Symmetric deformation of 3D face scans using facial features and curvatures

By Jeong-Sik Kim and Soo-Mi Choi*

Facial symmetry is one of the number of aesthetic traits associated with health, physical attractiveness and beauty of a person. In this paper, we present a novel method for enhancing the symmetry of a scanned 3D face automatically. To handle detailed areas of the face, we developed a new local 3D shape descriptor based on facial features and surface curvatures. Our shape descriptor can improve the accuracy when deforming a 3D face towards a symmetric configuration, because it provides accurate point pairing with respect to the plane of symmetry. It can also reduce the computational complexity by dividing the given face into partitioned slices and facial feature regions. In addition, we use point-based representation over all stages of symmetrization without generating a consistent triangle mesh or texture parameterization, which makes it much easier to support discrete processes such as symmetric deformation and to be combined with more sophisticated manipulations later such as cutting of surfaces. Finally, we performed a statistical analysis to assess subjects' preference for the symmetrized faces by our approach. Copyright © 2009 John Wiley & Sons, Ltd.

Received: 25 March 2009; Accepted: 25 March 2009

KEY WORDS: 3D symmetrization; face symmetry; 3D facial features; surface curvatures

Introduction

The human face plays a significant role in important tasks such as representing our feelings and choosing our mate.1 Many researchers in cognitive psychology and the social sciences have studied to find out the relationship between face preferences and factors, such as skin color and facial shape.2–4

Symmetry is an essential aspect of the human face, and a ubiquitous concept in many different other aspects of life. It is a significant component of visual perception and aesthetic sense.5 The symmetry of the frontal view of the face has been used as an important indicator in biometrics for a long time.6 It is now easy to construct 3D face models using scanning devices, resulting in an increasing need for automatic detection of symmetry. In addition, we would like to be able to make faces symmetrical for various applications, such as face recognition, virtual avatar creation, etc. A human face can have a range of shapes, reflecting sex, race, age, inherited characteristics. Therefore, a system to make faces symmetrical needs to include a method for detecting a wide range of symmetries, which can be affected by the scale of a face, its structural properties, and local surface shape. Recent research on making 3D models symmetric has focused on detecting partial and approximated symmetries in a shape and then constructing a simplified 3D model. This approach cannot capture small-scale symmetries such as facial features.

We propose an efficient method for automatic symmetrization of a 3D face scan. We make several contributions towards design and implementation of the 3D face symmetrization. This is more accurate than more generalized approaches, as we compute a set of symmetry pairs using a 3D shape descriptor which consists of 3D facial feature points and curvatures. We can then divide a face into feature-based and region-based components, and construct as the basis for symmetrization. Our approach uses only point-based structures, which allows it to be easily integrated with more sophisticated 3D manipulations such as deformation and cutting of surfaces. We analyze our method using a statistical analysis.

*Correspondence to: Professor S.-M. Choi, Department of Computer Engineering, Sejong University, 98 Gunja-Dong, Gwangjin-Gu, Seoul 143-747, Korea.
E-mail: smchoi@sejong.ac.kr

Copyright © 2009 John Wiley & Sons, Ltd.
**Related Work**

Early work on symmetry concentrated on finding perfect symmetries in point-sets.\(^7,8\) Sun *et al.*\(^9\) detected global reflective and rotational symmetries using an extended Gaussian image. Kazhdan *et al.*\(^10\) introduced a shape descriptor that concisely encodes global reflective and rotational symmetries, and used it for shape retrieval from database of 3D models. But these methods only consider global symmetry. More recently, Podolak *et al.*\(^11\) proposed a method of detecting the main reflective symmetry planes in a shape using a planar reflective symmetry transform. This captures the degree of symmetry of arbitrary shapes, but cannot detect local symmetries. Mitra *et al.*\(^12\) introduced an improved approach to detect partial and approximated shape symmetries in 3D models using surface curvature and Euclidean transformations. The result is a symmetry graph.

