Sketching of Mirror-Symmetric Shapes

Frederic Cordier, Hyewon Seo, Jinho Park, and Junyong Noh

Abstract—This paper presents a system to create mirror-symmetric surfaces from free-form sketches. The system takes as input a hand-drawn sketch and generates a surface whose silhouette approximately matches the input sketch. The input sketch typically consists of a set of curves connected at their endpoints, forming T-junctions and cusps. Our system is able to identify the skewed-mirror and translational symmetry between the hand-drawn curves and uses this information to reconstruct the occluded parts of the surface and its 3D shape.

Index Terms—Sketching interface, 3D modeling, and mirror-symmetric shape.

1 Introduction

Freehand sketching is a familiar, efficient, and natural way to visualize an idea in conceptual design. Sketches can be created quickly, and most people have a natural facility that permits basic drawing. In addition, drawing comprehension appears to be an inherent part of human perception.

Ideally, a software for sketched-based modeling of free-form shapes should take as input any freehand drawing and create a shape whose silhouette matches the input sketch and satisfies certain shape quality criteria, such as the maximum compactness constraint and the minimum surface constraint [15]. However, inferring a free-form shape from its drawing has proved to be very difficult. The most important problems are the interpretation of the sketch, the reconstruction of the occluded parts, and the computation of the 3D shape using the 2D data. Igarashi et al. [8] have presented the system Teddy, which can be considered as the seminal paper in the area of sketch-based modeling of free-form shapes. Although this work has been recognized as an important contribution, Teddy cannot process a sketch composed of several curves at once. Other researchers [11], [4] proposed approaches that can reconstruct shapes from drawings of higher complexity. The contribution of these works is mostly related to the reconstruction of the occluded parts of the shape.

The same problem of free-form modeling from sketches is addressed here. However, our work is dedicated to the reconstruction of mirror-symmetric shapes with a circular cross section (Fig. 1). Mirror-symmetric shapes are symmetric with respect to a central plane (also known as a symmetry plane). Symmetric shapes are invariant under reflection with respect to their symmetry plane. Many, if not most shapes in the real world are symmetric. Thus, we believe that a sketching interface for symmetric shapes would be useful.

In this paper, we show that the symmetry assumption can be used to simplify the 3D reconstruction considerably. In particular, we use it to compute the occluded parts of the shape and to estimate the 3D shape. Compared to previous work, our system is able to process much more complex sketches, as shown in Fig. 26. With this work, we make the following technical contributions:

- A method to identify the symmetry relationships between the input 2D curves and compute the orientation of the symmetry plane.
- A method to compute the occluded part of the input sketch using the symmetry assumption.
- A method to reconstruct the surface of the 3D shape using the symmetry relationship such that its 2D silhouette matches the input sketch.

2 Related Work

In what follows, we review previous works concerning sketching interfaces for 3D graphical modeling. The most common approach to 3D modeling with a sketching interface is to require the user to draw the visible and hidden contours of the rectilinear shape to be modeled. The reconstruction is usually formulated as an optimization problem. The variables of the objective functions are the missing depth of the vertices of the drawing (and possibly other parameters). Different objective functions have been proposed, such as minimizing the standard deviation of the angles (MSDA) in the reconstructed shape [18], minimizing the standard deviation of the segment magnitudes (MSDSM) [1], or minimizing the entropy of the angle distribution (MEAD) [20]. Leclerc and Fischler [14] also considered the planarity constraint of the faces of the reconstructed shape together with the MSDA. Lipson and Shpitalni [17] extended the work of Leclerc and Fischler [14] by taking into account the additional constraints of line parallelism, line verticality, isometry, corner orthogonality,

- *F. Cordier is with the LMIA Laboratory (EA 3993), University of Haute Alsace, 4-6 rue des Frères Lumière, Mulhouse 68093, France. E-mail: frederic.cordier@uha.fr.*
- *H. Seo is with the LSIIT Laboratory (UMR 7005), University of Strasbourg, Pôle API, Bd Sébastien Brant, BP 10413, Illkirch Cedex 67412, France. E-mail: seo@unistra.fr.*
- *J. Park is with the Department of Multimedia, Namseoul University, 21 Maeju-Ri, Seongwhan-eup, Seobuk-gu, Chonan, Chungnam 331-707, South Korea. E-mail: c2alpha@gmail.com.*
- *J. Noh is with KAIST, DKAIST GSCT, 335 Gwahangno, Yuseonggu, Daejeon 305-701, Korea. E-mail: junyongnoh@kaist.ac.kr.*

Manuscript received 10 Nov. 2009; revised 15 June 2010; accepted 6 Oct. 2010; published online 7 Dec. 2010.
Recommended for acceptance by W. Wang.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2009-11-0260.
Digital Object Identifier no. 10.1109/TVCG.2010.258.

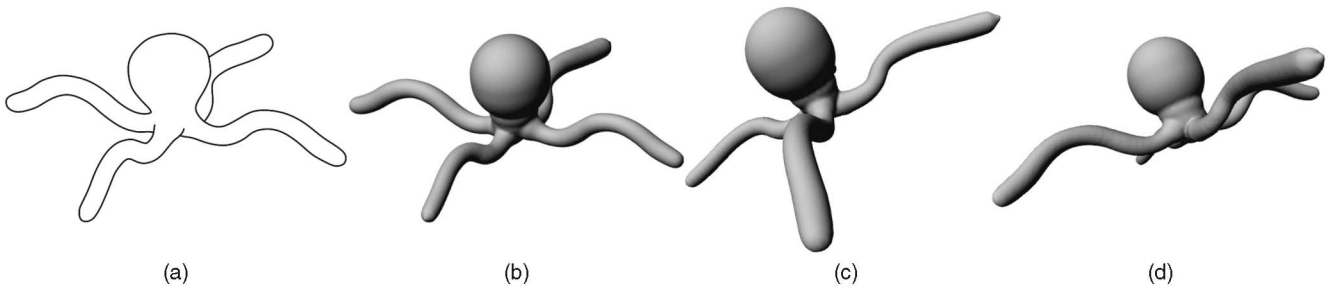

Fig. 1. Sketching of a symmetric shape. The sketch input by the user (a) and the corresponding model generated by our system (b), (c), and (d).

skewed facial orthogonality, and skewed facial symmetry. Later, Liu et al. [16] proposed a method in which the objective function is linear in the space RN, with N being the number of variables of the objective function. Compared to previous methods, this method can reconstruct more complex 3D objects from 2D line drawings and is computationally more efficient. All of these reconstruction techniques are particularly suitable for the design of CAD-like geometric shapes. However, the hypothesis they use allows modeling of rectilinear shapes only and is not suitable for free-form modeling.

Some other sketching tools use a purely gesture-based interface. For instance, "SKETCH," proposed by Zeleznik et al. [26], identifies gestures from the input strokes and interprets them according to a set of predetermined rules. The rules define the manner in which the user-supplied gestural symbols are mapped to the creation of primitive shapes or how operations are applied to existing shapes.

Another group [8] presented a sketching interface for free-form modeling. In their system, the user creates a shape by drawing its 2D silhouette; a 3D mesh is generated by inflating the region surrounded by the silhouette, making wide areas fat and narrow areas thin. The created model can be then modified interactively with a set of tools that cuts, extrudes, bends, or draws on the mesh.

Others [11], [4] have also proposed methods to create 3D models from 2D silhouette curves. Unlike the system proposed by Igarashi et al. [8], the user can create self-occluding objects (or multiple objects that possibly occlude each other). Another difference is that the curves of the 2D drawing are processed in conjunction, and no modification is allowed after the creation of 3D model. The aim of our approach is similar. However, our work is dedicated to the modeling of mirror-symmetric shapes. We use the symmetry assumption to reconstruct the occluded parts. Moreover, our system is able to create complex models with large occlusions, which is not possible with previous work.

One fundamental step in the 3D modeling from sketches is the completion of the hidden contours of the input sketch and its labeling. One of the seminal papers in this area is more than 30 years old [7]. In it, Huffman et al. proposed a labeling scheme for smooth objects, showing that the visible and invisible parts of the contours of a smooth object must have the corresponding sorts of labeling. Williams [24], [25] used Huffman labeling for figural completion. They computed the invisible parts of a drawing containing T-junctions and provided Huffman labeling for it. The result was a labeled knot diagram, which is a set of closed curves complying with the Huffman labeling. They also presented a method known as paneling construction to construct an abstract manifold that can be embedded in R3 so that its projection has contours matching the label-knot diagram. Karpenko and Hughes [11] extended the Williams' work to handle drawings with cusps. They proposed a method to formulate topological embedding from a labeled knot diagram, which is then used to create a smooth solid shape.

Nealen et al. [20] proposed an interactive design tool with which the user creates curves on the surface of the shape and uses them as handles to control the geometry. Schmidt et al. [22] proposed another interactive design tool to create 3D models. Using this tool, the user can define 3D constraints and to use these constraints to create complex curve networks. Gingold et al. [6] also proposed a system for the 3D modeling of free-form surfaces from 2D sketches. The 3D models are created by placing primitives and annotations on the 2D sketches. These three works are mostly based on a multiview incremental construction of complex surfaces, whereas our technique aims at the creation of surfaces from a single sketch.

