, in terms of precision of the reconstruction and of adaptation of the model to the rigid motion of the entire head. In [9], the motion of an ellipsoidal model is used to interpret the optical flow of the image sequence. This work is also the basis of [10], where the approach has been modified in order to cope with partial occlusions of the head. In [1] the motion of a textured polygonal model is used to register the rendered image of the model with the video images. The model used is complex and therefore it needs to be customized according to the face being tracked. Also [2] uses a textured head model. Each input image is projected onto the texture map of the model and the motion is reconstructed by image registration in the texture space. A different approach is presented in [8], where head pose is determined by means of a Kalman filter, which exploits the coordinates in the image plane of facial features like nostrils and eyes. The drawback is, again, the complexity of the model used which is obtained with a 3D scan of the user's head.

The approaches to facial expression reconstruction differ for the representation of the face. Usually the facial expression is described in terms of motion and shape of salient features, like brows, eye and lips. The work [14] first introduced the use of deformable templates. These templates are specified by a set of parameters, which define the feature's shape, and are fitted to an image sequence by minimizing a suitable energy function. The deformable templates have been extended to multi-state deformable models in [16]. Both permanent and transient features can have different templates according to the feature state. This allows describing more accurately the set of shapes the facial features can assume. A different approach has been followed in [19], where feature models are constructed with a network of springs. Tracking is achieved minimizing an energy function. Comprehensive references on many other techniques can be found in the recent survey [18].

The approach proposed in this paper is a two-stage process, which separates head tracking from reconstructing facial features, as outlined in Fig. 1. First, a 3D-textured polygonal model is fitted to an image sequence acquired using a single and non-calibrated camera. The model texture is initially reconstructed directly from the input image as the 2D texture which matches the rendered model and the video image. From this phase the head animation parameters, that is the sequence of translations and rotations of the head, are obtained.

After pose reconstruction, each input image is projected on the texture map of the model. The image obtained provides a stabilized view of the face, that is an image where the face has always the same position, orientation and size, which is processed to reconstruct significant face features. Feature detection and tracking uses a set of multi-state deformable templates which are fitted to the warped image. The shape parameters of the templates are then output as face animation parameters.

The outlined system has been designed in order to meet the following requirements:

1. it must be non-intrusive, that is no external devices must be worn by the subject
2. run-time reconstruction rates must be achieved (that is 25/30 frame/s)
3. images should be acquired with common PC cameras and interface cards; this reduces the dimension of the available images to 320·240 pixels arrays

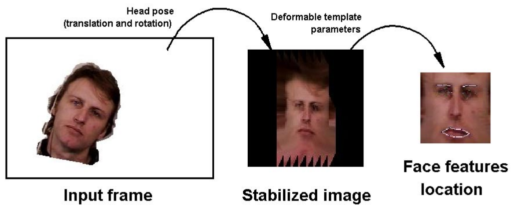

Fig. 1: reconstruction scheme

The remainder of the paper is organized as follows. In paragraph 2 the head tracking technique is introduced. Paragraph 3 details the multi-state models and the algorithm used for reconstructing facial expressions. Some results are presented in paragraph 4. Finally in paragraph 5 we report concluding remarks and outline the future developments of this work.

2. Head pose reconstruction

In the following sections we will detail the various components of the head pose reconstruction system.

2.1. The model

Choosing the right model is a critical problem for the tracking process. A simple model could be not adequate to track the head with precision; on the other hand, a complex model requires a precise initialization per user, a good initial fit and is computationally more expensive.

Our system uses a 3D ellipsoidal model defined by a polygonal mesh (see Fig. 2). This model is not able to reproduce head features, like nose, mouth or accurate face profiles, but allows fast computation and can be easily calibrated according to the user. However, any other 3D polygonal model can be used, regardless its complexity.

The dimensions of the three major axes of the ellipsoid are determined during the initialization process. Details of the startup procedure are given at the end of this paragraph.

