Fast head modeling for animation

WON-SOOK LEE, NADIA MAGNENAT-THALMANN
MIRALab, CUI, University of Geneva, Geneva, Switzerland
E-mail : {wslee, thalmann}@cui.unige.ch

Abstract

This paper describes an efficient method to make individual faces for animation from several possible inputs. We present a method to reconstruct 3D facial model for animation from two orthogonal pictures taken from front and side views or from range data obtained from any available resources. It is based on extracting features on a face in a semiautomatic way and modifying a generic model with detected feature points. Then the fine modifications follow if range data is available. Automatic texture mapping is employed using a composed image from the two images. The reconstructed 3D-face can be animated immediately with given expression parameters. Several faces by one methodology applied to different input data to get a final animatable face are illustrated.

Keywords: orthogonal picture, range data, reconstruction, modification, generic model, automatic texture mapping, and facial animation.

1 Introduction

3D modeling of human face has wide applications from video conference to facial surgery simulation. To clone a real person has been even more interesting in today's virtual world. However, cloning a real person's face has practical limitations in the sense of how long it takes to get a result, how simple the equipment is, and how much realistic shape can be obtained.

There are many approaches to enable the reconstruction of a realistic face in a virtual world such as using a plaster model [1], or interactive deformation [13], which can give nice results, but are time-consuming. Methods that are more efficient are classified into five categories.

Laser scanning. An example of commercial 3D digitizer based on laser-light scanning is Cyberware Color Digitizer™ [12].

Stripe generator. As an example of structured light camera range digitizer, a light striper with a camera and stripe pattern generator [11] can be used for face reconstruction with relatively cheap equipment compared to laser scanners.

Stereoscopy. The method uses the geometric relation over stereo images to recover the surface depth [22][2], which often results very noisy data.

Video sequences. It acquires a video stream, which needs a cheap and entirely passive sensor, such as an ordinary video camera [16]. More frames there are, better the results are.

Most of the above methods concentrate on recovering a good shape, but the biggest drawback is that they provide only the shape without structured information. Starting with a structured facial mesh, Lee et al. [10] developed algorithms that automatically construct functional models of the heads of human subjects from laser-scanned range and reflection data [12].

The approach based on 3D digitization to get a range data often requires special purpose (high-cost) hardware. More common way of creating 3D objects is reconstruction from 2D-photo information, which is accessible at the lowest price.

Feature points on pictures. It utilizes a 3D existing face model and very little information from few pictures. A generic model with animation structure in 3D is provided in advance and a limited number of feature points, which are the most characteristic points used to recognize people, is detected either automatically or interactively on the two (or more) pictures. Then the other points on the generic model are modified by a special function. Kurihara and Arai [9], Akimoto et al. [4], Ip and Yin [7], and Lee et al. [14] use an interactive, semiautomatic or automatic method to detect feature points and modify a generic model. Some have drawbacks such as too few points to guarantee appropriate shape from a very different generic head or an accurate texture fitting, while others utilize too much loose automatic methods such as simple filtering. Most of them do not provide either appropriate texture image generation with high resolution or correct texture coordinates.

We present *one* fast and efficient methodology to cover both range data and picture data to get an animatable face model reconstruction in a virtual world. The approach belongs to a modification a

generic model with ‘feature points on pictures’ category above, but also we generalize this method to range data input to give animation structure.

We organize this paper as follows. In Section 2, we present a fast method applied to two kinds of input to get an animatable cloning of a person. A semiautomatic feature detection is described to get rough shape of a given face from orthogonal picture data or range data (VRML format for our experiment). The methods to modify a generic model in one or two steps depending on data are followed and automatic texture mapping method is provided. Several resulting heads are illustrated in Section 3 with overview of our facial animation system. Finally in Section 4, conclusion and future research are given.

2 Face Reconstruction

This section dedicated into two goals. First to clone a person just using two orthogonal picture data and second to give an animation structure to a given range data. Two goals look very independent, but as shown in Figure 1, two methods follow the same modules, from feature detection to final animation.