Several researchers have worked on the 3D reconstruction of mirror-symmetric models from sketches. One recent reconstruction method [2] uses a predefined template. It assumes that the input sketch is topologically identical to the predefined template. Li et al. [15] proposed a computational model that uses planarity and compactness constraints to recover 3D symmetric objects from 2D images. They assume known correspondence of symmetric points. Jiang et al. [9] proposed an interactive method to create symmetric architecture. Their method is mostly dedicated to the modeling of buildings. In addition, it requires user interaction to specify the camera calibration and the geometric features of the buildings. Francois et al. [5] also worked on the 3D reconstruction of mirror-symmetric objects. Their work assumes that the calibration of the camera is known and that manually specified correspondences between symmetric points are required.

Other sketching tools have been developed for 3D curves modeling. Tolba et al. [23] describe a tool with which user can draw a scene with 2D strokes and then visualize it from different points of views. The 3D reconstruction is achieved by aligning the 2D curves on a "perspective grid." Cohen et al. [3] proposed another sketching interface for 3D curve modeling with which the user can model a nonplanar curve by drawing it from a single viewpoint and its shadow on the floor plane.

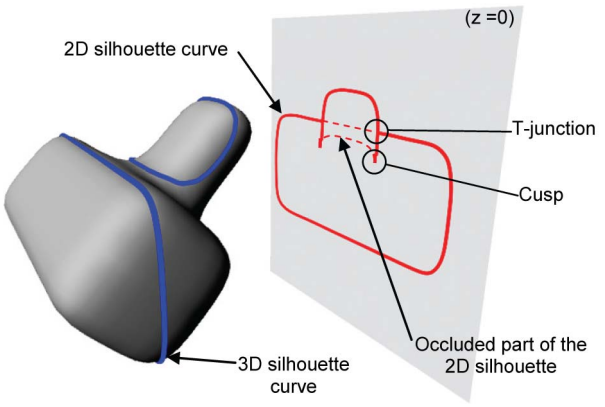

Fig. 2. The 2D silhouette curves are drawn by the user on the sketching plane ($z = 0$). These 2D silhouette curves are the orthogonal projection of the 3D silhouette curves of the shape.

Other researchers have worked on sketching interfaces to modify existing 3D shapes. In the system described by Kerautret et al. [12], user modifies a 3D surface by drawing its shading under different lighting directions. Similarly, with the tools proposed by Nealen et al. [19] and Kho and Garland [13], a 3D shape is deformed by fitting its silhouette to a curve given by the user.

3 Overview

Our system takes a user's sketch composed of a set of connected curves that represent the visible parts of the 2D silhouette and determines a 3D shape whose 2D silhouette matches the input sketch.

3.1 Assumptions

The user draws the 2D silhouette curves on the plane ($z = 0$) that we call the sketching plane. These curves are the orthogonal projection of the 3D silhouette curves of a shape onto the sketching plane ($z = 0$) (Fig. 2). This implies that the x- and y-coordinates of the shape are known. The z-coordinates have to be computed.

We assume the view of the sketch to be generic. The "generic view" assumption states that the view is not accidental (such as two 3D silhouette curves projecting onto the same 2D silhouette curve). This implies that the symmetry plane cannot be parallel to the sketching plane ($z = 0$).

Another assumption here is that the shape reconstructed from the input sketch is mirror symmetric. To simplify the reconstruction, we also assume that the 2D silhouette curves of the input sketch can be decomposed into a set of simple closed curves. In addition, we assume that the reconstructed 3D silhouettes curves are planar. These last two assumptions impose a number of important limitations on the type of shapes that are created using our approach. These limitations will be discussed in detail in Section 9.

3.2 Overview of the Approach

The driving idea is to use the symmetry assumption to compute the 3D shape. As shown in Section 6, the computation of the 3D position of a point and its mirror image is possible if we know the 2D position of their orthogonal projection onto the sketching plane ($z = 0$).

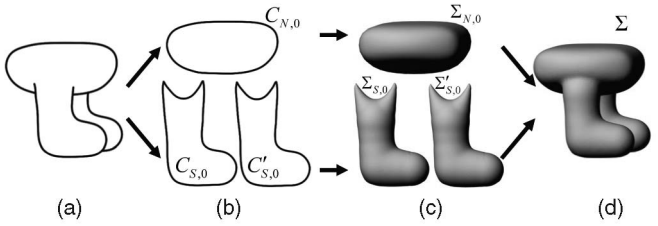

Fig. 3. Overview of the 3D reconstruction algorithm.

Therefore, we decompose the n 2D silhouette curves $\Gamma_I = \{C_{I,1}, \ldots, C_{I,n}\}$ into a set of m pairs of symmetric curves $\Gamma_S = \{C_{S,1}, C'_{S,1}, \ldots, C_{S,m}C'_{S,m}\}$ such that $C'_{C,m}$ is the mirror image of $C_{C,m}$ and a set of p curves $\Gamma_N = \{C_{N,1}, \ldots, C_{N,p}\}$ with no symmetry (Fig. 3b). To simplify the reconstruction further, we find these curves such that they are nonself-intersecting closed curves (Jordan curves). This step involves the detection of the symmetry relationship among the 2D silhouette curves and the computation of the hidden part of the curves. This will be explained in Section 5.

Once the two sets of symmetric and nonsymmetric curves have been constructed, we compute their relative depth order by analyzing the T-junctions and cusp of the input sketch. This information is then used to compute their 3D position. This will be explained in Section 7.

Finally, a closed surface homeomorphics to a sphere is generated for each curve of the two sets (Fig. 3c). Σ_S is the set of surfaces corresponding to the curve set Γ_S and Σ_N is the set of surfaces corresponding to the set of curves Γ_N. The reconstructed surface Σ is obtained as the union of all the surfaces of the sets Σ_S and Σ_N (Fig. 3d). This will be explained in Section 8.

4 Decomposing the 2D Silhouette Curves into Simple Closed Curves

The input sketch is composed of a set $\Gamma_I = \{C_{I,0}, \ldots, C_{I,n}\}$ of curves that represent the visible part of a 2D silhouette of the 3D shape. The first step is to compute the completion of the hidden curves of the drawing such that the completed drawing satisfies the Huffman's labeling scheme. The second step is to decompose the labeled drawing into a set of simple closed curves $\Gamma_{C,i} = \{C_{C,i,0}, \ldots, C_{C,i,m}\}$ by gluing together the hand-drawn curves and completion curves. The last step is to compute the intersections of these simple closed curves and the corresponding occlusion order.

At this point, it is important to note the two types of junctions that an input sketch may contain: a T-junction and a cusp. For a smooth manifold solid, these are the only types of singularities that can occur in the projection of their

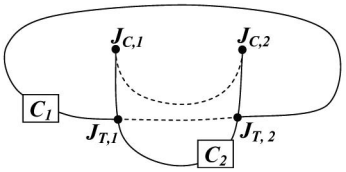

Fig. 4. Contour drawing with T-junctions ($J_{T,1}$ and $J_{T,2}$) where two contours cross and cusps ($J_{1,C}$ and $J_{C,2}$) where contours "reverse" direction. Occluded contours are dashes.

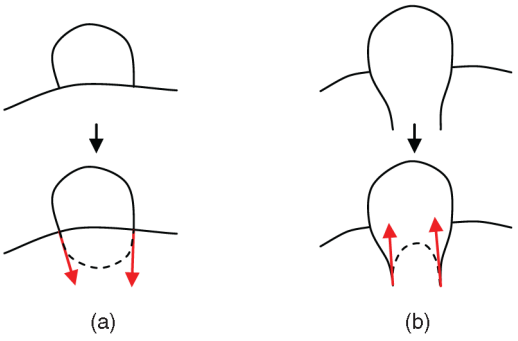

Fig. 5. Completion curves connecting two T-junctions (a) and two cusps (b). The dashed lines denote the completion curves and the red lines denote the tangent vectors.

3D contours, assuming a generic view. As shown in Fig. 4, T-junctions are the points where the curves form a "T-like" shape, one curve C_1 ending abruptly in the middle of another one C_2. Such points indicate that C_1 is partially occluded by C_2. Note that the T-junctions are where two points of the 3D silhouette curves project onto the same point in the sketching plane. A cusp is a point J_C on the surface S where the projector through J_C is tangent to C at J_C. The projection of a cusp appears as a point where the contour drawing "reverses direction" (see Fig. 4).

For the figural completion, we use an algorithm similar to that of Karpenko and Hughes [11]. This algorithm aims to reconstruct the entire 2D silhouette by finding its occluded parts such that the complete drawing complies with the Huffman labeling. This algorithm processes the input sketch in two steps: completion of the hidden silhouette curves and assignment of the Huffman labels.

4.1 Completion of the Hidden Silhouette Curves

Similarly to Karpenko and Hughes [11], we find a set of completion curves that connect pairs of endpoints of the input curves. These completion curves correspond to the occluded part of the silhouette. To compute the completion curve between two T-junctions, we first compute the direction of the tangent vectors at the endpoint. Subsequently, we compute the Bezier spline that joins the two endpoints with the specified direction (Fig. 5a). As we want this spline to approximate an elastica curve, we compute the length of the tangent vectors so that the curvature energy of the spline is minimized [25]. Finding the completion curve between two cusps is done in a manner similar to that used with the T-junctions, except that we use the opposite direction of the tangent vectors to compute the Bezier spline (Fig. 5b). Unlike Karpenko's method [11], we do not compute completion curves joining T-junctions and cusps.

