The Visual Computer

Modeling and animation of facial expressions based on B-Splines

Michael Hoch[1],
Georg Fleischmann[1],
Bernd Girod[2]

[1]Academy of Media Arts
Department of Computer Science/Audio-Visual Media
Peter-Welter-Platz 2, D-0676 Cologne,
Germany
e-mail: micha@khm.uni-koeln.de
[2] Institute for Telecommunications, University
Erlangen-Nuremberg, Erlangen, Germany

Abstract. This paper describes a technique for the automatic adaptation of a canonical facial model to data obtained by a 3D laser scanner. The facial model is a B-spline surface with 13×16 control points. We introduce a technique by which this canonical model is fit to the scanned data and that takes into consideration the requirements for the animation of facial expressions. The animation of facial expressions is based on the facial action coding system (FACS). Using B-splines in combination with FACS, we automatically create the impression of a moving skin. To increase the realism of the animation we map textural information onto the B-spline surface.

Key words: B-spline surfaces – Surface fitting – Facial animation – FACS – 3D digitizing

Correspondence to: M. Hoch

1 Introduction

Around the world research is being carried out towards new graphical user interfaces. Since face-to-face communication is the most natural method of human interaction, facial animation systems are attracting more and more attention. There are also many other applications in which facial animation is important and useful. An area of great interest is the production of computer-generated films using synthetic actors. In medicine, preoperative tests can help plan facial surgery. In the field of image coding, facial animation techniques are suitable for reducing the transmission rate of visual telephone calls. Further applications are the identification of criminals using a 3D face modeler or teaching lip reading to the deaf.

The human perceptual skills in recognizing faces are very refined (Harmon 1973). From a person's facial expression we can see whether he is happy or unhappy, angry or cheerful, etc. We are able to recognize a particular person out of a large group immediately and to distinguish very subtle changes in facial expressions. Facial expressions are very complex and are an important nonverbal communication channel. For this reason it is a difficult and challenging task in computer graphics to synthesize realistic looking facial expressions. In the computer synthesis of facial expressions we must deal with two main objectives: the *geometric representation* of the human face and the *control of the facial animation*. To produce convincing facial animations we must pay attention to both of these objectives.

Most facial animation systems are based on polygon meshes for representing the skin of the face (Aizawa 1989; Waters 1987, 1990; Williams 1990, etc.). Only a few systems (Nahas 1988; Waite 1989) use a B-spline model instead of a polygonal model. The accuracy of the model is determined by the number of polygons used. Polygonal models are easy to handle, and the data points can be controlled directly. A disadvantage of polygonal models is the fact that additional expenditure for realistic modeling of the facial skin is necessary. Furthermore, for highly curved surfaces like the human face, we find that using curved surfaces like B-splines instead of polygons is more appropriate. Other advantages of B-splines are that fewer data must be handled than for polygonal models, and moving a control point influences the surface only over a limited region (local control), which automatically creates the impression of moving skin.

The Visual Computer (1994) 11:87–95
© Springer-Verlag 1994

Independent of the geometric representation, the facial model should look realistic. Entering a person's geometric facial model by hand is a very difficult and time-consuming task. Fortunately, there are techniques for the automatic acquisition of 3D data of a human face. One method is 3D geometric reconstruction of a human face using 2D images, for example, photographs. Another method is the digitization of the face using a 3D scanner.

There are two completely different approaches for controlling facial animation: key-expression interpolation and parameterization. Key-expression interpolation requires the collection of all possible expressions of the face. Different data must be stored for every individual face. For this method, we either need to maintain a large database of expressions, or we must restrict the number of key expressions. In the later case, expressions that fall outside the key expressions are unattainable. Because of these difficulties with key-expression interpolation, models have been developed that allow the control of facial expression with parametricly based techniques. The fundamental concept of a parametric model is the control of a large number of objects by means of a small number of parameters (Parke 1982). Various facial expressions are obtained by manipulating only a few parameters. Parameterizations that are based not on a theoretical model, but on intuition and experience are called "ad hoc parameterizations". Most of the systematic parameterizations refer to the facial action coding system (FACS) developed by Ekman and Friesen (1977).

In this paper we describe a technique that allows the automatic adaptation of a canonical facial model to data acquired by scanning a person's head with a 3D laser scanner. This technique meets the requirements for a parametricly controlled animation based on FACS, i.e., B-spline control points are placed in areas that correspond to the action units of FACS. Thus, our approach combines the high quality of 3D laser scanner data with the advantages of B-splines and the acknowledged efficiency of FACS.