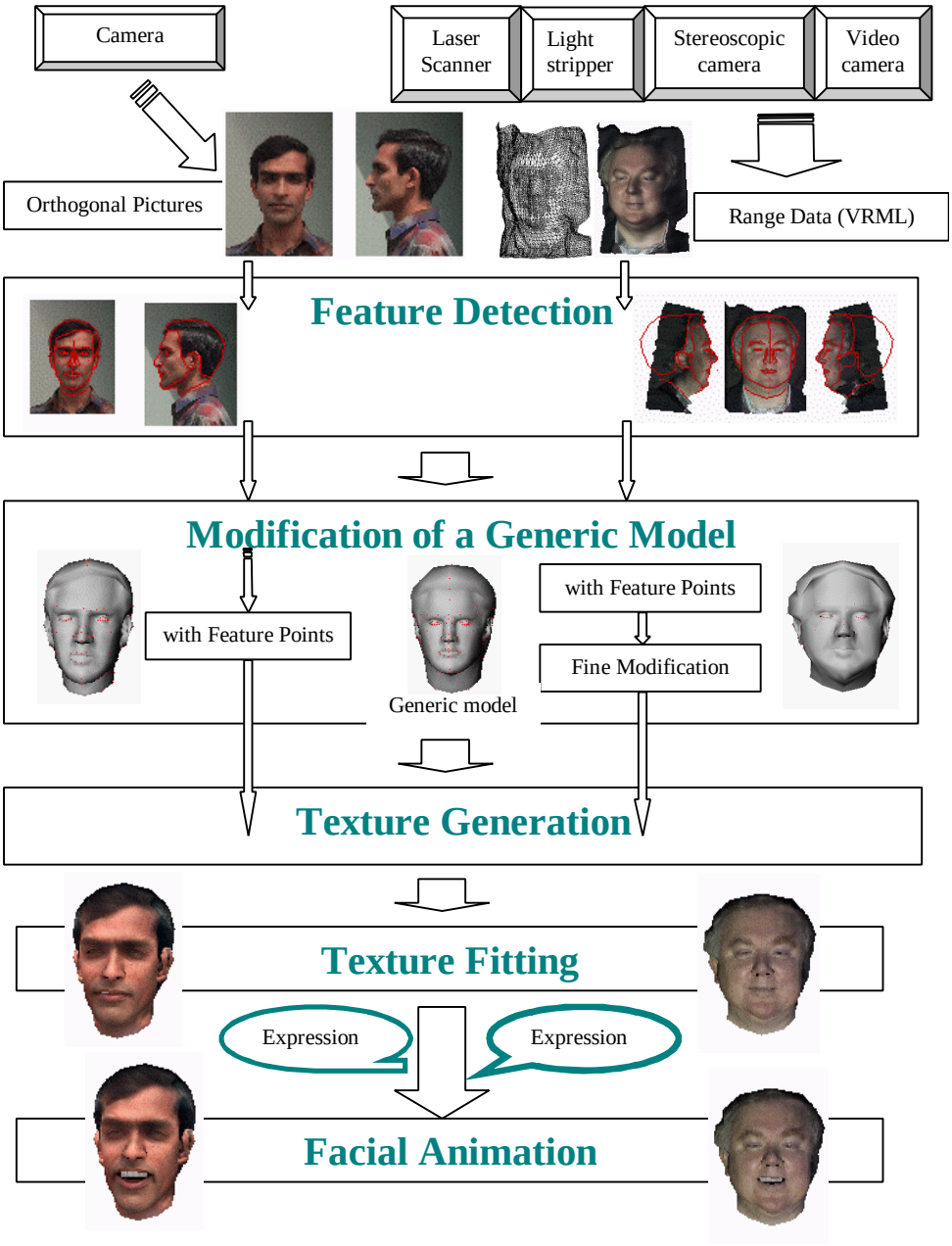


Figure 1: An overall flow diagram for face reconstruction from two kinds of input.

Two different inputs are processed in a very similar way. First we apply feature detection from 2D image data (orthogonal picture images or texture image for range data), and then we modify a generic model using DFFD for a rough matching. For range data, one more modification is followed. Then texture mapping is processed as a final step.

2.1 Preparation & Normalization

We prepare a 3D generic model with animation structure in and 2D frames used for the normalization and the feature detection. Normalization is used to bring input data into a standard space.

Orthogonal pictures First we prepare two 2D-wire frames composed of feature points with predefined relation for front and side views. The frames are designed to be used as an initial position for the snake method, which will be used later. To make the head size of side and front views the same, we measure the lengths of a face in two views. Then we choose one point from each view to match them with the corresponding points in prepared frame. Then we use transformation (scaling and translation) to bring the pictures to the wire frame coordinate, overlaying frames on pictures. Two photos with different size and position in Figure 2 are resulted from normalization process.

Range data For our experiment, we utilized data copied from the Turing Institute [3] where stereoscopic camera is used to generate range data of faces. We can convert some part of the range data in VRML format to appropriate to our need as input if necessary. We visualize the range data in front view as it is and rotate and translate to show the right and left views together as shown in Figure 3. The height checked in the front view is used to place three 2D frames for front, right and left views automatically.

2.2 Feature Detection

We provide a semi-automatic feature point extraction method with a user interface for interactive correction if it is required. There are methods to detect feature points just using special background information, predefined threshold [4][7], or image segmentation [4]. However, they are not very reliable since some boundaries are not easy to detect in many cases and many parameters are too sensitive depending on each individual's facial image. We use a snake method [14], which is more robust than simple thresholding method. To get correspondence between points from pictures and points on a generic model, which has a defined number, a snake is a good candidate. Above the conventional snake, we add some more functions called as structure snake, which is useful to make correspondences between points on a front view and ones on a side.

Orthogonal pictures Feature detection is applied for both front and side views. We get (x, y) from a front view and (y, z) from a side view. We can provide some functionality such that y coordinates on front and side views share the same or at least similar value. See Figure 2.