4.2 Assignment of the Huffman Labels

Once we have found a set of completion curves, we compute the Huffman labels [7] for all curves and check the validity of this labeling. We assign an orientation to each curve such that the surface is located on the left as one follows the orientation of the curve. We also compute a depth index for all curves. The depth index of a curve is the number of curves that lie between the camera pinhole and the curve itself. All visible curves receive the index 0. The Huffman labeling of other curves is computed with the Huffman rules, which indicate how the depth index changes at T-junctions and cusps. Once the labeling is complete, we check if the curve completion is valid or not, that is, if any of the labels of the complete drawing violates Huffman's rules.

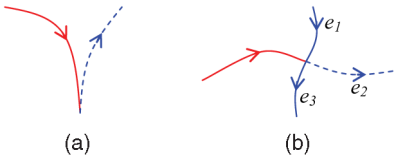

Fig. 6. The incoming edge is shown in red and the outgoing edges are shown in blue.

For a given set of curves Γ_I, there may be more than one means of computing the completion curves which satisfy the Huffman labeling scheme. In this case, we compute all of the solutions of the contour completion. Each solution i is the set of completion curves $\Gamma_{O,i} = \{C_{O,i,0}, \ldots, C_{O,i,m}\}$.

4.3 Computing the Simple Closed Curves and Their Occlusion Order Using the Labeled Drawing

At this point, we have a completed labeled drawing which is composed of the hand-drawn curves from set Γ_I and the completion curves from set $\Gamma_{O,i}$. The drawing partitions the plane into panels; a panel is bounded by a closed loop of consecutive hand-drawn and completion curves. Williams provided an algorithm known as paneling construction which computes the neighborhood of the panels and produces abstract manifolds corresponding to the anterior surfaces of the drawing. An anterior surface is defined as the locus of points on a 3D surface where the surface normal is defined with a positive component in the viewing direction.

Decomposition into simple closed curves is similar to finding anterior surfaces, although the regions bounded by the simple closed curves do not precisely correspond to the anterior surfaces. The difference is that we consider the completion curves connecting two cusps as curves having a depth identical to that of the two neighboring curves. The algorithm to compute the set of simple closed curves $\Gamma_{C,i} = \{C_{C,i,0}, \ldots, C_{C,i,m}\}$ works as follows: Using the labeled drawing, we first construct a directional graph where the nodes are T-junctions and cusps and where the edges correspond to the curves connecting them; the orientation of the curves determines the direction of the corresponding edge. Then, we take an edge which has not been processed and find the "next outgoing" edge. If the node joining the next outgoing edge is a cusp, there is only one edge to choose (Fig. 6a). If the node is a T-junction, we sort the three outgoing edges in a clockwise order and choose the middle one (edge e2 in Fig. 6b). This process is repeated until we find a cycle. Each cycle forms a closed curve $C_{C,i,j}$ which we put into set $\Gamma_{C,i}$.

Once all cycles have been found (i.e., all edges have been visited), we check if any curve of the set $\Gamma_{C,i} = \{C_{C,i,0}, \ldots, C_{C,i,m}\}$ self-intersects or is oriented clockwise. If so, the set $\Gamma_{C,i}$ is removed from the system. We only keep the sets $\Gamma_{C,i}$ whose curves are simple (Fig. 7) and oriented in a counterclockwise direction.

Using the labeled drawing, we also compute the location of the intersection regions (i.e., contiguous set of points located inside the two curves) of the curves of set $\Gamma_{C,i}$ and

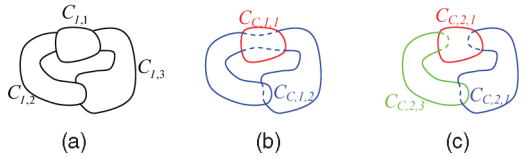

Fig. 7. Contour completion of the set $\Gamma_I = \{C_{I,1}, C_{I,2}, C_{I,3}\}$ (a) results in two solutions: $\Gamma_{C,I} = \{C_{C,1,1}, C_{C,1,2}\}$ (b) and $\Gamma_{C,2} = \{C_{C,2,1}, C_{C,2,2}, C_{C,2,3}\}$ (c). Only one solution, the one with nonintersecting curves (c), is kept in the system.

build an array. Each entry of this array contains the set of hand-drawn and completion curves that forms the intersection region and the occlusion order at the intersection (i.e., which curve is located behind the other). Note that a pair of curves may have more than one intersection region; all of these intersection regions are saved in the array. The occlusion order is computed by examining the change of the depth indices at the T-junctions. The array of intersections will be used in Section 7.1 to check the consistency of the 3D reconstruction.

Note that the method of processing the labeled drawing is completely different from that used by Karpenko. Karpenko's approach consists of gluing the panels together to form a topological manifold homeomorphic to the shape drawn by the user. In our case, the labeled drawing is decomposed into separate regions bounded by simple closed curves. The set of simple closed curves is not necessarily homeomorphic to the drawn shape (Fig. 8c). These simple closed curves are processed separately (Fig. 8d) and the union of the surfaces created from these closed curves is computed at the last stage of our system to create the final shape.

Another particularity of our approach is that the completion curves joining two cusps are assigned the same depth index used for the adjacent curves. In Fig. 8c, the front leg is composed of the hand-drawn curve whose depth index is 0 and the completion curve whose depth index is 1. According the Huffman labeling scheme, the completion curve should be located behind the hand-drawn curve; thus, the leg curve is not planar. In our system, this curve is considered as planar.

It is important to note that the decomposition into simple closed curves implies that several important limitations exist regarding the type of shapes generated with our system. In addition, we also assume that these simple closed curves are planar in the 3D space. This will be explained in Section 9.

Note that the completion is done without taking into account the symmetry relationship among the 2D silhouette curves. The shape of the completion curves, as computed with the Bezier spline, may not match what the user wanted. These completion curves can be modified later

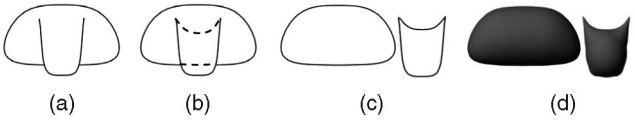

Fig. 8. Input drawing (a), labeled drawing with completion curves (b), simple closed curves corresponding to the leg and the body (c), the 3D shapes reconstructed from the closed curves (d).

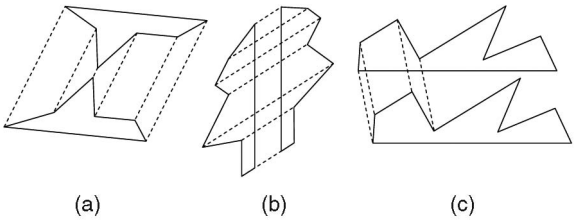

Fig. 9. Skewed-mirror symmetry (a) and (b) and translational symmetry (c). The dashes lines denote the lines of symmetry.

once the symmetry relationship has been found in the sketch. This process will be explained in detail in Section 5.

Depending on the complexity of the input ketch, there may be a large number of sets which can be a solution to the completion. However, only one or a few among them will lead to a valid reconstruction, i.e., 3D shapes whose silhouette matches the user's drawing. The sets which do not correspond to a valid 3D shape will be removed as we attempt to detect the symmetry relationship and compute the 3D positions of the shape. If there are several valid reconstructions, we choose the one that has the smallest number of nonsymmetric curves. The detail description on finding a valid reconstruction is given in Section 7.1.

5 Detection of Symmetric Curves

The next step is to identify the parts of the 2D silhouette curves that are symmetric to each other. As we assume that the shape S to reconstruct is mirror symmetric, there exists a symmetry relationship among the 2D silhouette curves of the shape that we must find. Our goal is to find the symmetry among the curves of the 2D silhouette and use this information to compute the occluded parts of the silhouette.

5.1 Skewed-Mirror and Translational Symmetries

To the best of our knowledge, the reconstruction of nonplanar 3D symmetric curves from their orthogonal projection remains an open problem. We restrain the search to the two following cases of symmetry: skewed-mirror symmetry and translational symmetry.

Skewed-mirror symmetry (Figs. 9a and 9b), as defined by Kanade [10], depicts a mirror-symmetric planar curve viewed from some (unknown) viewing direction. This implies that the detection of skewed-mirror symmetry between two 2D silhouettes is possible only if the corresponding 3D silhouette curves are planar and lie on the same plane. We use the method proposed by Posch [21] to detect skewed-mirror symmetry under orthogonal projection between the two curves C_i and C_j. In addition, to detect skewed-mirror symmetry, this method also provides the pointwise correspondence between the vertices of the curve C_i and the vertices of its mirror image C_j. In this case, the lines of symmetry (lines that connect the vertices to their mirror image) are parallel to each other.

Translational symmetry (Fig. 9c) results from the moving of a shape a certain distance in a certain direction. It is also known as translating by a vector. Translational symmetry is used to find pairs of 3D silhouette curves C_i and C_j lying in two different planes P_i and P_j such that P_i and P_j are parallel to each other. As with skewed-mirror symmetry, we also compute the pointwise correspondence between the

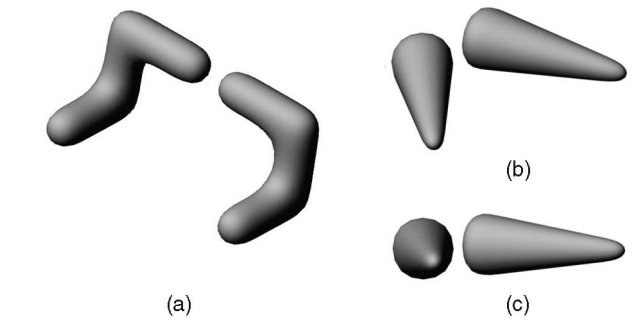

Fig. 10. A symmetric shape whose silhouette is not translational symmetric not or skewed-mirror symmetric (a). The 2D silhouette of a shape, whose skeleton is planar, is not always skewed-mirror symmetric (b), (c).

two symmetric curves. Akin to skewed-mirror symmetry, all of the lines of symmetry are parallel to each other.