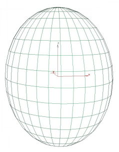

Fig. 2: 3D head model

2.2. Model motion

Let us consider the head as a rigid object with six degrees of freedom: three translations (t_x,t_y,t_z), and three rotations (r_x,r_y,r_z). Each pose is determined by a vector $\wp$ containing six values (where $\wp$ is $[t_x\ t_y\ t_z\ r_x\ r_y\ r_z]$). Also, let $\Gamma(\wp)$ be the matrix which projects the vertices of the model, written in homogenous coordinates, onto the image plane. Let $\mathbf{S}_0$ be the set of vertices $\mathbf{p}_\mathrm{i}$ of the model in their reference position. The projection (x_i,y_i) of each point $\mathbf{p}_\mathrm{i}$ on the image plane for pose $\wp$ is therefore $\Gamma(\wp)\cdot\mathbf{p}_\mathrm{i}$. The matrix Γ is given by:

$$\Gamma(\wp)=\mathbf{P}(f)\cdot\mathbf{T}(\wp)\cdot\mathbf{R}(\wp)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&f^{-1}&0\end{pmatrix}\cdot\mathbf{T}(\wp)\cdot\mathbf{R}(\wp)$$

where $\mathbf{R}$ is the rotation matrix and $\mathbf{T}$ the translation matrix. The values of the projection matrix $\mathbf{P}$ are functions of the focal length f, which is unknown, since the camera is non-calibrated. However, as demonstrated also by [1] and [2], using a rough estimate of its value does not influence substantially the final results.

Finally, let $\mathbf{N}_0$ be the set of normal vectors $\mathbf{n}_\mathrm{i}$ associated to the vertices of the model. The current normal vectors can be evaluated as $\mathbf{R}_{3x3}(\wp)\cdot\mathbf{n}_\mathrm{i}$, where $\mathbf{R}_{3x3}$ is the 3x3 sub-matrix of $\mathbf{R}$ containing the pure rotational values. When projecting the model in the image plane we consider only the points that are currently visible. Given the viewing direction v_d, a point is visible if:

$v_d\cdot(\mathbf{R}_{3x3}(\wp)\cdot\mathbf{n}_\mathrm{i})\le 0$

For simplicity, we approximate the real viewing directions all over the face with a constant vector. An example of the final result of the transformation applied to the model can be seen in Fig. 3.

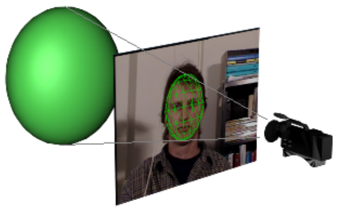

Fig. 3: projection of the model on the image plane

2.3. Pose reconstruction

The reconstruction problem involves finding for each frame n the vector $\wp_n$ which minimizes the differences between model and image features. These discrepancies are described by an error function E that comprises motion and texture errors.

The motion error function, E_{of}, summarizes the differences between the optical flow evaluated from two consecutive images and the motion of the model. The optical flow at each point of the image is the vector $[u,v]$ that describes the translation of the pixel from the previous image. We can also define the "model flow" as the translation on the image plane of the model vertices, that is the difference between the positions of the projected points between pose $\wp_n$ and candidate pose $\wp_{n+1}$. Since not all the points are visible for both poses, the flow can be computed only for the subset of common visible points. Let $\mathbf{V}_\mathrm{n}$ and $\mathbf{V}_{\mathrm{n+1}}$ be the two subsets, and $[u_{M,i},v_{M,i}]$ the i-th point estimated displacement vector.

The motion error function is hence defined as the norm of the difference between the estimated model flow, $\mathbf{V}_{n+1}$-$\mathbf{V}_n$, and the optical flow at the k common locations:

$$E_{of} = \frac{1}{k}\sum_{i=1}^{k}\left\| [u(x_i, y_i), v(x_i, y_i)] - [u_{M,i}, v_{M,i}] \right\|$$

In order to deal with fast motion between consecutive frames, optical flow is evaluated using a pyramidal version with reduced resolution of the Lucas-Kanade algorithm (see [15] for further details).

The texture error function, E_{txt}, describes the differences between the pixel values of the current frame and the texture map values associated to the projected model points. As in the previous case, the computation will be applied only to the two subsets of visible points $\mathbf{V}_n$ and $\mathbf{V}_{n+1}$. Each vertex of the textured model has an associated value $M(p_i)$ in the texture map. Let I be an image of the sequence; the texture error is then defined as the norm of the difference between the model texture and the current frame:

$$E_{txt} = \frac{1}{k}\sum_{i=1}^{k}\left\| M_n(p_i) - I_{n+1}(\mathbf{\Gamma}(\wp_{n+1}) \cdot p_i) \right\|$$

where L_2 norm is used for achromatic images and square distance in RGB space for color images.