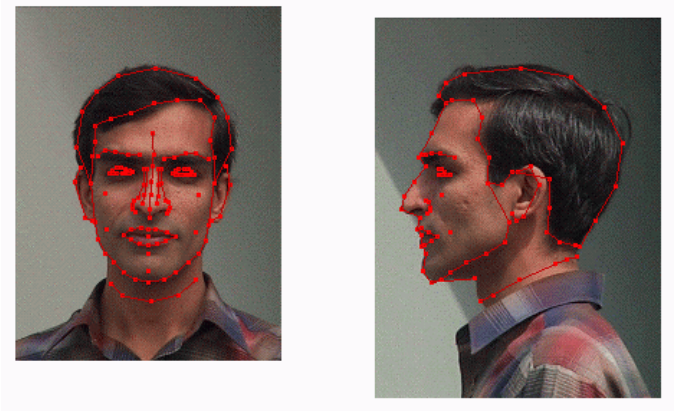


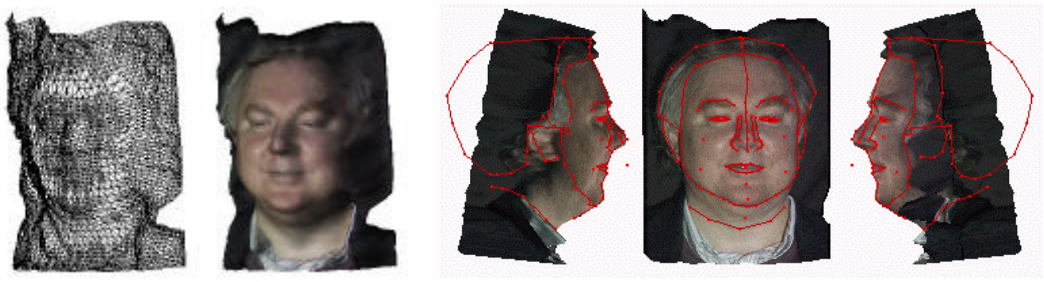
Figure 2: Normalization and semiautomatic feature detection on front and side views.

Range data Feature detection is applied only for front view. However right and left views are shown together as we can see in Figure 3. Whenever a point on a front view is detected, its depth z is calculated automatically and it visualizes the z value on a window. First we collect a certain number n of nearest points in (x, y) plane. Then apply

$$z = \sum_{i=1}^{n} \frac{\frac{1}{d^2(\vec{p},\vec{p}_i)}}{\sum_{i=1}^{n} \frac{1}{d^2(\vec{p},\vec{p}_i)}} \cdot z_i$$

where $\vec{p} = (x, y)$, $\vec{p}_i = (x_i, y_i)$ and $d^2(\vec{p},\vec{p}_i) = (x_i - x)^2 + (y_i - y)^2$.

Some feature points on side views are defined interactively since they are not provided in range data and most range data methods have problems obtaining proper hair shape because of the characteristic of high reflection.



(a) (b)

Figure 3: (a) Range data with details but without functional structure and backside head. (b) Feature detection on a front view and depth calculation on side view.

2.2.1 Structured Snake

First developed by Kass et al. [8] active contour method, called snakes, is widely used to fit a contour on a given image. Above the conventional snake, we add three more functions. First, we move a few points to the corresponding position interactively, and anchor them to keep the structure of points when snakes are involved, which is also useful to get more reliable results when the edge we would like to detect is not very strong. Since the correspondence between control points on a generic model, which will be used for head modification later, and feature points on pictures have to be provided, just to have edge detection is not enough. Anchoring some points to get sure-position for certain points helps this correspondence. We have several parameters such as elasticity, rigidity, image potential and time step, to manage the movement of snake. As our contours are modeled as polylines, we use a discrete snake model with elastic, rigid forces and image force acting on each pixel for color interest. We define different sets of parameters for hair and face according to their color characteristic. To adjust the snake on points of strong contrast, we consider

$$F_{ext,\,i} = n_i \cdot \tilde{N}\, E(v_i) \qquad (1)$$

where n_i is the normal to the curve at the node i, whose position is v_i and is given by

$$E(v_i) = |\, \tilde{N}\, I(v_i)\,|^2 \qquad (2)$$

where $I(v_i)$ represents the image itself. To estimate the gradient of the image, we use the Sobel operator. We then use color blending for a special area, so that it can be attracted by a special color [5]. Blending of the different color channels is changed to alter the snake's color channel sensitivity. For instance, we can make snakes sensitive to the excess of dark brown color for hair. Also we use clamping function to emphasize special interesting range of color. Snakes are useful in many circumstances, particularly in the presence of high contrast. The color blending and clamping function depend on the individual.

When the color is not very helpful and Sobel operator is not enough to get good edge detection, we use multiresolution techniques [6] to obtain strong edges. It has two main operators, REDUCE and EXAPND with Gaussian operator.

Let G_0 be the original image. Then for $0 < l < N$:

$$G_l = REDUCE(G_{i-1})$$

by which we mean:

$$G_l(i, j) = \sum_{m=1}^{5} \sum_{n=1}^{5} w(m,n) \cdot G_{i-1}(2i+m, 2j+n)$$

where the pattern of weights *w(m,n)* used to generate each pyramid level from its predecessor is called the generation kernel. These weights are chosen subject to four constrains, separable, symmetric, normalized, and each level *l* node must contribute the same total weight to level *l+1* nodes. So when $w(m,n) = \hat{w}(m)\,\hat{w}(n)$, $\hat{w}(0) = a$, $\hat{w}(-1) = \hat{w}(1) = b$ and $\hat{w}(-2) = \hat{w}(2) = c$ where $a + 2b + 2c = 1$ and $a + 2c = 2b$. So for a given *a*, $b = ¼$ *and* $c = ¼ - a/2$. We use $a = 0.4$ in our application.

