A Feature-based Approach for Smooth Surfaces

Shigeo Takahashi†, Yoshihisa Shinagawa‡, and Tosiyasu L. Kunii†
† Department of Information Science, Graduate School of Science, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan
‡ School of Computer Science and Engineering, The University of Aizu
Tsuruga, Ikkimachi, Aizuwakamatsu, Fukushima 965-80, Japan

Abstract

Feature-based representation has become a topic of interest in shape modeling techniques. Such feature-based techniques are, however, still restricted to polyhedral shapes, and none has been done on smooth surfaces. This paper presents a new feature-based approach for smooth surfaces. Here, the smooth surfaces are assumed to be 2-dimensional C²-differentiable manifolds within a theoretical framework. As the shape features, critical points such as peaks, pits, and passes are used. We also use a critical point graph called the Reeb graph to represent the topological skeletons of a smooth object. Since the critical points have close relations with the entities of B-reps, the framework of the B-reps can easily be applied to our approach.

In our method, the shape design process begins with specifying the topological skeletons using the Reeb graph. The Reeb graph is edited by pasting the entities called cells that have one-to-one correspondences with the critical points. In addition to the topological skeletons, users also design the geometry of the objects with smooth surfaces by specifying the flow curves that run on the object surface. From these flow curves, the system automatically creates a control network that encloses the object shape. The surfaces are interpolated from the control network by minimizing the energy function subject to the deformation of the surfaces using variational optimization.

1 Introduction

1.1 Background

Recent developments in computer hardware enable us to handle a large amount of data within a short period of time. These developments have enabled us to handle not only simple polyhedral objects such as mechanical parts and manufactured objects, but complicated smooth objects such as terrains, human organs, and virtual objects. In particular, designing virtual objects such as humans, animals, plants, etc. has become important for computer graphics (CG) animation and virtual reality (VR) applications. In this way, the need to handle smooth curved objects by computers has been increasing.

Contemporary CAD systems handle the smooth object shapes, however, by extending conventional polyhedral representation. This leads to the polyhedral decomposition of smooth object shapes that has no relations with the geometrical features of the smooth surfaces. In this situation, we encounter the following problems. The first is that we cannot avoid a large amount of design interactions caused by the inappropriate polyhedral approximations of complicated smooth shapes. Secondly, in contemporary CAD systems, none of the design operations characteristics of the smooth surfaces are taken into account. Furthermore, the CAD systems cannot provide the users with any efficient keys for shape databases due to the lack of information about the features of smooth surfaces. In order to remedy these problems, it is necessary to
construct a model for smooth surfaces based on shape features intrinsic to their smoothness.

1.2 Our Approach

This paper presents a new feature-based modeling method for smooth surfaces. As the shape features, critical points such as peaks, pits, and passes are used. The relations among the critical points are represented by critical point graphs (CPGs), which are defined as the graphs whose vertices correspond to the critical points. In particular, the Reeb graph [22], which is one of the CPGs, is used to represent the topological skeletons of a smooth object (Figure 1). The features such as the critical points and the CPGs serve as the upper level in the hierarchical representation of smooth objects. This means that our method provides intuitive feature-based operations for designing smooth objects by separating the topology from the geometry. In addition to this, the embeddings of the object in 3D space are also described in our method. This allows us to design surfaces with multi-layered inner structures as shown in Figure 17.

Figure 2 illustrates the rough classification of previous modeling methods and our method based on the object shapes and representation schemes. The leftmost column corresponds to the modeling methods for polyhedral shapes and the rightmost column corresponds to those for smooth surfaces. The middle column indicates the modeling methods for the objects that contain both polyhedral and smooth surfaces. The bottom row corresponds to the hierarchical representation schemes while the top row corresponds to the representation schemes without explicit hierarchies. As described above, conventional modeling methods cover the smooth objects by extending polyhedral representations. The CSG representation schemes are also extended to handle objects with free-form surfaces [38, 12, 20]. On the other hand, hierarchical representation schemes of polyhedral objects, such as constraint-based [7, 1, 23, 31] and feature-based [25, 14, 17] modeling methods, have been developed. Conversely, our approach directly handles the objects whose surfaces are smooth with hierarchical representations based on the features. Our modeling method also covers the three-dimensional (3D) surface shapes reconstructed from cross-sectional data, and also extends its target objects to smooth objects that contain flat surfaces partially.