To combine motion and texture information, the target error function is a weighted sum of the corresponding error functions. Convenient values of the weights have been found experimentally and their purpose is to equalize the contribution of the different sources of information. The error function E can thus be written as:

$$E = w_{of} \cdot E_{of} + w_{txt} \cdot E_{txt}$$

Given the error function E, we have to find the pose which minimizes the discrepancies between model and image features. The minimization of the function is obtained applying a steepest-descent based method. The method is iterative and, for each iteration, the function values corresponding to translations of $\pm\delta_t$ and rotations of $\pm\delta_r$ for x, y and z are evaluated. The transformation giving the best improvement of the error value is selected for the next iteration. When no improvement can be obtained, the values of δ_t and δ_r are reduced by a factor two and the process is iterated. The algorithm is stopped when the deltas are lower than a predefined threshold or a maximum number of iterations have been performed.

In order to cope with large head motion, we apply a head motion estimation procedure on each frame of the sequence before the error function minimization begins. The motion estimation works as follows. A 2D translation vector, given by the mean value of the optical flow of the visible model points, is evaluated in the image plane. This translation vector is projected on the plane passing through the center of the model and parallel to the image plane (see Fig. 4). The 3D vector obtained is used as initial translation of the model.

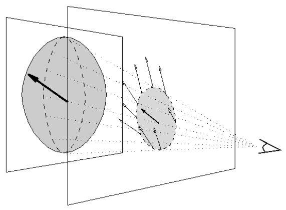
Fig. 4: head motion estimation

2.4. Image warping

When the optimal pose has been found, the content of the current image is warped into the texture map of the model. The warping function is the inverse of the texture mapping function. The warped image produces a stabilized view of the face, which will be used for reconstructing facial features. Some results of the tracking process can be seen in Fig. 5, where the input image, the reconstructed posture and the dynamic texture are shown for several frames.

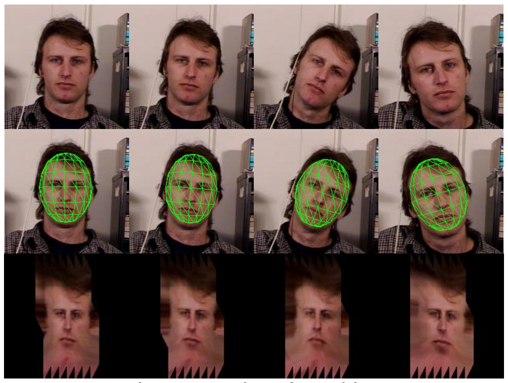
Fig. 5: results of tracking

2.5. Initialization

The outlined process needs to know the initial position and orientation of the model. So far, this step requires user intervention to align the face model to the head in the video images and to modify the size of the model. To avoid interactive procedures, automatic initialization techniques, as the ones suggested in [9], [13], [12] and [17], will be carefully verified in the development of this work.

3. Face features detection and tracking

Our approach to feature detection uses multi-state deformable templates for eyebrows, eyes and lips. Feature tracking involves several sources of information, like shape, texture and motion. In order to achieve real time reconstruction rates, template parameters are derived directly from the information extracted from the images, with no optimization or iterative procedures required to refine initial estimates. So far, we assume that the initial locations of the templates are given in the first frame. However, several works, such as [17] and [20], have dealt with automatic facial feature identification. Details about detection and tracking of the different templates are given in the following sections.

3.1. Brows tracking

A template with three feature points and two connecting segment models each brow (Fig. 6). The feature points are the inner and outer corner, (x_{ic},y_{ic}) and (x_{oc},y_{oc}), and the middle point of the brow, (x_m,y_m).

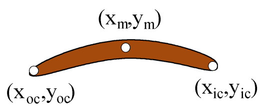

Fig. 6: brow model

The parameters of the template are obtained first tracking the feature points and then applying the template constraints. The inner corner is tracked precisely, while the middle point can shift along the brow and the outer corner is not always stable, due to warping distortions. Thus, the lengths of the segments connecting the inner corner, the middle point and the outer corner are constrained to lie in a range which is ±10% of their initial value. Assuming the tracked position of the inner corner is correct, the middle point and the outer corner are adjusted consequentially. Some results of brows tracking are shown in Fig. 7.