Let $G_{l,k}$ be the image obtained by expanding G_l, *k* times. Then $G_{l,0} = G_l$ and for $k > 0$,

$$G_{l,k} = EXPAND(G_{l,\,k-1})$$

by which we mean:

$$G_{l,k}(i,\,j) = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} G_{l,k-1}\left(\frac{2i+m}{2}, \frac{2j+n}{2}\right)$$

where only terms for which $(2i + m)/2$ and $(2j + n)/2$ are integers contribute to the sum.

The subtraction produces an image resembling the result after Laplacian operators commonly used in the image processing. More times the REDUCE operator is applied the stronger the edges become. The same method is used to remove boundary effect in texture generation in Section 2.4.1.2.

2.3 Modifying a Generic Model

We detected feature points on photo images. There are several ways to modify a given generic model either in 2D or in 3D. We provide a 3D modification using 3D control points, which makes the condition for perfect orthogonal image less critical than in 2D modifications. We produce 3D points from two 2D points on frames with predefined relation between points on a front view and on a side view. In our case, we use about 160 feature points and many points have values (it means they are visible on images) on both front and side views, while others have values only on a front view or on a side (for example, back part of a head is not visible in the front view). The problem is how to make (*x, y, z*) from a given set of 2D points. When points have x, y_f, y_s, z, (subscripts *s* and *f* mean side and front view) and we usually take y_f for *y* value, but sometimes y_s specially 'ear' region. Since the perfect orthogonal pair pictures are not easy to be realized, it may occur unexpected 3D shape if we take average of y_s and y_f for *y* coordinate. For other points with only x, y_f or x, y_s, we use a predefined relation from a typical face to get (*x, y, z*).

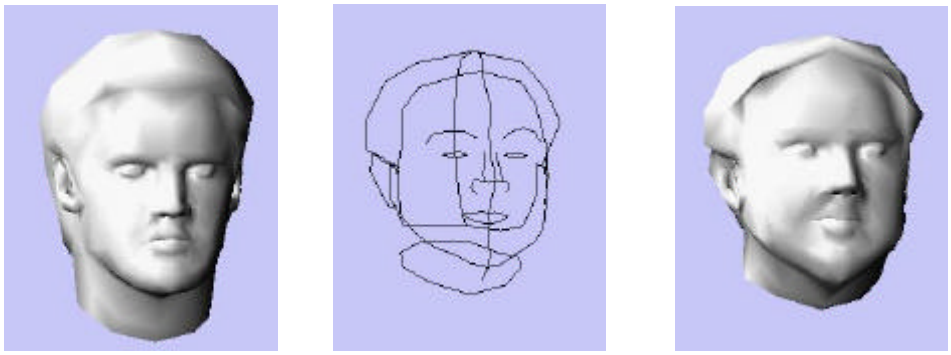
We have a certain set of 3D feature points, which has about 160 points. The question is how to deform a generic model, which has more than a thousand points to make an individualized smooth surface. Distance-related functions have been employed by many researchers [4][7][9] to calculate displacement of non-feature points related to the feature points detected. More sophisticated solution is to non-linear deformations as opposed to generally applied linear interpolation, which can give a smooth resulting surface. It uses 3D feature points as a set of control points for a deformation. Then the deformation of a surface can be seen as an interpolation of the displacements of the control points. Before applying deformation method, we use global transformations (translation, and scaling) to bring obtained 3D feature points to generic model's 3D space.

2.3.1 Dirichlet Free-Form Deformations (DFFD)

Free-Form Deformations (FFD)[17] belong to a wider class of geometric deformation tools. However FFD has a serious constraint in that control points boxes have to be rectangular, which limits the expression of any point of the surface to deform relative to the control points box. Farin [19] extends the natural neighbors' interpolant based on the natural neighbors' coordinates, the Sibson coordinate system [18] based on Voronoi/Dirichlet and Delaunay diagrams [19] for a scattered data interpolant, using the support for a multivariate Bezier simplex [21]. He defines a new type of surfaces with this extended interpolant called Dirichlet surfaces. Combining FFD's and Dirichlet surfaces leads to a generalized model of FFD's: Dirichlet FFD's or DFFD's.

In the Dirichlet-based FFD approach, any point of the surface to deform located in the convex hull of a set of control points in general position, is expressed relative to a subset of the control points set with the Sibson coordinate system. One major advantage of this technique is that it removes any constraint on the position and topology of control points. It also removes the need to specify the control lattice topology [23]. One control point is defined at the position of each surface point, so that any displacement applied to the control point will also be applied to the surface point.

