8 SELECTED APPLICATIONS

D. PAULUS

INTRODUCTION

In the previous six chapters we have introduced the methodology for model based image interpretation, computer graphics, as well as model based data visualization. In this chapter we apply subsets of the proposed methods to a variety of scenarios.

The expression of human faces are a major source of information in human communication. For image communication as well as for multi-media application it is of high importance to analyze human facial mimics. The appearance of a face is of course greatly influenced by the shape of the bones, such as skull and jaw. In Section 8.5 we describe how facial expressions can be synthesized on a graphical display for a simulated change of the bones. This will be used for surgery planning in medical applications. Due to spasms or paralysis, human facial expression is degraded severely in pathological cases. In Section 8.4 we outline how computer vision techniques can help the physicians to evaluate the degree of facial asymmetry. Model based techniques are used for training and rehabilitation. Section 8.6 describes how facial expressions can be modelled and how the parameters of the model can be used to generate mimics after image and parameter transmission.

One example of a solution for a complex visualization problem is given in Section 8.3 where flow fields are visualized in computational fluid dynamics. From the huge amount of data, that part has to be selected, which is of relevance for the observer.

At the present state of the art, images are presented to the observer on a flat display which provides a two-dimensional view of the data. Images captured by conventional cameras also record information on a two-dimensional grid. Since the real world

has a higher dimension, projection methods are used. Models of real-world objects often represent the three-dimensional structure by surface information. In the previous chapters we have learned how three-dimensional information can be recovered even from two-dimensional projected images. In the final two sections of this volume we describe how three-dimensional models are used in automotive design and how three-dimensional surface models can be created from several range images recorded from different directions. We describe reverse engineering and automatic object model generation that can be used in virtual reality environments. We also show how these techniques can be used for manufacturing of real objects in automotive design where esthetic criteria have to fulfilled. As in the application in Section 8.5, physical models which were used formerly are now replaced by computer simulations. By computer simulation, effects of changes in the shape of a car can be inspected which reduces the costs of car design.

8.1 DIGITIZING 3D OBJECTS FOR REVERSE ENGINEERING AND VIRTUAL REALITY

S. KARBACHER, N. SCHÖN, H. SCHÖNFELD, G. HÄUSLER

Optical 3D sensors are used as tools for *reverse engineering* and *virtual reality* to digitize the surface of real 3D objects. Common interactive surface reconstruction is used to convert the sensor point cloud data into a parametric CAD description (e.g. NURBS). We discuss an almost fully automatic method to generate a surface description based on a mesh of curved or flat triangles.

Multiple range images, taken with a calibrated optical 3D sensor from different points of views, are necessary to capture the whole surface of an object and to reduce data loss due to reflexes and shadowing. The raw data are not directly suitable for import into CAD/CAM systems, as these range images consist of millions of single points and each image is given in the sensor's coordinate system. The data usually are distorted, due to the imperfect measuring process, by outliers, noise and aliasing. To generate a valid surface description from this kind of data, several problems need to be solved:

- the *registration* (transformation) of the single range images into one common coordinate system (see Section 4.2),
- the *surface reconstruction* from the point cloud data to regain the object topology (see Section 4.4),
- the *modeling of the surface geometry* to eliminate errors introduced by the measurement process and to reduce the amount of data (see Section 4.3),
- and the *reconstruction of the texture* of the object from additional video images for virtual reality applications.

Commonly used software directly fit *tensor product surfaces* to the point cloud data [186, 180] (see Section 4.6). This approach requires permanent interactive control by the user. For many applications however, e.g. the production of dental prosthesis or

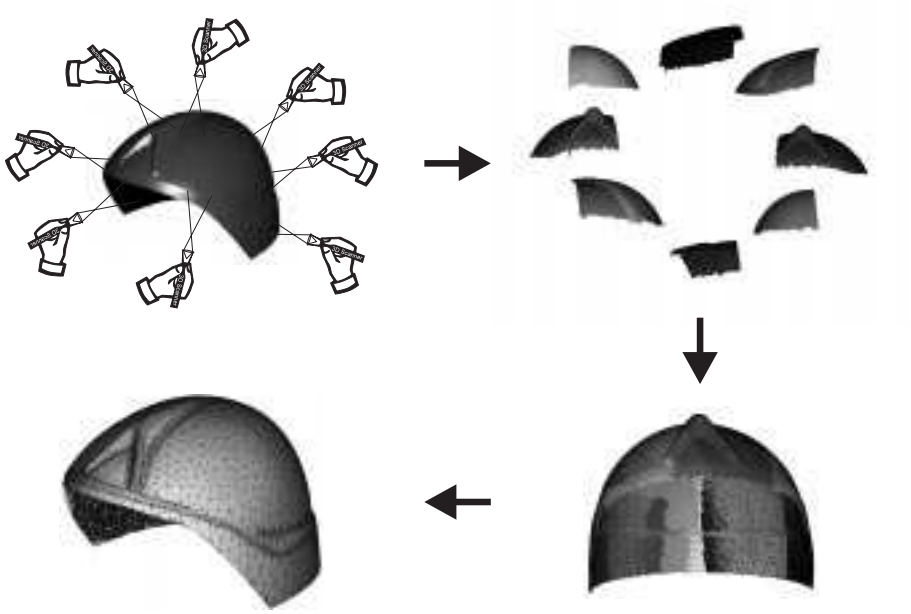


Figure 8.1. Data acquisition (upper left), registration (right) and mesh reconstruction (lower left) for a fire fighter's helmet.

the '3D copier', this kind of technique is not necessary. In this case the generation of meshes of triangles as surface representation is adequate [326, 199, 716, 586, 726, 145].

This section introduces a nearly automatic procedure covering the complex task from gathering data by an optical 3D sensor to generating meshes of triangles [617]. The whole surface of an object can be scanned by registering range images taken from arbitrary positions. The procedure includes the following steps (see Figure 8.1):

Data acquisition: Usually multiple range images of one object are taken to acquire the whole object surface. These images consist of range values arranged in a matrix as on the camera's CCD chip.

Calibration: By measuring a standard with an exactly known shape, a calibration function for transforming the pixel coordinates into metrical coordinates is computed. The coordinate triplets of the calibrated range image have the same structure as the original pixel matrix. This method calibrates each measurement individually. As a result each view has its own coordinate system.

Surface registration: The various views are transformed into a common coordinate system and are adjusted to each other. Since the sensor positions of the different views are not known, the transformation parameters must be determined by an accurate localization method. First the surfaces are coarsely aligned one to another with a feature based method. Then a fine tuning algorithm minimizes the deviations between the surfaces (global optimization).

Mesh reconstruction: The views are merged into a single object model. A surface description is generated using a mesh of curved triangles. The order of the original data is lost, resulting in scattered data.

Data modeling and smoothing: A new modeling method for triangle meshes allows to interpolate curved surfaces only from the vertices and the vertex normals of the mesh (see Section 4.3). Using curvature dependent mesh thinning it provides a compact description of the curved triangle meshes. Measurement errors, such as sensor noise, aliasing, calibration and registration errors, can be eliminated without ruining the object edges.

Texture reconstruction: For each range image a congruent normalized texture map is computed. In order to eliminate artifacts caused by illumination effects, two images of the same object view with illumination from different directions are merged.

8.1.1 Data Acquisition

Depending on the object to be digitized, that is its size and surface characteristics, the appropriate 3D sensor needs to be chosen. For details refer to Chapter 1.

8.1.2 Sensor Calibration

Optical sensors generate distorted coordinates $\boldsymbol{x}' = [x', y', z']^T$ because of perspective, aberrations and other effects. For real world applications a calibration of the sensor that transforms the sensor raw data $\boldsymbol{x}'$ into the metrical Euclidean coordinates $\boldsymbol{x} = [x, y, z]^T$ is necessary.

Our method uses an arbitrary polynomial $\boldsymbol{p}_{\mathrm{c}}$ as a calibration function [304]. The coefficients are determined by a series of measurements of a calibration standard with exactly known geometry. The standard consists of three tilted flat surface patches which intersect virtually at an exactly known position $\boldsymbol{x}_{\mathrm{s}}$. Due to aberrations, the digitized surface patches of the standard are not flat. Therefore *interpolating polynomials* $\boldsymbol{p}_k$, $k \in \{1, 2, 3\}$, (not to be mistaken with the above mentioned calibration polynomial $\boldsymbol{p}_{\mathrm{c}}$)[1] are approximated to find the virtual intersection point $\boldsymbol{x}'_{\mathrm{s}}$ with

$$\boldsymbol{x}_{\mathrm{s}} = \boldsymbol{p}_k(\boldsymbol{x}'_{\mathrm{s}}) \quad \forall\, k. \tag{8.1}$$

In order to fill the whole measuring volume with such calibration points $\boldsymbol{x}_i$, $1 \leq i \leq n$, the standard is moved on a translation stage and measured in n different positions. The intersection points can be calculated with high accuracy, since the information of thousands of data points is averaged by surface approximation. As a result, the accuracy of the calibration method is not limited by the sensor noise. This method is usable for any kind of sensor and, in contrast to other methods, requires no mathematical model of the sensor and no localization of small features, like circles or crosses.

[1] The calibration polynomial $\boldsymbol{p}_{\mathrm{c}}$ transforms arbitrary data points into real world coordinates, the interpolating polynomials $\boldsymbol{p}_k$ only the data points of the k-th calibration plane.

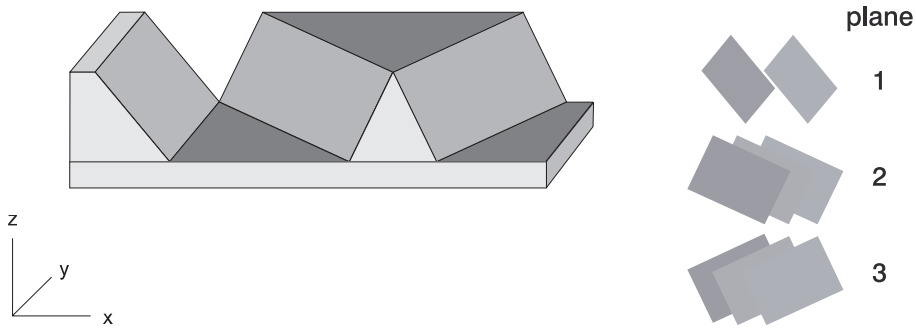


Figure 8.2. Calibration standard with three tilted planes (left) and 8 range images of these planes (right).

Figure 8.2 shows a block of aluminum with three tilted planes, which is used for calibration of our phase measuring (pmt) sensor. It is moved by small steps along y-direction. After each displacement three range images, one for each plane, are taken. Images of the same plane form a class of parallel calibration surfaces. Interpolating polynomials of order 5 are fitted to the deformed plane data. Each polynomial is intersected with polynomials of the other classes (see Figure 8.3). The intersection points are scattered throughout the whole field of view. Their real positions $\boldsymbol{x}_i$ are computed from the geometry of the standard and from its actual translation. The *calibration polynomial* $\boldsymbol{p}_{\mathrm{c}}$ that transforms the measured intersection positions $\boldsymbol{x}'_i$ to the known positions $\boldsymbol{x}_i$ by

$$\boldsymbol{x}_i = \boldsymbol{p}_{\mathrm{c}}(\boldsymbol{x}'_i) \tag{8.2}$$

is approximated by polynomial regression.

Calibration of a range image with 512×540 points takes about 3 seconds on an Intel Pentium® 166 CPU. The calibration error is less than 50% of the measurement uncertainty of the sensor. Since coefficients of higher order are usually very close to zero, it is sufficient to use a calibration polynomial of order 4.

8.1.3 Registration

Usually multiple range images from different views are taken. If the 3D sensor is moved mechanically to definite positions, this information can be used to transform the images into a common coordinate system. Some applications require sensors that are placed manually in arbitrary positions, e.g. if large objects like monuments are digitized. Sometimes the object has to be placed arbitrarily, e.g. if the top and the bottom of the same object are to be scanned. In these cases the transformation must be computed solely from the image data.

Coarse Registration. The following procedure for registration of multiple views is based on feature extraction. It is independent of the sensor that is used. Thus it may be applied to small objects like teeth, and to large objects like busts.

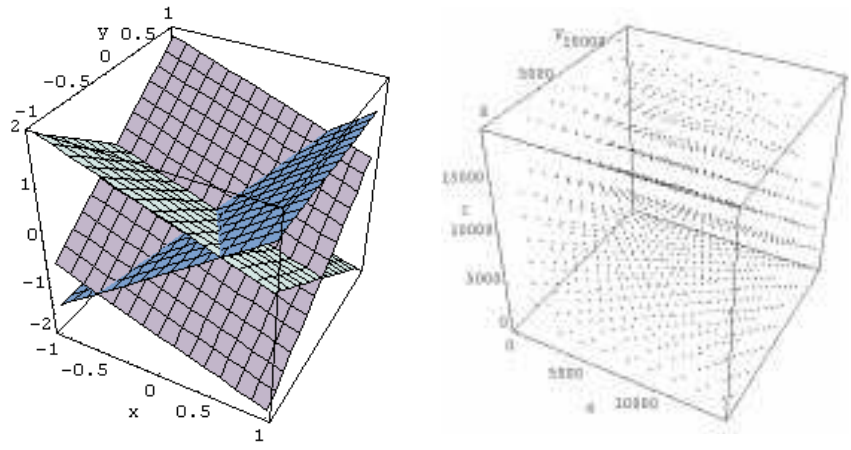


Figure 8.3. Intersection of three polynomial surfaces (one from each class) in arbitrary units (left) and field of view with all measured intersection points in metric coordinates (right).

Zero-dimensional intrinsic features, e.g. corners, are extracted from the range images or congruent gray scale images (see Section 4.2). The detected feature locations are used to calculate the translation and rotation parameters of one view in relation to a master image. Simultaneously, the unknown correlations between the located features in both views are determined by Hough methods. To allow an efficient use of Hough tables, the six-dimensional parameter space is separated into two- and one-dimensional sub-spaces (see [574]).

If intrinsic features are hard or impossible to detect (e.g. on the fireman's helmet in Figures 8.1 and 8.9), artificial markers, which can be detected automatically or manually, are applied to the surface. The views are aligned to each other by pairs. Due to the limited accuracy of feature localization, deviations between the different views still remain.

Fine Registration. A modified Iterated Closest Point (ICP) algorithm is used to minimize the remaining deviations between pairs of views (see Section 4.2.5). Error minimization is done by *simulated annealing*, so that in contrast to classic ICP, local minima of the cost function may be overcome. Since simulated annealing leads to a slow convergence, computation time tends to be rather large. A combination of simulated annealing and the Levenberg-Marquardt algorithm however shows even smaller remaining errors in much shorter time: The registration of one pair of views takes about 15 seconds on an Intel Pentium® 166 CPU. If the data is calibrated correctly, the accuracy (defined by the standard deviation between corresponding data points of both surfaces) is only limited by sensor noise.

Figure 8.4 shows the result of the registration for two views with 0.5% noise. One was rotated by approximately $50°$. The final error standard deviation was approximately σ_{noise} (standard deviation of the sensor noise).

Figure 8.4. Fine registration of two noisy views with very high initial deviation ($\alpha = -30°$, $\beta = 50°$, $\gamma = 40°$, $x = 30$ mm, $y = 60$ mm, $z = -40$ mm).

Global Registration. Solely using registration by pairs, closed surfaces can not be registered satisfactorily. Due to the accumulation of small remaining errors (caused by noise and miscalibration) frequently a chink develops between the surface of the first and last registered view. In such cases the error must be minimized globally over all views. One iteration fixes a single view and minimizes the error of all overlapping views simultaneously. About 5 of these global optimization cycles are necessary to reach the minimum or at least an evenly distributed residual error. Global registration of an object that consists of 10 views takes approximately 30 minutes on an Intel Pentium® II 300 CPU.

8.1.4 Mesh Reconstruction

The object surface is reconstructed from multiple registered range images. These consist of a matrix of coordinate triples $\boldsymbol{x}_{n,m} = [x, y, z]^T_{n,m}$. The object surface may be incompletely sampled and the sampling density may vary, but should be as high as possible. Beyond that, the object may have arbitrary shape, and the field of view may even contain several objects. The following steps are performed to turn this data into a single mesh of curved or flat triangles:

Mesh generation: Because of the matrix-like structure of the range images, it is easy to separately turn each of them into triangle meshes with the data points as vertices. For each vertex the vertex normals are calculated from the normals of the surrounding triangles.

First smoothing: In order to utilize as much of the sampled information as possible, smoothing of measuring errors like noise and aliasing is done before mesh thinning (see Section 4.3.2).

First mesh thinning: Merging dense meshes usually requires too much memory, so that mesh reduction must often be carried out in advance (see Section 4.1.3). The permitted approximation error should be chosen as small as possible, as ideally thinning should be only done at the end of the processing chain.

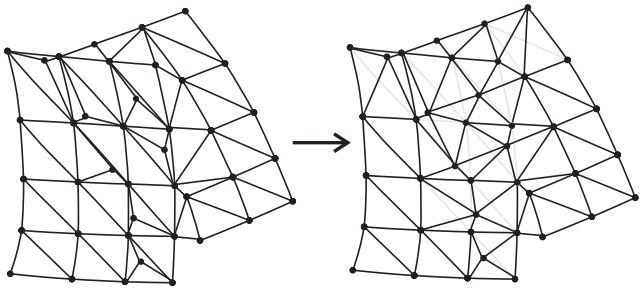


Figure 8.5. Topological optimization of the mesh from Figure 4.26 on page 178 using edge swap operations. Triangles that are as equilateral as possible were aspired.

Merging: The meshes from different views are merged by pairs using local mesh operations like *vertex insertion*, *gap bridging* and *mesh growing* (see Figure 4.26 and Section 4.1.7). Initially a master image is chosen. The other views are merged into it successively. Only those vertices are inserted whose absence would cause an approximation error larger than a given threshold.

Final smoothing: Due to calibration and registration errors, the meshes do not match perfectly. Thus after merging the mesh is usually distorted and has to be smoothed again.

Final mesh thinning: Mesh thinning is continued until the given approximation error is reached. For thinning purposes also a classification of the surfaces according to curvature characteristics is carried out.

Geometrical mesh optimization: Thinning usually causes awkward distributions of the vertices, so that elongated triangles occur. Geometrical mesh optimization moves the vertices along the curved surface in order to produce a better balanced triangulation (see Section 4.3.3).

Topological mesh optimization: At last the surface triangulation is reorganized using *edge swap* operations, in order to optimize certain criteria (see Figure 8.5). The interpolation error is usually minimized.

The result of this process is a mesh of curved triangles. A new modeling method is able to interpolate curved surfaces solely from the vertex coordinates and the assigned normal coordinates (see Section 4.3). This enables a compact description of the mesh, as modern data exchange formats like Wavefront OBJ, Geomview OFF, or VRML support this data structure.

8.1.5 Texture Reconstruction

The previous sections describe methods to reconstruct the shape of three dimensional objects. For visualization purposes it is often desirable to acquire the color texture of the object as well. Color video images of the object usually show artifacts caused by

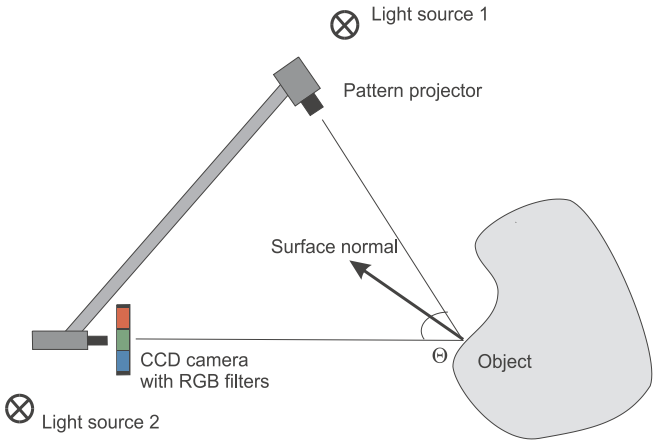


Figure 8.6. A 3D sensor with two additional light sources for texture reconstruction.

illumination effects, which are not part of the surface color itself, for example white spots due to shiny surfaces and brightness variations of diffusely reflected light. These artifacts can be eliminated if two images of the same object view with illumination from different directions are taken [616] (see Figures 8.6 and 8.7). The 3D sensor acquires a range image of the same view, so that for each camera pixel the 3D coordinates are known.

Variations of the orientation of the surface normals change the brightness in diffusely reflecting parts of the image. Assuming Lambertian reflection this effect can be compensated if the normals are known. Compensation factors for each pixel are

Figure 8.7. Color images of St. Georg ('Germanisches Nationalmuseum') taken from the same view and illuminated from left and right.