There are two limitations of our approach. First, our algorithm only works for 3D silhouette curves which are planar. Thus, we only detect the symmetry relationship of a certain class of 3D shapes, which are those that lie on a plane (i.e., whose skeleton is planar). Fig. 10a shows a pair of symmetric shapes whose 3D silhouette curve is not planar. This restriction does not appear to affect the performance of our sketching interface greatly. We leave this as future work.

Second, even for 3D shapes whose skeleton is planar, the 2D silhouette is not necessarily translational symmetric or skewed-mirror symmetric. This is shown in Figs. 10b and 10c.

5.2 Finding Pairs of Symmetric Curves

The detection of the symmetry from the 2D silhouette mostly consists of finding pairs of symmetric curves: a simple closed curve C_i is symmetric to another simple closed curve C_j.

The set of closed curves $\Gamma_{C,i} = \{C_{C,i,0}, \ldots, C_{C,i,m}\}$, which are a solution to the completion problem (Section 4), are composed of the hand-drawn curves and completion curves which have been computed for the occluded parts of the 2D silhouette. Given that these completion curves were generated using minimum energy curves, their shape may not match what the user wanted and we may not determine the symmetry between the curves correctly (Fig. 11a). Thus, our strategy is to compute symmetry matching only for the parts of the curves drawn by the user (Fig. 11b).

Our algorithm is composed of three steps: First, we find all pairs of symmetric closed curves (Fig. 12c). If all of the hand-drawn parts of a closed curve $C_{C,i,j}$ are symmetric to the hand-drawn parts of another closed curve $C_{C,i,k}$ and that correspondence is injective, these two curves form a pair of symmetric curves. Second, for each pair of curves whose hand-drawn parts are symmetric, we modify the shape of the completion parts such that these two curves are entirely symmetric (Fig. 12d). In the last step, we find the closed curves that are self-symmetric; these curves are divided into two simple closed curves such that they are symmetric to each other (Fig. 12e).

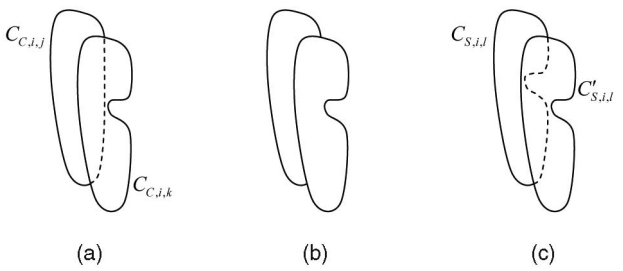

Fig. 11. Detection of translational symmetry fails if the occluded part of the curve (dashed line) is taken into account. (a) The detection of symmetry is possible only if the hand-drawn parts are taken into account (b); after the detection of the translational symmetry, we compute a new completion curve (c).

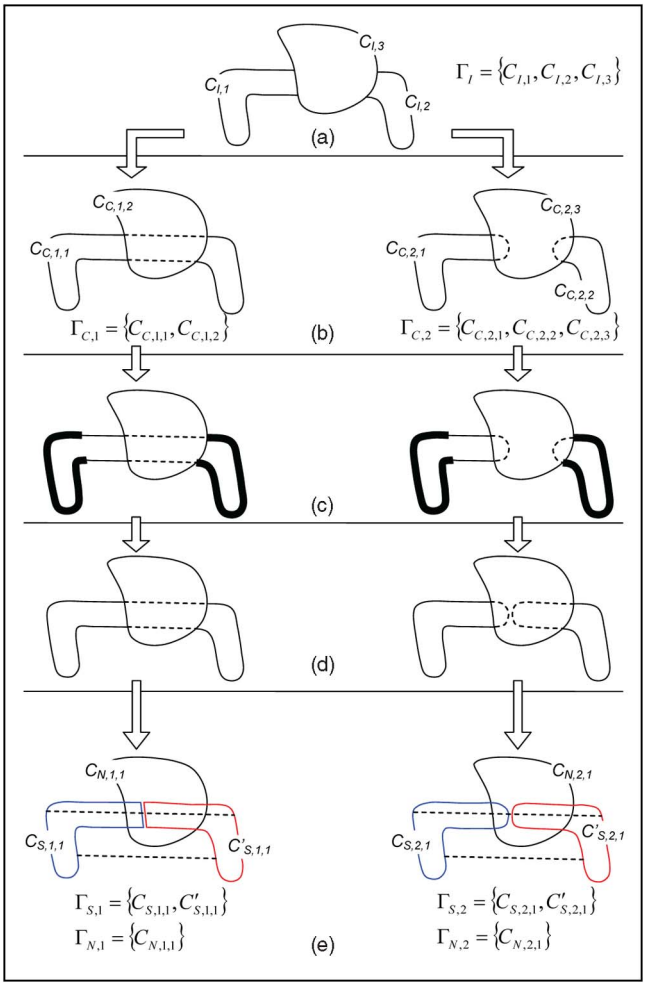

Fig. 12. Finding all pairs of symmetric curves and curves symmetric to themselves. (a) Input sketch. (b) Decomposition into a set of simple closed curves (see Section 4). (c) Finding the symmetric parts (bold lines) of the hand-drawn curves Γ_I. (d) Computation of the occluded parts using the symmetry relationship. (e) Output: a set of symmetric curves with symmetry lines (dashed lines) and a set of non-symmetric curves.

5.3 Output of the Algorithm

For each set of curves $\Gamma_{C,i} = \{C_{C,i,0}, \ldots, C_{C,i,m}\}$ that are solution of the completion problem, we obtain two sets:

- $\Gamma_{S,i} = \{C_{S,i,0}, C'_{S,i,0}, \ldots, C_{C,i,m}, C'_{C,i,m}\}$: the set of all possible pairs of symmetric closed curves (skewed-mirror or translational symmetry). For each pair of symmetric curves $C'_{S,i,j}$ and $C'_{S,i,j}$, we compute the pointwise correspondence between them and the direction of the lines of symmetry.
- $\Gamma_{N,i} = \{C_{N,i,0}, \ldots, C_{C,i,p}\}$: the set of closed curves for which no symmetry has been found.

Note that a curve $C_{C,i,j}$ of $\Gamma_{C,i}$ may be part of several pairs of symmetric curves of $\Gamma_{S,i}$. This case arises when the curve $C_{C,i,j}$ is symmetric to several other curves of $\Gamma_{C,i}$.

An example of the determination of pairs of symmetric curves is shown in Fig. 12.

One may consider that the detection of the symmetry relationship can be computed before the completion step. Indeed, the order in which the two processes are performed does not affect the reconstruction result. However, in practice, the computation time can be reduced significantly if we compute the completion prior to the symmetry detection, as the completion process can drastically reduce the number of curves in the 2D silhouette drawing, repetitively transforming several hand-drawn curves into a closed curve. Therefore, finding pairs of symmetric curves would require less time after the completion than before, whereas the completion process requires the same amount of time regardless of whether it is performed before or after the symmetry detection.

6 Mirror Symmetry

Now that we have computed the symmetry relationship among the closed curves of the sets, the next step is to compute the 3D surface using the symmetry relationship. In this section, we first analyze several properties of mirror-symmetric surfaces. In particular, we show how to compute the 3D position of pairs of symmetric points. In the next section, we show how to use these properties to compute the 3D surface.

6.1 Properties of 3D Mirror Symmetry

Mirror-symmetric surfaces are defined with a symmetry plane M. This plane is the set of all points v such that $\overrightarrow{N} \cdot (O - v) = 0$, where $\overrightarrow{N}$ is a nonzero normal vector of coordinates $(\mathrm{x_n}, \mathrm{y_n}, \mathrm{z_n})$ and O is a point in the plane. Without a loss of generality, we set the point O to the origin of the coordinate system.

Let $V = \{v_0, \ldots, v_i, \ldots, v_{n-1}\}$ be a set of n points of coordinates (x_i, y_i, z_i) and $V' = \{v'_0, \ldots, v'_i, \ldots, v'_{n-1}\}$ be a set of n points of coordinates (x'_i, y'_i, z'_i). Each point v'_i is the mirror image of v_i with respect to plane M. We assume that v_i and v'_i do not have same coordinates. The symmetry relationship between v_i and v'_i implies that the midpoint $(v_i + v'_i)/2$ is located in plane M and that the vector $(v'_i - v_i)$ is perpendicular to M. This gives us the following equations for all pairs of symmetric points v_i and v'_i:

$$\overrightarrow{N} \cdot (v_i + v'_i) = 0, \tag{1}$$

$$\overrightarrow{N} \times (v'_i - v_i) = \vec{0}. \tag{2}$$

Using (1) and (2), we express the z-coordinates of v_i and v'_i as a function of other coordinates and $\overrightarrow{N}$. Equation (1) gives the following result:

$$x_n(x'_i + x_i) + y_n(y'_i + y_i) + z_n(z'_i - z_i) = 0. \tag{3}$$

Equation (2) gives these two equalities:

$$y_n(z'_i - z_i) - z_n(y'_i - y_i) = 0, \tag{4}$$

$$z_n(x'_i - x_i) - x_n(z'_i - z_i) = 0. \tag{5}$$

By combining (3) and (4), the equations to compute z_i and z'_i are

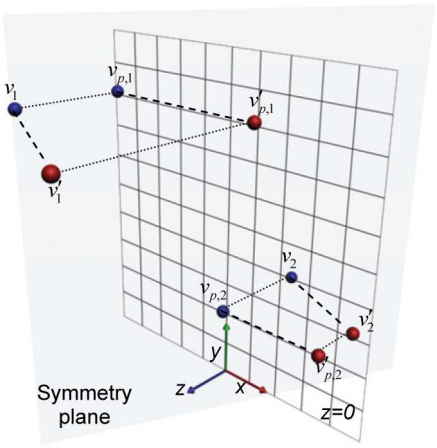

Fig. 13. Pairs of symmetric points (v_1, v'_1) and (v_2, v'_2) and their orthogonal projection $(v_{p,1}, v'_{p,1})$ and $(v_{p,2}, v'_{p,2})$, respectively, to the plane $(z = 0)$.