Therefore DFFD is used to get new geometrical coordinates for a modification of the generic head on which are situated the newly detected feature points. More in detail about DFFD algorithm we are using is described in reference [23], where DFFD is used for hand deformation. The same method is used for head modification here. All points on the head are located utilizing the feature points as constraint control points for DFFD. Since our feature points includes the front and side silhouette, the convex hull of control points contains most points on a 3D head, but there can be some missing points outside the convex hull. So we add 27 extra points on a rectanglar box surrounding the 3D head, which are control points for DFFD, but not feature points from image data. After applying DFFD on a head, eyes and teeth are recovered to the original shape since modifications may create unexpected deformation for them and then the correct positions of the eyes and teeth are assured through translation and scaling appropriate to new head. See Figure 4. This is a rough matching method; it does not attempt to locate all points on the head exactly, in constrast to range data from laser scanners or stereoscopic cameras, which is supposed to have more accurately matching shape. The result is, however, quite respectable considering the input data (pictures from only two views). Most important, it greatly limits the size of the data set associated with an individual head, as is necessary to accelerate animation speed. The problem of how to reduce the size of the data from range data equipment for animation purposes has not been satisfactorily solved.



(a) The generic model (b) Features in 3D (c) After DFFD

Figure 4: The shape modification using DFFD.

Our system provides a feedback modification of a head between feature detection and a resulted head.

2.3.2 Fine modification when range data available

(a) (b) (c)

Figure 5: (a) Voronoi triangles of feature points. (b) Points collected for fine modification using Voronoi triangles in (a). (c) After fine modification. Compare the chin lines in Figure 4(c).

We apply one more step to get accurate positions for not only feature points but also other points if the range data is available. Some of feature points are chosen for a fine modification. Not every feature point is candidate for fine modification since some points in range data, for example hair and ear regions, have non-proper position value. We collect feature points only when their positions on a modified head are inside certain limitation of corresponding points on original range data. Then we calculate Voronoi triangles of chosen feature points and collect non-feature points inside Voronoi

triangles. We check it using Barycentric coordinate. The Voronoi triangles and collected points on a surface are shown in Figure 5 (a) and (b). Then three kinds of projection of points are used depending on a normal vector of a corresponding triangle. It can be projected on either *(x,y), (y,z),* or *(z,x)* plane. Again the nearest points in 2D or 3D are calculated and a reverse distance function as in previous process for depth calculation is applied to get the corresponding accurate coordinate in a range data. Here we apply only reasonably big triangles (for example we consider a triangle is reasonably big if the area of it is above average of area of every triangle) for fine modification since if we apply projection to every triangles, it sometimes creates irregular shape, specially eye region. Figure 5 (c) is the final result after fine modification. Compare it with Figure 4 (c), especially in the parts of the chin lines. It recovered the original shape while the modification with only defined feature points is not able to do it due to lack of feature lines on the region. Figure 5 (c) has also a smoother surface. Without the first modification step using DFFD with feature points, it is difficult or takes much longer time to get detailed matching. Two steps modification approach, first rough matching with FFD and then fine matching with simple projections, obtains the result easier and guarantee the convergence for the fine shape.

2.4 Automatic Texture Mapping

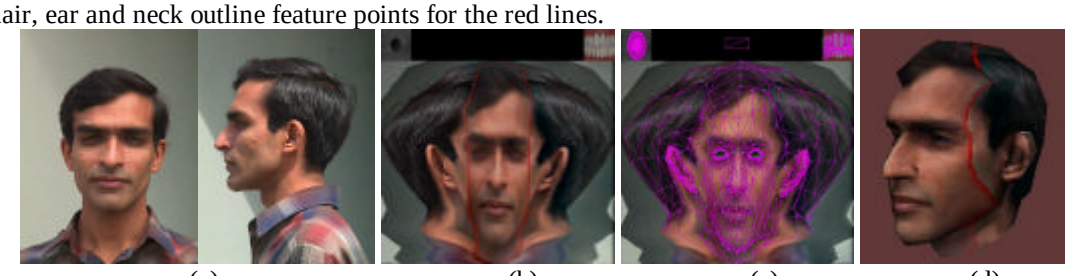
To increase realism, we utilize texture mapping. Texture mapping needs both a texture image and texture coordinates, where each point on a head needs one or several coordinates. The input images are not appropriate to be used for texture mapping; hence, generation of texture image is processed. In this section, we focus on orthogonal picture input case.

For the range data, we can extend some part of a front image. Since we know the location of boundary between hair and face from feature detection step, we produce new image with hair image and use it as side image like orthogonal photo input. Then we can apply the same algorithm as orthogonal picture input case.

2.4.1 Texture Image Generation

2.4.1.1 Image deformation

A front view is kept as it is and a side view is deformed to be connected to the front view. We define two sets of feature points (one for left part and the other for right part) on front view, intending to keep original (high) resolution for major part on a front view, which lies between two feature lines. There is a corresponding feature line on a side image. We use piecewise linear deformation for the side image to transform the feature line to the corresponding one on the front view. We use the side view for the right view and deform it with transformation to match to the right feature line on the front image. For a left image, we flip a side image across to the vertical axis and deform it with relation of the left feature line on the front image. **Figure 6** (b) shows how the side view is deformed to be connected to the front view with input images in (a). Red lines on **Figure 6** (b) are the defined sets of feature points, which can be changed easily. If the side image is not good enough, we take the hair, ear and neck outline feature points for the red lines.