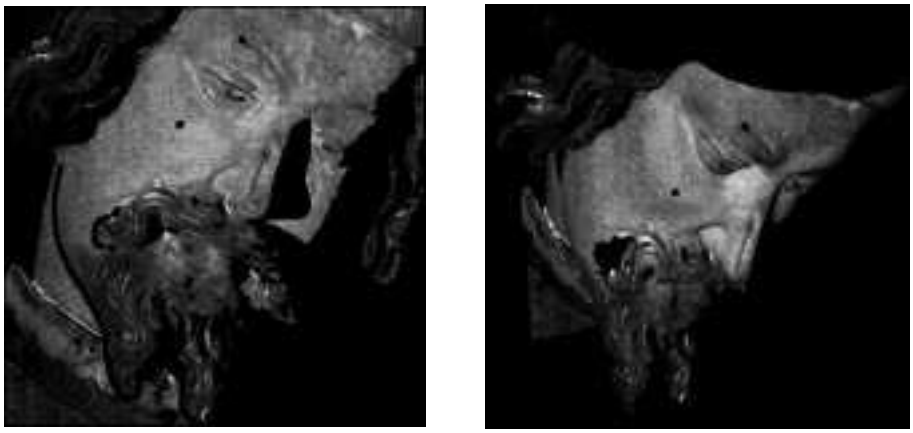


Figure 8.8. A normalized texture image is computed from the two color images of Figure 8.7 (left) and mapped onto the triangle mesh of the corresponding range image (right).

calculated from the angles between the normals and direction of illumination. The normals are computed from the range data and therefore are usually distorted by camera noise. Since the influence of this noise depends on the normal orientations, *confidence values* are assigned to each pixel which quantify the accuracy of that pixel. The following observations are considered:

- The lower the intensity the higher is the relative influence of the noise.
- The noise is amplified by the correction factor.

As a result, the confidence values increase with the intensity of the corresponding pixel and decrease with the deviation of the surface normal from the direction of illumination.

In order to remove shiny regions from the texture images, additional confidence values are defined. In contrast to the previous ones, these increase with the deviation of the observation direction from the directions of specular reflection. A mathematical model for specular reflection is used for that purpose. It depends on the spatial angles of the shiny regions, which are automatically computed by masking the highlights.

Finally, the two images with corrected diffuse reflection are merged by weighting the color of each pixel with its confidence value. The result is a normalized texture image without shading and highlights (see Figure 8.8). This texture can be mapped onto the triangle mesh that was generated from the corresponding range image (see Figure 8.8) or may be used for coloring the vertices of the final mesh (see Figure 8.11). The latter method requires meshes with texture dependent densities. Hence, a method to merge normalized texture maps from different viewpoints into a single one is currently being designed. The texture image can be mapped onto a mesh with arbitrary density without loss of details.

8.1.6 Results

The described digitizing method was tested with many different objects, technical and natural as well. The errors reinduced by the modeling process were smaller than the errors in the original data (measuring, calibration and registration errors). The smoothing method is specifically adapted to the requirements of geometric data, as it minimizes curvature variations. Undesirable surface waviness is avoided. Surfaces of high quality for visualization, NC milling and 'real' reverse engineering are reconstructed. The method is well suited for metrology purposes, where high accuracy is desired.

Figure 8.9 shows the results for a design model of a fire fighter's helmet. The reconstructed CAD-data is used to produce the helmet. For that purpose, the triangular mesh was translated into a mesh of Bézier triangles, so that small irregularities on the boundary could be cleaned manually. This work was carried out in collaboration of the Chair for Optics and the Computer Graphics Group of the University of Erlangen-Nürnberg. Eight range images containing 874,180 data points (11.6 MB) were used for surface reconstruction. The standard deviation of the sensor noise is 0.03 mm (10% of the sampling distance), the mean registration error is 0.2 mm. On a machine with an Intel Pentium® II 300 processor, the surface reconstruction took 7 minutes allocating 22 MB of RAM. The resulting surface consists of 33,298 triangles (800 kB) and has a mean deviation of 0.07 mm from the original (unsmoothed) range data.

For a virtual catalogue of teeth for medical applications we measured a series of 26 human tooth models of twice the natural size [417]. Figure 8.10 shows a reconstructed canine tooth and a molar. Ten views (between 15 resp. 13 MB) were processed. Surface reconstruction took 6 resp. 7 minutes.

For the 'Germanisches Nationalmuseum' in Nürnberg we measured parts of a statue of Saint George (15th Century). The results are used to improve the restoration by virtually visualizing each step. The 3D data allow to distinguish between relief and painted structures (texture) of the surface. For the visitors of the museum it is

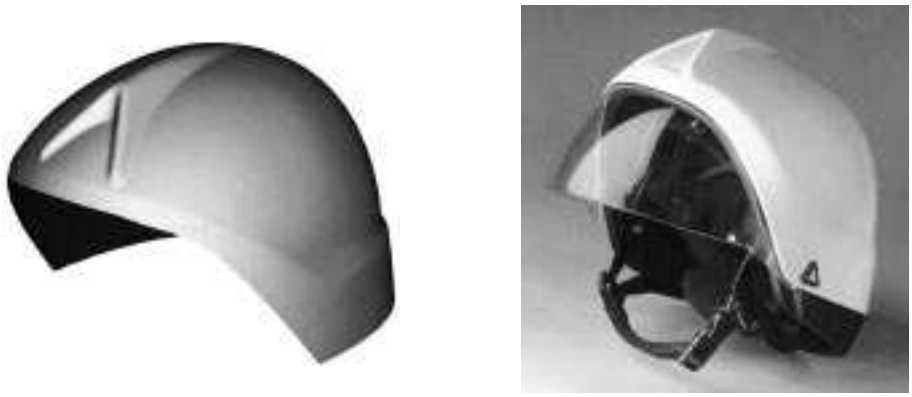


Figure 8.9. Reconstructed mesh of a fire fighter's helmet (left) and the helmet that was produced from the CAD data (right).

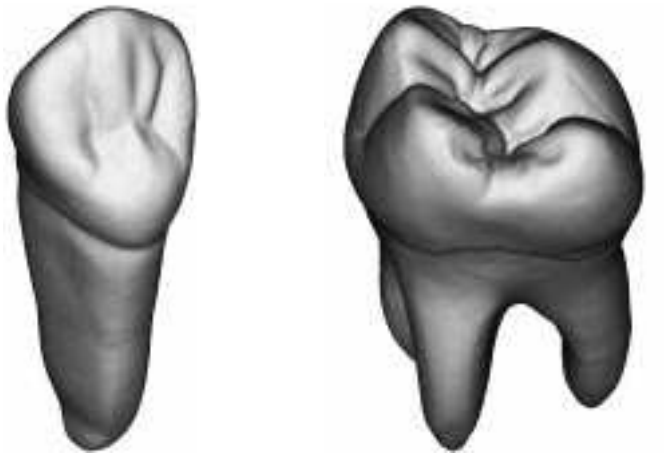


Figure 8.10. Reconstructed surfaces of plaster models of a human canine (left, 35,642 triangles, 870 kB) and a molar (right, 64,131 triangles, 1.5 MB).

interesting to watch a 3D simulation of the restoration process. Moreover, the obtained 3D model enables to view the statue in a virtual (e.g. ancient) environment. The 3D measurements were done using a phase measuring triangulation (pmt) sensor. Since the statue is partially very dark (e.g. the hair), 16 measurements for each view were

Figure 8.11. Reconstruction of the head of St. George: with plain surface (left) and with colored vertices (right).

overlayed. For each of altogether 26 views a normalized texture map was reconstructed. This was used to color the vertices of the final mesh (see Figure 8.11). In order to avoid loss of details, mesh thinning was carried out very carefully. The final mesh consists of 400,000 triangles. Hence, surface reconstruction took 6 hours. The final mesh approximates the raw sensor data with a mean deviation of 140μm.

Acknowledgement

Parts of this work were supported by 'Deutsche Forschungsgemeinschaft (DFG)' #1319/8-2 and by 'Studienstiftung des deutschen Volkes'.

8.2 SURFACE INTERROGATION IN AUTOMOTIVE DESIGN

G. GREINER

The competition on the global market forces car manufacturer to offer a great variety of different models. Moreover, redesign and updates of the models have to be done in shorter periods. The necessary speed-up of the development cycle of a car can be accomplished only by massive computer support. The slogan *digital car* has gained great popularity in the upper management of all car manufacturers. It expresses the desire to use CAx-technologies in order to achieve better quality, in less time and for less costs. In several areas of this complex process geometric modeling and computer graphics play a decisive role. For instance, car body design is based on free form surface modeling. In former time clay models have been used but for quite some time, the *master model*, which serves as reference for all steps in the construction is a digital model. It contains 3D-CAD-data of all components of the vehicle.

A historic fact is that many of the fundamental ideas of free form surface design have been developed in the sixties in the car industry. The theory of Bézier curves and surfaces (see Section 4.5) been established independently by de Casteljau at Citroën and Bézier at Renault. Coons at Ford and Gordon at General Motors used *transfinite interpolation methods* to interpolate a network of curves with fair surfaces. Extending the ideas of de Casteljau and Bézier naturally leads to B-splines and further on to non-uniform rational B-splines (NURBS), which are the de-facto standard of todays CAD systems [77, 215].

Free form surface design plays an important role in several stages of the development of a new car. It is particularly important in the design and construction of the car body. The latter can be subdivided in the following steps (see Figure 8.12):

Design. Stylists design new cars. At this stage none or only little computer support is used. Stylists still use traditional techniques, like hand drawings and modeling with clay.

Construction. Draftsmen use powerful computer hardware and sophisticated software to create a digital model. Reverse engineering techniques are used to build a 3D-CAD model of the stylists' concepts. In addition, fine detail as well as the overall appearance has to be checked for quality.

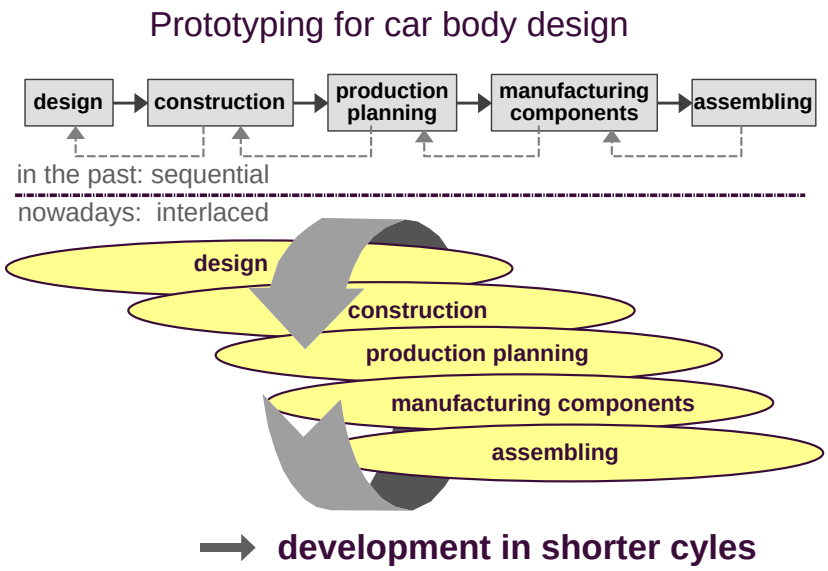


Figure 8.12. Car body: design and construction of prototypes.

Production planning. Possibility of manufacturing has to be checked, *digital mock-up* will widely replace *physical mock-up*. As a result of this stage, the master model is specified which is a reference for all further steps in the development and also later on in the regular production.

Manufacturing of components. At this stage geometric modeling is used for designing the tools that are necessary for the manufacturing.

Assembling of the components. Computer technology is used for process simulation. Finally prototypes are manufactured.

In order to speed-up the development cycle these steps are no longer executed sequentially but interlaced. This requires a lot of interaction between the different groups. Fast and meaningful visualization of the partial steps is indispensable. Moreover, adequate measures for checking quality and advanced tools for optimizing components are necessary. In the sequel we describe one of these procedures in more detail. In automotive industry there is a long tradition to judge the quality of the car body by inspecting reflection lines on the object. Formerly these lines were generated by placing the real object in front of a set of parallel light lines. Nowadays, reflection lines are simulated on the computer and then will be mapped as texture during display of the 3D-CAD model.

Besides reflection lines (see [388, 380] for more details), there are several other methods for surface interrogation, e.g. displaying color encoded curvatures on the surface [183], focal surfaces [271], isolines of curvatures [514] and isophotes (isolines of brightness) [548].

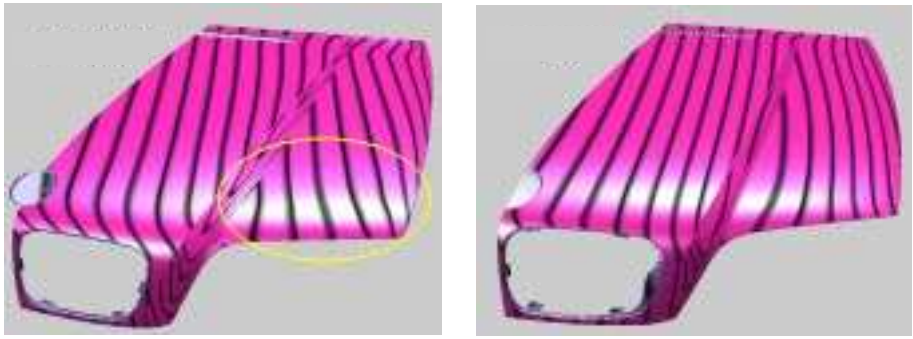


Figure 8.13. Part of a car (front hood): the right version has a more appealing reflection lines pattern.

Reflection lines are very popular in automotive industry probably because they are familiar to the designer from 'real' world experiments. Moreover, they are very sensitive to small shape deviations and, last but not least, modern graphics hardware allows rendering of reflection lines at interactive rates. In Figure 8.13 two version of a front hood are shown. The reflection lines are pre-calculated and then mapped by environment texture (see Section 5.3) on the Gouraud-shaded geometric model. Reflection lines are created by simulating the reflection of a family of parallel light lines in 3D-space at the surface. They are viewer dependent and assume a specular reflecting surface.

Due to the popularity of reflection lines, there is a need for a surface design tool that directly operates on these features, allowing the user to specify the desired lines and automatically generate the corresponding surface shape. In this section we outline a method which has been presented in [451] and constructs a surface geometry for a given reflection line pattern. Exact matching to the user specified data is in most cases inherently impossible (see [15, 16]). Therefore one tries to find an approximation which comes as close as possible to the desired surface shape.

8.2.1 Determination of Reflection Lines

For simplicity we assume that the surface F under consideration (describing a component of the car) is represented as a graph of a function, i.e. $F(u, v) = (u, v, f(u, v))$. Reflection lines are viewer dependent. Therefore we have to fix a viewer position. For simplicity, we assume that viewer and light source are located at infinity. Let $\boldsymbol{s}$ be the vector pointing towards the viewer.

An eye ray emerging from the viewer is reflected at the surface into a direction $\boldsymbol{r}$. Since the viewer position lies in infinity, $\boldsymbol{s}$ is constant. Thus $\boldsymbol{r}$ depends only on the surface normal $\boldsymbol{n}$ at the point of reflection (see Figure 8.14). A simple calculation shows

$$\boldsymbol{r} = 2\langle \boldsymbol{s} | \boldsymbol{n} \rangle \boldsymbol{n} - \boldsymbol{s} \tag{8.3}$$

where we assume $\|\boldsymbol{n}\| = \|\boldsymbol{s}\| = 1$ hence $\|\boldsymbol{r}\| = 1$ as well.

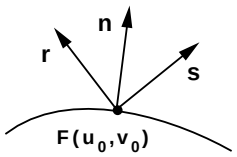


Figure 8.14. Reflection of the eye ray.

The light line which is 'seen' by $\boldsymbol{r}$ depends on the kind of light source and its orientation in space. There are two types of light source, which we refer to as *axial* or *radial* light lines. The axial light source is a family of straight (light)-lines, all parallel to a vector $\boldsymbol{z} \in \mathbb{R}^3$, which determines the orientation. Moving these lines to infinity (perpendicular to $\boldsymbol{z}$), the projection onto the unit sphere assigns a longitude to each light line (see Figure 8.15). By completing $\boldsymbol{z}$ to an orthonormal coordinate system $\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$ we can parameterize the longitudes from 0 to 2π as shown in the left image of Figure 8.15. Thus, every light line is determined by a unique angle $\phi \in [0, 2\pi[$. The radial light source is composed of circles in infinity (however, we still call them light 'lines'), all circles being perpendicular to $\boldsymbol{z}$. The projection onto the sphere maps them onto latitudes, which we number from 0 to π (right image in Figure 8.15). In this case each light line is determined uniquely by an angle $\theta \in]0, \pi[$. A different interpretation of this situation is the following statement: Light lines are isolines of the scalar function $\phi = \phi(u, v)$ and $\theta = \theta(u, v)$ respectively. These functions are obtained by inserting Equation 8.3 into the following identities

$$\tan\phi = \frac{\langle \boldsymbol{r}|\boldsymbol{y}\rangle}{\langle \boldsymbol{r}|\boldsymbol{x}\rangle} \quad \text{or} \quad \phi = \arctan\frac{\langle \boldsymbol{r}|\boldsymbol{y}\rangle}{\langle \boldsymbol{r}|\boldsymbol{x}\rangle}$$

and

$$\cos\theta = \langle \boldsymbol{r}|\boldsymbol{z}\rangle \quad \text{or} \quad \theta = \arccos\langle \boldsymbol{r}|\boldsymbol{z}\rangle \ .$$

These functions are the *height field of reflection lines* subsequently abbreviated by HFR. For a given surface F and specified viewing direction s, the HFR is considered as

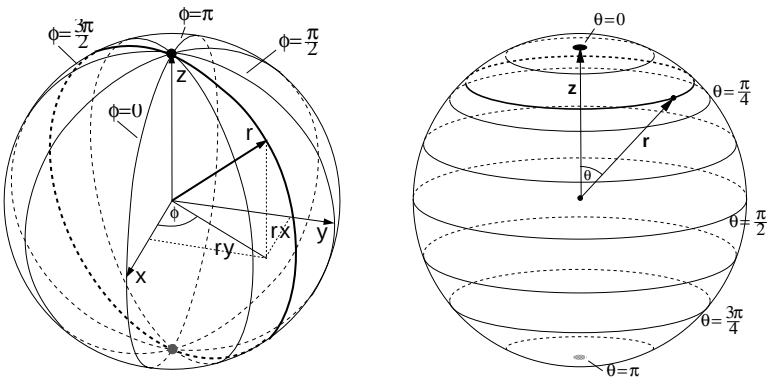


Figure 8.15. Model for axial (left) and radial (right) sources.

function defined on the parametric domain of F. To express specifically the dependence on F we will subsequently mark the HFR with a subscript $_F$, i.e. $I_F = I_F(u, v)$.

In order to manipulate reflection lines or to fit a geometry to a prescribed pattern of reflection lines we only consider the HFR. We outline the procedure by considering two examples: Fairing of reflection lines and surface reconstruction based on reflection lines.

8.2.2 Fairing of Reflection Lines.

If reflection lines are used as surface interrogation tool, it is more natural to perform the fairing procedure on the reflection lines or the corresponding HFR and to adapt the surface to the resulting HFR. The user smoothes the HFR first, obtaining better reflection lines while retaining the global look. Then the system computes the appropriate surface modification automatically. Figure 8.16 illustrates the fairing strategy. First the user specifies a region where the reflection lines and the surface should be faired. The outside region remains fixed.

For a given geometry F and a specified viewing direction s we proceed as follows:

- Compute the HFR of the surface F, thus obtaining a scalar function I_F. In order to obtain a B-spline representation of I_F, we evaluate I_F on a regular grid and interpolate the sampled data (see Section 4.6).
- Smooth the HFR I_F by a standard procedure, e.g. by minimizing an energy functional $J_{HFR}(I)$.

$$J_{HFR}(I) = \alpha J_{fair}(I) + (1 - \alpha) J_{fit}(I) \ ,$$

where J_{fair} is a simple quadratic fairing functional, e.g.

$$J_{fair}(I) = \int I_{uu}^2 + 2I_{uv}^2 + I_{vv}^2 \ .$$

The functional

$$J_{fit}(I) = \int (I - I_F)^2$$

measures the deviation of the new HFR obtained by minimizing J_{HFR} to the existing one. The weight $\alpha \in [0, 1]$ allows to control to which extent the lines should be smoothed. The result will be a scalar function I^*.