It must be noted that the objects handled in our modeling method are slightly different from those in the conventional modeling methods. While the conventional methods handle smooth objects indirectly by way of polyhedral approximation, our method aims at handling them directly.

Several CSG-based methods are presented for handling smooth surfaces [38, 12, 20]. In particular, Menon and Guo presented a method of handling sculptured solids using CSG boolean combinations [20], and Cavendish proposed a method of designing and deforming the free-form surfaces with their features [3]. However, his method is limited to the surfaces represented by single-valued functions.
Krishnan and Manocha presented a method of representing free-form surfaces by maintaining the connectivity of trimmed surfaces [12]. However, the methods cannot provide the operations based on the differential properties of smooth surfaces because they do not have the features of smooth surfaces.

Among the hierarchical representation schemes, feature-based modeling methods have been extensively studied recently [24]. In the methods, object features such as the slots and holes together with its parametric specifications are used as the upper levels of their hierarchical representations. What is important to note is that they provide bidirectional operations between object shapes and shape features, i.e., design by features and feature recognition. Users can design object shapes by their shape features, and can also extract features from existing objects. Our aim is to establish a smooth-surface version of the feature-based modeling method using the critical points and CPGs. In this paper, we focus on describing one of the bidirectional operations, i.e., the design by features.

1.3 B-reps and Our Representation

It is noted that critical points have close relations with the entities of B-reps. Figure 3 illustrates the relations. The entities of B-reps are simplices, i.e., faces, edges, and vertices. Mathematical theories tell us that the simplices have their surface-versions called the cells that are the entities of a smooth object in our modeling. Furthermore, according to the Morse theory [21], the cell has a one-to-one correspondence with the critical point of a height function of the object surface. Consequently, the entities of B-reps correspond to the entities of our representation, i.e., the critical points.

These relations enable us to apply the framework of B-reps to our modeling method. The first and the most important example is the Euler formula that maintains the validity of object shapes in B-reps. Let \( \chi \) be the Euler characteristic: a topological invariant of an object shape. From the above relation, the Euler formula of a polyhedral shape \( \#(\text{faces}) - \#(\text{edges}) + \#(\text{vertices}) = \chi \) can be converted to that of a smooth surface \( \#(\text{peaks}) - \#(\text{passes}) + \#(\text{pits}) = \chi \). To maintain the Euler formula in B-reps, the Euler operators are used. In our representation, on the other hand, we introduce a set of new operators called the Morse operators [28, 27].

1.4 The Design Steps in Our Method

The design of smooth surfaces is divided into two parts: topological design and geometrical design. For each design process, we provide an intuitive set of operations as follows.

The topological design of the smooth surfaces begins with specifying the skeletons of an object shape using the Reeb graph (cf. Figure 1). Users construct the Reeb graph by pasting cells that have one-to-one correspondences with the critical points of a height function. The Reeb graph is edited by the Morse operators [28] that specify the way of gluing cells. We also introduce the iconic representation of the Reeb graph [28] in order to specify the embeddings of smooth objects in 3D space. The system also provides macro operations for attaching a branch or a tube to the constructed surfaces by a pair of the Morse operators.

The geometry of a smooth object shape is outlined by the flow curves that run on the object surface. From these flow curves, the system automatically creates a control network that encloses the object shape. Local patches are assigned to the vertices of the control network and are then glued together to form the whole surface shape. The shape of each local patch is determined by minimizing the energy function subject to the deformation of the local patch using variational optimization. In order to discuss our modeling from the theoretical aspects in this paper, we assume that the smooth surface of an object is a 2-dimensional \( C^2 \)-differentiable manifold.

This paper is organized as follows: Section 2 shows how to design the topological skeletons of a smooth object using the Reeb graph and Morse operators. Section 3 presents geometrical design methods based on the variational optimization. Section 4 shows our implementation and design examples. Finally, Section 5 concludes this paper and refers to future work.