2 The facial model

The first step in the development of a facial animation is the acquisition of the 3D surface data of the face. The quality of this data is an important factor in realism and a convincing overall system. Since it is hopeless to attempt to enter a realistic looking geometric facial model manually, a variety of techniques have been developed for achieving data of satisfactory quality. Popular techniques are reconstruction from 3D photographs, reconstruction from cross-sectional scans, e.g., computer-aided tomography (CAT) scans, or 3D digitizing. Available 3D digitizing devices are based on various methods such as ultrasound, magnetic fields, or laser light (Hall 1992). The data used in this work are supplied by a scanner (Cyberware, Monterey, Calif.) that uses laser triangulation to acquire range as well as textural information. The 3D data obtained by scanning the face are very precise and of high resolution, i.e., 512×512 data points for a cylindrical scan or about 200×200 data points for the face. These data form an excellent basis for the geometric representation of a human face, which in our case is modeled as a B-spline surface.

FACS is widely accepted theoretical foundation for the control of a facial animation. FACS is based on an anatomical analysis of facial movements and has been developed by the psychologists Ekman and Friesen (1977). The human head consists mainly of bone, which is inanimate, apart from jaw rotation, skin, and muscles. Any facial expression results from the movements made by a combination of any of the 268 muscles of the face. Ekman and Friesen discovered that there is a finite set of possible basic actions of which the human face is capable. Each of these basic actions, known as action units (AUs), consists of a set of muscles that cannot be controlled independently for that motion, i.e., it cannot be broken up into smaller actions. There are 46 AUs that interact in several different ways to create facial expressions. FACS is a means of noting all the changes to the face through time that are caused by movements. However, it is not suitable to describe changes of the face that do not involve movements, such as blushing. For implementation on a computer, FACS is very suitable since each facial expression can be described as a list of the names of the AUs involved.

Besides FACS, the second theoretical basis of this work is the theory of B-spline surfaces. B-spline surfaces are parametric functions, $P(u, w)$, which are controlled by grid points $B_{ij} =$

$[B_{ijx}, B_{ijy}, B_{ijz}]$, $0 \leq i \leq n$, $0 \leq j \leq m$. Points on the surface of a B-spline patch are defined as a weighted sum of these control points B_{ij}.

$$P(u, w) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,j} N_{i,k}(u) M_{j,l}(w) \qquad (1)$$

$N_{i,k}$ and $M_{i,l}$ are basis functions of order k and l respectively. The basis functions $N_{i,k}$ (and $M_{i,l}$) can be calculated recursively by using the Cox-de Boor recursion formulas (Rogers 1990):

$$N_{i,1}(u) = \begin{cases} 1, & \text{if } x_i \leq u < x_{i+1} \\ 0, & \text{otherwise} \end{cases} \qquad (2)$$

$$N_{i,k}(u) = \frac{(u - x_i) N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u) N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}}, \quad k \geq 2.$$

The values x_i are elements of the so-called knot vector $[x_0, \ldots, x_n]$. Details about B-splines can, for example, be found in (Farin 1990; Rogers 1990). In the following section we combine these concepts of FACS, B-spline surfaces, and the Cyberware facial data to yield an efficient facial animation system.

With respect to the adaptation of the facial mask to the given data points, the degrees of freedom of B-splines are of particular interest. These parameters are

- The number and position of the control points
- The choice of the knot vector
- The estimation of the parameter values u, w for the given data points
- The order of the B-spline surface

Usually, an order of 3 or 4 is sufficient for B-splines. Higher-order curves allow more strongly curved surfaces to be modeled, but also require more calculations. Order 4 is a good trade-off between the expense of calculations and the modeling potential of the resulting B-spline. We choose $k = l = 4$. The number of control points $(n + 1) \times (m + 1)$ determines the overall variability of the surface. We use a B-spline mask of 13×16 control points. In Waite (1989) the general suitability of a similar mask with 12×16 control points is shown. Since our animation results were unsatisfactory for the mouth area, we introduced an additional row of control points. However, we have eliminated a few double control points. The number of control points is essential for the realism of the animation and depends, of course, on the particular application. For applications that require a talking actor, for example, a much finer model of the mouth area is desirable. In this case we would suggest using B-spline surfaces with local refinement in the mouth area, i.e., two or more B-spline patches are combined to get a finer resolution in critical areas. The canonical facial model used in this work is shown in Fig. 1.