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

Figure 6: (a) Input images (b) Combination with geometrical deformation. (c) Texture coordinates. (d) A textured head with a red line on it.

2.4.1.2 Multiresolution image mosaic

The three resulting images after deformation are merged using pyramid decomposition method using the Gaussian operator [6]. We utilize REDUCE and EXPAND operators to obtain the G_k (Gaussian image) and L_k (Laplacian image), which are described in detail in Section 2.2.1. Then we combine three L_k images on each level on the defined curves, which are feature lines used in Section

2.4.1.1. Then the combined image P_k is augmented with S_{k+1} to get S_k, which is the result for each level. The final image is S_0. Figure 7 shows the whole process of the Multiresolution technique to merge three images. **Figure 8** shows an example from level 3 to level 2. This Multiresolution technique is very useful to remove boundaries between the three images. Images of the eyes and teeth are added automatically on top to obtain full eyeball and teeth images as shown in **Figure 6** (b) and (c).

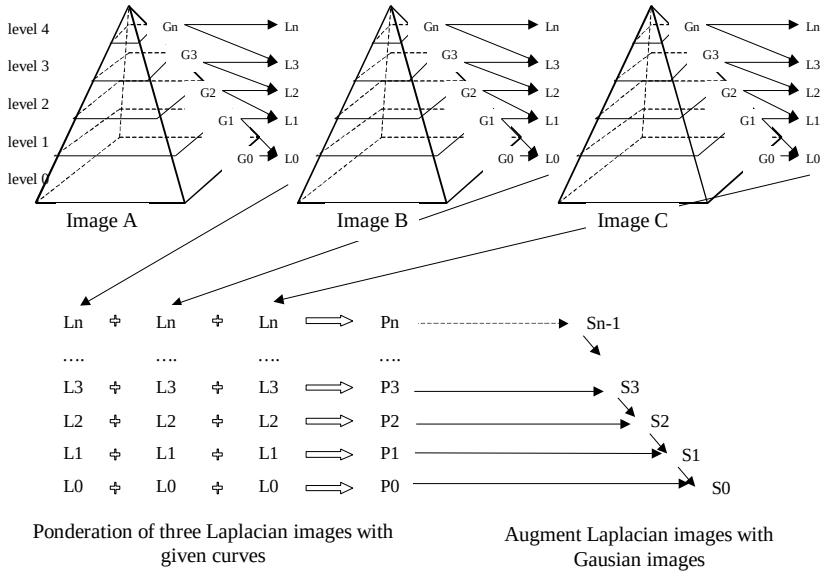


Figure 7: Multiresolution technique for image mosaic using pyramid decomposition.

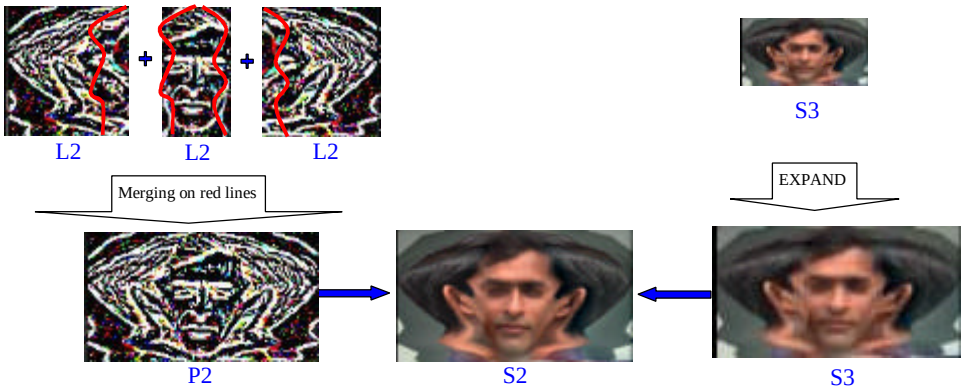


Figure 8: **Multiresolution techniques with an example from level 3 to level 2.**

2.4.2 Texture Fitting

To give a proper coordinate on a combined image for every point on a head, we first project an individualized 3D head onto three planes, the front (x, y), the left (y, z) and the right (y, z) planes. With the information of feature lines, which are used for image merging, we decide on which plane a 3D-head point is projected. The projected points on one of three planes are then transferred to one of 2D-feature points spaces such as the front and the side in 2D. Then they are transferred to the 2D-image space and finally to the combined image space. The eyes and teeth fitting process are done with predefined coordinates and transformation related to the resulted texture image size, which is fully automatic after one process for a generic model. **Figure 6** (c) shows the texture coordinates laid on the texture image while next image (d) shows how it results on a head with a red line (feature points used for image combination) on it. The left part of the red line in **Figure 6** (d) is from the front view image and the right part from the side (right) view image.

3 Results and animation

A final textured head is shown in **Figure 9** (b), where input images are shown in **Figure 6** (a). It has proper shape and texture on backside too, which makes full rotation of a head.