$$z_i = -\frac{1}{2}\left(\frac{x_n(x'_i + x_i)}{z_n} + \frac{y_n(y'_i + y_i)}{z_n} + \frac{z_n(y'_i - y_i)}{y_n}\right), \tag{6}$$

$$z'_i = -\frac{1}{2}\left(\frac{x_n(x'_i + x_i)}{z_n} + \frac{y_n(y'_i + y_i)}{z_n} - \frac{z_n(y'_i - y_i)}{y_n}\right). \tag{7}$$

Similarly, by combining (3) and (5), z_i and z'_i are given as follows:

$$z_i = -\frac{1}{2}\left(\frac{x_n(x'_i + x_i)}{z_n} + \frac{y_n(y'_i + y_i)}{z_n} + \frac{z_n(x'_i - x_i)}{x_n}\right), \tag{8}$$

$$z'_i = -\frac{1}{2}\left(\frac{x_n(x'_i + x_i)}{z_n} + \frac{y_n(y'_i + y_i)}{z_n} - \frac{z_n(x'_i - x_i)}{x_n}\right). \tag{9}$$

If a point v_i has no mirror image (v_i and v'_i have the same location), it is located on the symmetry plane. Its z-coordinate is given as follows:

$$z_i = -\left(\frac{x_n x_i}{z_n} + \frac{y_n y_i}{z_n}\right). \tag{10}$$

6.2 3D Reconstruction Using Mirror Symmetry

We now consider the orthogonal projection of V and V' to the plane $(z = 0)$. We define the set of points $V_p = \{v_{p,0}, \ldots, v_{p,i}, \ldots, v_{p,n-1}\}$ and $V'_p = \{v'_{p,0}, \ldots, v'_{p,i}, \ldots, v'_{p,n-1}\}$. $v_{p,i}$ and $v'_{p,i}$ are the orthogonal projection to the plane $(z = 0)$ of points v_i and v'_i, respectively (Fig. 13). The coordinates of $v_{p,i}$ and $v'_{p,i}$ are $(x_i, y_i, 0)$ and $(x'_i, y'_i, 0)$, respectively. We also define $\overrightarrow{N_p}$ of coordinates $(x_n, y_n, 0)$, which is the orthogonal projection of $\overrightarrow{N}$. Our goal is to compute the z-coordinates of the sets of points V and V' using their projections V_p and V'_p. We do this with (6), (7), (8), (9), and (10). In these equations, all of the variables are known using V_p, V'_p, and $\overrightarrow{N_p}$, except z_n. It follows that there is only one unknown parameter z_n to define the symmetry plane M completely. Once the value of z_n is set, we are able to compute the z-coordinates of the sets of points V and V'. How the value of z_n is chosen is explained in Section 7.

Given that the value of z_i is given by (6) and (8), the computation of these values is possible if and only if z_n

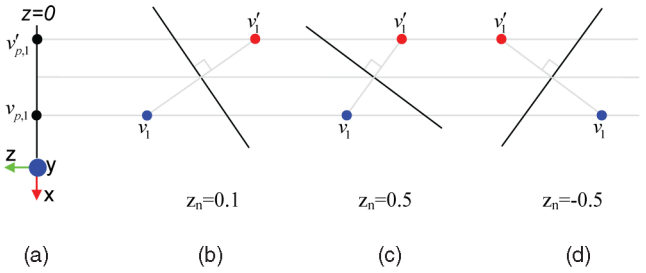

Fig. 14. Three possible solutions (b), (c), and (d) of the reconstruction of a pair of 3D points from their orthogonal projection (a).

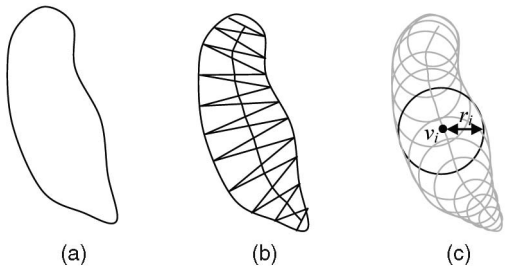

Fig. 15. Closed curve (a); computation of the medial axis (b); surface obtained by the union of the spheres located along the skeleton curve (c).

differs from 0 and is the coordinates of $V_{p,i}$ and $v'_{p,i}$ satisfying the following equality:

$$\frac{(x'_i - x_i)}{x_n} = \frac{(y'_i - y_i)}{y_n}.$$

This equality simply implies that the vector $\overrightarrow{N_p}$ must be parallel to all lines that connect point v_i to their mirror image v'_i. These lines are termed the lines of symmetry here. This gives us the following proposition:

Proposition 1. *Let be two sets of 2D points* $V_p = \{v_{p,0}, \ldots, v_{p,i}, \ldots, v_{p,n-1}\}$ *and* $V'_p = \{v'_{p,0}, \ldots, v'_{p,i}, \ldots, v'_{p,n-1}\}$, *each point* $v'_{p,i}$ *being the mirror image of* $v_{p,i}$. *These two sets are the orthogonal projection of the two sets of points* V *and* V', *which are mirror symmetric to each other if and only if all the lines of symmetry (lines that join* $v_{p,i}$ *and their mirror image* $v'_{p,i}$*) are parallel to each other.*

Fig. 14 illustrates the effect of choosing different values of z_n. A small value makes the symmetry plane nearly parallel to the plane $(z = 0)$, and the distance between symmetric points becomes large. Here, (6), (7), (8), (9), and (10) are not defined for $z_n = 0$; this is the case when the symmetry plane is the plane $(z = 0)$. A large value of z_n increases the slope of the symmetry plane with respect to the plane $(z = 0)$, and the distance between symmetric points becomes smaller (Fig. 14). It is also important to note that changing the sign of z_n modifies the depth order of the 3D points.

7 Computation of the Skeleton

We assume that the surfaces of the sets S_S and S_N have a circular cross section. We represent these surfaces with a skeleton curve defined by a set of vertices; each vertex v_i is associated with a radius r_i which is the thickness of the cross section at that vertex (Fig. 15d).

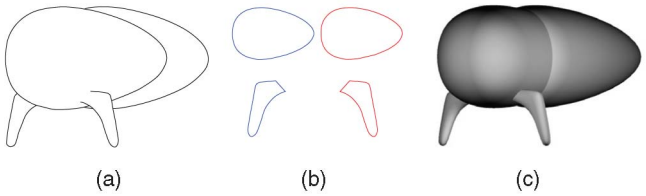

Fig. 16. A sketch that shows a depth ordering violation: the input sketch (a), pairs of symmetric curves (b), and the reconstructed shape (c).

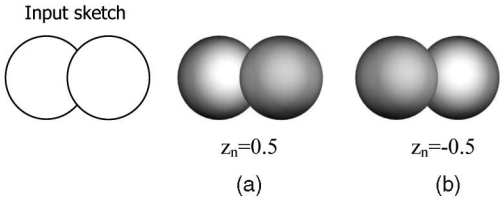

Fig. 17. Reconstruction with $z_n = 0.5$ (a) and $z_n = -0.5$ (b).

Using this surface representation greatly simplifies the computation of the 3D shape. It is not actually necessary to compute the 3D position of the silhouette curve; it is sufficient to compute the 3D position of the skeleton curves and generate the surface using the skeleton vertices and their associated radii. The surface reconstruction from the skeleton curves is explained in Section 8.

We compute the skeleton of a closed curve using the chordal axis. The chordal axis is the curve that connects the center of the "internal edges" of the Delaunay-triangulated closed curve (Fig. 15b). The skeleton curves are computed for all curves of $\Gamma_{S,i}$ and $\Gamma_{N,i}$. As there is a pointwise correspondence between symmetric curves, pointwise correspondence also exists for their corresponding skeleton curves.

7.1 Computation of the 3D Positions of the Skeleton

In Section 6, a method to compute the 3D positions of a set of pairs of symmetric points was presented. The method to compute the 3D position of the skeleton vertices appears to be straightforward. We select a set of pairs of symmetric curves such that the lines of symmetry are all parallel to each other. Given the value z_n provided by the user, we use (6), (7), (8), and (9) to compute the z-coordinates. We also use (10) for the curves that do not have any symmetry. However, by doing so, we define the depth ordering of these skeleton curves that may not be identical to the depth ordering computed using the T-junctions and cusps (see Section 4). We give several examples to explain the problems that may occur during the reconstruction process.

The sketch given by the user may not represent a mirror-symmetric shape. Such a case is illustrated in Fig. 16. Although the input sketch is composed of pairs of symmetric curves that satisfy Prop 1 (lines of symmetry are all parallel to each other), the 3D shape is not symmetric. Regardless of the value of z_n, the silhouette of the reconstructed surface does not match the 2D silhouette drawn by the user.