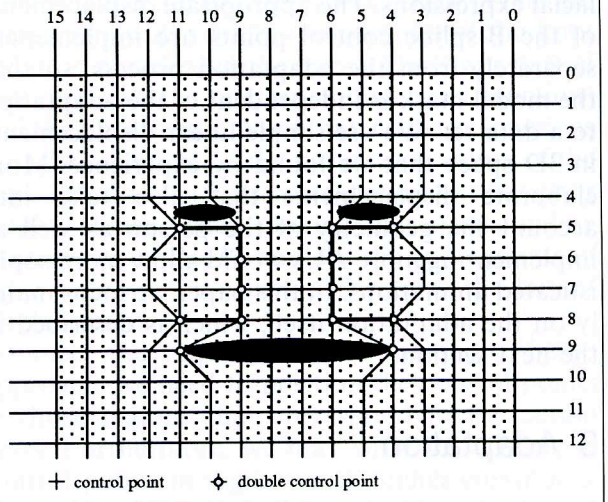


Fig. 1. Canonical facial model

The remaining degrees of freedom of B-spline surfaces, i.e., the positioning of the control points, the influence of the knot vector and the estimation of the parameter values u, w for the given data points, are discussed in the following section.

The last step in the development of a model-based facial animation is to define the animation itself, i.e., the implementation of the AUs to achieve the desired animations. Four AUs of FACS have been chosen and hard-wired to the model mask in order to achieve animations without further manual fine tuning of the model. We chose action units AU1 (inner brow raiser), AU2 (outer brow raiser), AU4 (brow lower), and AU12 (lip corner

puller), which are sufficient to generate simple facial expressions. The appropriate displacements of the B-spline control points are implemented separately from the adaptation process, so that the model mask is independent of the adaptation to a data set. We use simple linear displacements in 3D space that we found by experience. More elaborate displacements that also take into account the geometry of the person as well as implementing more AUs would allow more sophisticated animations. In this paper we focus mainly on the adaptation itself, which is described in the next section.

3 Adaptation

In order to achieve a realistic shape for the mask, we adapt it to the data points given by the laser scanner, minimizing the mean square error between the given data points and the surface points. The face region, consisting of 200×200 data points, has been determined by hand and extracted first. The model mask used is the B-spline mask shown in Fig. 1. Since the animation is hard-wired to the model mask, we must make sure that the control points are correctly positioned in regions where the respective AUs apply, i.e., the control points associated with an AU result in the deformation of the correct region of skin tissue. Hence, the adaptation process can be divided into two logical parts: (a) the surface fitting by minimizing the mean square error and (b) the constraints due to the model-based approach. Finally, the color information of the laser scanner is texture-mapped on the facial mask to add more realism.

3.1 Surface fitting

We will use a simple, direct method, described in Rogers (1990), that operates on topologically rectangular nets, i.e., the data can conceptually be arranged to form a rectangular grid. Figure 2 shows the problem schematically for a 4×4 control polygon and a 8×8 data matrix.
Recall that the B-spline surface to be determined is given by

$$P(u, w) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,j} N_{i,k}(u) M_{j,l}(w). \tag{3}$$

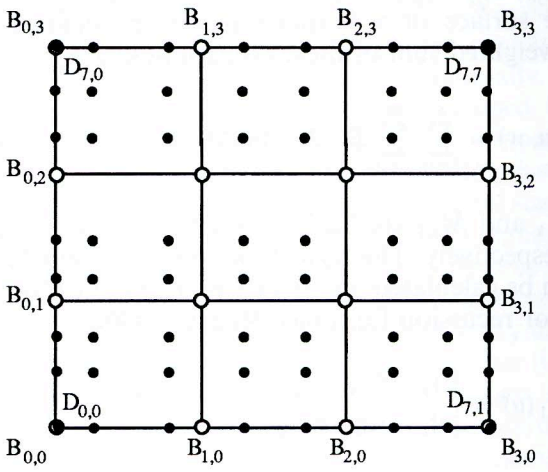


Fig. 2. Conceptual rectangular grid of the data and control points

$P(u, w)$ is a point on the B-spline surface for given parameter values u, w. $N_{i,k}(u)$ and $M_{j,l}(w)$ are the basis functions that can be determined for a known order and a known number of control points in each parametric direction. If a data point lies on the B-spline surface, it must satisfy Eq. 3. Hence, for each known data point $D_{g,h}$, this equation provides a linear equation for the unknown control points $B_{i,j}$. Writing out this equation for a single data point yields:

$$\begin{aligned} D_{g,h}(u_g, w_h) &= N_{0,k}(u_g)[M_{0,l}(w_h) B_{0,0} + M_{1,l}(w_h) B_{0,1} \\ &\quad + \cdots + M_{m,l}(w_h) B_{0,m}] \\ &\quad + N_{n,k}(u_g)[M_{0,l}(w_h) B_{n,0} + M_{1,l}(w_h) B_{n,1} \\ &\quad + \cdots + M_{m,l}(w_h) B_{n,m}]. \end{aligned} \tag{4}$$

A data field of size $r * s$ provides a system of linear equations, written in matrix form:

$$\underset{r \cdot s * 3}{D} = \underset{r \cdot s * (n+1)(m+1)}{C} \quad \underset{(n+1)(m+1) * 3}{B} \tag{5}$$

with $C_{i,j} = N_{i,k} \cdot M_{j,l}$,

where D contains the 3D coordinates of the data points, C contains the products of the basis functions and B contains the coordinates of the unknown control points. Usually, the problem is overdetermined, i.e., there are more data points than unknown control points, and a solution can only be obtained in a mean sense. The mean squared distance between data points and the B-spline surface are minimized by using the pseudo inverse method:

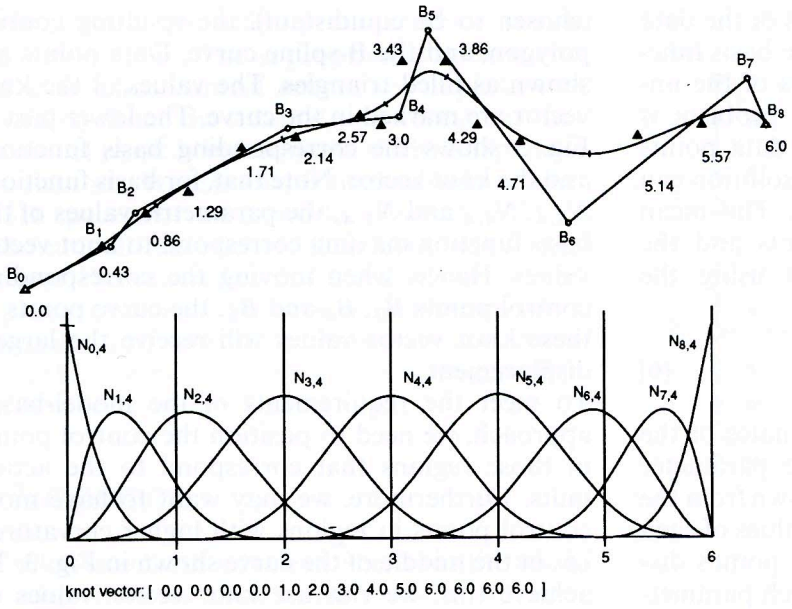
$$B = [C^T C]^{-1} C^T D. \tag{6}$$

Note that in Eq. 4 the x, y, z coordinates of the data points $D_{g,h}$ are given, but the parameter values u_g, w_h of each point are not known from the data. This is because the parameter values of each data point are a measure of the data point's distance along the B-spline surface in each parametric direction. Hence, the u and w parametric values for each data point must be estimated. They can be obtained by using equidistant values or by using a chord-length approximation (Rogers 1990). In the case of equidistant values, this will lead to an almost uniformly spaced grid of control points after the adaptation method just described. In the case of a chord-length approximation, the spacing will be adapted to the spacing of the data points. This is not appropriate in our case because we need the grid of control points that is shown in Fig. 1. The estimated parametric values will influence the position of the control points, which is the key problem in incorporating the contraints to achieve such a grid. We will show how these values are used to position the control points explicitly in order to meet the requirements of the model-based approach. This is a key problem of the adaptation method; we will explore it in detail in the following section.