---

\(^2\)The cells are not new to current shape models. Several papers proposed methods of representing non-manifold objects using the topological properties of the cells [19, 40].

\(^3\)The description of the theories can be found, for example, in [18].
Designing the Topological Skeletons

2.1 The Reeb Graph

The Reeb graph is defined as follows. Let \( f \) be a function of an object surface. Consider the cross-sectional contours of the object surface, i.e., \( f^{-1}(h) \) (\( h = \text{const} \)). By identifying the connected component of the object surface with a point at each cross section, we can obtain a set of points. The Reeb graph is then constructed by tracing these points at each cross section from the top to the bottom of the object. Figure 1 illustrates the Reeb graph of a height function defined on a torus. Here, the height function represents the point \((x, y, z)\) on the surface as the function \( z = f(x, y) \).

As mentioned earlier, the Reeb graph is one of the CPGs and its vertex corresponds to a critical point. From the above definition, the edge of the Reeb graph represents a tube that connects two critical sections. As can be seen from Figure 1, the Reeb graph represents the topological skeletons of the object shape. Since this skeletal representation is intuitive, we use the Reeb graph for designing the topological skeletons of the object shape.

In addition to the topological skeletons, the embeddings of the objects in 3D space are also specified. For this purpose, we employ the embedded Reeb graph proposed by Shinagawa, Kergosien, and Kunii [28]. The embedded Reeb graph is an iconic representation that represents the orientations of the surfaces and the inclusion relations of the cross-sectional surface profiles. Figure 4 illustrates the embedded Reeb graph of a double-layered torus, i.e., two tori where one of which contains the other. If we consider the outer torus as a solid object, the inner torus becomes a hollow object. The solid and hollow objects have different surface orientations. In the same way as in [28], we distinguish between solid and hollow objects by black and white colors in the iconic representation. The embedded Reeb graph serves as an interface for designing the topological skeletons of object shapes in our system.

2.2 The Morse Operators

In order to edit the Reeb graph, we use the Morse operators proposed in [28]. The Morse operators specify how the critical points are connected in the Reeb graph, which is equivalent to how the cells are glued to form the whole object.

The Reeb graph of a torus is constructed with the Morse operators as follows. As illustrated in Figure 1, a torus contains, from top to bottom, four critical points: a peak, a pass, a pass, and a pit. This means that a torus is constructed from its top to bottom with the four Morse operators that correspond to the critical points of the torus as illustrated in Figure 5. Firstly, we put a 2D cell \( e^2 \), i.e., a topological cap, that corresponds to the peak of the torus (Figure 5(a)). With this operation, a new circle appears at the cross section. Secondly, we attach a 1D cell \( e^1 \), i.e., a topological curve, to split the cross-sectional curve (Figure 5(b)). Thirdly, we attach another 1D cell \( e^1 \) to merge the two divided cross-sectional circles into one, which results in the hole of the torus (Figure 5(c)). Finally, we close the surface of the torus by attaching a 0D cell \( e^0 \), i.e., a point, to the existing surface (Figure 5(d)).

The changes of the embedded Reeb graph (iconic representation) are also illustrated at the bottom row of Figure 5.

In this way, the Morse operators describe the changes of cross-sectional contours at critical sections. In our implementation, the system stores the inclusion relations of cross-sectional contours as a tree, which can be seen in Figure 6. Here, the contour \#0, the virtual root of the contour tree, is introduced for convenience. Consider the relation between the contours \#2 and \#4, for example. Since the contour \#2 contains the contour \#4, \#2 is the parent of \#4 in the contour tree.

Similar to the Euler operators in B-reps, the Morse operators maintain the validity of smooth object shapes. Here, the validity of object shapes means that they can be embedded in 3D space without self-intersections. To maintain the validity of the object shape, six types of operators are required as described in [28]. The changes in cross-sectional contours by these six operators are shown in Figure 7.

In Figure 7(a), the downward arrow indicates the change of a peak, while the upward arrow indicates that of a pit. In this paper, the corresponding operators are denoted by \( E_2 \) (which is named after \( e^2 \)) and \( E_0 \) (which is named after \( e^0 \)), respectively. In the

---

Figure 4. The embedded Reeb graph of a double-layered torus

---

This construction is depicted in detail in [21].