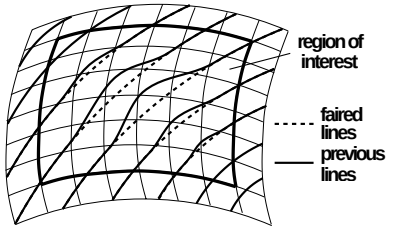


Figure 8.16. Local fairing of reflection lines for surface.

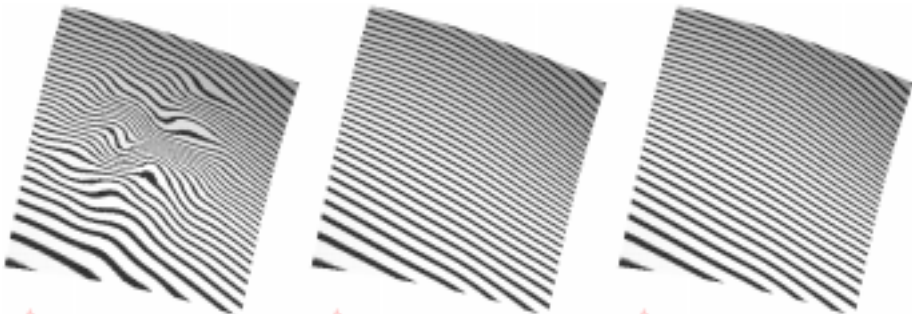


Figure 8.17. Initial surface (left), faired lines (middle), faired surface (right).

- Determine a new surface F^* such that the HFR I_{F^*} satisfies $I_{F^*} \approx I^*$. This will also be achieved by minimizing an appropriate error functional. Good results will be obtained using the following functional

$$\begin{aligned} J(F) &= \int (\mathrm{grad} I_F - \mathrm{grad} I^*)^2 \\ &= \int \left((I_F)_u - (I^*)_u\right)^2 + \left((I_F)_v - (I^*)_v\right)^2. \end{aligned} \tag{8.4}$$

  We additionally impose F^* to interpolate at the boundary the derivatives up to order two of the initial surface F. This ensures that the reflection lines of F^* connect tangent continuous to the lines in the outside region.

An example for the described fairing method is given in Figure 8.17. The left picture shows the initial surface with obvious curvature perturbations. The picture in the middle shows the smoothed lines, being texture mapped onto the initial surface. The right picture shows the faired surface and its reflection line pattern, which has been computed by approximating the line pattern of the middle picture.

The reason for incorporating gradient deviation in the error functional in Equation 8.4 is the following. Isolines of a scalar function (which represent the reflection lines in our situation) are closely related to the gradient of the function: isolines and gradient are orthogonal at every point. Thus in order to make sure that the reflection lines of the resulting surface F^* closely follow the faired isolines of F the gradient deviation of the corresponding height field should be minimal.

8.2.3 Surface Reconstruction

In the previous example we tried to interpolate reflection lines which may not correspond to a surface geometry. We had to find an approximation. If however, a set of reflection lines, which is created from an existing surface is used as input, the algorithm is capable to reconstruct the original surface. Since mechanical measurements often do not provide satisfying accuracy, it seems reasonable to use optical measuring based

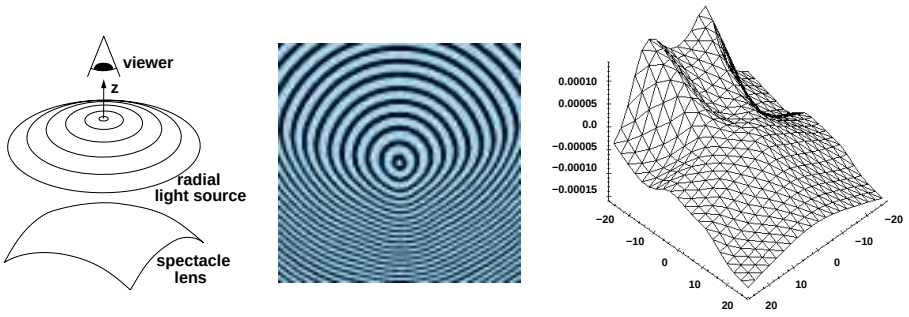


Figure 8.18. Configuration of radial light lines and viewing position (left), reflection line pattern of a progressive lens (middle), error plot (right).

on specular reflection. As an example (see Figure 8.18) the geometry of a progressive addition lens is reconstructed. We used this type of lens, because it has an increasing mean curvature along its vertical axis, and therefore exhibits a more interesting reflection pattern than uni-focal lenses. As input we computed the HFR from CAD data of such a lens (see [465, 450]).

The arrangement of camera, light lines and lens is illustrated in the left image of Figure 8.18. The orientation of the radial light lines is given by the direction $\boldsymbol{z} = (0, 0, 1)$, which also points to viewer position. The arrangement is adopted from Halstead et.al. [278], where it was used to reconstruct the topography of a human cornea. The reconstruction algorithm used there is based on fitting a set of normal vectors which are obtained by backward ray tracing. We use the following strategy to reconstruct the geometry of the lens.

- For the known HFR I we determine a surface F such that the error measured by the functional

$$J(F) = \int (\mathrm{grad} I_F - \mathrm{grad} I^*)^2$$

  is minimal.

The error plot in the right image of Figure 8.18 shows the difference between the original lens geometry and the surface which was reconstructed by adaption to the reflection line pattern shown in the middle image.

8.3 FLUID DYNAMICS

F. SCHÄFER, M. BREUER, C. TEITZEL

In Sections 6.4 and 6.5 of this book, methods for the visualization of three-dimensional scalar data (volume data) and vector fields have been described. An important field of application of these methods is the visualization of fluid dynamical data, which can be obtained by computer simulations of flows or detailed flow measurements.

Fluid dynamical data can contain three-dimensional, time-dependent scalar fields as well as vector fields and second-order tensor fields (e.g. pressure, velocity and stress, respectively). Hence, a variety of different methods is needed for the visualization of these different types of flow data.

In the following we want to indicate exemplarily how data visualization can be used in the field of fluid dynamics. We use the term 'data visualization' to distinguish the subject of this section from 'experimental visualization', where, for example, smoke is photographed while being advected in a flow. In Section 8.3.1 the purpose of data visualization in fluid dynamics is discussed, followed by a brief summary of data visualization techniques suitable for flow field visualization (see Section 8.3.2). The application of selected techniques to two different flow problems of actual research will be discussed in Sections 8.3.3 and 8.3.4. Here the main focus will be on computational fluid dynamics, since the spatial and temporal resolution of the flow data is usually higher for simulations than for measurements. But all of the visualization techniques are, in principle, applicable to experimental data as well, provided that the data resolution in space and time is sufficiently high.

To reveal the time-dependent behavior of fluid flows, animations are especially important. For the stirred vessel example discussed in Section 8.3.4 video sequences are available in the world wide web. Details about the online resources can be found in Section 8.3.5.

8.3.1 Purpose of Data Visualization in Fluid Dynamics

The progress in computational fluid dynamics (CFD) and experimental measuring techniques has led to increasingly detailed information about the investigated flow fields such as velocity, pressure, temperature and more. Experimental methods such as laser-Doppler anemometry [193], hot-wire anemometry [95] and particle-image velocimetry [3] have been developed allowing detailed measurements of the velocity field in a flow. Moreover, by solving the governing equations of fluid dynamics numerically, CFD can provide values of the flow field variables at a high resolution in space and time [221, 321].

In particular for CFD this has been possible due to increasing computer power and the development of efficient numerical algorithms. The most general description of a fluid flow, on which computer simulations can be based, is obtained from the full system of Navier-Stokes equations expressing the conservation of mass, momentum and energy. These equations may be written as

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho \boldsymbol{v}) &= 0 \\
\frac{\partial (\rho \boldsymbol{v})}{\partial t} + \operatorname{div}(\rho \boldsymbol{v}\boldsymbol{v} + \boldsymbol{\tau}) &= -\operatorname{grad} p + \boldsymbol{F} \\
c_p \left(\frac{\partial (\rho T)}{\partial t} + \operatorname{div}(\rho \boldsymbol{v} T) \right) &= \operatorname{div}(k \operatorname{grad} T) + \frac{\partial p}{\partial t} + \boldsymbol{v} \operatorname{grad} p + S,
\end{aligned}$$

where ρ denotes the density, $\boldsymbol{v}$ the velocity, p the pressure, T the temperature, $\boldsymbol{\tau}$ the stress tensor, c_p the specific heat at constant pressure and k the thermal conductivity

of the fluid. $\boldsymbol{F}$ is a volume specific force acting on the fluid (for instance gravity), and S is a volume specific source term for heat energy. Additional equations may be formulated for further scalar quantities such as the concentration of chemical species.

In addition to the Navier-Stokes equations, also lower approximation levels are in use such as Euler equations or potential flow models. Regardless of the underlying set of equations, numerical flow simulations are generally based on a spatial and temporal discretization of the computational domain, using different approaches such as finite elements, finite volumes or finite differences methods. In combination with different grid arrangements such as structured, block-structured and unstructered grids, this leads to a variety of different CFD methods. They all have in common that an approximate solution of the governing equations is determined. A special problem arises in the case of turbulent flows, since the resolution of the smallest vortices in the flow requires such a fine discretization in space and time that the direct numerical simulation (DNS) of the governing equations leads to prohibitive long computation times. To cope with this problem, often the Reynolds-averaged Navier-Stokes equations are used for the simulation of turbulent flows together with statistical turbulence models.

Depending on the flow problem and the computational method applied, a wide range of spatial and temporal discretizations can be used in CFD applications. Nowadays, three-dimensional grids containing 10^4 to more than 10^7 grid points are quite common, and the resulting output data sets contain just as many velocity values, pressure values, etc. The information output is even larger in the most general case of a time-dependent simulation, where for every time step such a voluminous data set is produced.

Having this in mind, the question arises how to get useful information from the columns of numbers produced by CFD applications and high resolution measurements. Of course, the numerical values of the calculated or measured flow field variables are important for comparison with theoretical data, as far as they are available, and for comparison between simulations and measurements. Furthermore, integral quantities such as drag or lift coefficients can be derived from the flow field data, leaving details of the flow information unconsidered. But the trend is to analyse flow field data in more detail, and if one wants to *understand* the physical behavior of the flow, pure columns of numbers are not very useful. Here visualization offers the possibility for a detailed exploration of flow field data. By using the visual pattern recognition facility of the human brain, visualization can provide a deeper insight into the dynamical behavior of fluid flows. Therefore, visualization is an important research tool for improving our physical understanding of fluid flows and for communicating these physical insights within the scientific community [162].

8.3.2 Data Visualization Methods

The field variables that are of interest in fluid dynamics, such as pressure, temperature, velocity and stress, are quite different with respect to their information content, since they include scalar, vector and second-order tensor quantities. Going from scalar to vector and from vector to tensor fields, there is an increase of information contained in the data, and its visualization becomes more and more sophisticated. Accordingly, there is a need for different visualization techniques depending on the kind and amount of information that has to be visualized.

The following list is a compilation of methods suitable for the visualization of scalar, vector and second-order tensor fields. It is certainly not complete, but it is thought to contain the most important techniques used for flow field visualization. For details about the methods we refer to Sections 6.4 and 6.5 in this book and to the cited references. A survey of visualization methods frequently used in fluid dynamics is also given in [555].

- Visualization of scalar fields
  - Contour plots [589, 99]
  - Isosurfaces [452]
  - Volume rendering [433]
  - Color coding (see Section 6.5.3)
- Visualization of vector fields
  - Arrow plots (see Section 6.5.1)
  - Glyphs [426]
  - Line integral convolution (LIC) [103]
  - Particle tracing
    - Basic trace types: stream lines, streak lines, path lines, time lines [23, 418]
    - Variants: lines, bands [51], tubes [177]; time surfaces [649]; stream surfaces [347]; flow volumes [475]
  - Vector field topology [316]
- Visualization of second-order tensor fields
  - Hyperstreamlines [165]

It is worth noticing that some methods for vector field visualization such as glyphs, stream bands, stream surfaces and flow volumes are also capable of displaying information about tensor quantities (for instance shear). Depending on the application this may be sufficient, but the more specialized methods for tensor field visualization are more powerful.

Whatever method is used for visualization, special attention has to be paid to the accurate and careful interpretation of the visualization results. One always has to bear in mind what kind of information is displayed by a specific visualization method, otherwise the obtained images can be misleading. For example, stream lines are well suited for depicting the fluid's path of motion in a steady flow, but applied to an unsteady flow only the local flow directions are displayed. The same is true for other methods generating a representation of the instantaneous velocity field (e.g. LIC, arrow plots). Other visualization methods have their specific limitations as well. Nevertheless, they can be useful even in cases where their limitations take effect, as long as the restrictions are taken into account during interpretation.

8.3.3 Simulated Flow in a Melting Crucible

The first application example we want to discuss is the flow in a melting crucible filled with pure silicon. The crucible is part of an arrangement used for the synthesis of monocrystalline silicon according to the Czochralski process, which is the most popular method in the semiconductor industry for producing large crystals for electronic and photonic devices. In this process a rotating seed crystal is placed on top of the melt's surface so that an almost cylindrical single crystal starts growing and can be dragged from the melt. The quality of this crystal depends on many parameters, in particular on those determining the silicon flow in the crucible. In order to investigate the influence of different parameters on the flow, three-dimensional time-dependent simulations of the turbulent flow and heat transfer in the crucible have been performed within the scope of a research project [45, 209]. The calculation considered here is a direct numerical simulation using a very fine discretization in space and time. Thus, we can expect that very small flow structures are resolved by the simulation.

For visualization we have selected a data set representing the flow field after a simulated time of two minutes, with zero velocity as initial condition. Besides Coriolis and centrifugal forces, the flow in the crucible is driven by buoyancy forces due to the temperature gradients in the heated melt. From the contour plot of the temperature field in Figure 8.19, it can be seen that there is an increase in temperature from the inner top of the crucible to the outer bottom. The reason for this temperature profile is twofold. First, the crucible is heated radiatively from the sides and the vertical heating elements extend below the crucible. This causes that the highest temperature is reached at the outer bottom. On the other hand, the cylindrical crystal is placed at the inner top of the crucible so that the fluid is cooled down there to allow crystallization of the silicon. The diameter of the single crystal is almost 60 percent of the crucible diameter, and its 'fingerprint' can be seen clearly in the contour plot.

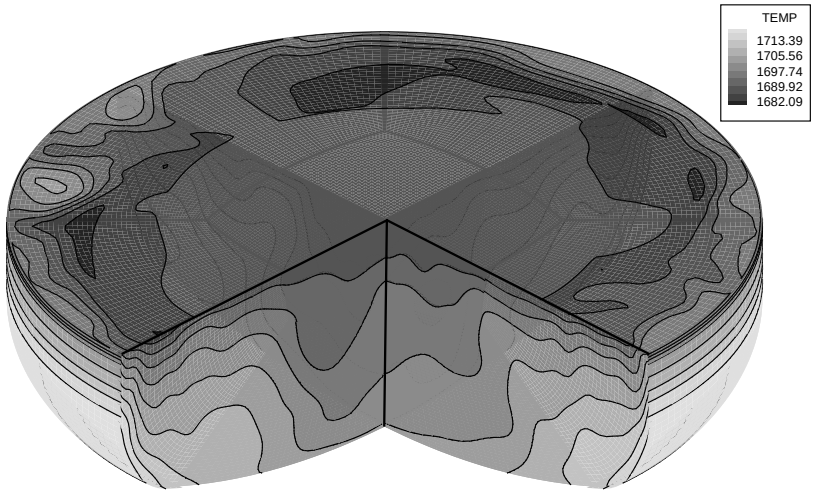


Figure 8.19. Contour plot of the temperature field in the crucible.

Figure 8.20. Line integral convolution applied to the turbulent flow in the crucible: LIC of the velocity field in a vertical cutting plane.

We can get an idea of the fluid flow from the LIC image in Figure 8.20, which is a visualization of the velocity field in a vertical cutting plane (compare Section 6.5.1 for a description of the method). It gives a very detailed impression of the instantaneous velocity field parallel to the plane without regarding the velocity component perpendicular to the plane or the evolution of the velocity field in time. However, it can be observed that there are very small vortical structures in the flow, as we had to expect for the direct numerical simulation. To take a closer look at the flow field, a magnified clipping of the cutting plane is depicted in Figure 8.21. Three different visualization techniques have been applied to the velocity field: LIC, arrow plot, and stream lines restricted to the cutting plane. Again, the LIC image shows very clearly the structures in the flow, whereas the magnitude and direction of the velocity cannot be determined. On the other hand, the arrow plot is particular good at depicting the magnitude and direction of the velocity components parallel to the plane, whereas the flow structures cannot be recognized as good as in the LIC image. Hence, combining LIC and arrow plot can give a good idea of the instantaneous velocity field in the cutting plane.

The stream line plot in Figure 8.21 is also capable of displaying the main flow structures in the plane, but it is much less detailed than the LIC image. This is because only a few stream lines can be used in order to avoid cluttering of the plot. Since the starting points of the stream lines have to be selected manually, small flow structures are only found by chance and can be missed easily.

Taking a closer look at the arrow plot of the velocity and the contour plot of the temperature in Figure 8.21, the interaction between both fields can be understood. Most clearly the upwards moving stream along the right wall of the crucible is caused by buoyancy due to the high temperature in this region compared to the surrounding, confirming that the flow is partly driven by temperature gradients. On the other hand, the flow is carrying thermal energy along and is likely to deform the temperature contour lines in this region to an S-like shape.

Concluding Figure 8.21, this example shows the advantages and disadvantages of three different techniques for the visualization of the velocity field in a cutting plane. In order to extract more information from the data, it is often useful to combine different visualization techniques revealing different information about the field quantities and their interaction. Furthermore, it is found that the two-dimensional cutting plane

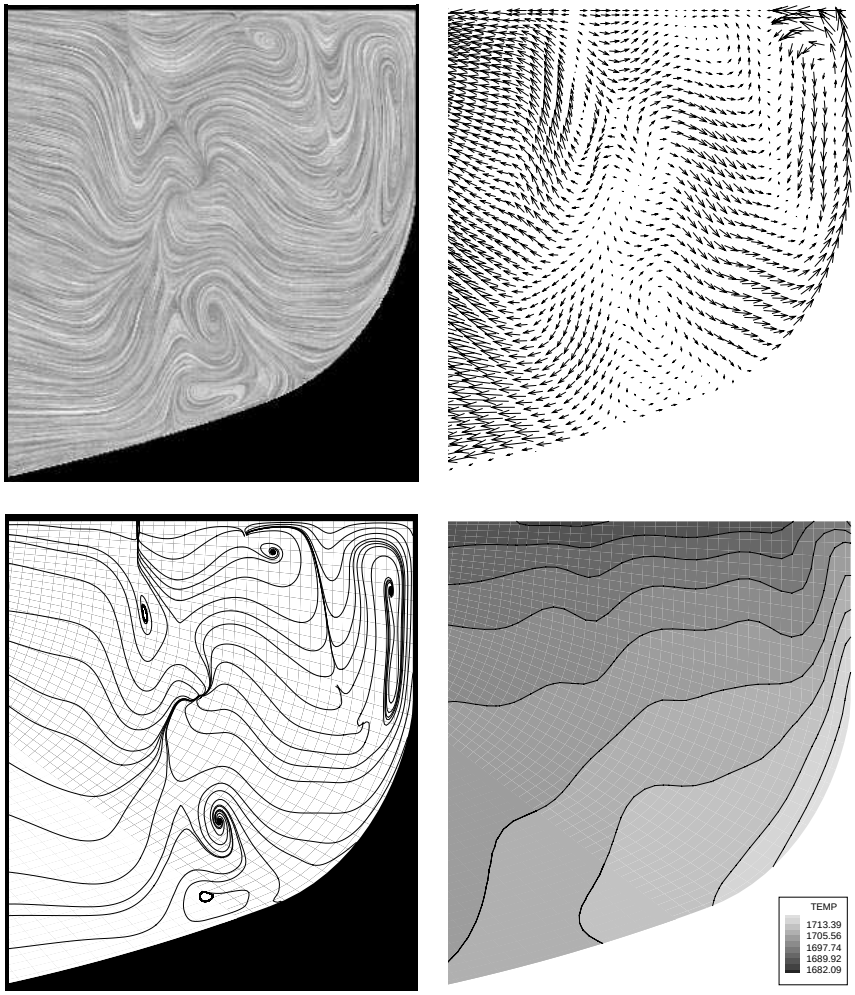


Figure 8.21. Visualization of the velocity and the temperature field in the crucible. The region shown is a magnified clipping of the cutting plane in Figure 8.20. Top left: LIC of the velocity field. Top right: velocity field depicted as an arrow plot. The arrow length is proportional to the magnitude of the velocity component parallel to the plane. Bottom left: 2D stream lines restricted to the cutting plane. Bottom right: contour plot of the temperature.

visualization is able to resolve very small vortical structures in the three-dimensional flow. Similar investigations can be made to examine the influence of the velocity component perpendicular to the cutting plane and the evolution of the vortical structures in time. This can be done by using several cutting planes with data sets representing different time steps. However, there are more specialized methods for the visualization of time-dependent flows in three space dimensions, as will be shown in the next example.