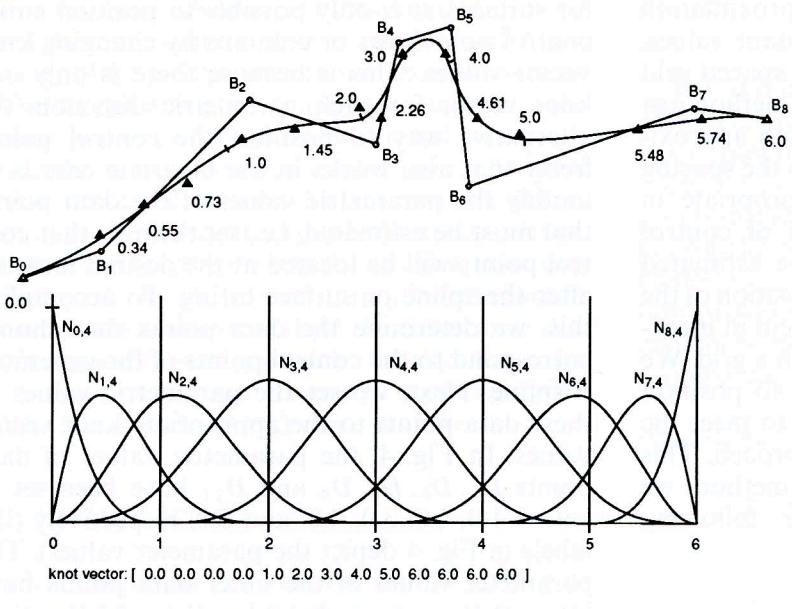
3.2 Constraints

In this section we show how the u and w parametric values for the data points can be used to position the control points in the adaptation process. For simplicity, we use a simple curve fitting example (15 data points, 9 control points), but this can easily be extended to surfaces. Figure 3 shows the data points with their parametric values (chosen to be equidistant), the resulting control polygon, and the B-spline curve. Data points are shown as filled triangles. The values of the knot vector are marked in the curve. The lower part of Fig. 3 shows the corresponding basis functions and the knot vector. Note that, for basis functions $N_{3,4}$, $N_{4,4}$ and $N_{5,4}$, the parametric values of the basis function maxima correspond to knot vector values. Hence, when moving the corresponding control points B_3, B_4 and B_5, the curve points of these knot vector values will receive the largest displacement.

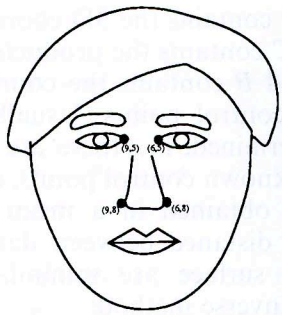
To meet the requirements of the model-based approach, we need to position the control points in those regions that correspond to the action units. Furthermore, we may want to have more control points in regions with higher curvatures, i.e., in the middle of the curve shown in Fig. 3. To achieve this, we can use knot vector values or choose parametric values for the data points differing from those that were used in Fig. 3. Altering knot vector values is appropriate for curves, but for surfaces, it is only possible to position entire control point rows or columns by changing knot vector values. This is because there is only one knot vector for each parametric direction. An alternative way to position the control points freely that also works in the bivariate case is to modify the parametric values of the data points that must be estimated, i.e., set them so that control points will be located at the desired location after the spline or surface fitting. To accomplish this, we determine the data points that should correspond to the control points of the generated B-spline. Next, we set the parametric values of these data points to the appropriate knot vector values. In Fig. 4, the parametric values of data points D_4, D_6, D_8, D_9 and D_{11} have been set to values 1.0, 2.0, 3.0, 4.0, and 5.0, respectively (the labels in Fig. 4 depict the parameter values). The parameter values of the other data points have been set by using a chord-length approximation. Thereafter, the surface-fitting method (here curve fitting) described in the previous section has been applied. Note that the curve points at parametric values of the knot vector (marked in the curve in Fig. 4) are located next to the desired data points. Furthermore, the control points B_3 through B_6 are positioned so that the desired regions will be influenced when these control points are moved.