This means that our model does not permit objects that cannot be embedded in 3D space. The Klein bottle is an example of such an invalid object. The model extended for such objects is described in [30].
Figure 5: The Morse operators for constructing a torus: (a) putting a 2D cell $e^2$, (b) attaching a 1D cell $e^1$, (c) attaching a 1D cell $e^1$, and (d) attaching a 0D cell $e^0$ [28].

Figure 6: A contour tree based on inclusion relations [28].

In our implementation, these six operators are provided as fundamental tools for editing the Reeb graph. Since the system holds the contour trees as illustrated in Figure 6, it automatically rejects illegal Morse operators that result in generating invalid objects. This will be described in Section 2.4 in detail.

2.3 Macro Operations

With the Morse operators, we can edit the Reeb graph that represents the topological skeletons of a smooth object. The order of these operations, however, must follow the height order of their corresponding critical points. This means that it is necessary to describe the changes of cross-sectional contours from the top to the bottom of the object. Since this limitation is not intuitive to design a whole object, it should be avoided.

In our implementation, we introduce macro operations for avoiding this limitation. The macro operation inserts a pair of new critical points to an existing object shape while maintaining the topological validity of the object shape. Note that the topological validity of the object shape can be examined also in this case because the macro operation is equivalent to applying a pair of the Morse operators. Figure 8
the upper section

<table>
<thead>
<tr>
<th>critical section</th>
</tr>
</thead>
<tbody>
<tr>
<td>the lower section</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>for a peak and a pit</th>
<th>for a pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>E1IN</td>
</tr>
<tr>
<td>E1N</td>
<td>E1SI</td>
</tr>
<tr>
<td>EOUT</td>
<td>E1PC</td>
</tr>
</tbody>
</table>

(a) (b) (c)

Figure 7. The effects of the six Morse operators: (a) E2 and E0, (b) E1IN and E1SI, and (c) E1OUT and E1PC

2.4 Data Structures

The data structure in the system is illustrated in Figure 9. The system holds the Reeb graph for representing the topological skeletons of the object, and the contour trees for representing the inclusion relations among the cross-sectional contours. Since the topology of the cross-sectional contours changes at critical sections, it is sufficient to hold the contour trees at sections each of which lies between a pair of adjacent critical sections. Figure 9(a) shows an object shape, Figure 9(b) shows its graph data, and Figure 9(c) shows its corresponding iconic representation. As shown in this figure, each vertex of the Reeb graph has its corresponding Morse operator.

In order to maintain the correct embeddings of the object shape, the system automatically constructs closed surfaces by adding appropriate virtual pits to the object shape if necessary. Since our theoretical framework is based on the assumption that the object consists of closed surfaces, this implementation is convenient for verifying the consistency of the object embeddings. In Figure 9, the lower parts with a shaded region represent the virtual parts that are automatically added by the system.

When the object data is modified, the system updates the contour trees by scanning the Morse operators from the top to the bottom of the object, and checks whether its embeddings are correct or not. If the newly designed object has incorrect embeddings, the system rejects the latest operation. Note that we can easily establish the geometric correctness of the object boundary because the system represents the object as closed surfaces whichever the object is open or closed.

3 Designing Geometry using Variational Optimization

We are now able to design the topological skeletons of a smooth object using the Reeb graph. Our next step is to design the detailed geometry of the object surface. First, the geometry of the object surface is outlined by flow curves that run on the object surface. From the flow curves, the system automatically cre-

Figure 8: Macro operations: (a) an operation for attaching a new branch, and (b) an operation for attaching a new tube

illustrates examples of macro operations. Figure 8(a) shows a macro operation for attaching a new branch to a topological sphere. This macro operation is equivalent to applying the E2 and E1SI operations to the existing surface, which means that a peak and a pass are inserted. As illustrated in Figure 8(a), the Reeb graph is also modified with the macro operations. Another example of attaching a new tube to a topological sphere is shown in Figure 8(b). This macro operation amounts to applying the E1IN and E1SI operations to the existing surface. Note that the possible set of such macro operations is E2-E0, E2-E1SI, E2-E1PC, E1IN-E0, E1IN-E1SI, E1IN-E1PC, E1OU1-E0, E1OU1-E1SI, and E1OUT-E1PC. We can also provide more complex operations by combining these macro operations. With these macro operations, we can overcome the limitation due to the height order of the Morse operations.
Figures 9: Data structure in the system: (a) object shape, (b) graph data, and (c) iconic representation