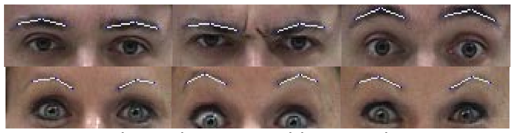
Fig. 7: brows tracking results

3.2. Lip tracking

To define the lip shape, we use a model similar to [16]. Two parabolic arcs which describe the outer lip contours define the lip template. The template parameters are six: (x_c,y_c), the template centre, w, its width, h_{up} and h_{down}, the upper and lower parabolic arc height, and θ, the lip orientation.

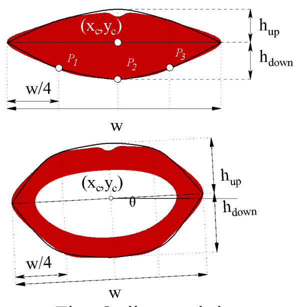

Fig. 8: lip model

Eight feature points are tracked to reconstruct the lip shape: the two lip corners and, for each parabolic arc, three other points, whose abscissas are at a distance *w/4*, *w/2*, and *3w/4* from the left lip corner (that is, points P_1, P_2, and P_3 in Fig. 8). The lip corners are usually tracked precisely, and they allow to evaluate (x_c,y_c), w and θ. The lip contours are more sensible to noise and tend to shift in wrong position during tracking. Their shape is then computed by finding the best fitting parabolas through the corresponding tracked points and constrained to pass through the right and the left lip corner. Some results of lip tracking can be seen in Fig. 9.

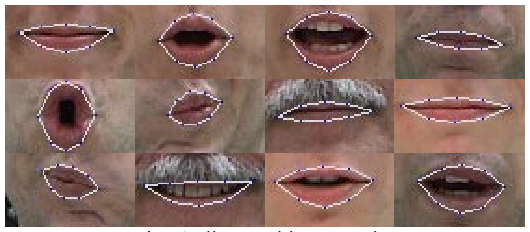
Fig. 9: lip tracking results

The template used is accurate, but can only cope with sequences where both lip contours are visible in every image. Also, since it is intrinsically symmetric, is not able to reconstruct accurately asymmetrical shapes. A more complex mouth template (as in [14]) which is also able to model different mouth states (as in [16]) might overcome these problems.

3.3. Eyes tracking

A multi-state template, like the one presented in [16], describes each eye. The two eye states are “open” and “closed”. The open eye template consists of two parabolic arcs describing the eyelids and a circle corresponding to the iris. The parameters of the template are nine: (x_e,y_e), the eye centre, w, its width, h_{up} and h_{down}, the upper and lower eyelid arc height, θ, the eye orientation, (x_i,y_i), the iris centre and r, the iris ray. For closed eye state, the template is described by the position of the inner and outer corners, P_{ic} and P_{oc}, and by the line connecting them (see Fig. 10).

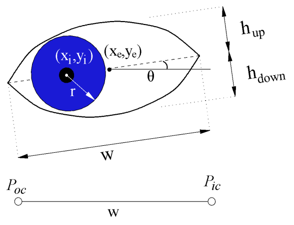

Fig. 10: open and closed eye model

3.3.1. **Iris detection and tracking.** The iris provides important information about the eye state. If the eye is closed, the iris is hidden, while it is usually visible when the eye is open. Therefore, the capability to detect the iris in the eye region can be used to classify the eye as open or closed. Usually, the iris is the darkest region of the eye. A labelling criterion based on a reasonable intensity threshold can be applied to classify the pixels of the eye region as iris and not iris pixels. The threshold is different for each eye and it is evaluated in the first frame analysing the distribution of intensities of sclera and iris pixels.