3



4



5

Fig. 3. Curve fitting with equidistant parameter values. *Filled triangles* are data points

Fig. 4. Positioning control points by setting the parameter values of the data points. *Filled triangles* are data points

Fig 5. Four characteristic points that are determined by hand

In order to position the control points as shown in Fig. 1, we determine a set of characteristic points of each face that correspond to feature points for the action units. In our case, we determine the four characteristic points shown in Fig. 5 by hand [which correspond to control points (6, 5), (9, 5), (6, 8), and (9, 8) in Fig. 1]. All other points that are necessary for the four implemented AUs

like the lip corners and the brow regions, are determined relative to these four control points. This method worked well for the data sets used. It should be possible to determine the characteristic points automatically by using pattern recognition techniques. Next, we automatically set the parametric values of the data points that correspond to AU regions to the appropriate knot vector values. The control points will then be located in the center of these regions after the surface fitting. The parametric values of all other data points have been set by using equidistant values.

4 Results

Because of the model-based approach, the resulting masks can be animated without any further fine tuning of the animation. The quality of the adaptation process can be seen in Fig. 6, where the laser scanner data (shown as a shaded image) and the corresponding B-spline masks are shown. For simplicity, the eyes and the mouth are implemented as holes (trim curves).

Fig. 7 shows four different facial expressions generated by FACS. For the first row of examples, the same B-spline mask as shown in Fig. 6

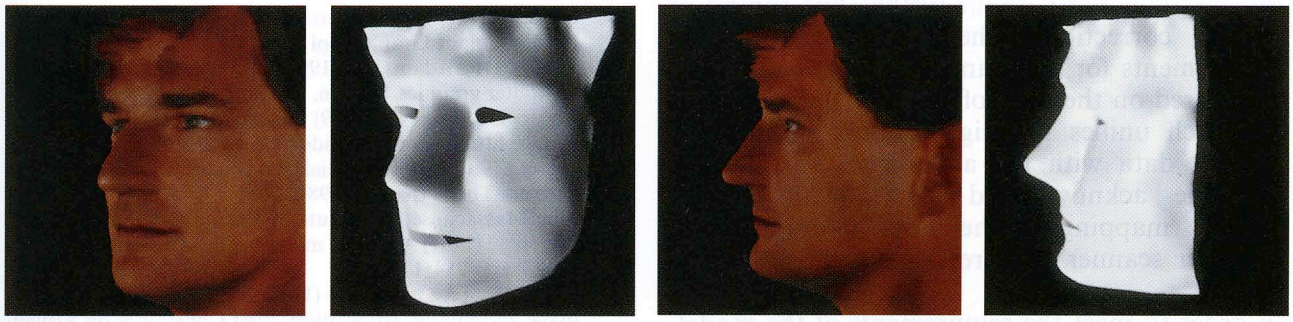


Fig. 6. Laser-scanner data and the resulting B-spline mask

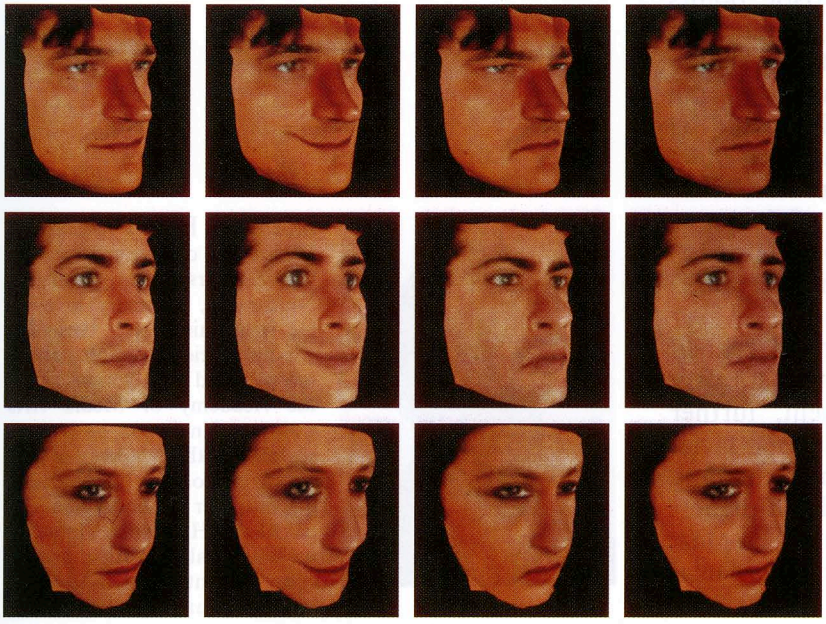


Fig. 7a–d. Various facial expressions with B-spline-mask and texture-mapped color information: **a** neutral; **b** friendly; **c** angry; **d** sad

has been used, and texture mapping of the color information supplied by the laser-scanner has been added. Figure 7 shows three different masks without trim curves, but with texture. For this work, a total of four data sets have been used that were supplied by the MIT Media-Lab.