Figure 10 shows a result with range data. It shows several animated faces with given expression parameters. The head in Figure 3 (a) has a nice shape, but it does not have a good shape for back part and ear part. In addition, it does not have any functional information how to animate. Through our process using two-steps modification with a given generic model (adapting the shape and passing an existing animation structure), it keeps nice shape and enables to animate with given expression.

We employ an approach for deformation and animation of a face, based on pseudo muscle design. Muscle actions to stretch, expand, and compress the inside geometry of face are simulated by displacing or changing the weight of the control points. This deformation model for simulating muscle is simple and easy to perform, natural and intuitive to apply and efficient to use for real time applications. In our multi-level approach [15], motion parameters as Minimum Perceptible Action (MPA) are used to construct practically any expression and viseme. At the highest level, animation is controlled by a script containing speech and emotions with their duration.

Figure 9: **Rotation of a reconstructed head from picture data. A backside has proper texture too.**

Figure 10: The result of giving functional structure on a range data. This reconstructed head is able to animate with given expression parameters.

Other examples from orthogonal picture data and range data are shown in **Figure 11**. Females and males in several different ethnic groups covering wide range of ages are reconstructed using only two picture images being modified from only one generic model in Figure 4 (c). The resulted head from range data shares the same generic model. As our generic model has short hairstyle and has only a few polygons on hair region, a reconstructed virtual human can have different hairstyle from the hairstyle in the input image, especially for the case of long hair. Some models are not easy to detect

features even in semiautomatic way. In this case, interactive feature detection is only one solution. In practice, the whole process takes a few minutes including the time for image calculation and interactive correction if necessary. For a child in **Figure 11**, it is not very easy to keep him in the same position without movement even for one second. Our algorithm allows a quite flexible solution to adapt this kind of case. The total amount of data for the reconstructed heads in *OpenInventor* format is small considering their realistic appearance. The size of Inventor format (corresponding to *VRML* format) is about 200 KB. The texture image is stored in *JPEG* format and has sized about 5~50 KB depending on the quality of pictures; all examples shown in this paper have size less than 45 KB. The number of points on a head is 1257, where 192 of them are for teeth and 282 of them are used for the two eyes. This leaves only 783 points for the individualization of the face surface aside from eyes and teeth.

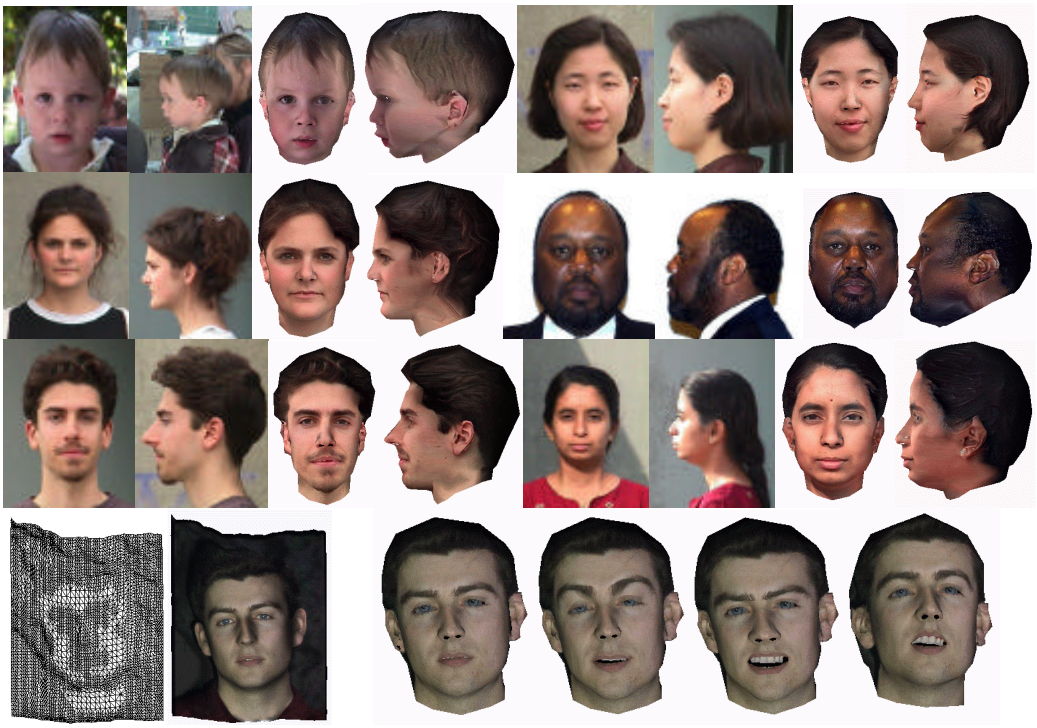


Figure 11: Examples of reconstructed heads from pictures or from range data. They can be animated immediately in virtual world.