The iris position and the eye state are then reconstructed according to the following steps:

- in the first frame the mean intensity I_0 of the iris and the number n_0 of pixels classified as iris pixels in the iris area, are evaluated; in the following frames, the pixels of the eye region are classified as iris or not iris pixels
- the iris centre (x_i,y_i) is evaluated as the position of the circle containing the highest number of iris pixels n
- given the iris centre, the mean intensity I of the iris region is computed

- the iris is not detected, and the eye state is classified as "closed", if $(I - I_0)/I_0 > T_i$ and $n/n_0 < T_n$, where T_i and T_n are the thresholds of the intensity and iris pixels; in the current implementation $T_i = 0.29$ and $T_n = 0.72$

Some results of iris tracking can be seen in Fig. 11.

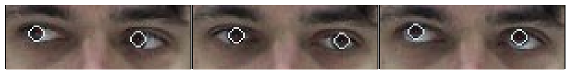
Fig. 11: iris tracking results

3.3.2. **Eye shape tracking.** The eye shape is reconstructed by means of different tracking techniques. The inner corners, the outer corners of the eyes and the middle point of the segment connecting the two inner corners (P_{nose}), are tracked. Usually, the inner corners and P_{nose} are tracked correctly while the outer corners are less stable. In order to recover their correct position, we assume that inner and outer corners of both eyes lies on the same line, which, for us, is the best fitting line through the tracked point. Then, given the eye shape information obtained in the first frame (eye widths and inner corners distance), the eye corners are set along the found line.

If the eye state is "open", eyelids are reconstructed by means of an eyelid detector. A search region between the eye corners is used. For each vertical line of the search window an upper eyelid and a lower eyelid pixel are extracted, on the basis of the following consideration. Usually, the eyelids show a peak in the change of intensity, since a skin pixel is close to a sclera pixel or to an iris pixel. Therefore, for each pixel of the search column, we compute the intensity gradient and the absolute value of the intensity difference of adjacent pixels. By finding common relevant peaks of the two quantities in the upper and lower part of the column, we identify the corresponding eyelid pixels (see Fig. 12). Spurious identifications are removed by discarding points whose distance from both neighbours is higher than four pixels. The upper and lower parabola parameters are then calculated by finding the best fitting parabolas through the corresponding detected eyelid points and constrained to pass through inner and outer corners. Some results of eyelid detection can be seen in Fig. 13.

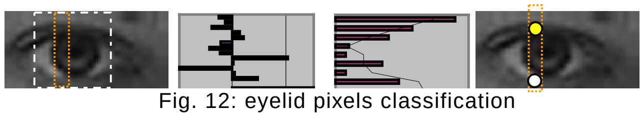
Fig. 12: eyelid pixels classification

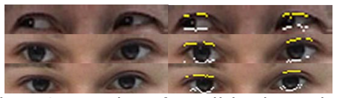
Fig. 13: results of eyelids detection

3.3.3. **Eye tracking results.** Examples of eye tracking can be seen in Fig. 14. The proposed reconstruction algorithm works well for most of the cases. However we experienced some problems with subjects having very bright iris. In this case the mean intensity of the iris and of the sclera are similar, while the "dark region" of the eye is reduced to the pupil. Therefore, the eyelid tracker gives several wrong responses. To build a robust eye tracker, we are planning to develop different labelling schemes based on pixel chromaticity rather then simple intensity.

4. Experimental results

Head tracking algorithm has been tested on the same input sequences used in evaluating the system [2], which are available by courtesy of the Image and Video Computing Group of the Boston University. Ground truth data for position and orientation of the head for each sequence have been acquired using a magnetic sensor. Only the rotational values have been used, since the magnetic marker is hidden in most of the sequences. Thus, we have no way to guess its position with a sufficient precision. Also, no data are available about the accuracy of the specific sequences, which does not allow us to make a comparison between the two systems.

The reconstruction looks very stable for *xyz* translation and for rotation around *z* axis, while the system error increases when reconstructing large head rotations around *x* and *y*. This is due to the fact that translations in the *xy* plane and *x* or *y* rotations produce similar effects in the image plane. Another drawback is that, for large *x* and *y* rotations, the discrepancies between head profile and ellipsoidal model become relevant. Better results might be obtained using synthesized head-like surfaces, as the one used in [11]. We are currently investigating this point.