5 Conclusion and future work

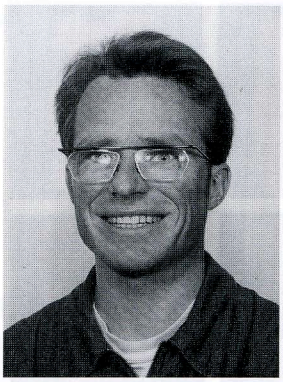
We have introduced a technique that adapts a B-spline facial mask consisting of 13×16 control points to the data of a 3D laser scanner. Due to the model-based approach, the resulting mask can be animated without further fine tuning or manual corrections. The technique meets the requirements for a parameter-controlled animation based on the AUs of the FACS system. The approach unifies the high quality of 3D laser scanner data with the advantages of B-splines and the acknowledged efficiency of FACS. Texture mapping of the color information of the laser scanner adds realism to the generated masks.

In order to meet the requirements of the model-based approach, the parameter values of the data points have been set to specific values. These settings insure that the control points will be located in the correct regions after the surface fitting. In this work, four characteristic points of the face have been determined by hand, and the relative parametric values are set in a simple way in order to constrain the position of the control points in specific regions. This could be enhanced by an automatic determination of characteristic points with pattern recognition techniques and by optimizing the setting of the parameter values, depending on the input data, i.e., optimizing the position of the control points (Rogers 1989).

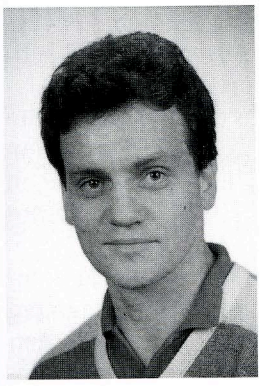
The eye and mouth areas need to be refined because of their importance and the subtlety of their movements. At the moment, further investigation is being done to create a facial model by hierarchical refinement and extending the adaptation process to this model. Furthermore, the animation of a hierarchical B-spline model must be investigated concerning facial animation.

References

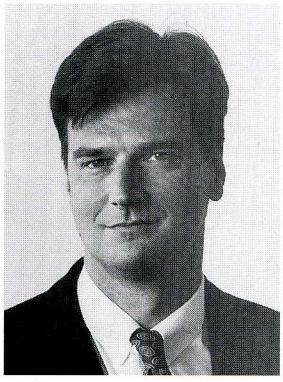
1. Aizawa K, Harashima H (1989) Model-based analysis synthesis image coding (MBASIC) system for a person's face. Signal Process: Image Commun 1:139–152
2. Ekman P, Friesen W (1977) Facial action coding system. Consulting Psychologists Press, Palo Alto, Calif
3. Farin G (1990) Curves and Surfaces for Computer Aided Geometric Design. 2nd Edn, Academic Press, San Diego, Calif
4. Hall V (1992) Speech driven facial animation, Thesis for Bachelor of Science, Department of Computing Science, Curtin University of Technology, Perth, Western Australia
5. Harmon LD (1973) The recognition of faces. Sci Am 229:70–82
6. Nahas M, Huitric H, Saintourens M (1988) Animation of a B-spline figure. Visual Comput 3:272–276
7. Nahas M, Huitric H, Rioux M, Domey J (1990) Facial image synthesis using skin texture recording. Visual Comput 6:337–343
8. Parke FI (1982) Parameterized models for facial animation. IEEE Comput Graph Appl 2:61–68
9. Rogers DF, Adams JA (1990) Mathematical elements for computer graphics. 2nd En, McGraw-Hill, New York, USA
10. Rogers DF, Fog NG (1989) Constrained B-spline curve and surface fitting. Comput Aided Design 21:641–648
11. Waite CT (1989) The facial action control editor, FACE. Masters Thesis (partial) 1989, Media Arts and Science Section, Massachusetts Institute of Technology, Boston Mass.
12. Waters K (1987) A muscle model for facial animation. Comput Graph 214:17–24
13. Waters K, Terzopoulos D (1990) A physical model of facial tissue and muscle articulation. First Conference on Visualization in Biomedical Computing, IEEE Computer Society Press, Atlanta, pp 77–82
14. Williams L (1990) Performance-driven facial animation. Comput Graph 24:235–242



MICHAEL HOCH received his Diploma in computer science at the University of Erlangen-Nuremberg, Germany, in 1992. He wrote his master thesis for computer science diploma about model based facial animation at the Academy of Media Arts Cologne. Since 4/92 he is a researcher at the Academy of Media Arts Cologne, Department of Computer Science/Audio-Visual Media. His research interests include facial animation, human computer interfaces, interactive media and artificial live.