Various symmetrization techniques are known in classical geometry.\(^13,14\) In 2007, Mitra *et al.*\(^15\) proposed a symmetrization algorithm for geometric objects which enhances approximate symmetries in a model while making only small changes to its shape. This algorithm uses a local shape descriptor based on curvatures, geometric information, and clustering technique. But curvature-based pairing algorithm does not find corresponding points in small-scale features or small differences in local geometry. Like Mitra *et al.*,\(^15\) we use point-pair correspondences and a coupling of transformation space and object space to symmetrize the parts of the surface. But we find a more accurate set of symmetric point-pairs using a 3D shape descriptor based on surface curvatures and facial feature points. We simplify the symmetrization process by using point-based techniques rather than a mesh.

**Overview**

Our face symmetrization process, shown in Figure 1, begins with the acquisition of a 3D face scan. From this a 3D shape descriptor is automatically created from surface curvatures and feature points. Using this descriptor, the face is then partitioned into facial feature regions, as well as horizontal and lateral slices. Then we compute a set of point-pairs from horizontal slices through the face, as well as dealing individually with the eyes, eyebrows, nose, mouth, and ears. We then compute a transformation for each point-pair, and map all the point-pairs into transformation space. We can then symmetrize the local parts of the surface representing the slices and the facial feature regions by moving each pair’s transformation to the ideal symmetry transformation with respect to the local plane of symmetry in the transformation space. We then perform a global symmetrization step in which we align the ideal symmetry planes assigned to the horizontal slices to a fixed \(yz\)-plane. Finally, we use optimization methods to relax the surface and high-frequency noise and adjust irregularly distributed surface points.

**A 3D Shape Descriptor for Facial Symmetry**

In order to symmetrize small-scale features as well as the global shape of a face, we need a local 3D shape descriptor that reflects not only the unique structural properties of the face but also various Euclidean transformations, such as translation, rotation, reflection, and uniform scaling, for each point of the input scan. We therefore propose a new local 3D shape descriptor based
on curvatures and 3D facial feature points. We will first introduce the shape representation that we use, and then new shape descriptor.

**Fundamental Shape Representation**

A 3D face obtained from a scanner consists of a point cloud. This can be triangulated, but we use simple surface elements (surfels) from computer graphics and their one-ring neighborhoods as our fundamental shape representation. A surfel is a point with various attributes such as position, normal, and shading factors. It allows efficient 3D geometric manipulation and rendering. The one-ring neighborhood encodes a symmetrical relationship between neighboring surfels, which facilitates geometric manipulation even on low-resolution or irregular surfels. To capture the neighborhood proposed by Guennebaud et al., we first determine Euclidean neighbors of each point on the surface, which are the points within a range of a point.

\[
N_P = \{ i | p_i \in P, p_i \neq p, \| p - p_i \| < r \}
\]

Then we remove any indices do not satisfy the following restriction:

\[
\left| (n_0 + n_1) \cdot \frac{p_0 p_1}{\| p_0 p_1 \|} \right| > 1 + n_0 \cdot n_1
\]

where \( n_0 \) is the appropriate normal at the current point \( p_0 \) and \( n_1 \) is the normal at the Euclidean neighbor \( p_1 \) of \( p_0 \). We complete the pertinent one-ring neighborhood \( N_P \) by removing all false neighbors of \( p \).

At a general point \( p \) on a surface, a normal plane contains a unique direction tangent to the surface and cuts the surface in a plane curve. Different normal planes in general have different curvatures. The principal curvatures at \( p \), denoted by \( k_1 \) and \( k_2 \), are the maximum and minimum values of this curvature. We use principal curvatures for computing the local shape index in the automatic extraction of 3D face feature points, and also to control the pruning of unnecessary point samples in the detection of symmetrical point-pairs. We find the principal curvatures using an algorithm proposed by Rusinkiewicz, which is robust against the artifacts caused by scanning. The input to the original version of this algorithm is the vertex normals of a triangular mesh model, but we have applied it to a set of point samples using one-ring neighborhoods.