8.3.4 Simulated and Measured Flow in a Stirred Vessel

Impeller stirred reactors play an important role for production processes in the chemical and process engineering industry. In order to permit reliable process optimization, a lot of experimental and numerical research studies were carried out to increase the understanding of the mixing process in stirred vessels. Owing to several problems encountered in experimental investigations, CFD has gained increasing interest during the past few years.

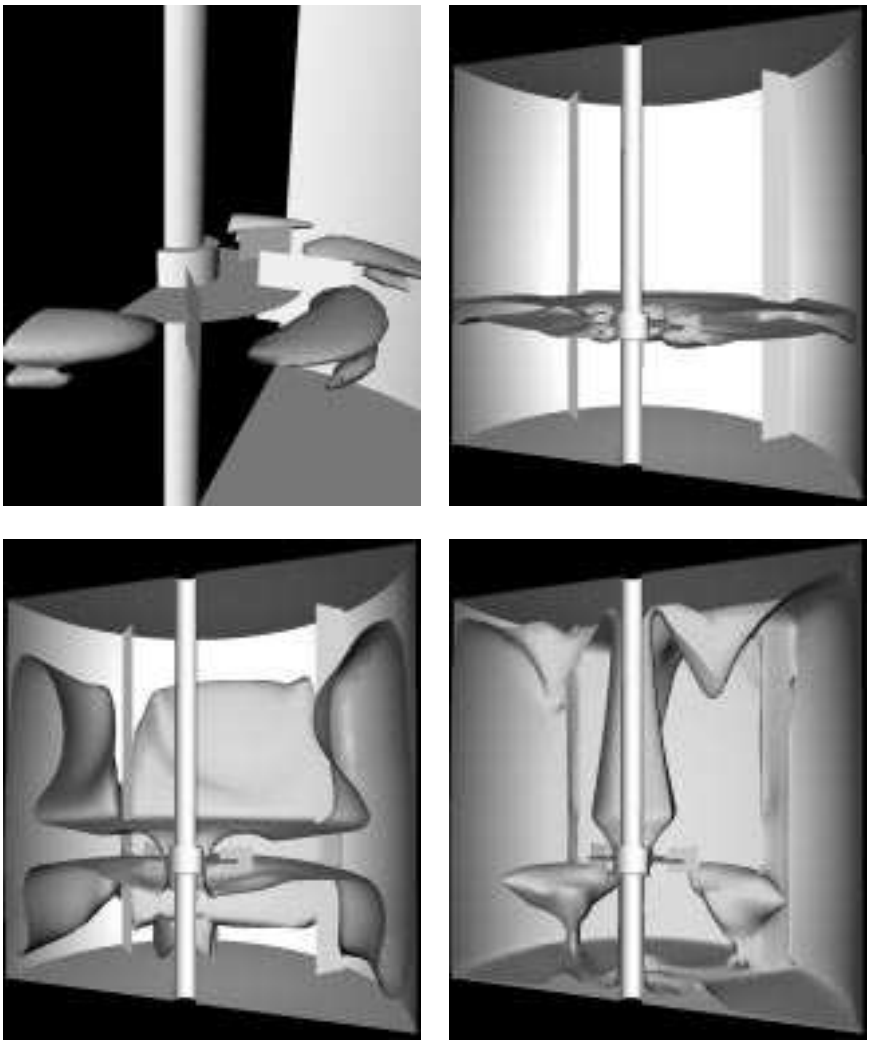


Figure 8.22. Isosurfaces of the turbulent kinetic energy for different isovalues (4, 1, 0.25, 0.1 J/kg, from top left to bottom right, respectively). Flow in a vessel stirred by a Rushton turbine (only half of the vessel is depicted).

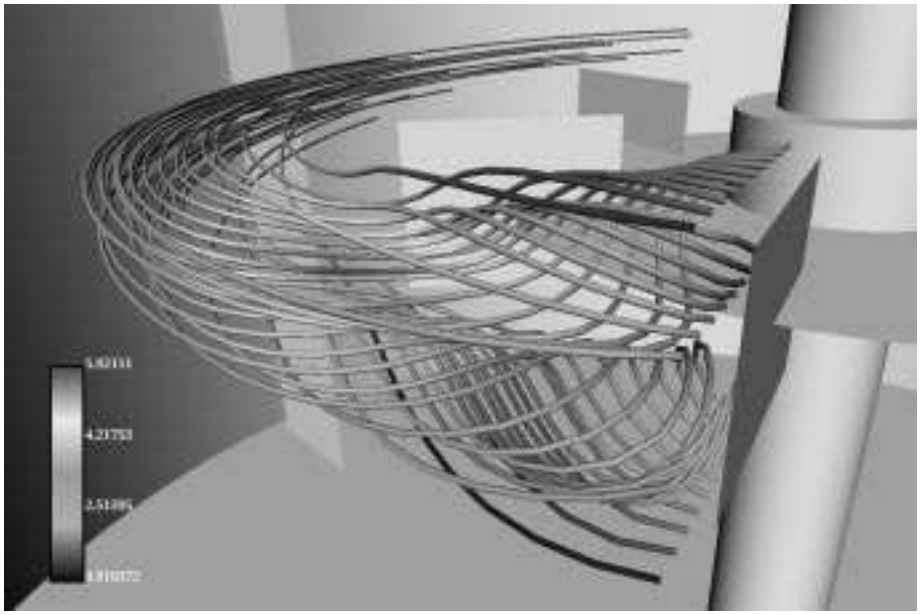


Figure 8.23. Streak lines starting behind a blade of the Rushton turbine (the impeller is rotating counterclockwise). Color coding has been used to colorize the lines according to the turbulent kinetic energy at the particle positions. The lines are represented by pipes in order to obtain a three-dimensional depth cue.

The flow field in a stirred vessel is complex and three-dimensional with vortical structures and high turbulence levels in the vicinity of the impeller requiring large computational efforts. However, based on efficient numerical algorithms and fast supercomputers, it is nowadays possible to predict the flow in the vessel accurately. Within a joint European research programme detailed numerical and experimental investigations have been carried out for different impeller types [603, 604, 739]. The vessel considered here for visualization is stirred by a Rushton turbine, i.e. the impeller consists of a disc carrying six vertical blades.

The underlying numerical simulations of the vessel flow are three-dimensional and time-dependent. They are based on the Reynolds-averaged Navier-Stokes equations combined with a statistical two-equation turbulence model, which does not resolve the small scale vortices but introduces the so-called turbulent kinetic energy and the dissipation to represent their kinetic energy content and their dissipation rate, respectively. Since turbulence comes along with a very efficient mixing of the fluid, the turbulent kinetic energy is of great interest for the investigation of the flow in the stirred vessel. Isovalue surfaces of constant turbulent kinetic energy are a very powerful tool for exploring the turbulence distribution in the vessel. Figure 8.22 shows a sequence of isosurfaces for different isovalues of the turbulent kinetic energy. As it can be seen from the first image, the highest values occur at a short distance behind the blades of the Rushton turbine. Furthermore, the distribution of the turbulent kinetic

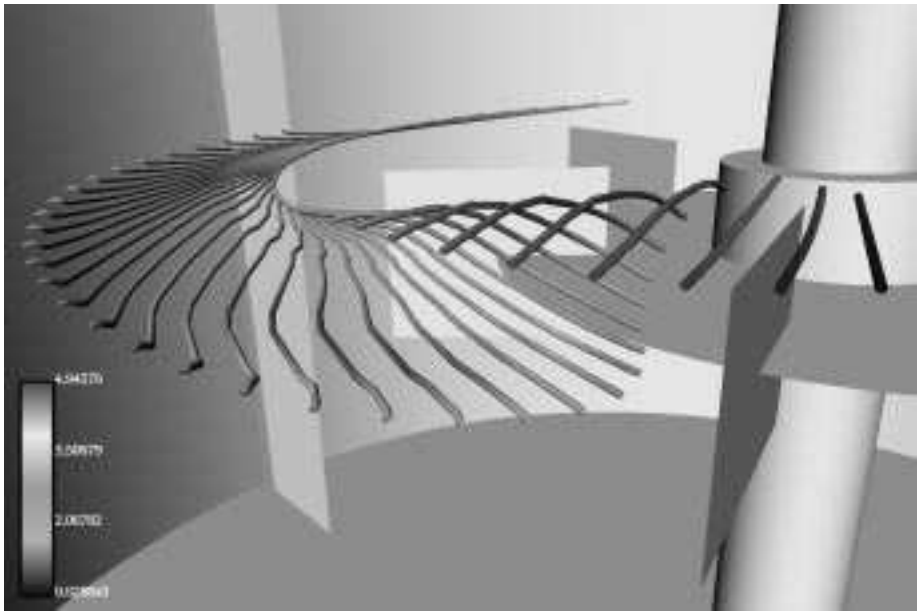


Figure 8.24. Time lines starting in front of a blade of the Rushton turbine. Color coding according to the turbulent kinetic energy.

energy is clearly asymmetric with respect to the horizontal impeller plane. Below the plane there is only a small volume enclosed by the isosurface, indicating that the turbulence level is lower than above the plane. Going to a four times lower value of the turbulent kinetic energy, the isosurface encloses almost the complete vessel cross-section in the height of the impeller. Just the innermost region between the blades and the rotating axle is spared out. Decreasing the isovalue by a factor of four again, the isosurface extends upwards and downwards along the vessel walls. Only for even smaller isovalues the isosurface reaches out into the outer regions of the vessel volume. As a result, the isosurface sequence shows that the turbulent fluid motion is concentrated more or less in the horizontal plane around the Rushton turbine.

An investigation of the three-dimensional time-dependent velocity field can be performed by using a particle tracing technique such as streak lines or time lines. Streak lines are generated by starting particles at each time step at some fixed seed positions and connecting all particles coming from the same seed position. Since this visualization method is time-dependent, animations are most powerful. However, for animations we must refer to our web-pages (see Section 8.3.5), since here we can only show single snapshots depicting the streak lines at one instant in time. This is the case in Figure 8.23, with the seed positions located behind an impeller blade. In order to investigate the vortices being dragged along with the blade, the seed positions are fixed with respect to the blade, i.e. they are rotating with the same angular velocity. The streak lines reveal two vortices behind the blade, one above and one below the impeller plane. This vortical region is restricted to a relatively small distance behind

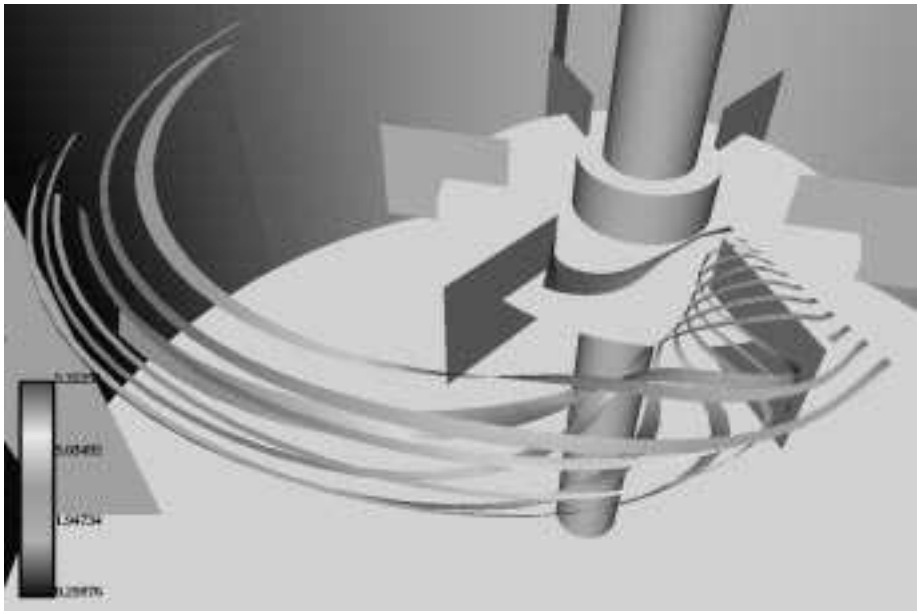


Figure 8.25. Streak bands starting in front of a blade of the Rushton turbine. Color coding according to the turbulent kinetic energy.

the blade. In addition, it can be seen that the streak lines are converging towards the impeller plane when passing through the vortices.

To take a more detailed look at the vortex region trailing the blade, a time line visualization is shown in Figure 8.24. Time lines are constructed by connecting the particles released at the same time step. To avoid cluttering of the image we restrict ourselves to the upper vortex by starting the particles only above the impeller plane. In the vortex region the time lines are stretched and rotated while their front end is turned to the back and the back end to the front. Furthermore, the time lines are moving vertically towards the impeller plane. We will come back to this vertical movement later, since it is part of a large scale vortex in the vessel.

Another possibility to reveal rotation in the flow is shown in Figure 8.25. The streak bands in this image are contructed by connecting two neighboring streak lines. When passing through the vortex behind the impeller blade, the bands are twisted and stretched, in accordance with the time line visualization in Figure 8.24. Moreover, from this perspective it can be seen very clearly that the particles are moving in a spiral outwards to the vessel wall after leaving the vortex region.

Besides the small vortices trailing the blades also a large scale vortical motion exists in the vessel. As has been shown in Figures 8.23 and 8.24, the fluid is moving vertically towards the impeller plane when passing through the vortices behind the blades. In addition, Figure 8.25 has demonstrated that there is a trailing spiral behind the vortex region. Both movements are part of a large scale flow, which can be seen in the LIC image in Figure 8.26. This is a velocity field visualization in a vertical cutting

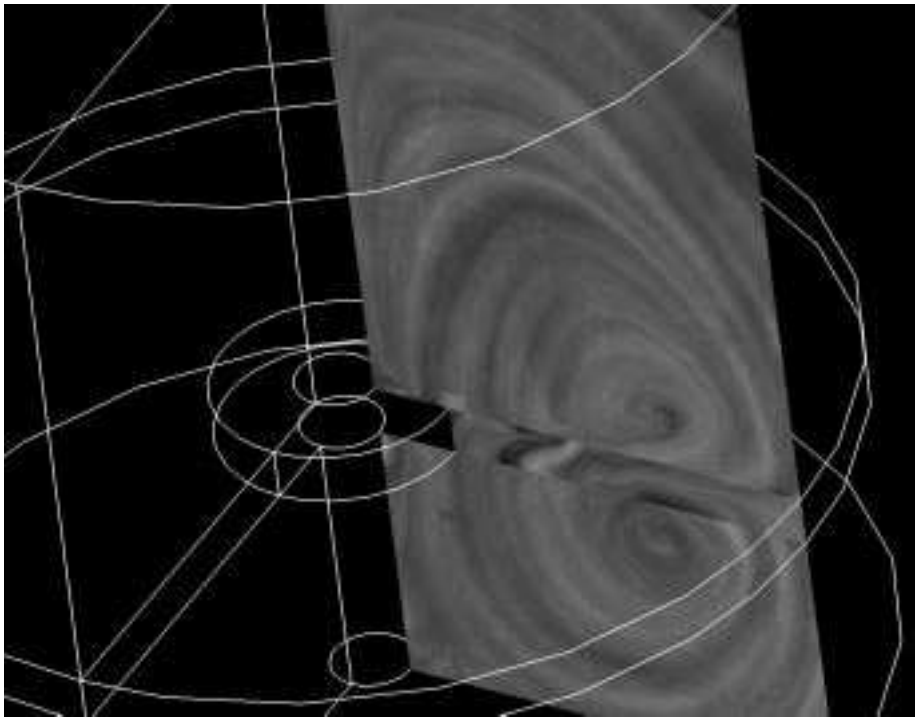


Figure 8.26. Line integral convolution applied to experimental velocity data obtained for the stirred vessel [689]. The white lines are representing the vessel geometry. The impeller is indicated by the white circles next to the gap in the LIC image. There are discontinuities between the middle and the adjacent top and bottom parts of the LIC image, because the LIC algorithm has been applied to the three parts independently.

plane of the vessel based on experimental data. The velocity has been measured using laser-Doppler anemometry. There are two large vortices filling the vessel volume, one above the impeller plane and one below. These vortices can be found in CFD data as well.

To conclude this example, the isosurface visualization has proved to be powerful for the investigation of the spatial distribution of some scalar field in the vessel. Some main flow structures have been revealed by using particle tracing methods in combination with line integral convolution applied to CFD data and experimental data, respectively. However, we want to emphasize that time-dependent visualization methods such as streak lines, streak bands and time lines are most powerful when they are animated.

8.3.5 Online Resources

For the stirred vessel example discussed in Section 8.3.4, video sequences are available in the world wide web (http://www.lstm.uni-erlangen.de/SFB603_C3/applications/). All images presented in this section can be found there as well, a colored version is

provided where available. Moreover, links have been added to lead you to further material created by our and other research groups.

Acknowledgements

Without data there is no data visualization. We would like to thank the following people for making the data sets available to us which have been visualized in this section. The crucible simulation has been performed by Sven Enger at the Institute of Fluid Mechanics (LSTM), University of Erlangen-Nürnberg. Klaus Wechsler at LSTM provided the stirred vessel simulation data, which have been generated within the scope of the BRITE EURAM project "Chemical Reactor Modeling for Fast Exothermic and Mixing Sensitive Reactions". The experimental stirred vessel data have been provided by Marcus Schäfer at LSTM.

8.4 DIAGNOSIS SUPPORT OF PATIENTS WITH FACIAL PARESIS

A. GEBHARD, D. PAULUS

Computers are used more and more in medical applications. Well-known examples are the analysis of radiographs or MR images [536, 445]. A special field is the analysis of human faces and facial features. Already realized applications exist e.g. for face recognition [89, 90] or face analysis [594, 317].

With the presented system we consider the problem of diagnosis support of patients with facial paresis. Approximately 350 patients per year are registered in the Department of Otorhino-Larygology of the University of Erlangen with new occurrences of this type of paresis (cf. [763] as an overview). The current way to diagnose the paresis is a subjective judgement of the functionality of the face muscles by a physician. The patient performs mimic exercises such as closing the eyes or showing the teeth, while he is observed by the physician. The subjective observations of the physician are then graded by means of two medical indexing systems [343, 671].

One part of the rehabilitation of the patients is to perform specific mimic exercises with the face's musculature. The success of the therapy is also observed by a physician or educated clinic personal. For every observation the patient has to travel to the clinics. Furthermore, educated personal is needed for the diagnosis and rehabilitation observation of the paresis.

The applications of our system are on the one hand diagnosis support in the clinics. We want to improve the subjective judgments of a physician by objective measurements and numerical features from the face. On the other hand the supervision of the rehabilitation process of the patient will be enhanced by placing the system to the patient's home. The patient can use the system in a convenient environment and does not have to travel to the clinics. The patient does *not* have to pay for the more convenient observation environment by wearing any artificial markers inside the face and he is allowed to move in front of the system's camera.

We present two different approaches for the analysis of facial features which will measure the face during the performance of mimic exercises. The result is a value for every picture with the level of asymmetry of the eye and mouth region which can

be used to classify the facial paresis. The asymmetry of the face can be used as we handle just patients with paresis in one half of the face. This class of patients is the most frequent class. Patients with double sided pareses are approximately 1% of the face paresis patients.

This contribution is organized as follows: In Section 8.4.1 we compare the use of different data sources (2D vs. 3D images). A survey about related work on the field of facial image processing is given in Section 8.4.2. The localization of facial features will be described in detail in Section 8.4.3. It is followed by the description of the analysis approaches (see Section 8.4.4). In Section 8.4.5 we show how the analysis results can be used to classify the facial paresis inside the observed regions. Results are shown in Section 8.4.6. Finally, Section 8.4.7 will recapitulate the contents of this paper with the main results in a short way.

8.4.1 Sensor Selection

The analysis of facial images can be performed using either the frontal view in a 2D projection, or taking into account the 3D structure of the head. The first approach can be motivated by human perception. For human interaction and human interpretation of mimics, 2D information seems to be sufficient. We can clearly tell whether a person smiles, frowns, or whether his eyes fixate ours, even when we close one eye. Paralyzed parts in the face are less perceived when they are in the lateral part of the cheeks. In the following we concentrate on 2D images and use a model of the projected face.

Three-dimensional information on the face is available when stereo information is taken into account. However, it is disparity is difficult to estimate for facial images, since these images exhibit neither significant texture—except for areas with facial hair—for area based matching, nor many lines or prominent points. Traditional stereo algorithms on the face thus result in sparse or coarse depth data.