Figure 10: Flow curves of a torus

3.1 Flow Curves

The first step of the geometrical design is to specify the shape of the flow curves that run on the object surface. Figure 10 shows an example of the flow curves that run on the surface of a torus. As illustrated in Figure 10, the flow curves are assumed to go down monotonously with respect to the height value in our implementation. The flow curve used in the system is the curve that goes from one critical point to another in order to outline the rough shapes of the object surface. In particular, the flow curves can be used to guide the cross-sectional shapes of the object surface. For later convenience, a pass is assumed to have four incident flow curves, two of which come to the pass from the upper side and two of which go out of the pass to the lower. Note that the configuration of flow curves is based on those of critical points of an object surface, the properties of the surfaces around the critical points are reflected in the processes of the surface generation.

The flow curve is represented by a cubic B-spline curve defined on a knot sequence that is uniformly spaced everywhere except for its ends, where its knots have multiplicity 4. The basis functions of such B-spline curves are called endpoint-interpolating B-splines [4]. The freedom of the B-spline curve is controlled by inserting or deleting the internal knots of the knot sequence. The shape of the flow curve is designed by imposing point-position constraints and tangent constraints of the curve. While preserving the imposed constraints, the system determines the curve shape by minimizing its deformation. This can be implemented by using the techniques of Welch and Witkin [30] that optimize the energy function subject to the deformation of the curve. In determining the shape using these techniques, the freedom of the curve can be adjusted in proportion to the degree of the constraints and is also specified manually by users when necessary. Note that the endpoint-interpolating B-splines are well suited to the multiresolutional representation of shapes with spline wavelets [5, 6]. This scheme is also used for the design of local patches in our system (cf. Section 3.4).

3.2 Control Network

In order to generate an object surface from the given flow curves, the system automatically creates a control network that encloses the object shape. This control network is created by adding appropriate cross-sectional curves to the flow curves. The control network of a torus is shown in Figure 11. By default, the cross-sectional curves are added to the cylindrical parts of the object that correspond to the edges of the Reeb graph. Of course, such cross-sectional curves can be modified by adding constraints or moving control points in the system.

A peak vertex, a pit vertex, and a pass vertex are defined to be the vertices of the control network that correspond to a peak, a pit, and a pass of the object surface, respectively. Other vertices of the control network are called regular vertices. The control network decomposes the object surface into regions surrounded by its curve segments. In our implementation, the decomposed faces are topologically equivalent to three-sided regions (i.e., triangles), four-sided regions (i.e., quadrilaterals), or five-sided regions (i.e., pentagons) as illustrated in Figure 11. In particular, the faces around a peak vertex or a pit vertex are three-sided regions, the faces around a pass vertex are five-sided regions, and the others are four-sided regions. The rule of the network construction also implies that the regular vertices and pass vertices have only four incident curve segments in the control network. Since this surface decomposition takes into account the configuration of the critical points on the object surface, the surfaces around the critical points are made smooth without any special manipulations.
3.3 Overlapping Local Patches

In the next three subsections, we describe how to determine the surface shape of an object from its control network. Several techniques are proposed for generating smooth surfaces from an irregular network [10, 15, 37]. For an irregular curved network, in particular, Kuriyama [13] proposed a method of generating a smooth $n$-sided patch from its $n$ boundary curve segments by blending the swept subsurfaces of the $n$ curve segments.

In our implementation, we perform the following procedures. First, we assign a local patch to each vertex of the control network. We then design the local patches by minimizing its deformation while satisfying the constraints derived from the control network. Finally, we glue the local patches together to form the whole surface of the object so that the adjacent patches are blended in their overlapping parametric domains.