**Automatic Extraction of 3D Facial Feature Points**

Facial feature extraction is important in many face-related applications. But current applications still encounter difficulties in handling facial variations due to head poses, lighting conditions, and facial expressions, so there have been many approaches to overcome their problems using a multi-modal scheme to integrate 3D and 2D information from a face scan. We aim to detect 19 facial feature points and two bounding boxes surrounding the eyebrow regions, combining 3D (a range image and principal curvatures) and 2D (an intensity image) information. As shown in Figure 2, our method consists of the following steps:

1. Generation of a range image and an intensity image.
2. Detection of facial regions, such as eyes, nose, and mouth in the intensity image.
3. Finding the tip of the nose in the range image by cross profile analysis.
4. Extraction of 18 further feature points using curvatures and a Harris corner algorithm.
5. Finding the eyebrow regions by intensity-based segmentation.
6. Recovering the 3D facial surface points from the extracted 2D feature positions.

The tip of the nose is located by analysis of vertical and horizontal profiles of the range image, which consists of depth values \( z(c, r) \), where \( c \) and \( r \) indices of the columns.

Figure 2. 3D facial feature points extraction diagram.
and rows, respectively. The upper row of Figure 3 illustrates the main steps in this process. The tip of the nose can be expected to be close to the plane of vertical symmetry, and points on the nose ridge will provide many of the extreme $z$ values along horizontal cross-sections of the face. For each row, we therefore find the pixel position with the maximum $z$ value, as shown in Figure 3(a). Next, the number of these positions is counted for each column, producing a histogram. The column corresponding to the peak of the histogram becomes center-line shown in Figure 3(b). The search for the tip of the nose can now be reduced to the points along this mid-line. From these points, we determine the depth profile shown in Figure 3(c), in which the bridge of the nose can be expected to correspond to a strong consecutive increase in $z$ values. We now determine the gradients $g(r) = z(r + 1) - z(r)$, where $r$ is the row index, from the depth values of the profile and the number of consecutive positive signs (run-length) of the gradients for each column. The tip of the nose can then be found at the peak at which the run-length reaches a local maximum shown in Figure 3(d). We find the extreme tip by analyzing the profile at the position of each peak along each row.

Then we extract other 18 feature points from the intensity image: eight on the eyes, two on the nose, four on the mouth, and four on the facial outline. This process, which is shown in the lower row of Figure 3, starts with the application of the AdaBoost cascade classifier$^{20}$ to the intensity image to detect the feature regions. Then we compute the shape index and the curvedness, which respectively represent local surface shape and scale, at each vertex. These shape descriptors are robust under Euclidean transformations, and are computed from the maximum and minimum principal curvatures $k_1(p)$ and $k_2(p)$, as follows:

$$S(p) = \frac{1}{2} - \frac{1}{\pi} \arctan \frac{k_1(p) + k_2(p)}{k_1(p) - k_2(p)}, \quad \text{C}(p)$$

$$= \frac{2}{\pi} \ln \sqrt{\frac{(k_1(p))^2 + (k_2(p))^2}{2}}$$

(3)

$$R(p) = \frac{\frac{\partial^2 I}{\partial x^2} \frac{\partial^2 I}{\partial y^2} - \left( \frac{\partial^2 I}{\partial x \partial y} \right)^2}{\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}}$$

(4)

The shape index takes a value in the interval $[0,1]$, while curvedness is in the range $[0,10]$. We use $\text{SC}(p)$, which is the product $S(p)$ and $\text{C}(p)$, to extract local feature points. $\text{SC}(p)$ in each search region is identified as a feature point because many facial features, such as the corners of the eyes, nose, and the mouth, are cup-like.
shapes with low SC(\(p\)) values, within the interval \([0, 0.25]\). We also use the Harris corner-detection algorithm to find points with strong corner-like patterns in the intensity image. The algorithm computes the Hessian matrix \(H\) of an intensity image function \(I\) in an eight-connected local neighborhood of point \(p(x, y)\). If the two eigenvalues of \(H\) are large, then a small motion in any direction will cause a significant change of gray level at point \(p\). This indicates that \(p\) is a corner. Equation (4) shows the corner response function \(R(p)\). A total of 18 feature points are found where the points with low values of \(SC(p)\) and the corners coincide.