GEORG FLEISCHMANN is currently head of the Department of Computer Science/Audio-Visual Media at the Academy of Media Arts Cologne, Germany. He studied computer science and mathematics at the University of Erlangen-Nuremberg were he graduated in 1986 and received his Dr.-Ing. in 1990. His PhD research concerned performance evaluation of massively parallel computer systems while his current research interests include human computer interfaces, facial animation and human figure animation, interactive media and generative techniques in graphical computer art.



BERND GIROD is Professor of Telecommunications in the Department of Electrical Engineering at University of Erlangen-Nuremberg, Germany. In previous appointments, has been a Professor of Computer Graphics at the Academy of Media Arts in Cologne, Germany, and an Assistant Professor of Media Technology at the Media Laboratory, Massachussetts Institute of Technology, Cambridge, Mass., USA. He has received his Doctoral degree in Electrical Engineering from University of Hannover, Germany. Professor Girod is also co-founder and Chief Scientist of Vivo Software, Inc. in Boston. His research interests include multidimensional signal processing, information theory, video signal compression, human and machine vision, sensory computing, computer graphics and animation, as well as interactive media.

The Visual Computer

3D-line clipping algorithms – a comparative study

Ivana Kolingerová

Department of Informatics and Computer Science, University of West Bohemia, Americká 42, 306 14 Plzeň, Czech Republic
e-mail: kolinger@kiv.zcu.cz

Some well-known line-polyhedron intersection methods are summed up and new accelerating modifications presented. Results of comparison of known and newly developed methods are included. New methods use the fact that each line can be described as the intersection of two planes.

Key words: Line clipping – Convex polyhedron – Line–polyhedron intersection – Dual representation – Intersection detection – Computer graphics

1 Introduction

The problem of finding intersections of 3D geometrical objects with lines or line segments occurs very frequently in computer graphics. If the geometrical object can be considered a convex polyhedron, the solution seems easy and deserves no special attention. However, if the computation must be repeated many times, as, for example, in ray tracing, special effort to speed it up is not wasted.

In this paper, some accelerating methods for computing convex polyhedra-line or line-segment intersections are suggested, and results of their implementation are compared. The methods considered in this article (with one exception) have a complexity of $O(N)$, where N is a number of polyhedron facets. Algorithms are written for searching both points of intersection of the polyhedron and a line.

The following notation is used:

The given line L: $\mathbf{x} = \mathbf{x}_A + \mathbf{s}^T t$,

where $\mathbf{x}_A$ is a point on L, and $\mathbf{s}$ is a directional vector

The given convex polyhedron P:

vertices $\{\mathbf{x}_i, i = 1, 2, \ldots, M\}$

facets $\{f_j, j = 1, 2, \ldots, N\}$

2 The Cyrus–Beck algorithm

This algorithm (Cyrus and Beck 1979) uses the fact that the convex polyhedron in E^3 can be understood as the intersection of halfspaces. Boundaries of these halfspaces are formed by planes in which facets of the polyhedron lie.

Suppose we have a convex polyhedron and a line with some parametrization. Searching for the intersection of these geometrical objects, we can divide the bounding planes of the polyhedron into two groups according to the orientation of their normal vectors. Among the planes oriented towards the observer, we search for the point of intersection with the maximal parameter value t_1. Among planes of the other group, the minimal parameter value t_2 is found. If $t_1 > t_2$, the intersection of the polyhedron with the given line does not exist. If $t_1 \leq t_2$, we compute the points of intersection.

The method is stable, easy to implement and fast. Therefore, it is suitable for comparison. The greatest disadvantage is that the algorithm cannot stop after finding the desired intersections – all facets must be processed.

The Visual Computer (1994) 11:96–104
© Springer-Verlag 1994