One of the advantages of this framework is the locality of the surface design. In other words, we can design the local patches without modifying the overall shape of the object. Another advantage is the flexibility in assembling the local patches. It means that it is unnecessary to take care of the continuity in connecting the patches because the adjacent local patches have overlaps where the patches are blended while preserving the continuity. In addition, this implementation enables us to introduce multiresolutional design of object surfaces using wavelets. We will mention this again in Section 3.4.

The remainder of this subsection describes how to establish the mappings between the object surface and the decomposed local patches. In order to construct such mappings, we use the Varady's parametrization techniques [37]. Following the Varady's notation, we call the local patch a vertex patch. According to the Varady's techniques, each vertex patch has a vector-valued parametric form that maps the rectangular bivariate parametric domain onto 3D space. On the other hand, the $n$-sided region decomposed by the control network is defined on a regular $n$-gon, i.e., on a regular triangle when $n = 3$, on a square when $n = 4$, and on a regular pentagon when $n = 5$. This implies that we have to define the mapping between a vertex patch and a surface region by defining the mapping between the corresponding parametric domains, i.e., the mapping between the bivariate parametric domain and the regular $n$-gon.

In our implementation, we use the parametrization based on a planar biquadratic Bézier patch [37]. Figure 12 illustrates how to map the bivariate parameter $(u, v)$ onto an $n$-gon using this parametrization. If we denote the control points of the Bézier patch by $T_{ij}$ where $i, j = 0, 1, 2$, we can map the bivariate coordinates $(u, v)$ onto the coordinates in the polygon $(\mu, \nu)$ using the following equation:

$$
\begin{pmatrix}
(1-v^2) & 2v(1-v) & v^2 \\
2u(1-u) & u^2 & 2uv
\end{pmatrix}
$$

Figure 12(a) shows the bivariate parametrization in a regular triangle ($n = 3$), and Figure 12(b) shows that in a regular pentagon ($n = 5$). In this parametrization, $T_{00}$ lies at the base vertex $(u = 0, v = 0)$, $T_{22}$ and $T_{20}$ lie at the vertices adjacent to $T_{00}$, and $T_{10}$ and $T_{12}$ lie at the midpoints of the edges emanating from $T_{00}$. $T_{11}$ lies at the center of the regular polygon. $T_{21}$ and $T_{12}$ lie at the midpoint of the edge next to the edge emanating from the base point $T_{00}$. $T_{22}$ is generally put in the middle of the polygon boundary between $T_{21}$ and $T_{12}$.

From the explanation in Section 3.2, we see that the peak and pit vertices of a control network can have more than four incident curve segments. For these cases, we use the polar parametrization instead of the Varady's one. This is made possible because the curve segments and adjacent faces around the peak (pit) constitute a spider's web in the control network.

3.4 Designing Local Patches

Having defined the mappings between the object surface and the vertex patches, our next step is to determine the shapes of the vertex patches. As described above, the vertex patch is a bivariate parametric patch. In our implementation, the vertex patch
is represented by the tensor-product B-spline surface whose basis functions are cubic endpoint-interpolating B-splines (cf. Section 3.1).

In determining the shape of the vertex patch, we would like to make the patch fit to the curve segments of the control network. This means that we would like to use the curve segments as the underlying geometric constraints of the vertex patches. In addition to this, the surface areas where no constraints are imposed should be interpolated smoothly without unnecessary local bumps and dents. For this purpose, we use the variational techniques developed by Welch and Witkin [39]. The techniques enable us to determine the adequate shape of smooth surfaces by minimizing the deformation of the surface that satisfies the imposed constraints. In order to find such an adequate shape of the surface, an energy function subject to the stretching and bending of the surface is defined and it is optimized while satisfying the imposed constraints.

Now we review briefly the mathematical models of the techniques. Let us denote the tensor-product B-splines by \( b_{ij}(u,v) \) \((i = 0,1,\ldots, j = 0,1,\ldots)\). We also define the vector \( b(u,v) \) as \( (b_{00}(u,v), b_{01}(u,v), \ldots, b_{ij}(u,v), b_{ij}(u,v), \ldots) \), which is a row-major representation of the matrix \( B(u,v) = (b_{ij}(u,v)) \). Hence, the tensor-product B-spline surface \( s(u,v) \) is represented by

\[
s(u,v) = p^T b(u,v),
\]

where \( p \) is a coefficient vector of \( b(u,v) \). In the following, we determine the \( x-, y-, \) and \( z \)-coordinates of the coefficients (i.e., the elements of \( p \)) independently.