We do not try to find feature points in the eyebrow regions. Instead we create two boxes that bound the eyebrows using intensity-based segmentation. Since the eyebrows are located above the eyes and can be easily discriminated from facial skin by gray-level intensity, we first separate sub-images that include the eyebrows. Then we extract pixels which have the gray-level intensity of the eyebrows. Finally, we compute axially aligned boxes around these pixels. We now can recover 3D points on the surface from the extracted 2D feature positions using an OpenGL’s \texttt{gluUnProject\(()\)} function which inverts the projection calculation instead of going from a 3D point to a 2D point. To find a corresponding 3D point, it requires grabbing three matrices from the OpenGL system: model-view, projection, and viewport.

### Segmentation of 3D Facial Regions

A lot of research on facial symmetry has concentrated on detecting bilateral symmetry using an axis or plane. For 3D face symmetrization, it is not enough to identify various types of overall symmetry. Instead, we segment a 3D face model into both vertical and lateral parts, and into facial feature regions using the face feature points. This allows us to symmetrize different face models locally and globally in different directions.

We will consider three types of face: (1) a face with a straight center-line; (2) a face with a bent center-line; and (3) a face with a deformed lower jaw. Our method aims to discriminate between these types and analyze their individual structural characteristics to enhance the accuracy of the face symmetrization. So we need to obtain some sub-symmetry planes from the segmented facial parts, not just a single symmetry plane.

We start by dividing the face into three horizontal slices: an upper part \(Part^u\) (forehead, eye-brows, and eyes), a middle part \(Part^m\) (nose and ears), and a lower part \(Part^l\) (mouth and jaw). Then we again divide each part into a left and right part, cutting the 3D face model into a total of six parts. Figure 4(a) shows how we divide up a 3D face. We first define two planes, \(\pi_{hu}\) and \(\pi_{hl}\). The upper plane \(\pi_{hu}\) is constructed by finding the mid-point \(m_1\) between \(e_3\) and \(e_7\), and then by projecting the vector \(\bar{m}_1\) on to the \(xy\)-plane to create the vector \(v_u\), and the vector \(v_u^\perp\) perpendicular to \(v_u\) on the \(xy\)-plane. Next, we compute the angle \(\theta_{hu}\) between the normal vector \(n_u\) to the \(xz\)-plane and \(v_u^\perp\). The upper plane \(\pi_{hu}\) is defined by \(m_1\) and \(n_{hu}\), rotating the normal vector \(n_u\) by \(\theta_{hu}\) about the \(z\)-axis. The lower plane \(\pi_{hl}\) is defined a similar way by finding the mid-point \(m_2\) and a vector \(v_l\). Then we calculate the vector \(v_l^\perp\) which is perpendicular to \(v_l\) on the \(xy\)-plane. Next we find the angle \(\theta_{hl}\) between the normal vector \(n_l\) and \(v_l^\perp\). The lower plane \(\pi_{hl}\) is defined by \(m_2\) and \(n_{hl}\), rotating the normal vector \(n_l\) by \(\theta_{hl}\) about the \(z\)-axis. Finally we can classify all the points into the three parts.

![Figure 4. Segmentation of facial parts using separating planes.](image-url)
As shown in Figure 4(b), the upper part \( Part''u \) is separated into left and right parts by the plane \( \pi_{vu} \), which is created from a normal vector \( \mathbf{n}_u \) and an angle \( \theta_{vu} \) between \( \mathbf{n}_u \) and a vector created by projecting a vector \( \mathbf{m}_1 \mathbf{b}_1 \) on to the \( xy \)-plane. The plane \( \pi_{vu} \) is defined by \( \mathbf{m}_u \), which lies on it, and the normal vector to \( \pi_{vu} \) which is created by rotating the normal vector of the \( xz \)-plane \( \mathbf{n}_u \) by an angle \( \theta_{vu} \) about the \( z \)-axis. Finally, using the equation of \( \pi_{vu} \), each point in \( Part''u \) can be classified into the left-hand part \( Part''ul \), for which \( \pi_{vu} \) is positive, or into the right-hand part \( Part''ur \). We can divide \( Part''u \) and \( Part''l \) in a similar way by computing \( \pi_{vu} \) from \( \mathbf{n}_{m1} \) and \( \mathbf{m}_1 \), and \( \pi_{v1} \) from \( \mathbf{n}_{m1} \) and \( \mathbf{k}_3 \).