8.4.2 Related Work

In this section we give a survey of related areas of the field of face image processing.

Face Localization and Tracking. One of the basic components of many systems which process face images is a face localization module. Different approaches can be found in the literature. In the following we present exemplary two different approaches.

One approach was introduced by De Silva et al. [651]. A person is expected to sit in front of a homogenous background. A gray-level image of the head and the shoulders is captured. The edge strength representation of the image (cf. Section 3.1) is used to find an elliptic region containing a high amount of edge strength which is supposed to be the face.

Another approach was introduced by Oliver et al. [520] and bases on the segmentation of *blobs*. A blob is a "compact set of pixels that share a visual property that is not shared by the surrounding pixels" (see [520, p. 123]), and is thus a special kind of region segmentation (see Section 3.1.2). Oliver et al. use the normalized color as the visual property of a pixel. Every image coordinate is combined with color and brightness information by generating a vector $(i, j, \frac{r}{r+g+b}, \frac{g}{r+g+b})^T$. A model for skin

color was trained by thousands of samples that is valid for a broad spectrum of users. With an adaptive strategy, face color regions are grown on the complemented image data. The classification accuracy is close to 100%.

Face Recognition. There are two major classes of face recognition operations. On the one hand there are methods using geometrical features or template-matching and on the other hand there is the processing on grey-level information.

The techniques basing on features or templates where analyzed in detail by Brunelli and Poggio [91]. They show the extraction of features or the template-matching with the goal of recognition. The results achieved by template-matching were better.

A prominent technique which bases on the processing of grey-level information is the *Eigenface* approach (e.g. [717]). The basic idea of the Eigenface approach is to encode a face image, and compare one face encoding with a database of models encoded similar. The encoding is done in the following way: A face image of size $N \times N$ is interpreted as a point in an N^2-dimensional space. A set of training images of faces with similar overall configuration (e.g. face as the dominant image region) can be described by a relatively low dimensional subspace of the huge image space. Principal Component Analysis used with different images of one person gives vectors that best account for the distribution of face images within the entire image space. Those vectors are called *eigenfaces* (comp. the *eigenspaces* approach in Section 3.4.3).

For recognition, a new face image is transferred into its eigenface components and assigned to the eigenface with minimal distance.

Face Coding. The principal component representation of face images can also be used to encode faces, transmit the encoded information via a relative slow line, and decode and view then at the destination. Moghaddam and Pentland [485] propose this proceeding for the video telephony task.

Another model-based approach for this task was presented by Tao and Huang [684]. A basic articulation model can be influenced by articulation parameters such as facial action coding system (FACS) parameters of MPEG-4 facial animation parameters (FAPs).

Medical Applications. In [318] a system for the automatic diagnosis of craniofacial dysmorphic signs is presented. Different multi-layer perceptrons were trained with a training set of 31 images using back-propagation as learning algorithm. The classification of a face whether craniofacial dysmorphic signs were present or not was done with a correct classification rate of 95%.

8.4.3 Localization and Tracking of Faces and Face Features

In the application presented in this contribution facial asymmetries are judged by means of asymmetries in the eyes, nose and mouth regions. The approaches we chose for the analysis of the facial paresis base on the localization of the mentioned facial features and of their surrounding (in case of the eyes the surrounding covers eyebrows and zygomatic). In our system the localization is done with a parametric face model, which is fitted to the image data by an energy-minimization process.

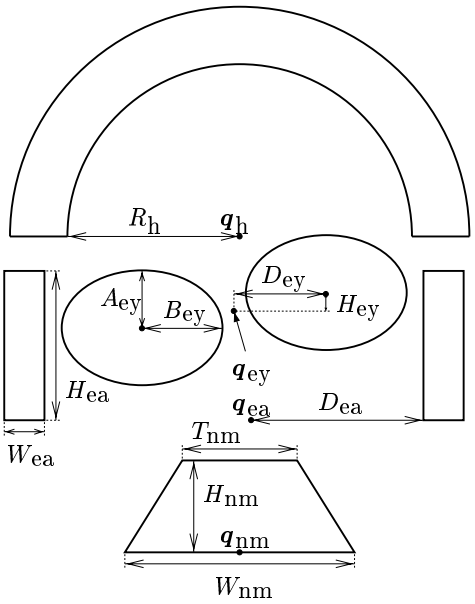


Figure 8.27. Parametric Face Model. All shown parameters are optimized during face localization.

Face Model. We assume that the patient's face is the dominant region inside the image. The background is expected to be either homogeneous or a background image is captured prior to the analysis which is used to part foreground from background. The localization is executed by calculating parameters of the face model shown in Figure 8.27. All calculations are performed on the edge strength part e_s of a Sobel filtered image $\boldsymbol{f}$ (see Section 3.1).

Localization. The localization is performed as a four-step process. The *first* step is to localize the upper arc of the head. A ring segment with fixed width C_R is found in the image which is expected to be the top of the patient's head. There are three parameters which have to be estimated: the x-coordinate x_h and the y-coordinate y_h of the origin $\boldsymbol{q}_\mathrm{h}$ and the radius R_h of the arc. We expect much edge energy between patient and background. That lets us find the parameters of the circle and of the following part of the model in the edge strength representation of the original image. Equation 8.5 gives the edge-energy inside the head arc in the image:

$$E_\mathrm{h} = \int_0^\pi \int_{R_\mathrm{h}}^{R_\mathrm{h}+C_\mathrm{R}} e_\mathrm{s}(x_\mathrm{h} + r\cos\phi, y_\mathrm{h} + r\sin\phi) dr d\phi. \tag{8.5}$$

To get the optimal model parameters the ratio of edge energy inside the head arc and the area of the head arc must be optimized. This optimization is done by the following

maximization process:

$$(x_{\mathrm{h}}^*, y_{\mathrm{h}}^*, R_{\mathrm{h}}^*) = \underset{(x_{\mathrm{h}}, y_{\mathrm{h}}, R_{\mathrm{h}})}{\operatorname{argmax}} \frac{E_{\mathrm{h}}}{\left(\frac{(R_{\mathrm{h}}+C_{\mathrm{R}})^2 - R_{\mathrm{h}}^2}{2}\right)^{C_{\mathrm{h}}}}. \tag{8.6}$$

The constant C_{h} influences the energy to area ratio. The values of the constants are $C_{\mathrm{R}} = 20$ and $C_{\mathrm{h}} = 1.4$.

The optimization is performed by an adaptive random search with a subsequent local simplex method [210]. To speed up the optimization there are restrictions for plausible parameters: The parameter x_{h} can vary form $N_{\mathrm{y}}/4$ to $3N_{\mathrm{y}}/4$, y_{h} form $M/4$ to $M/2$, and R_{h} from $N_{\mathrm{y}}/10$ to $N_{\mathrm{y}}/3$.

The *next* step is to localize the ears which are modeled as rectangles positioned below the arc of the head. The parameters which have to be determined are the position of the origin $\boldsymbol{q}_{\mathrm{ea}}$, the width W_{ea} and height H_{ea} of the ears, and the distance $2D_{\mathrm{ea}}$ between the two ears. The equation to calculate the edge-energy inside the ears' region is

$$\begin{aligned} E_{\mathrm{ea}} &= \int_0^{W_{\mathrm{ea}}} \int_0^{H_{\mathrm{ea}}} \{e_{\mathrm{s}}(x_{\mathrm{ea}} + w + D_{\mathrm{ea}}, y_{\mathrm{ea}} + h) \\ &\quad + e_{\mathrm{s}}(x_{\mathrm{ea}} + w - D_{\mathrm{ea}}, y_{\mathrm{ea}} + h)\} dh dw \end{aligned} \tag{8.7}$$

and similar to Equation 8.6 the calculation of the optimal ear parameters bases on the ratio of edge energy inside the ears regions and the area of this regions. They can be determined as

$$(x_{\mathrm{ea}}^*, y_{\mathrm{ea}}^*, D_{\mathrm{ea}}^*, W_{\mathrm{ea}}^*, H_{\mathrm{ea}}^*) = \underset{(x_{\mathrm{ea}}, y_{\mathrm{ea}}, D_{\mathrm{ea}}, W_{\mathrm{ea}}, H_{\mathrm{ea}})}{\operatorname{argmax}} \frac{(E_{\mathrm{ea}} D_{\mathrm{ea}})^{C_{\mathrm{ea}}}}{(W_{\mathrm{ea}} H_{\mathrm{ea}})} \tag{8.8}$$

with the ratio influencing constant $C_{\mathrm{ea}} = 1.4$. Additionally the distance between the left and right ear region is involved in the optimization as they are used as the horizontal boundaries of the face and therefore the distance D_{ea} should be as big as posible.

There are anatomic restrictions for the ear parameters. The origin of the ears $\boldsymbol{q}_{\mathrm{ea}}$ must have a lower horizontal distance than $R_{\mathrm{h}}/3$, the vertical position of $\boldsymbol{q}_{\mathrm{ea}}$ must be below the origin of the head arc $\boldsymbol{q}_{\mathrm{h}}$, but with a lower distance than $R_{\mathrm{h}}/2$. D_{ea} must be between $0.9R_{\mathrm{h}}$ and $N_{\mathrm{y}}/3$.

The eyes, which are found in the *third* step, are modeled as ellipses. The parameter $\boldsymbol{q}_{\mathrm{ey}} = (x_{\mathrm{ey}}, y_{\mathrm{ey}})^T$ is the position of the origin of the eyes, A_{ey} and B_{ey} the length of the ellipses axis, $2H_{\mathrm{ey}}$ the vertical and $2D_{\mathrm{ey}}$ the horizontal distance of the eye centers to each other. We use the following equation to calculate the edge energy inside the eyes' regions:

$$\begin{aligned} E_{\mathrm{ey}} &= \int_0^{2\pi} \int_0^{1} e_{\mathrm{s}}(x_{\mathrm{ey}} + A_{\mathrm{ey}} r\cos\phi + D_{\mathrm{ey}}, y_{\mathrm{ey}} + B_{\mathrm{ey}} r\sin\phi + H_{\mathrm{ey}}) \\ &\quad + e_{\mathrm{s}}(x_{\mathrm{ey}} + A_{\mathrm{ey}} r\cos\phi - D_{\mathrm{ey}}, y_{\mathrm{ey}} + B_{\mathrm{ey}} r\sin\phi - H_{\mathrm{ey}}) dr d\phi. \end{aligned} \tag{8.9}$$

With the following optimization we get the eye parameters:

$$\left(x^*_{\mathrm{ey}}, y^*_{\mathrm{ey}}, D^*_{\mathrm{ey}}, {A_{\mathrm{ey}}}^*, {B_{\mathrm{ey}}}^*, H^*_{\mathrm{ey}}\right) = \underset{(x_{\mathrm{ey}}, y_{\mathrm{ey}}, D_{\mathrm{ey}}, A_{\mathrm{ey}}, B_{\mathrm{ey}})}{\operatorname{argmax}} \frac{E_{\mathrm{ey}}}{\left({A_{\mathrm{ey}}}^2 + {B_{\mathrm{ey}}}^2\right)^{C_{\mathrm{ey}}}} \quad (8.10)$$

with a constant $C_{\mathrm{ey}} = 1.4$ influencing the ratio of energy to area of the eyes' regions.

The anatomic restrictions here are: The horizontal distance of the origin of the eyes $\boldsymbol{q}_{\mathrm{ey}}$ must be less than $R_{\mathrm{h}}/3$, the vertical position must be greater than y_{h} but lower than $y_{\mathrm{h}} + R_{\mathrm{h}}$. A_{ey} and B_{ey} must be lower than $0.4R_{\mathrm{h}}$ and D_{ey} lower than D_{ea}. The two eye regions must not overlap and the eye regions must not overlap the ears' regions.

Finally the nose/mouth region is to be found. It is modeled as a triangle stump with parameters: origin $\boldsymbol{q}_{\mathrm{nm}} = (x_{\mathrm{nm}}, y_{\mathrm{nm}})^T$, height H_{nm}, length of base line W_{nm} and the length of top line T_{nm}.

The amount of edge strength in the nose mouth region can be determined by the following equation:

$$E_{\mathrm{nm}} = \int_0^{H_{\mathrm{nm}}} \int_{-1}^{1} e_{\mathrm{s}}(x_{\mathrm{nm}} + w(W_{\mathrm{nm}} - \frac{h}{H_{\mathrm{nm}}}(W_{\mathrm{nm}} - T_{\mathrm{nm}})), y_{\mathrm{nm}} + H_{\mathrm{nm}}) dw dh. \quad (8.11)$$

The optimal parameters are found as

$$(x^*_{\mathrm{nm}}, y^*_{\mathrm{nm}}, {W_{\mathrm{nm}}}^*, {T_{\mathrm{nm}}}^*, {H_{\mathrm{nm}}}^*) = \underset{(x_{\mathrm{nm}}, y_{\mathrm{nm}}, W_{\mathrm{nm}}, T_{\mathrm{nm}}, H_{\mathrm{nm}})}{\operatorname{argmax}} \frac{(E_{\mathrm{nm}})^{C_{\mathrm{nm}}} H_{\mathrm{nm}}}{W_{\mathrm{nm}} + T_{\mathrm{nm}}}. \quad (8.12)$$

Here the constant to influence the ratio of edge energy to area is $C_{\mathrm{nm}} = 1.2$. The height of the nose/mouth region appears in the numerator of the ratio in Equation 8.12 to avoid that the optimization result is a region that covers just the nostrils or the mouth and not both of them.

The horizontal distance of the origin $\boldsymbol{q}_{\mathrm{nm}}$ must be lower than $D_{\mathrm{ey}}/3$ from $\boldsymbol{q}_{\mathrm{ey}}$. The vertical position must be between $y_{\mathrm{ey}} + 2D_{\mathrm{ey}}$ and $y_{\mathrm{ey}} + 3D_{\mathrm{ey}}$. W_{nm} must be between $2D_{\mathrm{ey}}$ and $3D_{\mathrm{ey}}$, T_{nm} between D_{ey} and $2D_{\mathrm{ey}}$, and H_{nm} between D_{ey} and $3D_{\mathrm{ey}}$.

All the restrictions to the optimized parameters were imposed because of observations of anatomic facts. The constants C_{R}, C_{h}, C_{ea}, C_{ey}, and C_{nm} which were used for the calculations were determined experimentally by localization and tracking of patient faces and facial features in 1000 images.

Tracking. When the face and the facial features are localized in one image (i.e. the parameters of the face model are calculated) the face and features can be found in the following image in approximately the same position as we postprocess a video stream with 25 frames per second and a relatively slow moving patient.

We initialize the simplex optimization with the parameters from image i and a set of parameters which are normally distributed with mean old parameter and variance depending on the expected parameter variation.

That reduces the search area very much and we initialize the simplex optimization with less parameter sets (instead of 5000 we use 500) to get the optimal parameter in a reliable way. The reduction of the initialization set also results in a noticeable decrease of calculation time (instead of 40 sec we need 8 sec per image).

8.4.4 Analysis of Facial Paresis

As written in the introduction we operate with patients with single sided facial pareses. In this case asymmetries can occur in the eye and mouth region when specific mimic exercises are performed. This asymmetries are considered to be symptoms for the present paresis. We will give two different data-driven analysis methods for the analysis of the eye region and mouth region of a human face.

The first mimic exercise is to lift up the eyebrows such that wrinkles will appear on the forehead. In the following we will call this exercise 'frowning'. Depending on the grade of paresis, some patients are not able to lift the eyebrow. This will result in certain asymmetries in the eye/eyebrow-region. Also the second exercise, the closing of the eyes, can generate asymmetries in this face area, as some patient are not able or have severe problems to close their eyes.

The other two exercises tell us something about the patient's mouth region. Asymmetries can arise when patients try to point their mouth or when they show the teeth. We record images of the patient while he is performing the mentioned exercises. To grade the asymmetries we need an additional view, the relaxed face.

Comparison of Intensity Values and Edge Strength. The first attempt is the direct comparison of gray-level values of mouth and eyes. To get a feature of the eye region, we take the gray-values of the left eye, mirror the single lines and match the gray-levels with the right eye by varying the x- and y-coordinate to find the minimal absolute sum D_1^* of the pixel differences.

$$D_1^* = \min_{x_t, y_t} \int\limits_{right\ Eye} |(f(x,y) - f(2(B_{\mathrm{ey}} - D_{ey}) - x + x_t, y - 2H_{\mathrm{ey}} + y_t)|dxdy. \quad (8.13)$$

The absolute sum divided by the area of the right eye (in our face model both eyes have the same size) is taken as feature c_1.

$$c_1 = \frac{D_1^*}{(A_{\mathrm{ey}} B_{\mathrm{ey}})}. \quad (8.14)$$

To analyze the mouth region we find the row index r_{min} inside the mouth region that will give the minimal sum of absolute differences when matching the pixels on the left of row r_{min} with those on the right side. Row r_{min} represents the vertical symmetric axis of the mouth which is the elongation of the nasal labial fold

$$D_2^* = \min_{r} \int\limits_{Mouth\ left}^{r} \int\limits_{0}^{H_{\mathrm{nm}}} |(f(x,y) - f(2r - x, y)|dxdy. \quad (8.15)$$

D_2^* is devided by the area of the analyzed region to get a feature for the asymmetry of the mouth region.

$$c_2 = \frac{D_2^*}{r_{min} H_{\mathrm{nm}}}. \quad (8.16)$$

To keep the following more predictable: Odd-indexed features c_{2i+1} result from comparisons of the eye regions, even-indexed features c_{2i} result from the mouth region. As mentioned before interesting areas inside the face are often regions where changes of the gray-values occur. Such regions appear emphasized in the edge-strength representation of the image. For that reason we additionally applied the methods not only on the gray-levels f but also on the edge-strength part e_{s} of the Sobel-filtered image. That gives us the next two features c_3 for the eye region and c_4 for the mouth region.

Using Averaging Filter Responses for Face Analysis. The second class of approaches for the analysis of facial asymmetries arises from the theory of steerable filters [770]. We use the response of averaging rotated wedge filters to characterize the direction information in the environment of certain key points (see Figure 8.28). The key points here are the corners of the eyes and the mouth and the extracted information contains the opening angle of those facial features (see Figure 8.29). The disadvantage of this approach is that the positions of the angles of the eyes and the mouth have to be determined as exact as possible. This additional localization is of course another source for errors. The localization of these features can be very hard even for a exercised person and it is often the case that just an unprecise estimation can be performed.

To get an estimate for the position of the angles of eyes and mouth we use the columns of the surrounding boxes of eyes and the mouth. We noticed that the angles often appear noticeable darker than the surrounding intensity values. The search for the angle positions is started at the columns of the outer borders of the localized facial feature regions (cf. Section 8.4.3). We calculate the average intensity value of one

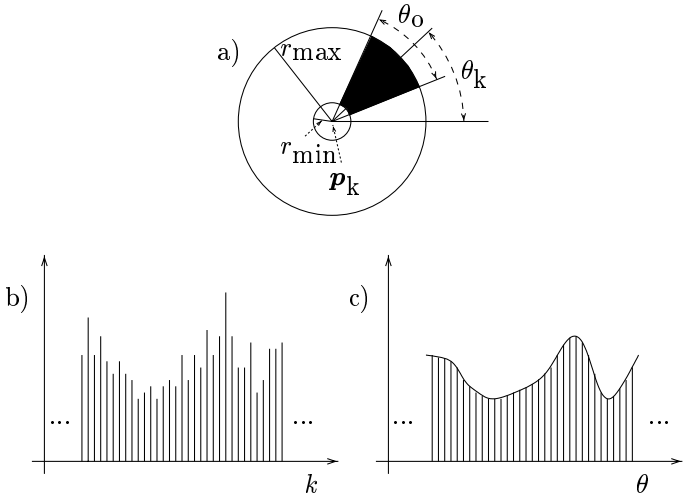


Figure 8.28. a) an averaging mask centered at key point $\boldsymbol{p}_{\mathrm{k}}$, b) the response of the individual wedge filters, c) the reconstructed (Gaussian smoothed) filter response.