Some results are plotted in Table 1. The first three columns show the mean angular reconstruction errors in degrees, while the last three columns show the maximal reconstruction error for all the sequences. As can be seen the best average results range between 1.32 and 2.76 degrees. It should be noted, however, that the mean errors on *x* and *y* increase drastically when the sequence analyzed contains large and fast variations of their values (sequences 5 to 8). On the contrary, conspicuous *z* rotations are reconstructed with good precision (sequences 1 and 9). This observation is also underlined by the fact that the worst average reconstruction error for *z* value does not exceed 4 degrees.

Seq.	X	Y	Z	X_{max}	Y_{max}	Z_{max}
1	1.87	**2.76**	2.85	5.32	9.07	11.17
2	2.34	7.84	2.26	6.17	26.47	**5.36**
3	1.81	5.83	**1.71**	6.71	9.74	5.37
4	2.79	7.05	3.72	6.73	12.08	9.03
5	3.12	*14.90*	2.00	7.39	*34.07*	7.15
6	*12.12*	3.06	2.85	*25.58*	**7.49**	9.59
7	**1.32**	10.73	2.23	**4.78**	21.65	6.61
8	4.22	10.02	*3.44*	19.64	31.75	*13.90*
9	3.83	5.30	3.05	11.06	12.65	9.67

Table 1: mean and maximal reconstruction errors in degrees

The reconstruction rate of the current implementation, discarding the time spent in reading and decoding the video stream, is approximately 30 frame/s on a Pentium III 600 MHz. This value is nearly constant for different sequences, despite the iterative nature of the reconstruction algorithm. Considerable improvements can be expected exploiting graphical hardware to perform model transformation and image warping, which account for 20% of the execution time.

The facial feature tracking algorithm has been tested on several image sequences with subjects of different age and sex. A first series of experiments has been performed directly on the input images, with subjects front facing the camera and not moving the head, in order to avoid possible distortions due to incorrect head pose reconstruction. Finally, the whole system has been tested on sequences showing both head motion and significant changes of face expressions. The size of camera images is 320·240 pixels, while the texture map used is a 256·256 pixels array. The results presented are based only on a qualitative comparison given by the superimposition of the templates on the incoming images. Examples of the final reconstruction can be seen in Fig. 14 and Fig. 15.

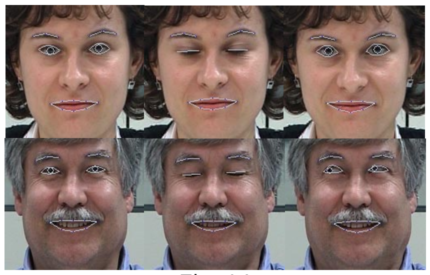
Fig. 14

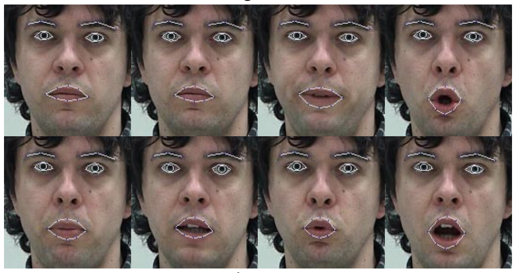
Fig. 15

The facial features reconstruction process runs at 40 frame/s on a Pentium III 600 MHz using as input images the full 320·240 frames. The 256·256 texture map is reconstructed at about 55 frame/s, and the whole reconstruction process (head pose and facial features reconstruction) runs at about 18 frame/s.

5. Conclusion and future work

In this paper we have presented an approach which is capable of reconstructing head pose and facial expression from a monocular view in real-time. The proposed system separates head and facial feature tracking. Our approach to head-pose reconstruction is model-based. The system exploits both motion and texture information, which are combined to strengthen tracking accuracy. Reconstruction is achieved finding for each frame the pose that minimizes the differences between model and image features. The advantage of this model based reconstruction process is that motion and texture registrations are based only on the image features being observed, which correspond to the locations of the projected model points. Hence, disturbing motion or similar textures in other parts of the input image are completely ignored by the system. Moreover, unlike other feature based approaches, the type of analyzable motion is not constrained by features vanishing for some views.

After the correct head pose has been found, the current frame is warped into the model texture, obtaining a stabilized view of the face that is used to enhance the reconstruction of facial expressions. Deformable multi-state templates are used to detect and track significant facial features, like brows, eyes and lips, exploiting shape, texture and motion information.