To segment the facial feature regions from the face scan, we compute bounding boxes surrounding the main facial features, such as the eyes, nose, mouth, and ears. In the case of the mouth region, we first define tangential planes \( h_{upper}, h_{lower}, v_{left} \) and \( v_{right} \) from the four corner points shown in Figure 5. In order to create these planes, we first compute two lines \( h_{mid} \) and \( v_{mid} \), which are defined by connecting each point-pair, such as \( (I_u, I_d) \) and \( (I_l, I_d) \). Then \( h_{upper} \) and \( h_{lower} \) are constructed from two lines parallel to \( h_{mid} \), which intersect at the upper and lower point \( I_1 \) and \( I_3 \), respectively. The planes \( v_{left} \) and \( v_{right} \) can also be defined in the same way.

### 3D Face Symmetrization

We propose three complementary techniques for face symmetrization: (1) We use a local 3D shape descriptor based on the curvatures and the 3D facial feature points to obtain an accurate point-pair mapping of the set of surface points set. (2) We apply different types of point-pair mapping to the feature regions and the separation plane parts. (3) We simplify the overall process of symmetrization by using ideal symmetry planes.

#### Detecting Symmetry Point-Pairs

Given a set of surface points \( P \), we discard an unnecessary point \( P \), by applying a threshold \( g < 1 \) to the ratio of the maximum curvature \( k_{i,1} \) and the minimum curvature \( k_{i,2} \) of \( P \), if \( |k_{i,1}/k_{i,2}| \geq g \). To reduce the amount of computation, we only consider meaningful candidate pairs with distinct principal curvatures. In addition, we find more precise point-pair candidates from the pruned set of point samples by comparing local frames \( (c_{i,1}, c_{i,2}, \mathbf{n}_i) \) defined by the principal directions \( c_{i,1} \) and \( c_{i,2} \) and the normal vector \( \mathbf{n}_i = c_{i,1} \times c_{i,2} \). We then apply our pairing algorithm based on the 3D facial feature points. We consider the parts \( Part''u \), \( Part''l \), and \( Part'l \) in turn. We make one of each part the source point-set \( P_{Source} \) and the other side becomes the target point-set \( P_{Target} \). Since the number of points in \( P_{Source} \) and \( P_{Target} \) do not necessarily match, we select the part that contains fewer points from among the two parts to be \( P_{Source} \). Then we find the set of one-to-one symmetry pairs. In this process, points not selected from \( P_{Source} \) are excluded from the symmetrization.

Figure 6(a) shows the procedure of detecting the point-pairs for the upper part \( Part''u \). The detection of symmetry pairs starts by computing the interval distances between the facial feature points projected on to the \( xy \)-plane, denoted by L1 to L4 and R1 to R4, and the slopes of the two lines created when the ideal symmetry planes \( \pi_{vu} \) and \( \pi_{vu} \) intersect the \( xy \)-plane are denoted by slope_{vu} and slope_{vu}. Then we find a set of symmetry point-pair candidates \( Q \) from the \( P_{Target} \) for each point \( p \) in the \( P_{Source} \). Next, we identify the corresponding sections R3 and L3, which contain of the point \( p \) and a mirror point \( p \), respectively, reflected by the symmetry plane \( \pi_{vu} \) from \( p \). Then we compute the distance ratio \( d_r = (L3/R3) / (H3/R3) \), slope_{vu} and slope_{vu}. After that, in order to match the relative position of the mirror point \( p \) in section L3 with the relative position of \( p \) in section R3, we compute the best paired-point \( q \) by adjusting the displacement of \( p \), using slope_{vu} and slope_{vu}. Finally, we search for the paired point which is closest to \( q \).