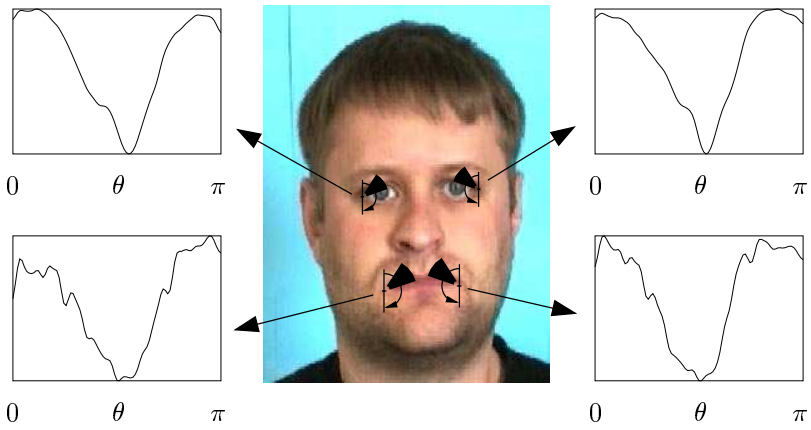


Figure 8.29. Smoothed responses of the wedge filters applied to the angles of the eyes and the mouth.

column, and compare the minimum of the column to it. If the quotient is below a threshold θ_e of θ_m, we stop the process and consider the angle as found. This simple method gives useful results which are used in the analysis process.

We take the localized eye angles and mouth angles as the key points of the wedge filters. The filters are rotated not the whole 2π but only π over the face feature in 2 degree steps which gives 91 averaged gray-values for every localized four facial feature angle. The opening angle of the wedge is 6 degree. The results are the four vectors $\boldsymbol{w}_i, i = 1, \ldots, 4$ of filter responses with 91 entries each $\boldsymbol{w}_i = [w_{i,1}, \ldots, w_{i,91}]^T$. Feature $\boldsymbol{w}_1$ results from the filtering of the corner of the right eye, $\boldsymbol{w}_1$ from the left eye, $\boldsymbol{w}_3$ from the right corner of the mouth, and $\boldsymbol{w}_4$ from the left corner. The filters are also applied to the edge-strength part e_s of a Sobel-filtered image. That gives another 4 vectors $\boldsymbol{w}_5$ to $\boldsymbol{w}_8$, resulting from the respective facial feature corners. With these eight vectors we can calculate another eight features values to analyze the facial asymmetry. c_5 is the absolute component difference sum of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$

$$c_5 = \sum_{i=1}^{91} |w_{1,i} - w_{2,i}|. \tag{8.17}$$

The same is done with $\boldsymbol{w}_3$ and $\boldsymbol{w}_4$ to calculate the feature c_6, c_7 with $\boldsymbol{w}_5$ and $\boldsymbol{w}_6$, and c_8 with $\boldsymbol{w}_7$ and $\boldsymbol{w}_8$. The feature c_9 is generated after matching $\boldsymbol{w}_1$ to $\boldsymbol{w}_2$. This is done by translations (t_1) and scalings (s_1) of the vector components of $\boldsymbol{w}_2$

$$c_9 = \min_{s_1,t_1} \sum_{i=1}^{91} |w_{1,i} - s_1 w_{2,i+t_1}|. \tag{8.18}$$

In analogy to the feature c_9 the features c_{10}, c_{11} and c_{12} can be calculated.

8.4.5 Classification of Facial Paresis

To classify a face whether a paresis is present or not, we proceed as follows. First we calculate the features c_1 to c_{12} while the person's face is in the following states:

1. relaxed face: all face muscles in a relaxed state,
2. lifting up the eyebrows: the result of this motion are wrinkles on the forehead,
3. closing the eyes: patients with paresis in the eye region have problems to do this,
4. pointing of the mouth,
5. showing the teeth: the last two exercises give information about a potential paresis in the mouth region.

That gives five sets of the twelve parameters each which are used for classification:

1. $c_{1,\mathrm{norm}}, \ldots, c_{12,\mathrm{norm}}$: extracted parameters while the face is in a relaxed state,
2. $c_{1,\mathrm{frown}}, \ldots, c_{12,\mathrm{frown}}$: extracted parameters while frowning,
3. $c_{1,\mathrm{close}}, \ldots, c_{12,\mathrm{close}}$: extracted parameters while closing the eyes,
4. $c_{1,\mathrm{point}}, \ldots, c_{12,\mathrm{point}}$: extracted parameters while pointing the mouth,
5. $c_{1,\mathrm{teeth}}, \ldots, c_{12,\mathrm{teeth}}$: extracted parameters while showing the teeth.

To analyze the facial paresis we are interested in asymmetries which are caused by the performation of the mimic exercises. To subtract the asymmetry information from the extracted features which is caused not by the paresis but other factors like the illumination or anatomic reasons, all the parameters $c_{1,\mathrm{frown}}, \ldots, c_{12,\mathrm{frown}}$, $c_{1,\mathrm{close}}, \ldots, c_{12,\mathrm{close}}$, $c_{1,\mathrm{point}}, \ldots, c_{12,\mathrm{point}}$, and $c_{1,\mathrm{teeth}}, \ldots, c_{12,\mathrm{teeth}}$ are normalized by the parameters $c_{1,\mathrm{norm}}, \ldots, c_{12,\mathrm{norm}}$, which mainly contain asymmetry information not generated by the mimic exercises.

This normalization is done by calculating the ratios

1. $\dfrac{c_{1,\mathrm{frown}}}{c_{1,\mathrm{norm}}}, \ldots, \dfrac{c_{12,\mathrm{frown}}}{c_{12,\mathrm{norm}}}$.
2. $\dfrac{c_{1,\mathrm{close}}}{c_{1,\mathrm{norm}}}, \ldots, \dfrac{c_{12,\mathrm{close}}}{c_{12,\mathrm{norm}}}$.
3. $\dfrac{c_{1,\mathrm{point}}}{c_{1,\mathrm{norm}}}, \ldots, \dfrac{c_{12,\mathrm{point}}}{c_{12,\mathrm{norm}}}$.
4. $\dfrac{c_{1,\mathrm{teeth}}}{c_{1,\mathrm{norm}}}, \ldots, \dfrac{c_{12,\mathrm{teeth}}}{c_{12,\mathrm{norm}}}$.

The four sets of normalized parameters with 12 values each are finally used to detect a facial paresis. Again, the odd-indexed normalized parameters contain information about the eye region, even-indexed normalized parameters about the mouth region. In the present state of the system the normalized features are thresholded to get the

information whether facial paresis exists or not. For every set of features we calculate if facial paresis is present. E.g. if $\frac{c_{1,\text{frown}}}{c_{1,\text{norm}}}$, $\frac{c_{1,\text{close}}}{c_{1,\text{norm}}}$, $\frac{c_{1,\text{point}}}{c_{1,\text{norm}}}$, or $\frac{c_{1,\text{teeth}}}{c_{1,\text{norm}}}$ are greater than the thresholds $\theta_{1,\text{frown}}$, $\theta_{1,\text{close}}$, $\theta_{1,\text{frown}}$, or $\theta_{1,\text{close}}$ we consider a paresis in the eye region. If $\frac{c_{2,\text{frown}}}{c_{2,\text{norm}}}$, $\frac{c_{2,\text{close}}}{c_{2,\text{norm}}}$, $\frac{c_{2,\text{point}}}{c_{2,\text{norm}}}$, or $\frac{c_{2,\text{teeth}}}{c_{2,\text{norm}}}$ are over the thresholds $\theta_{2,\text{frown}}$, $\theta_{2,\text{close}}$ $\theta_{2,\text{point}}$, or $\theta_{2,\text{teeth}}$ facial paresis in the mouth region is supposed.

This gives six ways to detect facial paresis in the eyes region and another six ways to diagnose the mouth region. The classification results of all twelve feature sets are presented in Section 8.4.6.

8.4.6 Experimental Results

The evaluation of the whole diagnosis support system was performed with 16 patients with different grades of paralysis and 4 healthy persons. In this section we present the results of all our experiments.

We started with the generation of image sequences of the 20 persons. The persons performed mimic exercises in front of a homogeneous background. and the image series including 20 images were taken when the extremal positions of the exercises described in the last section were reached. That gives five image sequences of length 20 of every person.

In every first image of the series the face was located by means of the parametric face model and tracked in the remaining 19 images. In Figure 8.30 the localization results are shown graphically. In that example the input image was of size 384×288. The calculated parameters are shown in Table 8.1.

Totally the localization was successful in 82% of the eyes and in 73% of the mouth and whenever the facial features where localized correctly the tracking was done error-free.

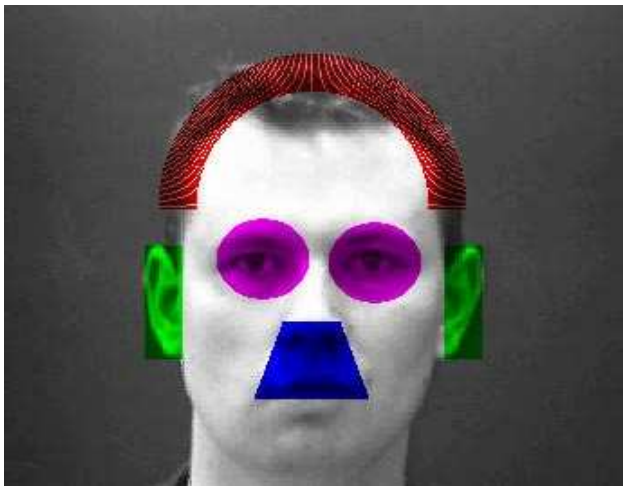


Figure 8.30. Localization example of face and facial features.

Head: $(x^*_{\mathrm{h}}, y^*_{\mathrm{h}}, R^*_{\mathrm{h}})$	(191,123,65)
Ears: $(x^*_{\mathrm{ea}}, y^*_{\mathrm{ea}}, D^*_{\mathrm{ea}}, W^*_{\mathrm{ea}}, H^*_{\mathrm{ea}})$	(192,144,20,62,93)
Eyes: $(x^*_{\mathrm{ey}}, y^*_{\mathrm{ey}}, D^*_{\mathrm{ey}}, {A_{\mathrm{ey}}}^*, {B_{\mathrm{ey}}}^*, H^*_{\mathrm{ey}})$	(194, 153, 31, 26, 22, 1)
Nose/Mouth $(x^*_{\mathrm{nm}}, y^*_{\mathrm{nm}}, {W_{\mathrm{nm}}}^*, {T_{\mathrm{nm}}}^*, {H_{\mathrm{nm}}}^*)$	(190,229, 63, 31, 43)

Table 8.1. Computed parameters (see Figure 8.27) for real face.

The localization of the eye and mouth angles (cf. Figure 8.31) was performed correctly with a rate of 50% for the eye angles and 45% for the mouth angles. In the case of an error the correct position was hand-segmented.

With the localized facial features the classification of the facial paresis was performed.

In Figure 8.32 characteristical evaluations of the features c_1 (eye region) and c_2 (mouth region) are shown. The features were calculated while the observed person performed the five mimic expressions described in the last section. The numbers at the x-axis show the exercise which was performed to calculate the feature. A patterned entry shows a change of asymmetry that might be originated by a facial paresis.

Then we generate the ratio of the parameters belonging to the relaxed face with the corresponding other ones. That produces for every of the twelve features c_1 to c_{12} four ratios. The ratios are compared with a threshold and if a certain number of ratios are greater than the threshold the face is classified to contain paresis or not.

The extracted features were used to classify face images into healthy persons and patients with facialis paresis. The selected threshold for all ratios was $\theta = 0.7$. Table 8.2 shows the classification results. The correctness of a classification was detected by comparation with the grading of a medical specialist.

8.4.7 Conclusion

We presented the basics of a system for diagnosis support and rehabilitation supervision of patients with facial paresis. The advantages of this system are an objective method to diagnose facial paresis. This is performed by means of measurements and numerical features from the face.

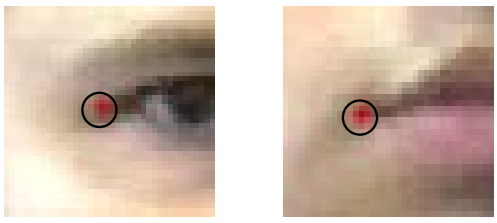


Figure 8.31. Localization example of the eye angle and mouth angle.

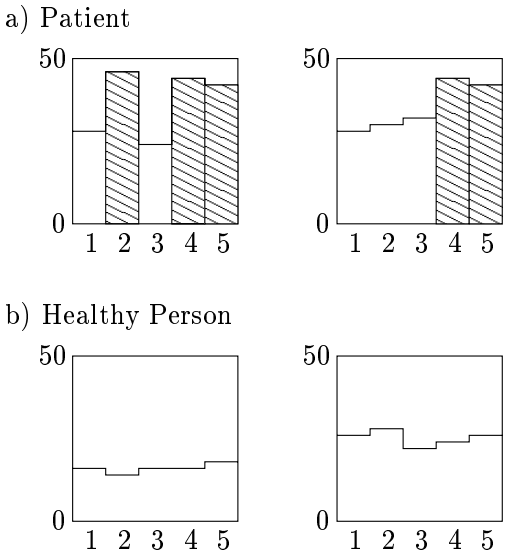


Figure 8.32. Results of face analysis: a) Patient, left: feature c_1 (eye), right: feature c_2 (mouth); b) Healthy Person, left: feature c_1 (eye), right: feature c_2 (mouth).

In a first step we localized the face and facial features using a dynamic face model and a energy maximization Simplex approach. The segmented model parameters where used to analyze the eye and mouth regions of the face towards asymmetries as asymmetries are symptoms for facial paresis. The extracted numerical features made us grade the facial paresis in mouth and eye regions. We evaluated the system by 15 patients and 5 healthy persons.

Eye						
	c_1	c_3	c_5	c_7	c_9	c_{11}
healthy	0.75	0.75	0.75	0.50	0.75	0.25
paresis	0.6	0.6	0.67	0.67	0.73	0.73
Mouth						
	c_2	c_4	c_6	c_8	c_{10}	c_{12}
healthy	1.0	1.0	0.25	0.25	0.50	0.50
paresis	0.87	0.87	0.73	0.73	1.0	0.93

Table 8.2. Classification results.

8.5 SURGICAL PLANNING

M. TESCHNER, S. GIROD, B. GIROD

Although 'virtual aircrafts' are commonly used as a training method for pilots, there exists no 'virtual surgery room' to educate physicians. When physicians finish their theoretical studies they gain surgical experiences on real patients. This fact is a strong motivation to develop computer-based simulation methods for surgical procedures. These methods can be used in medical education, they can improve diagnosis, surgical planning, and surgical treatment by providing the ability to perform 'virtual surgeries'.

The development of simulation methods for surgical procedures is a very complex task and only very specific problems have been solved in recent years. In order to describe a 'virtual patient' huge data sets consisting of different sensory modalities are required and interdisciplinary methods have to be applied to the data sets in order to build a surgical simulation system.

The basis for developing simulation methods for surgical procedures has been given in 1973, when Hounsfield invented computed tomography (CT) [342]. This technology enabled the generation of three-dimensional models that might be used in surgical simulation. There are two principal methods of simulating surgeries based on models:

One method is to build physical models. The idea of manufacturing lifelike models has been introduced in 1980 [9]. These models are generated from CT data using stereolithography and milling. In order to segment the bone structure a semiautomatic contour detection is applied to each CT slice. Further processing of contour data is done by a milling machine or by a stereolitography machine. While the milling machine builds the model from a block of polyurethane, the stereolitography machine builds the model slice by slice in a basin filled with liquid, photosensitive resin. Realistic, lifelike models improve osteotomy planning and allow accurate manufacturing of transplants. Moreover, these models can be used for educational purposes and demonstrations. However, lifelike models are expensive and their production can take several days. Furthermore, stereolitography machines have problems to reconstruct small bone structures that are not connected to the main part of the model. Stereolitographic models shrink up to six months after their production. Milling machines cannot reproduce hollows, undercuts and small holes. While physical models provide information on the bone structure, they do not contain knowledge of other structures.

The second method of simulating surgical procedures is to utilize medical imaging for generating computer models. The generation of computer models takes less time and is cheaper compared to stereolitographic and milled models. Additional inaccuracies caused by the manufacturing process of physical models are avoided. Computer models are more flexible than physical models and are able to provide more information by integrating several sensory modalities. A multimodal computer model of the bone structure and soft-tissue can be used for simulating osteotomies as well as for assessing the resulting soft-tissue changes. Various simulations can be performed with less additional effort compared to physical models. This is especially helpful in cases where various surgical options are possible.

The improvement of three-dimensional visualization techniques of tomographic data sets in the late 80's promoted the development of surgical procedures based on computer models. The *Marching Cubes algorithm* by Lorensen, 1987 [452] provided the ability to generate iso-surface triangle meshes from volume data sets. Based on this algorithm research focussed on simulating the manipulation of bone structure. In 1984 first results on modeling of deformable objects were published by Barr [43], in 1986 modeling of skin deformation using finite element models was proposed by Larrabee [420]. Deng (1988) [166] attempted to simulate wound closing using a three-layer soft-tissue model. Yasuda (1990) [768] and Pieper (1991) [545] introduced first craniofacial surgery systems. These systems were able to perform simple facial surgical simulations and to estimate the corresponding soft-tissue changes roughly. The idea of estimating soft-tissue changes due to bone realignment was formulated by Vannier in 1983 [722]. In 1992 further approaches to craniofacial surgery simulation were introduced by Kikinis [387], followed by Delingette 1994 [164], Bohner 1996 [75], Koch 1996 [398], and Bro-Nielsen 1998 [82].

At the Telecommunications Laboratory, University of Erlangen-Nürnberg, methods for craniofacial surgery simulation based on 3D computer models are investigated since 1993 [245, 383, 382]. In this section, an overview of components of the current surgical planning system is given [693].

The system uses an optimization approach for fast soft-tissue simulation. The section is organized as follows. In the next subsection the generation of the 3D computer models of the bone structure and the face surface is described. In Section 8.5.2 mesh simplification is described. In Section 8.5.3 the simulation of bone realignment is explained. In Section 8.5.4 the structure and parametrization of the soft-tissue model is described. In Section 8.5.5 optimization methods are compared that are used to estimate the soft-tissue deformation due to bone realignment. Simulation results are presented in Section 8.5.6.

8.5.1 Data Acquisition

In order to plan surgeries various sensory modalities are commonly used, e.g. radiographs, CT, MRI. Figure 8.33 illustrates the process of acquisition and model generation in case of craniofacial surgery simulation. Triangle meshes that describe the surface of the face and the bone structure of the head are the basic elements of the craniofacial simulation process. These meshes are built using two different sensory modalities.

A CT scan provides the anatomically correct representation of the bone structure and a laser scanner records a photorealistic, 3D model of the patient's face. The triangle mesh that represents the surface of the bone structure is generated by segmenting bone from the CT scan and applying the *Marching Cubes algorithm* [452] to the result. Although the *Marching Cubes algorithm* extracts isosurfaces from volume data sets it is useful to segment the bone structure in advance to reduce artifacts of the CT scan. The triangle mesh that represents the face surface is computed from the depth and color map of the laser scan.

Both modalities are registered by exploiting corresponding cephalometric landmarks of the laser scan and the skin surface taken from the CT scan [382].

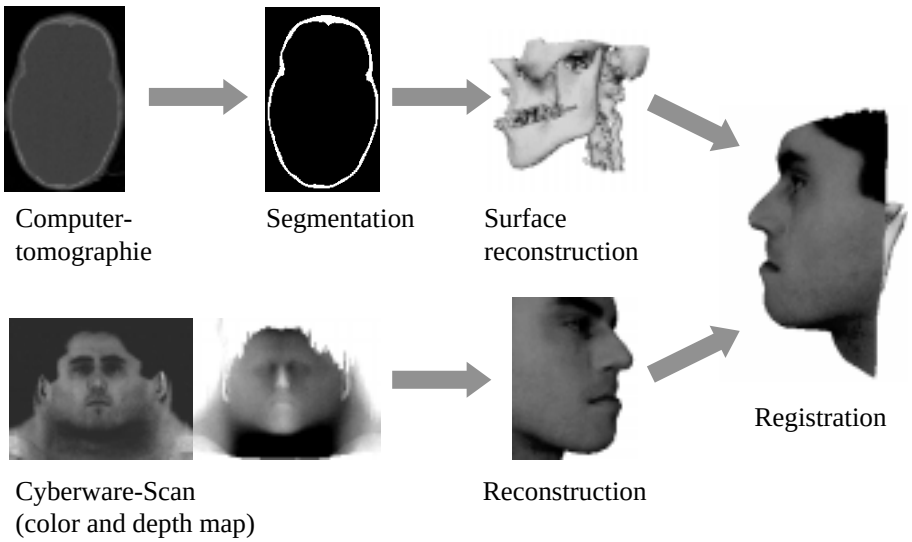


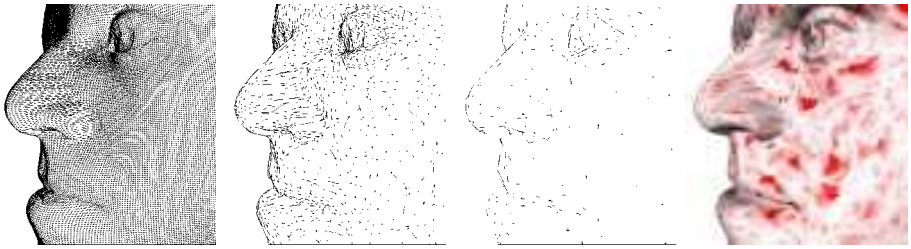
Figure 8.33. Data acquisition for craniofacial surgery simulation.