In other cases, reconstruction of the symmetric surface is possible only for certain values of z_n. Fig. 17 shows an

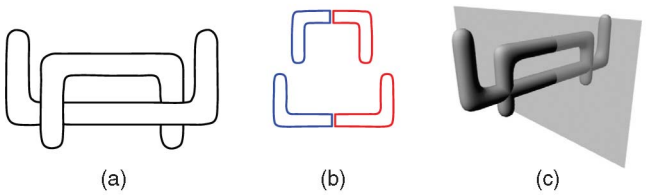

Fig. 18. A figure that shows a depth ordering violation. The gray rectangle is the symmetry plane.

example for which the reconstructed surface does not match the input silhouette for $z_n < 0$.

Another problem may arise when the reconstructed shape has several symmetry planes. Fig. 18 shows such an example. The 3D shape is composed of two surfaces that are self-symmetric with respect to different symmetry planes. The 2D silhouette is composed of two curves that are both self-symmetric with the lines of symmetry all parallel to each other. Here, reconstruction is not possible if the shapes are computed such that they are symmetric with respect to the same symmetry plane. The solution is to consider one curve as self-symmetric with the other curve assumed to be lying on the symmetry plane.

To compute a mirror-symmetric surface whose silhouette matches the input sketch, we use the following algorithm. This algorithm takes as input a set of symmetric curves $\Gamma_{S,i}$ and a set of nonsymmetric curves $\Gamma_{N,i}$.

Step 1. We construct $\Gamma_{SR,i}$, a set of pairs of symmetric curves and $\Gamma_{NR,i}$, a set of nonsymmetric curves such that:

- The two sets $\Gamma_{SR,i}$ and $\Gamma_{NR,i}$ represent the entire input sketch.
- The lines of symmetry of all of the symmetric curves of $\Gamma_{SR,i}$ are parallel to each other. The set $\Gamma_{SR,i}$ should satisfy Prop 1.
- $\Gamma_{NR,i}$ contains the curves which are considered as being not symmetric. In particular, it includes the pairs of symmetric curves whose lines of symmetry are not parallel to those of $\Gamma_{SR,i}$.

Note that there may exist several means to constructing the sets $\Gamma_{SR,i}$ and $\Gamma_{NR,i}$ for the given sets $\Gamma_{S,i}$ and $\Gamma_{N,i}$. This arises when the shape S has several axes of symmetry.

Step 2. We write a set of inequalities for the z-coordinates of the skeleton vertices corresponding to the relative occlusion order at the intersections of the closed curves (see Section 4). There are three different cases for two closed curves intersecting each other: in the first case, the curves intersect each other at four points or more with T-junctions only (Figs. 19a and 19b); the corresponding surface should not intersect in the 3D space. In the second case, the intersecting region of the two curves contains cusps; the two surfaces intersect each other (Fig. 19c). Concerning the third case for which the curves intersect at two points with T-junctions only (Fig. 19d), we interpret this drawing as two surfaces that may intersect each other. This type of drawing is usually used for the drawing of legs (see the caterpillar legs in Fig. 26).

For the case of nonintersecting surfaces (Figs. 19a and 19b), we write a set of inequalities for the z-coordinates of the skeleton vertices to define the minimum distance between

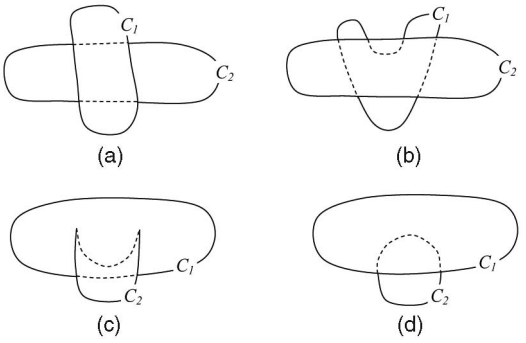

Fig. 19. The surfaces corresponding to C_1 and C_2 do not intersect in 3D (a); the surfaces intersect each other at the locus of the cusp (b).

the skeleton vertices so that the two surfaces do not intersect. For two intersecting closed curves C_i and C_j that belong to $\Gamma_{SR,i}$ and/or $\Gamma_{NR,i}$, we compute the region R which is the intersection of C_i and C_j and find all the skeleton vertices connected to this region (the red dot in Fig. 20a).

Given the skeleton vertices $v_i = (x_i, y_i, z_i)$ of curve C_i and adjacent to R and $v_j = (x_j, y_j, j_j)$ of the curve C_j and adjacent to C_j, the minimum distance along the z-axis (Fig. 20b) is given as follows:

$$d_{z,l} = \sqrt{\left(r_i + r_j + d_{MinSurf}\right)^2 - l_l^2}.$$

Here, r_i and r_j are, respectively, the radius of the cross section at the vertices v_i and v_j, l_l is the distance along the sketching plane between v_i and v_j, and $d_{MinSurf}$ is the minimum distance between the boundaries of the reconstructed shapes. The final value is provided by the user. Finally, we define the linear inequality constraint for the two skeleton vertices:

$$z_i - z_j \geq d_{z,l}. \tag{11}$$

Note that the sign of $d_{z,l}$ is determined by the relative depth order of the two overlapping skeletons.

For pairs of curves whose surfaces intersect in 3D (Figs. 19c and 19d), we use a linear inequality of the same form used in (11). For all pairs of skeleton vertices $v_i = (x_i, y_i, z_i)$ and $v_j = (x_j, y_j, z_j)$ such that

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq r_i + r_j,$$

we define the inequality constraint:

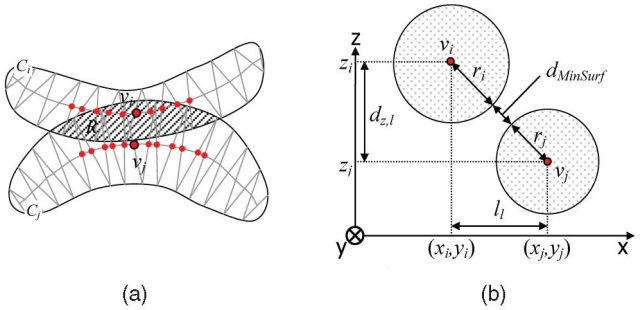

Fig. 20. The minimum distance between two vertices $v_i(x_i, y_i, z_i)$ and $v_j(x_j, y_j, z_j)$.

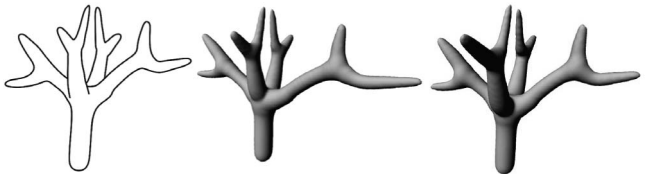

Fig. 21. Reconstruction with different values of Z_n.

$$z_i - z_j > 0. \tag{12}$$

r_i and r_j are, respectively, the cross-section radii at the vertices v_i and v_j. The z-coordinate of the skeleton vertex v_i of the front curve should be larger than the z-coordinate of the skeleton vertex v_j of the other curve.

The variables z_i and z_j in (11) and (12) are written as function of z_n using either (6) to (9) or (10). These inequalities are written for all intersections of the curves of $\Gamma_{SR,i}$ and $\Gamma_{NR,i}$. If the set of inequalities has a solution, reconstruction of the symmetric shape is possible with the sets of symmetric curves $\Gamma_{SR,i}$ and nonsymmetric curves $\Gamma_{NR,i}$. Here, the variable z_n is not uniquely defined. It can be any value within the interval $[z_{n,\min}, z_{n,\max}]$ which is a solution to the set of inequalities. A reconstruction with a small value of z_n increases the size of the reconstructed shape along the z-axis; in contrast, with a large value, its size decreases. By default, the smallest value of z_n is chosen. If this solution is not satisfactory, the user may directly modify the value (see Fig. 21). Using the z_n value, we then compute the z-coordinates of the skeleton curves of the two sets $\Gamma_{SR,i}$ and $\Gamma_{NR,i}$.

If the set of inequalities has no solution, we move to Step 3.

Step 3. The set of inequalities has no solution. This indicates that reconstruction of the symmetric shape S is not possible. There are two reasons. One is that the shape drawn by the user is not mirror symmetric (Fig. 16a). The other reason is that the symmetry relationship among the simple closed curves was not correctly computed. One example is shown in Fig. 23. The algorithm described in Section 5 would decompose the two simple closed curves (Fig. 23a) into two pairs of symmetric curves (Fig. 23b); this is because the symmetry lines of the two simple closed curves are parallel to each other. The reconstruction of these two pairs of symmetric curves would give the shape in Fig. 23c.

As described in Step 2, we construct the set of inequalities using (11) and (12) for the relative occlusion order of the skeleton vertices and (6) to (10) to express the z-coordinates of the skeleton vertices as a function of z_n. The z-coordinates of the skeleton vertices are different whether the corresponding curve is symmetric or not. As shown in Fig. 22, we can compute the z-coordinates of a pair of symmetric curves $C_{S,i,j}$ and $C'_{S,i,j}$ either with (6) to (9) or with (10). In the first case, the reconstructed surface is symmetric with respect to the symmetry plane (Fig. 22b). In the second case, the reconstructed surface is located in the symmetry plane (Fig. 22c).