Computation of the facial feature regions is somewhat easier than the detection of the bilateral parts. We first divide each facial feature part into a left and right region using the symmetry plane associated with that part, as shown in Figure 6(b). Then we calculate the distances \( (W_L, W_R) \) and \( (H_L, H_R) \) in the directions of the \( x \)- and \( y \)-axes of the local frames of the two bounding boxes. Given these distances, we can now find a point \( q \) which has the ratio \( r_L = (W_q/W_L, H_q/H_L) \) that is closest to the ratio \( r_R = (W_p/W_R, H_p/H_R) \).

#### Local and Global Symmetrization

Our approach to local and global symmetrization builds on the approach proposed by Mitra et al., which...
couples an object space containing the positions of each set of paired-points with a transformation space in which we can create a transformation which maps one paired-point to the other. Any pair of points \((p, q)\) on the surface defines a unique reflection with respect to the bisector plane through \(m = (p + q)/2\) with normal direction \(n = (q - p)\). This transformation can be expressed as a point \(T = (d, \theta)\) in a 2D transformation space, where \(\theta\) is the angle with respect to some fixed reference plane, and \(d\) is the distance to the origin. If we now want to move the transformation \(T\) to \(T' = (d', \theta')\), which is the transformation defined by the ideal symmetry plane, the points \(p\) and \(q\) need to be displaced by some vectors \(d_p\) and \(d_q\), which are determined as follows:

\[
d_p = \frac{T'^{-1}(q) - P}{2}, d_q = \frac{T'^{-1}(p) - q}{2} \tag{5}
\]

Having determined the displacements of each pair, we use an objective function to determine the optimal symmetrizing transformation of the set of point-pairs \(\{(p_1, q_1), \ldots, (p_m, q_m)\}\). We find the transformation \(T\) and corresponding displacements that make the point-set symmetric with respect to \(T\), using the following symmetry cost function:

\[
E = \sum_{i=1}^{m} \left( \|d_{p_i}\|^2 + \|d_{q_i}\|^2 \right) = 2 \sum_{i=1}^{m} \|d_{p_i}\|^2 \tag{6}
\]

Mitra et al.\textsuperscript{15} updated the good pairs iteratively and determined the best transformation using a clustering technique. But we create a transformation in a pre-processing step.

Global symmetrization can now be performed by aligning the ideal symmetry planes \(\pi_{uv}, \pi_{vu}, \pi_{vu}\) and \(\pi_{uv}\) with the fixed \(yz\)-plane \(\pi_x\). This alignment is achieved by creating a global symmetrizing matrix \(T_{global}\) which is defined as follows:

\[
T_{global} = T^{-1}(R(T(P_i))), (k = u, m, l) \tag{7}
\]

We can apply \(T_{global}\) to each of the point-sets \(P_u, P_m, \) and \(P_l\), where \(T\) is a translation matrix which moves the point-set to the origin, and \(R\) is a rotation matrix which rotates the translated points about \(z\)-axis, aligns the plane associated with the translated points to the plane \(\pi_x\).

### Surface Refinement

Surface refinement consists of surface smoothing and interpolatory surface refinement. Its aim is to remove the surface artifacts caused by the symmetrization, and allow high-quality rendering of the symmetrized face. We employ a lambda-mu smoothing technique based on ideas to diffuse high-frequency errors in the surface.\textsuperscript{22} This uses a low-pass filtering of the points to eliminate high frequencies represented by small-scale displacements of neighboring points. An addition step eliminates the shrinking effect produced by many smoothing techniques. After the smoothing, we apply an interpolatory surface refinement\textsuperscript{17} to improve the distribution of the points. Given a set of surface points \(P\), we compute the one-ring neighborhood \(N_p\) and the implicit triangle fan formed by the sorted neighborhood at each point \(p\). Then, we interpolate the points using point-normal triangulation based on a Bézier triangle.