The described system is currently in its development stage. The goal of the project is to build a completely automatic approach. So far, both head pose and facial feature reconstruction requires user intervention in the initialization phase. Several works presented in literature describing automatic feature finding procedures will be carefully analyzed in order to find an approach compatible with the outlined system. Further studies will be carried out on making the system more robust. This will require to evaluate some sort of confidence level of the tracking results and to restart the initialization phase when it drops off its acceptable level. Also, since one the final goals of the system is to become a virtual device for other applications, it is necessary to reduce the amount of CPU time required to extract animation parameters.

6. Acknowledgments

The author would like to thank Prof. Aldo Laurentini for his suggestions on this paper. This work has been supported by grants from "*JOE - Join Our Experience*" research project of TILab laboratories.

REFERENCES

[1] A. Schodl, A Haro, I. Essa, "Head tracking using a textured polygonal model", Proc. Workshop on Perceptual UI, 1998

[2] M. La Cascia, S. Sclaroff, V Athitsos "Fast, reliable head tracking under variant illumination: an approach based on registration of texture-mapped 3D models", IEEE Trans. PAMI, Vol. 22(4), April 2000: 322-336

[3] A. Azerbayejani, B. horowitz, A. Pentland, "Recursive estimation of structure and motion using the relative orientation constraint". Proc. CVPR'93, 1993

[4] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms". Proc. CVPR'98, 1998, pp. 232 -237

[5] N. Oliver, A. Pentland, F. Bérard, "LAFTER: a real-time face and lips tracker with facial expression recognition". Pattern Recognition, vol. 33, 1999, pp. 1369-1380

[6] R. Stiefelhagen, J. Yang, A. Waibel, "Tracking eyes and monitoring eye gaze". Proc. Workshop on Perceptual UI, 1998

[7] G.D. Hager, P.N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination". IEEE Trans. PAMI, Vol. 20, 1998, pp. 1025-1039

[8] S. Valente, J. L. Dugelay, "Face tracking and realistic animation for telecommunicant clones". IEEE Multimedia Computing and Systems, Jan 2000, pp. 34-43

[9] S. Basu, I. Essa, A. Pentland, "Motion regularization for model-based head tracking", Proc. of ICPR'96, Vienna, Austria, August 1996

[10] Y. Zhang, C. Kambhamettu, "Robust 3D head tracking under partial occlusion", IEEE, 2000

[11] M. Malciu, F. Prêteux, "A robust model-based approach for 3D head tracking in video sequences", Proc. of FG 2000, pp. 169-174, 2000

[12] P.M. Antoszczyszyn, J.M. Hannah, P.M. Grant, "Accurate automatic frame fitting for semantic-based moving image coding using a facial code-book". Proc. ICIP 96, 1996, Vol 1 , 1996: pp. 689 –692

[13] G.R. Bradsky, "Computer vision face tracking for use in a perceptual user interface", Intel Technology Journal Q2, 1998.

[14] A. Yuille, P. Haallinan, D.S. Cohen, "Feature extraction from faces using deformable templates", Int. Journal of CV, Vol. 8, 1992: pp. 89-111

[15] J.Y. Bouget, "Pyramidal implementation of the Lucas Kanade feature tracker. Description of the algorithm", Intel corporation, Internal Rept.

[16] Y. Tian, T. Kanade, J.F. Cohn, "Recognizing action units for facial expression analysis". IEEE trans. PAMI, Vol. 23 (2), 2001, pp. 97 -115

[17] K. Jin, J. Cho, "Automatic feature tracking", Proc. IEEE TENCON, 1999, pp. 68-71

[18] M. Pantic, L.J.M. Rothkrantz, "Automatic analysis of facial expressions: the state of the art", IEEE trans. PAMI, vol. 22 (12), 2000, pp. 1424-1445

[19] M. Malciu, F. Prêteux, "Tracking facial features in video sequences using a deformable model-based approach", Proc. Conf. on Mathematical Modeling, Estimation and Imaging, San Diego, CA, 2000, pp. 51-62

[20] S. Bernögger, L. Yin, A. Basu, A. Pinz, "Eye tracking and animation for MPEG-4 coding", Proceedings of 14th ICPR (2), 1998, pp. 1281-1284