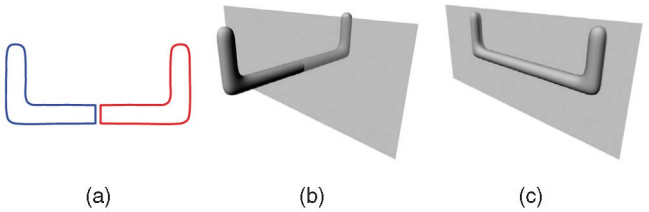

Fig. 22. A pair of symmetric curves (a) can be considered either as being symmetric (b) or nonsymmetric (c).

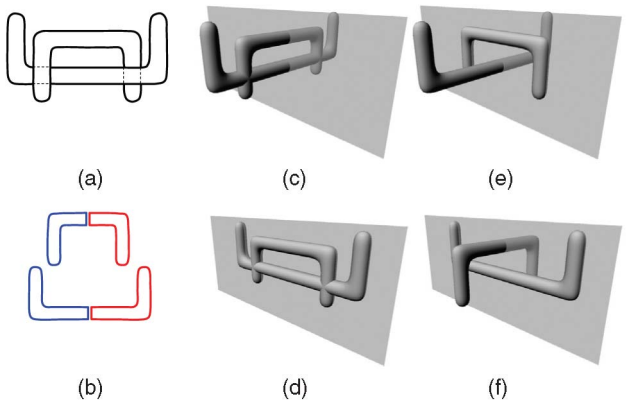

Fig. 23. Given the set of symmetric curves (b), we compute all possible solutions (c), (d), (e), and (f). The gray rectangle is the symmetry plane. The reconstructed shape (f) is the only one whose silhouette matches the input sketch (a).

The idea is to compute the sets of inequalities corresponding to all the possible combinations of symmetric and nonsymmetric curves until we find one that has a solution: each pair of symmetric curves of $\Gamma_{S,i}$ is alternately considered to be symmetric by using (6) to (9) and nonsymmetric by using (10). This is illustrated in Figs. 23c, 23d, 23e, and 23f.

We compute two new sets $\Gamma_{SR,i}$ and $\Gamma_{NR,i}$ corresponding to a different combination of symmetric and nonsymmetric curves and move to Step 2.

Note that our system uses only a single symmetry plane for the reconstruction and does not consider any additional symmetry planes that may be contained in the object. In Section 5, we demonstrated that one symmetry plane suffices to reconstruct the 3D shape.

8 Generating the 3D Shape from the Skeleton

Thus far, we have described an algorithm to compute the z-coordinates of the skeleton curves of the two sets $\Gamma_{SR,i}$ and $\Gamma_{NR,i}$. The output of the algorithm is a set of skeleton curves with 3D coordinates. The final step of the 3D reconstruction is the computation of the surfaces surrounding the skeleton curves. To do this, we use the surface modeling method described by Cordier and Seo [4] due to the simplicity of its implementation. Briefly, the surface is generated by a blend of spherical implicit surfaces whose centers are located along the skeleton curves. The radii of these spherical implicit surfaces are computed such that the silhouette of the resulting surface matches the curves drawn by the user. Σ_S is the set of pairs of symmetric shapes computed for $\Gamma_{SR,i}$ and Σ_N is the set of shapes computed for $\Gamma_{NR,i}$. The reconstructed shape is obtained as the union of the shapes of the two sets Σ_S and Σ_N (See Figs. 3c and 3d).

TABLE 1
Complexity of the Input Drawing and Computation Time of the Models Shown in Figs. 1 and 26

	Number of hand-drawn curves	Number of simple closed curves	Computation time in seconds
Octopus	3	3	9
Caterpillar (front view)	18	14	36
Caterpillar (side view)	13	11	24
Tetrahedron	21	10	42
Basket	52	13	142
Tree	179	85	454

9 Results, Limitations, and Conclusion

Our sketch-based modeling tool has been implemented as a plug-in to Maya. The user can draw the silhouette of the shape, request our plug-in to compute the free-form shape, and visualize the reconstructed shape.

Our method is demonstrated with several examples corresponding to different cases of sketching, showing its versatility. The number of curves that constitute these sketches varies from 3 to 179, as summarized in Table 1. Two caterpillar models are used to demonstrate the reconstruction of the same shape, but with a sketch drawn from different viewpoints. These two models were created from a sketch of naïve users without any assistance. The octopus and the basket examples show the reconstruction of curved shapes.

The model with highest complexity is the tree model, as shown in Fig. 26. The visible and invisible parts of the silhouette were created from an orthogonal projection of an existing 3D model. For a silhouette of such a high level of complexity, enforcing the symmetry constraint is difficult from the user's perspective. In our current system, reconstruction of models of high complexity is facilitated by referring to the silhouette from 2D photos of symmetric objects. Assisting the user to draw the symmetric shape remains as future work.

One of the main advantages of our system is that it can reconstruct occluded contours with less information. In previous methods, the completion is achieved by connecting free endpoints with curves of minimum energy. In our system, we use the symmetry relationship between the curves in the foreground and background to compute their completion. As shown in Fig. 24b, even a partial view of the leg in the background is sufficient to compute its entire silhouette. This is not possible with previous methods.

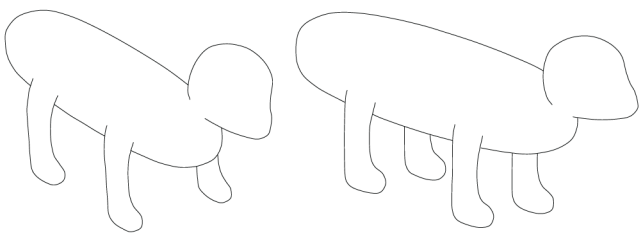

Fig. 24. The reconstruction is possible with the sketch (b) but is not possible with the sketch (a).

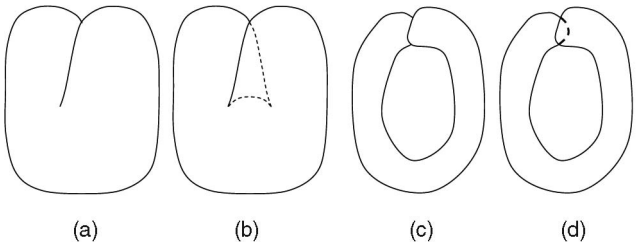

Fig. 25. Example of a sketch (a) that cannot be processed with our system; its silhouette curve contains a hidden cusp (b). The second example contains a self-intersecting curve (d). The dashed curve denotes the occluded part of the silhouette.

9.1 Computation Time

The computation time required to generate the 3D shapes ranges from a few seconds to several minutes depending on the number of the curves that compose the hand-drawn sketch (see Table 1). This slow computation time results from the large number of solutions that are computed with the contour completion algorithm (see Section 4). Let n be the number of hand-drawn curves, with each solution of the contour completion approximately an ordered arrangement of the hand-drawn curves. Thus, the maximum number of completion solutions is $n!$. For example, the number of completion solutions of the bug model, which is composed of 18 curves, is 382. Each completion solution must then be processed to find the one that can be used for the 3D reconstruction.

9.2 Limitations

One requirement of our reconstruction method is that the viewpoint has to be chosen specifically such that both the features and their mirror images are visible or partially visible. Fig. 24a shows a sketch for which the reconstruction of the back legs is not possible because the left back leg is not visible. Another requirement is that the drawing should not contain any hidden cusps (Fig. 25b) or self-intersecting curves (Fig. 25d). We also assume that the silhouette curves of the reconstructed shape are planar in the 3D space.

Another limitation is that the silhouette of the final shape may differ slightly from the input sketch, as the final shape is obtained as the union of the shapes of the two sets Σ_S and Σ_N. Differences between the actual silhouette and the input sketch may appear at the location of the cusps in the sketch, which are the junctions between the shapes of Σ_S and Σ_N. Another source of mismatch between the silhouette and the hand-drawn curves is our inflation algorithm which we use to compute the surface from the skeleton curves. The system does not guarantee that the partial depth order implied by the T-s and cusps in the sketch is preserved.

Future work would be to extend our system to include the sketching of shapes that are approximately mirror symmetric. Such system would be very useful to reconstruct shapes of animals or humans with different leg and arm postures.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. Frederic Cordier has been supported by