### Results and Discussion

We designed three experiments to evaluate our methods for 3D face symmetrization. (1) We assessed the accuracy with which facial feature points are extracted from a 3D face scan. (2) We evaluated the effect of our

---

Figure 6. (a) 3D facial feature- and curvature-based pairing between left- and right-hand parts. The surface is rendered by color-coding the principal curvatures of the points. \(P_{Source}\) and \(P_{Target}\) are the right and left parts, respectively. (b) Detection of point-pairs for the facial feature regions. The yellow line is the symmetry plane that divides the left- and right-hand regions.
local 3D shape descriptor, and the method of partitioning a face into the facial feature-based and separation plane-based regions on the accuracy of the local and global symmetrization and the point-pairing algorithm.

(3) We tested the hypothesis that our face symmetrization can enhance viewer preference using a statistical survey. In these experiments, we used a face scan from the Max Planck Institute for Biological Cybernetics. This consists of a $512 \times 512$ texture image and 75,870 surface points. We also generated two artificial faces by deforming this model. Figure 7(a) shows the three experimental face models: “face 1” has a straight center-line, “face 2” has a bent center-line, and “face 3” has a deformed lower jaw. We converted these faces into surfel models for symmetrization.

Figure 7(b) shows examples of the principal curvatures and the feature extraction results. Figure 7(c) shows the result of the segmentation procedure and subsequent processing. Figure 7(d) compares the curvature-based pairing with our technique using 3D facial feature points. In the curvature-based method each point-pair is computed using a threshold on the range of the curvature. This can enhance approximate symmetries in various types of 3D model, but it cannot capture small-scale features like eyes, eyebrows, and mouth. It can also cause points not included in a symmetry pair to be omitted because of the threshold value, as you can see the dotted black circle in the left-hand image of Figure 7(d). Otherwise, our pairing algorithm is more accurate.

Figure 8(a) shows the result of local and global symmetrization. On the left and middle images, we can see the results of local symmetrization in the horizontal, lateral, and facial feature regions. On the right images, we can see that the ideal symmetry planes separating the locally symmetrized point-pairs are well aligned to the $yz$-plane.

We attempted to assess whether our method enhances the accuracy of symmetrization of the local facial feature regions. The left images of Figure 8(b) compare the nose and mouth of face 2 before symmetrization with those of the symmetrized face, and suggest that local symme-

Figure 7. (a) Three experimental face models: face 1 is undeformed; face 2 has a bent center-line; and face 3 has a twisted lower jaw; (b) (left-hand images) color-coded by curvatures and (right-hand images) 19 facial feature points and bounding boxes extracted from the intensity and depth images; (c) face 2: (left) partitioned into six regions and (right) results of processing segmented regions, as labeled; (d) results of (left) curvature-based pairing and (right) our pairing technique.
trization was successful. The right images of Figure 8(b) compare curvature-based symmetrization with our method of symmetrization. Applying the curvature-based method to face 2 produced the poor result because of inaccurate pairing and artifacts introduced by the symmetrizing deformation, whereas our method creates a smooth and symmetric face. Figure 8(c) shows that symmetrization has restored face 2 and face 3, so that they resemble face 1, from which they were generated. This demonstrates the potential of our approach to enhance symmetry.

Figure 9 shows the effect of the two optimization techniques that we apply to the symmetrized faces to remove artifacts such as high-frequency ripples and unevenly distributed points.

We also examined the idea that our symmetrization enhances the appeal of faces to a panel of viewers, using an analysis of variance (ANOVA) technique. We looked specifically at two hypotheses: “Our symmetrization enhances people’s preference for a face model” and “The effect of symmetrization will depend on the combination of the facial features”. Then we built two datasets. The first set consists of face 1, face 2, and face 3 before symmetrization and after symmetrization. From these images, we created a second set of 24 pictures by separating and combining some facial feature regions, such as nose, mouth, nose-mouth, and mouth-jaw. We asked 35 subjects to look at these pictures. We considered face preference as a dependent variable and the combination of the local facial features as an independent variable. We asked the subjects to score...
their preference for each of the 30 pictures from 1 to 5, where 1 is the least, and 5 the most favorable score. Using the data that the subjects provided, we assessed each hypothesis separately, using the F-test, which is appropriate for ANOVA. We used the repeated measure ANOVA tool in the SPSS statistical software package.