8.5.2 Data Reduction

Models represented as triangular meshes consist of a large number of triangles. Extraction of isosurfaces from volume data sets (MRI, CT) or reconstruction of surfaces from range scanner data sets (Cyberware) easily generates hundreds of thousands of triangles. In order to enable interactive visualization and handling of these triangle meshes they must be decimated. Several triangle mesh decimation algorithms have been developed in recent years. A classification of these algorithms is given in [130]. Basic operations of simplification algorithms are:

- *Vertex removal*: Vertices are removed and the resulting polygon is retriangulated.
- *Vertex clustering*: Vertices are grouped in clusters. These clusters are replaced by new representative vertices.
- *Edge / face collapsing*: Successive collapse of edges into vertices or triangles into edges or vertices.
- *Coplanar triangle merging*: Coplanar or nearly coplanar triangles are merged and the resulting polygon is retriangulated into fewer triangles.

In most cases mesh simplification is an iterative process that generates a sequence of meshes $M_0 \rightarrow M_1 \rightarrow \cdots \rightarrow M_{i-1} \rightarrow M_i$. A criterion for choosing an item for the reduction step $M_{i-1} \rightarrow M_i$ can consider the current mesh M_{i-1} (*local criterion*) or the initial mesh M_0 (*global criterion*) to compute a certain error measure. These error measures are defined as distances, angles or as a combination of various measures described by an energy function.



a) b) c) d

Figure 8.34. Mesh simplification. a) Original mesh, 100.0%. b) Simplified mesh, Hausdorff distance 0.1mm, 7.8%. c) Simplified mesh, Hausdorff distance 0.5mm, 1.8%. d) Hausdorff distances between a) and c). White: 0mm deviation. Black: 0.5mm deviation.

Due to the fact that local error criteria consider the current, partially simplified mesh, they do not describe the difference of the initial and the simplified mesh. However, in case of medical applications a desired accuracy of the decimated mesh compared to the initial mesh should be guaranteed by the algorithm.

In [105] a simplification method is proposed that incorporates the one-sided Hausdorff distance as global error criterion. The one-sided Hausdorff distance is defined by

$$d_h(X,Y) = \max_{\boldsymbol{x}\in X}(\min_{\boldsymbol{y}\in Y}(\|\boldsymbol{x}-\boldsymbol{y}\|)), \tag{8.19}$$

with X denoting the set of vertices of the simplified mesh and Y denoting the surface of the original mesh. The decimation algorithm guarantees that the global deviation of the original and the decimated mesh is not larger than the given Hausdorff distance. Figure 8.34 illustrates an original mesh generated from a laser range scan, two reduced triangle meshes and the Hausdorff distance between the original and a reduced mesh.

8.5.3 Simulation of Bone Movement

The process of surgery planning requires a realistic simulation of the transformation of certain bone structures, e.g. craniofacial surgery, knee surgery, hip surgery. While graphical environments provide the functionality to transform bone models they do not check for penetration of these models. This problem is addressed by collision detection algorithms. Basically, these algorithms work on surface models and check for interferences between triangles of different geometric objects. To speed up the collision test some approaches are restricted to a class of triangle meshes, e.g. convex meshes. However, these algorithms are not suitable for medical applications. Another approach to speed up the collision test and to avoid considering all triangles in the interference check is to employ hierarchical object representations.

In [255] a collision detection algorithm is presented that can be applied to objects represented as unstructured triangle meshes. The objects do not have to be convex and they are not considered to have any special properties. As an initialization step the

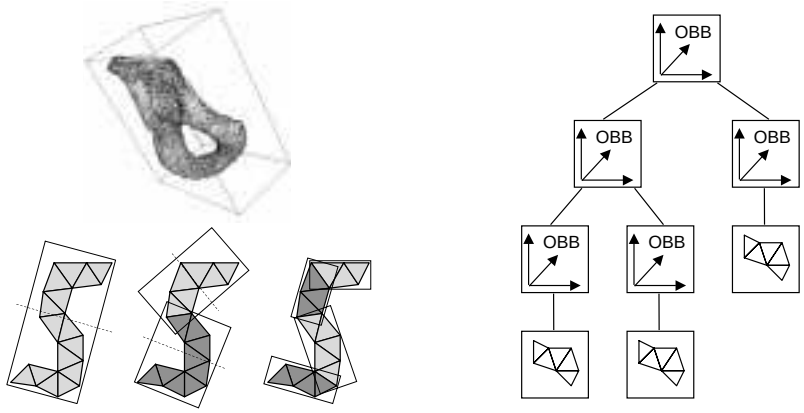


Figure 8.35. OBB-tree representation.

algorithm computes a hierarchical tree-structure of oriented bounding boxes (OBB). In contrast to axis-aligned bounding boxes (AABBs), OBBs are aligned with the principal axes of an object and grant a better approximation of the object. Although the computation of OBBs is more time-consuming than the computation of AABBs OBBs do not have to be recomputed if the object is transformed. In case of transformation OBBs only have to be transformed with the object. AABBs would have to be recomputed in case of rotation. Following certain rules a generated OBB is separated into two areas with quite the same number of triangles. For each of these areas a new OBB is generated and so on. This leads to a hierarchical OBB tree-representation (see Figure 8.35).

One important reason for the speed of the interference check of this algorithm is that only the OBBs are transformed instead of the whole object. Only if two OBBs are not separated, the next layer of OBBs is transformed and tested. If leaves are reached in both structures then all triangles that are represented by these leaves are transformed and tested for interference. This method optimizes the amount of transformation as well as the amount of interference tests (see Figure 8.36).

Collision detection based on this algorithm can be performed in real-time and enables interactive physiological bone movement and bone realignment.

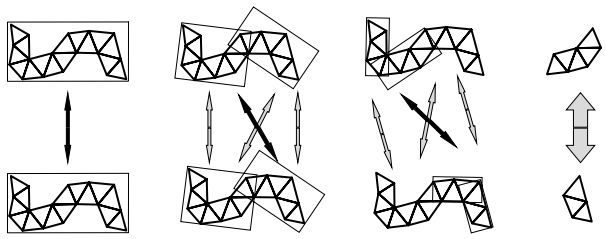


Figure 8.36. Collision detection based on an OBB-tree.

8.5.4 Soft-Tissue Model

Given the triangle meshes that represent the skull and the face, the soft-tissue model is generated. In recent years, several soft-tissue models based on springs or finite elements have been developed [164, 82, 382, 474, 163, 398]. As computational costs for finite-element methods are high and these methods seem to be less suitable for interactive applications, in our work, a mesh of springs is utilized. The springs are categorized according to their location and function (see Figure 8.37):

- *Layer springs* represent soft-tissue layers. In order to model differentiated elasto-mechanical properties of soft-tissue layers, each layer is represented by a particular class of springs. The number and the thickness of soft-tissue layers are variable (see Figure 8.38). Simulations have been performed with one, three, and five layers.

- *Bone springs* represent connections between bone and soft-tissue. Only some regions of the soft-tissue are connected to the underlying bone structure. To mimic sliding contact, these connections are modeled using springs.

- *Boundary springs* prevent the soft-tissue model from undergoing global transformation. Due to the fact that the face model does not include the complete head surface but only the facial region, these springs anchor the face and the underlying soft-tissue in space.

A spring is characterized by a spring constant k, which describes its stiffness and by a length l (see Figure 8.38). Every spring class is parametrized with a particular spring constant to model the elasto-mechanical properties of the corresponding soft-tissue layer. Bone springs and boundary springs are parametrized with a comparatively large spring constant. The length of bone springs and boundary springs is zero. The springs that represent the skin surface are given a certain strain. This strain corresponds to the skin turgor. Setting the natural length of all surface springs to $c_{\text{turgor}} \cdot l$, with $0 < c_{\text{turgor}} < 1$, introduces a certain strain. The difference of l and $c_{\text{turgor}} \cdot l$ corresponds to the desired skin turgor.

A *soft-tissue position* is characterized by a location $P \in \mathbb{R}^3$ and a mass m in order to enable simulation of gravity (see Figure 8.38). Every soft-tissue layer is parametrized by an overall mass, which is distributed according to the topology of the representing soft-tissue positions.

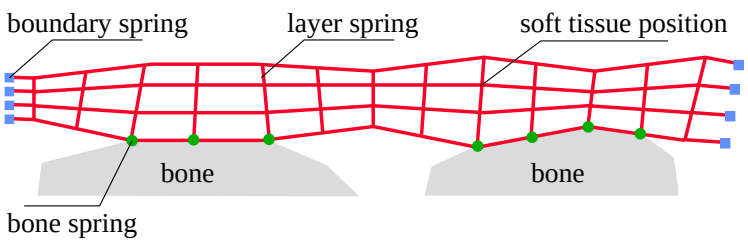


Figure 8.37. Soft-tissue model.

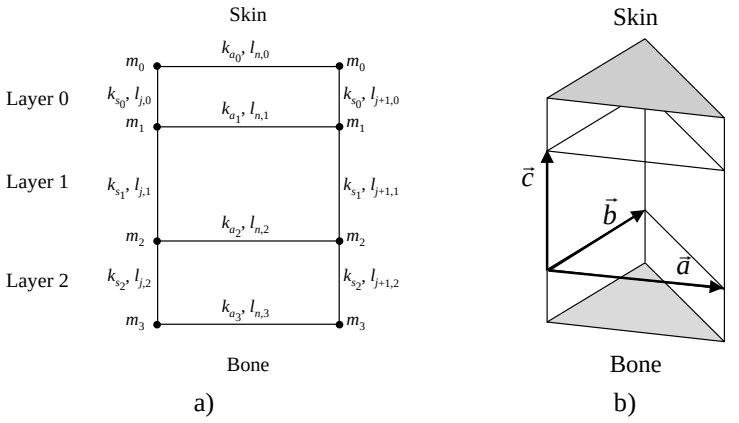


Figure 8.38. Soft-tissue representation. a) Parametrization. b) Basic element.

Due to the masses and the strain of the surface there are forces at each soft-tissue position. In order to prevent the model from changing without performing any bone realignment and obtaining a stable equilibrium of the mesh, the sum of these forces has to be zero. This is achieved by determining appropriate strains for all springs, given the strain of the skin surface.

8.5.5 Soft-Tissue Deformation

The described soft-tissue model is used to estimate soft-tissue deformation due to simulated bone realignment. Basically, the soft-tissue deformation is computed by applying an optimization method that minimizes the energy of the spring mesh. In the initial state of the simulation process the energy is zero. The energy is increased by performing bone transformation. An optimization process deforms the spring mesh in order to minimize the energy. The energy function

$$g(P_0, P_1, \ldots, P_{N-1}) = \lambda \sum_i k_i (l_{0i} - l_i)^2 + (1 - \lambda) \sum_j (v_{0j} - v_j)^2 \qquad (8.20)$$

depends on $3 \cdot N$ independent variables determining N soft-tissue positions $P_i \in \mathbb{R}^3$. It mainly captures differences between initial spring lengths l_{0i} and current spring lengths l_i and differences between initial volumes v_{0i} and current volumes v_i. The values k_i are spring constants, and λ $(0 < \lambda < 1)$ weights the influence of both terms of the function. The values v_i and v_{0i} are volumes of basic elements of the soft-tissue. Figure 8.38 illustrates the basic elements. The volume of a basic element is approximated by a cross product $0.5 \cdot (\boldsymbol{a} \times \boldsymbol{b}) \cdot \boldsymbol{c}$.

The initial state of the mesh is characterized by $l_i = l_{0i}$ for every spring. Bone movement leads to $l_i \neq l_{0i}$ and $(l_{0i} - l_i)^2 > 0$ for certain bone springs and to $g(P_0, P_1, \ldots, P_{N-1}) > 0$. Now, new soft-tissue positions P^* are computed by minimizing g. These values P^* describe the deformed soft-tissue:

$$P_0^*, P_1^*, \ldots, P_{N-1}^* = \operatorname{argmin} g(P_0, P_1, \ldots, P_{N-1}). \qquad (8.21)$$

Optimization method	Order of additional memory	Requires partial derivatives	Time $[s]$	$\min g$
Conjugate gradient, parabolic interpolation	N	yes	0.60	4.33
Conjugate gradient, derivative based	N	yes	2.21	4.33
Direction set (Powell)	N^2	no	81.43	4.33
Variable metric (quasi-Newton)	N^2	yes	2.91	4.30

Table 8.3. Comparison of optimization methods using a synthetic data set, the energy function in Equation 8.20, and performing an exemplary bone movement. 138 soft-tissue positions P_i, 986 springs, 144 volumes (SGI O2, R10000, 175 *MHz*). N is the number of parameters of the energy function.

During the minimization process there are no additional restrictions applied to the soft-tissue positions P apart from the energy function. All soft-tissue positions are considered in the minimization process, regardless of the simulated bone realignment.

Four optimization methods have been compared with regard to computational costs and robustness of the result (see Table 8.3). All optimization methods are iterative processes. They terminate if the difference of two P^* or the difference of two evaluations of g in successive steps is tolerably small. This tolerance can be chosen. On one hand, it influences the accuracy of the minimum, on the other hand it has an effect on the computation time. The slightly different minima found by the optimization algorithms (see Table 8.3) are due to this tolerance. Some methods require the calculation of partial derivatives. The methods differ in the amount of allocated memory. The order

a) b) c)

Figure 8.39. Craniofacial surgery simulation for a patient. a) Preoperative appearance. b) Simulated postoperative appearance. c) Postoperative appearance. Figure 8.40 shows the corresponding bone realignment.

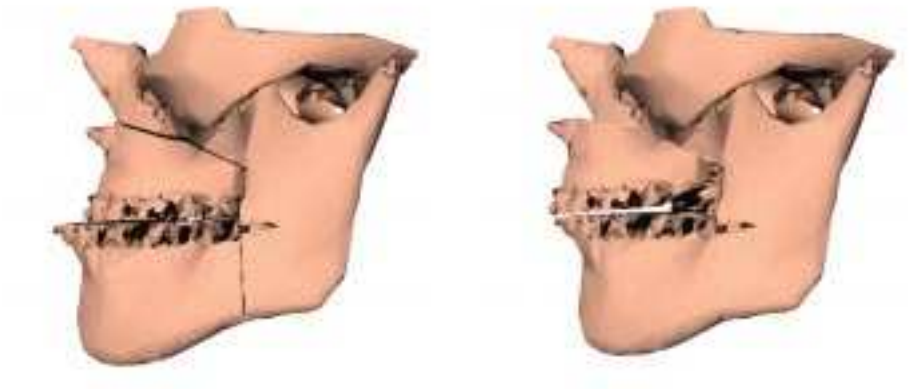
Figure	Number of soft-tissue positions	Number of springs	Number of volumes	Max. simulation time $[s]$
8.39, 8.40	954	6103	874	1.1
8.40, 8.41	1838	11763	1696	4.5

Table 8.4. Model parameters and simulation speed.

of additional memory that is needed by an optimization method is important due to the fact that its amount is dependent on the number of soft-tissue positions. If the model consists of 3000 positions, then the energy function in Equation 8.20 has 9000 parameters and an optimization method that requires memory in order of N^2 would need a multiple of 81 *MByte* memory instead of a multiple of 9 *kByte* for an algorithm with order of N. All optimization methods are described in [560].

Tests have shown that the conjugate gradient method provides reliable results and is very efficient with regard to memory and computational complexity. Parabolic interpolation is used for 1D sub-minimization due to the quadratic form of g. Although partial derivatives of the energy function are calculated by this optimization method, its computational expense is comparatively low because of the similarity of the energy function and its partial derivatives. The partial derivatives are responsible for fast convergence of the optimization process and fast convergence reduces the number of function evaluations.

In addition to computational costs another important criterion of an minimization algorithm is the quality of the minimum found. It cannot be guaranteed that the



a) b)

Figure 8.40. Craniofacial surgery simulation for a patient. a) Preoperative bone structure. b) Simulated postoperative bone structure. The upper jaw is repositioned 4mm forward and 2mm upward anteriorly and 4mm posteriorly. The lower jaw is moved backwards 5mm. The corresponding soft-tissue changes are illustrated in Figure 8.39.

Figure 8.41. Simulated physiological lower jaw movement for a patient.

minimum P^* is the global minimum, and it is difficult to prove that fact in a space with $> 10^3$ dimensions. A method to check the robustness of the minimum P^* is to perform a certain bone movement in different ways and to compare the results. If only a small movement is performed, the distance of the initial soft-tissue positions P and P^* is small, g is comparatively small, and the global minimum is likely to be found. For example, translating a bone by 0.1mm ten times or translating a bone by 1mm once should lead to the same P^*. Several tests using the multidimensional conjugate gradient method have been performed and all minima have been reliable.

8.5.6 Results

Table 8.4 and Figures 8.39–8.44 show examples of simulations performed. The soft-tissue prediction is tested with three individual patient data sets. Several simulations of bone movement have been applied to each model. The last column of Table 8.4 shows the maximum time needed by the optimization process. All tests have been performed on a standard workstation SGI O2, R10000, 175MHz. Figure 8.45 shows parts of the simulated planning process in case of a craniosynostosis. In this case, only cutting of the bone structure and its realignment has been simulated.

Figure 8.42. Simulated jaw movement for a patient. Figure 8.43 shows the corresponding bone realignment.

Figure 8.43. Simulated bone realignment for a patient. The corresponding soft-tissue changes are illustrated in Figure 8.42.

8.5.7 Ongoing Work

In this section, a craniofacial surgery simulation system has been presented. It simulates bimaxillary osteotomies, physiological jaw movement and predicts the soft-tissue changes caused by bone realignment. The soft-tissue prediction is based on an optimization method and has been tested with several individual patient data sets. Interactive collision detection and collision response has been integrated into the system to enable a realistic simulation of bone movement. Ongoing work focuses on realistic simulation transplants. As well as estimating the patient's static postoperative appearance and simulating physiological bone movement, the visualization of the patient's post-operative facial expressions is very useful. Therefore, it is planned to add muscles to the existing soft-tissue model. Furthermore, it is intended to register very accurate measurements of the jaws with the CT scan, in order to consider the occlusion of the jaws in the planning process.

Acknowledgement

This work is supported by the Deutsche Forschungsgemeinschaft DFG (SFB 603, Project C4). Patient data have been provided by the Maxillofacial Surgery Department of the University of Cologne, by the Maxillofacial Surgery Department of the University of Erlangen-Nürnberg, and by the Childrens' Hospital of the University of Erlangen-Nürnberg.

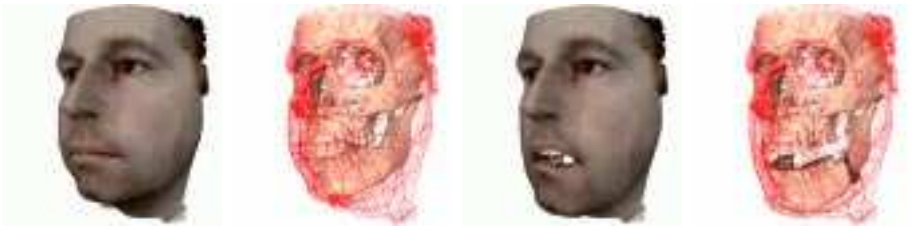


Figure 8.44. Simulated jaw movement for a patient.

a) b) c)

Figure 8.45. Planning process in case of a craniosynostosis. a) Reconstructed pre-operative skull. b) Segmentation of the Cranial Vault. c) Reconstruction of the Cranial Vault.

8.6 IMAGE COMMUNICATION

P. EISERT, B. GIROD

Source models play an important role in image coding. Knowledge that is available a priori and that can be represented appropriately need not be transmitted. Rate distortion theory allows us to calculate a lower bound for the average bitrate of any coder, if a maximum permissible average distortion may not be exceeded. Many of today's sophisticated coding schemes probably operate very close to their rate distortion theoretical bound. It may not be concluded, however, that this fundamentally prevents us from inventing even more efficient coding schemes. Rate distortion theoretical bounds are valid only for a given source model. Another source model might result in a lower rate at a given distortion. Better source models are the key to more efficient image compression schemes.

The majority of images are the result of a camera pointing to a three-dimensional scene. The scene consists mostly of surfaces reflecting the illumination towards the camera according to well understood physical laws. *Three-dimensional models* throughout this chapter are models capturing the three-dimensional spatial structure of a scene in front of the camera.