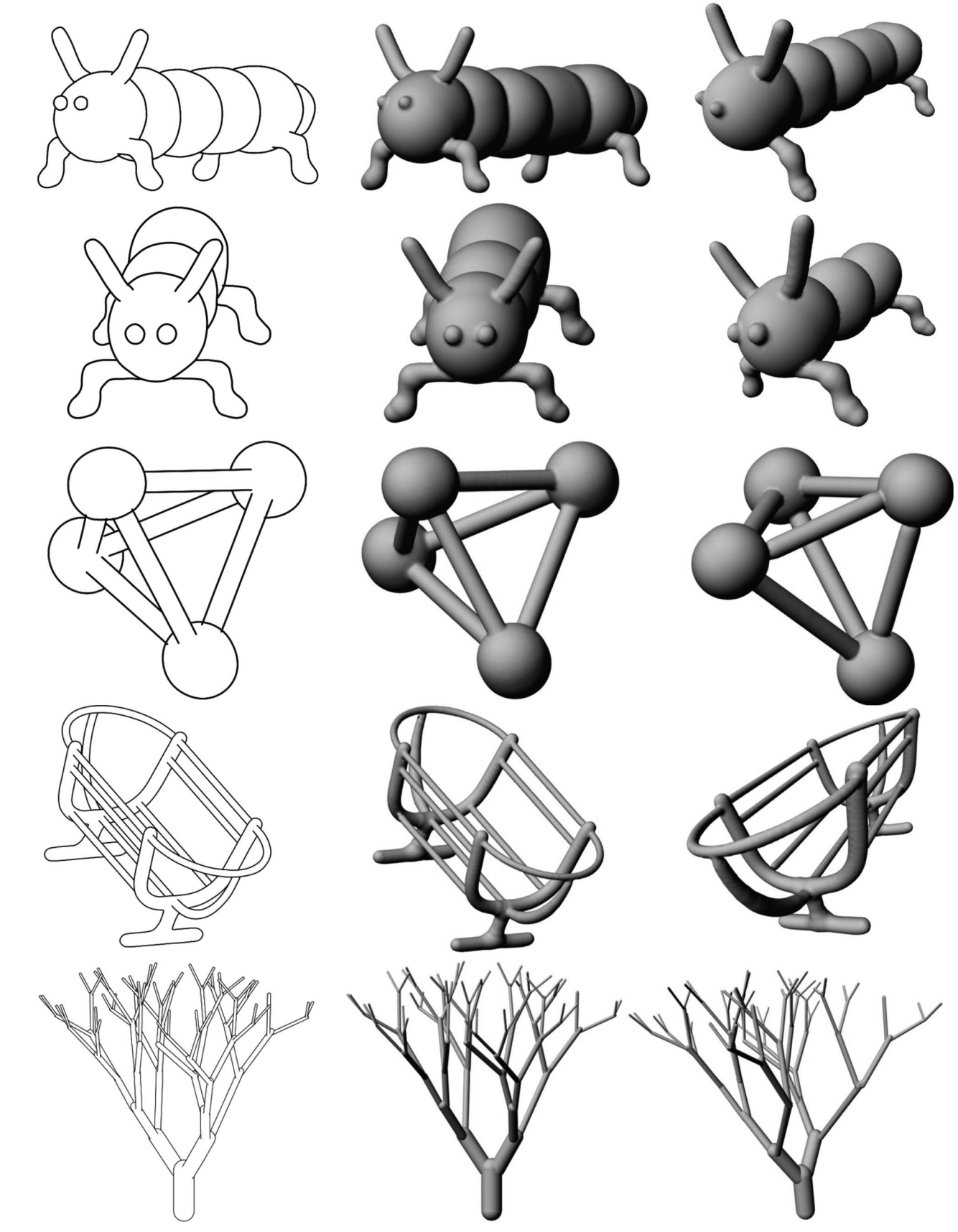

Fig. 26. Several models created using our system. The input sketches are shown in the first row. The reconstructed models seen from two different viewpoints are shown in the two other rows.

the LMIA-EA 3993. Hyewon Seo has been supported by the Centre National de la Recherche Scientifique (CNRS) and the LSIIT-UMR 7005.

REFERENCES

[1] E. Brown and P. Wang, "3D Object Recovery from 2D Images: A New Approach," *Proc. SPIE Robotics and Computer Vision,* vol. 2904, pp. 138-145, 1996.

[2] S.-U. Cheon and S. Han, "A Template-Based Reconstruction of Plane-Symmetric 3D Models from Freehand Sketches," *Computer-Aided Design,* vol. 40, no. 9, pp. 975-986, 2008.

[3] J. Cohen, L. Markosian, R. Zeleznik, J. Hughes, and R. Barzel, "An Interface for Sketching 3D Curves," *Proc. 1999 Symp. Interactive 3D Graphics (I3D '99),* pp. 17-21, 1999.

[4] F. Cordier and H. Seo, "Free-Form Sketching of Self-Occluding Objects," *IEEE Computer Graphics and Applications,* vol. 27, no. 1, pp. 50-59, Jan./Feb. 2007.

[5] A. Francois, G. Medioni, and R. Waupotitsch, "Reconstructing Mirror Symmetric Scenes from a Single View Using 2-View Stereo Geometry," *Proc. Int'l Conf. Pattern Recognition (ICPR),* 2002.

[6] Y. Gingold, T. Igarashi, and D. Zorin, "Structured Annotations for 2D-to-3D Modeling," *Proc. ACM SIGGRAPH Asia '09 Papers,* Dec. 2009.

[7] D.A. Huffman, "Impossible Objects as Nonsense Sentences," *Machine Intelligence 6,* B. Meltzer and D. Michie, eds., Am. Elsevier Publishing Co., 1971.

[8] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design," *Proc. SIGGRAPH '99 Conf.,* pp. 409-416, 1999.

[9] N. Jiang, P. Tan, and L.-F. Cheong, "Symmetric Architecture Modeling with a Single Image," *Proc. ACM SIGGRAPH Asia '09 Papers,* 2009.

[10] T. Kanade, "Recovery of the Three-Dimensional Shape of an Object from a Single View," *Artificial Intelligence,* vol. 17, pp. 409-460, 1981.

[11] O. Karpenko and J. Hughes, "SmoothSketch: 3D Free-Form Shapes from Complex Sketches," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 589-598, 2006.

[12] B. Kerautret, X. Granier, and A. Braquelaire, "Intuitive Shape Modeling by Shading Design," *Proc. Int'l Symp. Smart Graphics,* pp. 163-174, 2005.

[13] Y. Kho and M. Garland, "Sketching Mesh Deformations," *Proc. Symp. Interactive 3D Graphics and Games (I3D),* pp. 147-154, 2005.

[14] Y. Leclerc and M. Fischler, "An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames," *Int'l J. Computer Vision,* vol. 9, no. 2, pp. 113-136, 1992.

[15] Y. Li, Z. Pizlo, and R.M. Steinman, "A Computational Model that Recovers the 3D Shape of an Object from a Single 2D Retinal Representation," *Vision Research,* vol. 49, no. 9, pp. 979-991, May 2009.

[16] J. Liu, L. Cao, Z. Li, and X. Tang, "Plane-Based Optimization for 3D Object Reconstruction from Single Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 2, pp. 315-327, Feb. 2008.

[17] H. Lipson and M. Shpitalni, "Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing," *Computer-Aided Design,* vol. 28, no. 8, pp. 651-663, 1996.

[18] T. Marill, "Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects," *Int'l J. Computer Vision,* vol. 6, no. 2, pp. 147-161, 1991.

[19] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A Sketch-Based Interface for Detail-Preserving Mesh Editing," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1142-1147, 2005.

[20] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "FiberMesh: Designing Freeform Surfaces with 3D Curves," *Proc. ACM SIGGRAPH '07 Papers,* 2007.

[21] S. Posch, "Detecting Skewed Symmetries," *Proc. Int'l Conf. Pattern Recognition,* pp. 602-606, Aug. 1992.

[22] R. Schmidt, A. Khan, K. Singh, and G. Kurtenbach, "Analytic Drawing of 3D Scaffolds," *ACM Trans. Graphics,* vol. 28, no. 5, 2009.

[23] O. Tolba, J. Dorsey, and L. McMillan, "A Projective Drawing System," *Proc. I3D Symp. Interactive 3D Graphics,* 2001.

[24] L.R. Williams, "Topological Reconstruction of a Smooth Manifold-Solid from Its Occluding Contour," *Int'l J. Computer Vision,* vol. 23, no. 1, pp. 93-108, 1997.

[25] L.R. Williams, "Perceptual Completion of Occluded Surfaces," PhD dissertation, Dept. of Computer Science, Univ. of Massachusetts, 1994.

[26] R.C. Zeleznik, K.P. Herndon, and J.F. Hughes, "SKETCH: An Interface for Sketching 3D Scenes," *Proc. ACM SIGGRAPH,* vol. 96, pp. 163-170, 1996.

Frederic Cordier received the PhD degree in computer science from the University of Geneva, Switzerland. He is an assistant professor at the University of Haute Alsace. His research interests include 3D modeling and texturing, human-computer interaction, and physics-based simulation.

Hyewon Seo received the BSc and MSc degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST). After obtaining the PhD degree from the University of Geneva in 2004, she became an assistant professor and supervisor of the Computer Graphics Laboratory in the Computer Science and Engineering Department at the Chungnam National University, South Korea. She has been a CNRS (Centre National de la Recherche Scientifique) research fellow at the University of Strasbourg, France, since 2009. She has worked on human body modeling, virtual prototyping, and sketch-based modeling.

Jinho Park received the BS and MS degrees in applied mathematics in 1999 and 2001, respectively, and the PhD degree in computer science in 2007 from Korea Advanced Institute of Science and Technology. He is a full-time lecturer in the Department of Multimedia at Namseoul University, South Korea. His research interests include fluid animation and scientific visualization.

Junyong Noh received the PhD degree in computer science from the University of Southern California (USC) in 2002 where his research focus was on facial modeling and animation. He is an associate professor in the Graduate School of Culture Technology at the Korea Advanced Institute of Science and Technology (KAIST). He is also affiliated with KAIST Institute of Entertainment Engineering (KIEE). His research relates to human facial modeling/animation, character animation, fluid simulation, and stereoscopic visualization. Prior to his academic career, he was a graphics scientist at a Hollywood visual effects company, Rhythm and Hues Studios. He performed R&D for movie post productions including Superman Returns, Happy Feet, The Chronicles of Narnia, Garfield, Around the world in 80 days, and The Chronicles of Riddick. He had also participated in implementation of fluid simulation software, which received an Academy Award in 2008. He has been doing consulting for or collaborative work with many companies such as Weta Digital, SKT, ETRI, Macrograph Olive studio, and KAI studio.