4 Conclusion & Future Research

We described a fast and efficient method to create a virtual animatable face from two kinds of input such as orthogonal pictures and range data. *One* methodology is applied to two different kinds of input to have a final animatable face. One generic model with animation structure is prepared in advance to make any individualized head. It is based on feature detection both for modification of shape and for texture mapping. Using DFFD (and fine modification for range data), the modified shape has smooth surface applied to several ethnic groups in various aging. A seamless texture image generation combining two separate images is presented. Two steps modification for range data input, which is first based on features and second based on detailed matching, guarantees the convergence of a generic model to match the data to get higher accuracy. The difficulties of automatic feature detection come from various colors depending on each person, many shadows on face and also wrinkles and spots. Our semiautomatic method brings reliable results. It has shown its efficiency and robustness in a 5-days demonstration in ORBIT'98 in Basel, Switzerland, where more than 65 people were reconstructed from orthogonal picture data. The reconstruction from two pictures needs low-cost commercial equipment and takes just a few minutes. The result from the range data has the better visual result for output, but it also has limitation for equipment since it requires usually either expensive or sophisticated equipment.

An extension of our face reconstruction to individualized body reconstruction from two views in our ongoing research topic.

5 Acknowledgment

The authors would like to thank every member of MIRALab, especially Laurent Moccozet, Pierre Beylot, and Hyewon Seo. We are grateful to Chris Joslin for proof reading this document. This project is funded by an European project eRENA and Swiss National Research Foundation.

6 Reference

[1] Meet Geri: The New Face of Animation, Computer Graphics World, Volume 21, Number 2, February 1998.
[2] Exhibition On the 10th and 11th September 1996 at the Industrial Exhibition of the British Machine Vision Conference.
[3] http://www.turing.gla.ac.uk/turing/copyrigh.htm
[4] Takaaki Akimoto, Yasuhito Suenaga, and Richard S. Wallace, "Automatic Creation of 3D Facial Models", *IEEE Computer Graphics & Applications*, Sep., 1993.
[5] P. Beylot, P. Gingins, P. Kalra, N. Magnenat Thalmann, W. Maurel, D. Thalmann, and F. Fasel, "3D Interactive Topological Modeling using Visible Human Dataset", *Computer Graphics Forum*, 15(3):33-44, 1996.
[6] Peter J. Burt and Edward H. Andelson, "A Multiresolution Spline With Application to Image Mosaics", *ACM Transactions on Graphics*, 2(4):217-236, Oct., 1983.
[7] Horace H.S. Ip, Lijin Yin, "Constructing a 3D individual head model from two orthogonal views", *The Visual Computer*, Springer-Verlag, 12:254-266, 1996.
[8] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, pp. 321-331, 1988.
[9] Tsuneya Kurihara and Kiyoshi Arai, "A Transformation Method for Modeling and Animation of the Human Face from Photographs", *Proc. Computer Animation'91*, Springer-Verlag Tokyo, pp. 45-58, 1991.
[10] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters, "Realistic Modeling for Facial Animation", *Computer Graphics (Proc. SIGGRAPH)*, pp. 55-62, 1996.
[11] Marc Proesmans, Luc Van Gool. "Reading between the lines - a method for extracting dynamic 3D with texture", *Proc. of VRST'97*, pp. 95-102, 1997.
[12] http://www.viewpoint/com/freestuff/cyberscan
[13] LeBlanc. A., Kalra, P., Magnenat-Thalmann, N. and Thalmann, D. "Sculpting with the 'Ball & Mouse' Metaphor", *Proc. Graphics Interface '91*. Calgary, Canada, pp. 152-9, 1991.
[14] Lee W. S., Kalra P., Magenat Thalmann N, "Model Based Face Reconstruction for Animation", *Proc. Multimedia Modeling (MMM) '97*, Singapore, pp. 323-338, 1997.
[15] Kalra P, Mangili A, Magnenat-Thalmann N, Thalmann D, "Simulation of Facial Muscle Actions Based on Rational Free Form Deformations", *Proc. Eurographics'92*, pp. 59-69, NCC Blackwell,1992.
[16] P. Fua, "Face Models from Uncalibrated Video Sequences", *Proc. CAPTECH'98*, pp. 215-228, 1998.
[17] Sederberg T. W., Parry S. R., "Free-Form Deformation of Solid Geometric Models", *Computer Graphics (Proc. SIGGRAPH'86)*, pp. 151-160, 1986.
[18] Sibson R., "A Vector Identity for the Dirichlet Tessellation", *Math. Proc. Cambridge Philos. Soc.*, 87, pp. 151-155, 1980.
[19] Aurenhammer F., "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure", *ACM Computing Survey*, 23, 3, September 1991.
[20] Farin G., "Surface Over Dirichlet Tessellations", *Computer Aided Geometric Design*, 7, pp. 281-292, North-Holland, 1990.
[21] DeRose T.D., "Composing Bezier Simplexes", *ACM Transactions on Graphics*, 7(3), pp. 198-221, 1988.
[22] P. Fua, "From Multiple Stereo Views to Multiple 3-D Surfaces", *International Journal of Computer Vision*, 24(1), pp. 19-35, 1997.
[23] L. Moccozet, N. Magnenat-Thalmann, "Dirichlet Free-Form Deformations and their Application to Hand Simulation", *Proc. Computer Animation'97*, IEEE Computer Society, pp. 93-102, 1997.