The attempt to explicitly recover 3D structure for a still image and use this information for coding is not very promising. The projection of the 3D scene on the image plane is certainly an enormous data reduction, and a 3D reconstruction has to overcome many ambiguities. How, e.g., to encode a flat photograph in a 3D scene? We can nevertheless hope for enormous data reduction using 3D models for image sequences. Even sequences with a lot of change from one frame to the next can often be described by only a few parameters in the domain of 3D objects and 3D motion. Using such *model-based coding* techniques, extremely low bitrates can be achieved for particular video sequences. For example, for an image sequence resulting from a camera moving through a static room, we would ideally transmit a texture-mapped 3D model of the room once, and then only update the 3D motion parameters of the camera. The decoder includes a 3D computer graphics renderer in this scenario. The

problem of rendering photorealistic images from 3D models is basically solved today. The difficult part which is addressed in this section is the automatic estimation of the object's deformation and 3D motion from the original images.

In the following sections, we first present the concept of model-based coding of head-and-shoulder video sequences. This technique uses a three-dimensional head model to describe the appearance of a talking person sitting in front of a camera. The head model together with the modeling of facial expressions are discussed in the next section. We then describe how the 3D facial motion and deformation can be estimated from 2D images. Finally, experimental results are shown that demonstrate the efficiency of model-based coding techniques.

8.6.1 Model-Based Video Coding of Head-and-Shoulder Scenes

In model-based coding of image sequences [8, 437, 533, 744], 3D models are needed for all objects in the scene describing their shape and texture. Therefore, we have to put some restrictions on the scene content. In this context, we focus on head-and-shoulder scenes, which are typical for video telephone and video conferencing applications. Encoding of other classes of sequences makes basically no difference provided that an explicit 3D model is available at the codec.

For videotelephony, we want to transmit the head-and-shoulder view of a talking person recorded with a single camera. Model-based coding is used as illustrated in Figure 8.46. The encoder analyzes the incoming frames and estimates the 3D motion and the facial expressions of the person. A set of facial expression parameters is obtained that describes (together with the 3D model) the current appearance of the person. Only a few parameters have to be encoded and transmitted, resulting in very low bitrates, typically less than 1 kbit/s [201]. At the decoder, the parameters are decoded and then used to deform the head model according to the person's facial expressions. The original video frame is finally approximated by simply rendering the 3D model at the new position.

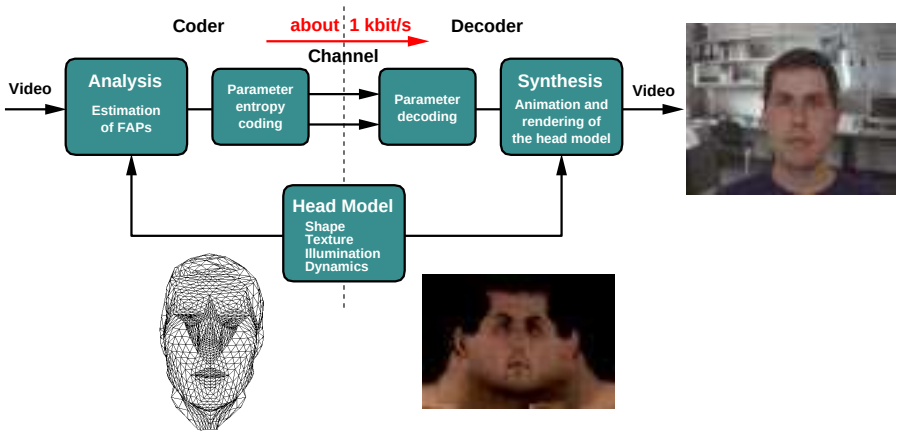


Figure 8.46. Basic structure of a model-based video codec.

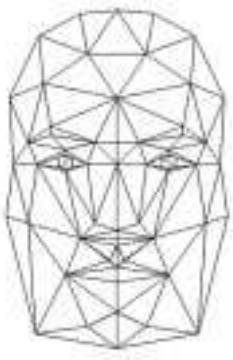


Figure 8.47. Wireframe of the CANDIDE face model.

8.6.2 Head-and-Shoulder Models

The modeling of the objects in the scene is an important part of model-based coding, since both encoder and decoder strongly depend on the shape, color and motion information from the 3D model. Many researchers have made progress modeling head and shoulders of a talking person in the past years [526]. One of the first parameterized head model was the CANDIDE model [587] that is shown in Figure 8.47. Similar to the face model developed by Parke [525] it consists of a three-dimensional triangular mesh that can be deformed to show facial expressions. Geometric details like eyes and the mouth are visible but due to the use of shaded single color triangles they typically lack reality.

A large step towards photorealistic modeling of human heads is the use of texture mapping [109, 73] and today almost all models have textured surfaces [8, 122, 201]. Figure 8.48 shows a head model in wireframe representation and the corresponding textured version.

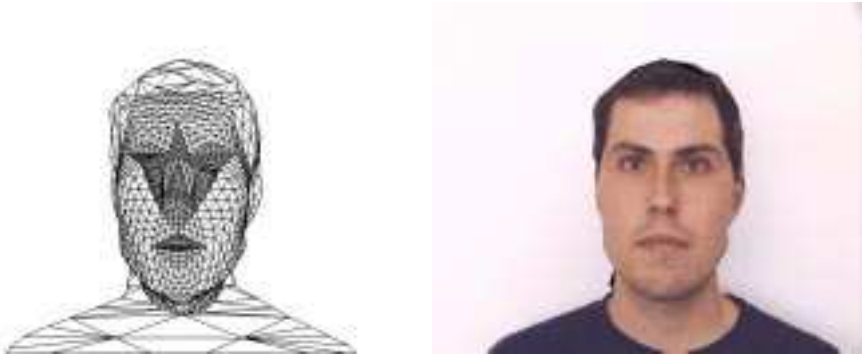


Figure 8.48. left: wireframe representation of a head model, right: corresponding textured version.

Eyes, teeth and the interior of the mouth, including the tongue, can be modeled similarly with textured polygonal meshes but a realistic representation of hair is still not available. A lot of work has been done in this field to model the fuzzy shape and reflection properties of the hair. For example, single hair strands are modeled with polygonal meshes [737] and the hair dynamics are incorporated to model moving hair [17]. However, these algorithms are computational intensive and are not practical for real-time applications in the next future. Image-based rendering techniques [435] might provide new chances for realistic hair modeling.

Modeling a human head with polygonal meshes results in a representation with a large number of triangles and vertices, which have to be moved and deformed to show facial expressions. To reduce the large number of degrees of freedom in the shape, splines [227] can be used to describe the surface, exploiting the smoothness of the facial tissue. Hoch et al. [322] have used B-splines with about 200 control points for this purpose. To allow a local refinement for the spline topology in areas that are more curved, triangular B-splines [201] or hierarchical splines [229] are used.

Rather than defining the shape of a human head by the position of the vertices or control points of a polygonal mesh or spline surface, a parameterized view-based modeling technique can be used [728]. A large database with the shape and texture of different people is established that spans a high-dimensional space of faces. Using the knowledge about the point correspondences between all 3D models, new shape and texture can be created by linearly interpolating or morphing between the models in the database. Even if not all individual characteristics of a person can be reconstructed exactly, the advantage of this approach is the small number of parameters (coordinates in the face space) that are sufficient to describe the shape and texture of a person.

8.6.3 Facial Expression Modeling

Once a 3D head model is available, new views can be generated by rotating and translating the 3D object. However, for the synthesis of facial expressions the model can no longer be static. In general, two different classes of facial expression modeling can be distinguished: the clip-and-paste method and algorithms based on the deformation of the 3D model.

Clip-and-Paste Method. For the clip-and-paste method [7, 744], templates of facial features like eyes and the mouth are extracted from previous frames and mapped onto the static 3D shape model. The model is not deformed according to the facial expression but remains rigid and is only used to compensate the global motion given by head rotation and translation. All local changes in the face must therefore be compensated by changing the texture of the model. During the video sequence, a codebook containing templates for the different facial expressions is built. A new expression can then be synthesized by combining several feature templates that are specified by their position on the model and their template index from the codebook.

Deformation Method. With the clip-and-paste method, a discrete set of facial expression can be synthesized. However, the transmission of the template codebook to the decoder consumes a large number of bits which makes the scheme unsuitable for

coding [744]. Beyond that, the localization of the facial features in the frames is a hard task. The pasting of templates extracted at inaccurate positions leads to unpleasening jitter in the resulting synthetic sequence.

The deformation method avoids these problems by using the same model for all facial expressions. The texture remains basically constant and the facial expressions are generated by deforming the 3D surface. In order to avoid the transmission of the changed position of all vertices of the triangular mesh, the facial expressions are parameterized using high-level expression parameters. Deformation rules associated to the 3D head model describe how certain areas in the face are deformed if a parameter is changed. The superposition of many of these local deformations is then expected to lead to the desired facial expression.

One system for facial expression parameterization that is widely used today [7, 122, 322, 438] is the facial action coding system (FACS) developed by the psychologists Ekman and Friesen [204]. Any facial expression results from the combined action of the 268 muscles in the face. Ekman and Friesen discovered that there are only 46 possible basic actions performable on the human face. Each of these basic actions, which they call *action units*, consists of a set of muscles that cannot be controlled independently. To obtain the deformation of the facial skin that is caused by a change of an action unit, the motion of the muscles and their influence on the facial tissue can be simulated using soft-tissue models. Head models which exploit the properties of the tissue and the muscles are called muscle-based models [692]. Due to the high computational complexity of the simulation, in many applications the surface deformation is modeled directly [7, 322] using heuristic transforms between action units and surface motion.

Very similar to the FACS is the parameterization in the recently determined *synthetic and natural hybrid coding* (SNHC) part of the MPEG-4 video coding standard [493]. SNHC allows the transmission of a 3D face model that can be animated to generate different facial expressions. Rather than specifying groups of muscles that can be controlled independently and that sometimes leads to deformations in larger areas in the face, in this system the single parameters directly correspond to locally limited deformations of the facial surface. There are 65 different facial animation parameters (FAPs) that control both global and local motion. Examples of different facial expression synthesized using this scheme are depicted in Figure 8.49.

Figure 8.49. Images synthesized using the MPEG-4 facial expression modeling scheme.

8.6.4 Analysis

The analysis of facial expressions from the image data itself is a hard task. Three-dimensional motion and deformation in the face have to be estimated. Due to the loss of one dimension in the camera projection, this ill-posed problem can only be solved by using additional assumptions and restrictions on the motion characteristics. In the case of head-and-shoulder scenes, a parameterized 3D head model provides us with information about shape, color and motion constraints. The motion constraints are defined by the particular facial expression modeling. If, for example, the FACS system is used, the complete motion in the face is described by the superposition of 46 well defined 3D displacement fields each controlled by a single parameter. Instead of estimating the motion vector of every surface point on the face, only 46 parameters corresponding to the action units have to be determined.

The motion constraints are assigned to the objects using 3D and 2D motion models as described in Section 2.2. Global head motion, characterized by head translation and rotation, is often estimated prior to the local motion [397, 438]. For the global motion, the rigid body model in Equation 2.13 can be used that leads to a motion constraint in the 2D image plane as shown in Equation 2.34. Equivalently, the local motion in the face caused by facial expressions is modeled using a flexible body motion model as in Equation 2.17 with the corresponding 2D representation given by Equation 2.38. Assuming that 2D displacements $(x_2 - x_1,\ y_2 - y_1)$ of object points between two frames recorded at the time instants t_1 and t_2 are given, we can solve for the unknown motion and deformation parameters. Since the underlying motion model is valid for the whole object and each displacement provides two additional equations, the number of point correspondences must be at least half the number of parameters to be estimated.

However, the displacements in the 2D images have to be determined first. Based on the way this is accomplished, the algorithms for estimating motion and deformation parameters from 2D images are typically classified into two groups. First, methods that are based on the tracking of discrete feature points and, second, algorithms that use the complete image incorporating gradient-based or optical-flow-based techniques.

Facial Parameter Estimation using Feature Points. One common way for determining the motion and deformation in the face between two frames of a video sequence is the use of feature points [7, 4, 346, 373]. Highly discriminant areas with large spatial variations, e.g. like areas containing the eyes, nostrils, or mouth corners are searched and tracked over the frames. If the corresponding features are found in two frames the change in position can be taken as the displacement.

How the features are searched depends on their properties like color, size, or shape. For facial features, extensive research has been performed especially in the area of face recognition [116]. Templates [91], often used for finding facial features, are small reference images of typical features. They are compared at all positions in the frame to find a good match between the template and the current image content. The best match is then said to be the desired feature. Problems with templates arise from the wide variability of captured images due to illumination changes or different viewing position. To compensate these effects, Eigen-features [486], which span a space of

possible feature variations, or deformable templates [771], that reduce the features to their contours, can be utilized.

Rather than estimating single feature points, also the whole contour of features can be tracked [346, 533] using snakes. Snakes [378] are parameterized active contour models that are composed of several energy terms. Internal energy terms account for the shape of the feature and smoothness of the contour while the external energy attracts the snake towards feature contours in the image.

Optical-Flow Based Estimation. When using feature based algorithms, single features like the eyes can be found quite robustly. Dependent on the image content, however, only a small number of feature correspondences is typically determined. As a result, the estimation of the 3D motion and deformation parameters from the displacements is completely wrong, if only one feature is associated to a different feature in the second frame.

In contrast, optical flow based techniques [160, 201, 397, 438] utilize the complete image information leading to a large number of point correspondences. The single correspondences are not as reliable as the ones for feature-based methods, but due to the large number of equations, their influence is not critical. Additionally, possible outliers can generously be removed without obtaining an under-determined system of equations.

Gradient-based algorithms utilize the optical flow constraint [333]

$$\frac{\partial I(x,y)}{\partial x}(x_2 - x_1) + \frac{\partial I(x,y)}{\partial y}(y_2 - y_1) = I_1(x,y) - I_2(x,y), \tag{8.22}$$

where $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are the spatial derivatives of the image intensity at image position (x, y). This equation, obtained by Taylor series expansion up to first order of the image intensity, can be set up anywhere in the image. It relates the unknown 2D motion displacement $(x_2 - x_1, y_2 - y_1)$ with the content of the images.

The solution of this problem is under-determined since each equation has two new unknowns for the displacement coordinates. For the determination of the optical flow or motion field, additional constraints are required [333]. Exploiting knowledge about the shape and motion characteristics of the object, any 2D motion model of Section 2.2.3 can be used as an additional motion constraint. Inserting the 2D motion model into Equation 8.22 reduces the number of unknowns to the number of motion parameters of the corresponding model. In that case, it is assumed that the motion model is valid for the complete object. An over-determined system of equations is obtained that can be solved robustly for the unknown motion and deformation parameters in a least-squares sense. The computational complexity is moderate, since the 2D motion models are linear in the parameters.

In the case of facial expression analysis, the motion models are taken from the motion characteristics of the head model description according to Equation 2.15. Global motion can be estimated first, followed by local motion refinement [122, 397] or both global and local motion are determined in a uniform framework [160, 201].

Since the optical flow constraint Equation 8.22 is derived assuming the image intensity to be linear, it is only valid for small motion displacements between two

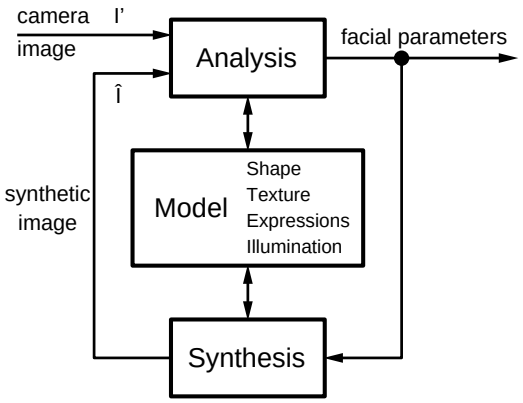


Figure 8.50. Analysis-synthesis loop of the model-based coder.

successive frames. To overcome this limitation, a hierarchical framework can be used [201]. First, a rough estimate of the facial motion and deformation parameters is determined from sub-sampled and low-pass filtered images, where the linear intensity assumption is valid over a wider range. The 3D model is motion compensated and the remaining motion parameter errors are reduced on frames having higher resolutions.

The hierarchical estimation can be embedded into an analysis-synthesis loop [438] as shown in Figure 8.50. In the analysis part, the algorithm estimates the parameter changes between the previous synthetic frame $\hat{I}_1$ and the current frame I' from the video sequence. The synthetic frame $\hat{I}_1$ is obtained by rendering the 3D model (synthesis part) with the previously determined parameters. This approximative solution is used to compensate the differences between the two frames by rendering the deformed 3D model at the new position. The synthetic frame now approximates the camera frame much better. The remaining linearization errors are reduced by iterating through different levels of resolution. By estimating the parameter changes with a synthetic frame that corresponds to the 3D model, an error accumulation over time is avoided.

8.6.5 Experimental Results

As an example for facial expression analysis we present some results from the algorithm proposed in [201] that uses the model-based concept for encoding of head-and-shoulder video sequences.

Video sequences are recorded of a talking person at a frame rate of 25 Hz and CIF resolution (352 x 288 pixels). The camera is calibrated using Tsai's camera calibration technique [709]. The obtained internal camera parameters like aspect ratio of the image and viewing angle form the parameters of the perspective projection model that describes the relation between 3D and 2D coordinates.

A triangular B-spline based 3D head model as depicted in Figure 8.48 provides us with information about shape and color of the person. To individualize the generic model, a 3D laser scan of the person is utilized. The control points of the splines, defining the shape of the surface, are optimized to adapt the model to the measured

Figure 8.51. Original (left) and corresponding synthetic frame (right) of the sequence 'Peter'.

data of the scan. The texture for the 3D model is extracted from the first video frames of the corresponding sequence.

For the estimation of the facial expression parameters that represent the 3D motion and deformation of the 3D head model, a hierarchical gradient-based algorithm is used as described above. In our experiments, 23 parameters are estimated. These parameters include global head rotation and translation (6 parameters), shoulder motion (4 parameters), movement of the eyebrows (4 parameters), two parameters for eye blinking, and 7 parameters for the motion of the mouth and the lips. These parameters are transmitted to the decoder, where they determine the position and deformations of the 3D head model. The original images are reconstructed by simply rendering the model. Figures 8.51 and 8.52 show examples of the resulting synthetic frame for two video sequences.

Figure 8.52. Original (left) and corresponding synthetic frame (right) of the sequence 'Eckehard'.

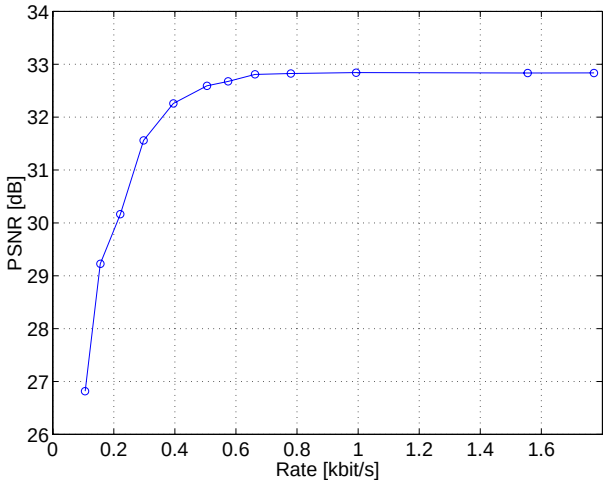


Figure 8.53. Rate-distortion plot for the model-based coder.

To measure the quality of the resulting synthetic sequences we use the PSNR (peak signal to noise ratio) as an error measure which is defined as

$$PSNR = 10 \log \left(\frac{255^2}{\frac{1}{N} \sum_{i=0}^{N-1} (I_{orig,i} - I_{synth,i})^2} \right). \tag{8.23}$$

In this equation, $I_{orig,i}$ is the luminance component of pixel i of the original camera image with values from 0 to 255, and $I_{synth,i}$ is the corresponding value in the synthetic image. N denotes the number of pixels in the image. Evaluating this error measure in the facial area of the two sequences, having a length of 230 and 220 frames, leads to an average value of 32.8 dB and 32.6 dB, respectively.

To determine the bitrate necessary to transmit the 23 parameters over a channel, we predict the current values from the previous frame, scale and quantize them. In this experiment, only every third frame is encoded leading to a frame rate of 8.33 Hz. An arithmetic coder that is initialized with experimentally determined probabilities is then used to encode the quantized values. The training set for the arithmetic coder is separate from the test set. Figure 8.53 illustrates the rate-distortion curve, where the average image quality is plotted over the average bitrate. Note that the model failures of the 3D head model lead to a saturation of the curve at about 32.8 dB. However, the plot shows that it is possible to transmit head-and-shoulder video sequences at bitrates of below 1 kbit/s.