Figure 10(a) shows how our symmetrization enhances the subjects' preferences for the three experimental faces. As we would expect, the subjects preferred “face 1” and disliked “face 2” and “face 3” before symmetrization. There was a statistically significant increase in preference for the faces after symmetrization. The $F_0$ values of all face types are larger than $F(1, 34) = 4.17$ when the level of significance is 5% ($F_{10}$: face 1 = 10.1, face 2 = 89.1, face 3 = 56.1). In addition, the standardized effect size $\eta^2$ is very large ($\eta^2$: face 1 = 0.23, face 2 = 0.72, face 3 = 0.62). These values are interpreted as small (0.01), medium (0.06), and large (0.14). Figure 10(b)–(d) show how the combination of the facial features (nose, mouth, nose-mouth, and mouth-jaw regions) influences on the viewers' preferences for three types. Results showed that there was also a statistically significant increase in preference. We identified that all $F_0$ values are larger than $F(4, 136) = 2.58$ when the level of

Figure 10. (a) The effect of the face symmetrization. A1 to A3 are respectively face 1, face 2, and face 3 before symmetrization, and B1 to B3 are the faces after symmetrization. (b–d) The effects of the combination of the local facial features. Abbreviations N, M, and J mean the nose, mouth, and the jaw region, respectively.
significance is 5% ($F_0$: face 1 = 5.1, face 2 = 3.0, face 3 = 2.6). However, the effect size was not large as compared to the symmetrization of a whole face, since faces are perceived holistically ($q^2$: face 1 = 0.13, face 2 = 0.08, face 3 = 0.07).

**Conclusions**

We have proposed a new method for more accurate symmetrization of local facial feature regions, as well as of the overall shape of a face, using a 3D shape descriptor based on 3D facial feature points and curvatures derived from 3D face scans. We evaluated the performance of our techniques by symmetrizing three face scan models, and compared our results to previous methods. We also showed that faces with their symmetry restored by our technique were more attractive to observers. Because we employ point-based techniques throughout our symmetrization process, we respect that it will be relatively easy to integrate it with more sophisticated 3D manipulations such as physical deformation and the cutting of surfaces. This work focused on guaranteeing the accuracy of symmetry point-pairs to enhance both local and global symmetrization. We need to consider more empirically how to preserve the characteristic of the main features of a face. We are also planning to assess the performance of our approach more extensively using a variety of scanned faces.

**ACKNOWLEDGMENTS**

The authors give special thanks to Professor Markus Gross and all the members of Computer Graphics Lab at ETH Zurich. This work was supported in part by the Seoul R&D (Research and Business Development) program (10581) and in part by a grant from Korea Science and Engineering Foundation and Swiss National Science Foundation (IZAKZ0-124034).

**References**

Authors’ biographies:

Jeong-Sik Kim is currently a Post-doctoral Researcher in the Center for Computer Graphics and Virtual Reality at Ewha University of Korea. He received his B.S., M.S. and Ph.D. degrees in Computer Engineering from Sejong University of Korea in 2000, 2002, and 2009, respectively. His current research interests include computer graphics, virtual reality, medical image processing, HCI, and ubiquitous computing.

Soo-Mi Choi is currently an Associate Professor in the Department of Computer Engineering at Sejong University, Seoul, Korea. She received her B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Ewha University of Korea in 1993, 1995, and 2001, respectively. Between June 1998 and December 1998 she was with the Fraunhofer Institute for Computer Graphics Research (IGD) of Germany as a Visiting Researcher. After she received her Ph.D., she joined the Center for Computer Graphics and Virtual Reality at Ewha University as a Research Professor. Since 2002, she has been working at Sejong University. Currently, she is staying at ETH Zurich of Switzerland as a Visiting Scholar for the academic year 2008–2009. She serves as a member of program committees of many conferences on computer graphics and human-computer interaction. Her current research interests include computer graphics, virtual reality, human-computer interaction, and ubiquitous computing.