The constraints are specified by attaching point or curve constraints to the surface. According to the techniques developed by Welch and Witkin, such constraints are reduced to a system of linear equations with respect to the coefficient vector \( p \):

\[
M p = q.
\]

Here, each row of the matrix \( M \) represents a single linear constraint and the corresponding component of \( q \) represents its value.

As the energy function for the adequate surface design, we use the function that measures how much the surface is stretched and bent by looking at its differential area and curvatures at each point of the surface. The energy function is defined as follows:

\[
E(s) = \sum_{i,j} \alpha_{ij} D_i s D_j s + \beta_{ij} (D_i D_j s)^2 = p^T H p,
\]

where \( D_i s \) represents the partial derivative of the surface \( s(u,v) \) with respect to the \( i \)-coordinate. The first and the second terms of the energy function correspond to the stretching and the bending of the surface, which are controlled by the values of \( \alpha_{ij} \) and \( \beta_{ij} \). From the above equation we see that this energy function is finally reduced to the product of the coefficient vector \( p \) and the matrix \( H \), where \( H \) represents the integral of the energy function over the parametric domain. Consequently, we can find the coefficient vector \( p \) by minimizing the sum of the energy function and the constraint term with the Lagrange multiplier \( y \)

\[
\| \frac{1}{2} p^T H p + (M p - q)^T y \|.
\]

By default, the shape of the vertex patch is determined by solving the constraints of the curve segments in the control network. The geometry of the curve segments is transformed to that of a local vertex patch using the coordinate mappings described in Section 3.3. Furthermore, additional constraints can be attached to the face of the control network and they are then shared by the vertex patches that have effects on the shape of the face. Note that this scheme enables us to specify geometric constraints such as dimensions of the feature parts. For example, we can create systematic bumps and dents on the object surfaces with this framework.

It should be noted that we can introduce multiresolutional representation to the design of vertex patches by incorporating spline wavelets into our representation [32] (cf. Section 3.1). We can also extend this multiresolutional scheme to the whole surface of an object by the patch assembling technique described in Section 3.5 [33].

### 3.5 Blending Local Patches

The final step of generating the object surface is to assemble the vertex patches using the previously described mappings. Since any point of an object surface is covered with more than one vertex patch, the object surface can be generated by blending the vertex patches in their overlapping parametric domains.

Let us calculate the coordinates of the point whose parameter vector is \( x \) as shown in Figure 13. Here, \( x \) is contained in the domains of five vertex patches because \( x \) lies in a five-sided region. Note that Figure 13 illustrates only two of the five vertex patches for simplic-
ity. The parameter vector $x$ is mapped onto the two parametric domains by $x_i$ and $x_j$, where the mapped vectors are denoted by $x_l$ and $x_r$ as illustrated in Figure 13. Since we have already determined the shapes of the local vertex patches, we can obtain the coordinates of $x_l$ and $x_r$ as $s_i(x_l)$ and $s_j(x_r)$. For interpolating the vertex patches, the system uses a blending function $B(x)$, which is a polynomial function in our implementation. The corresponding coordinates $s(x)$ are calculated from

$$ s(x) = \frac{\sum_k B_k(x_k) \cdot s_k(x_k)}{\sum_k B_k(x_k)}. $$

In this way, we can obtain the whole surface of a smooth object.

### 3.6 Other Geometric Operations

Our system also provides the following geometric operations.

**Interference checking.** The system provides operations that find the illegal interferences among surface layers in order to maintain the predefined topological skeletons of the object.

**Flat-surface generation.** In order to support flat surfaces perpendicular to the height axis, the system provides operations that set the height of the surface patches along the height axis to zero. This means that we first design a smooth surface and then press it by setting its height to zero. The basic ideas of these operations are presented in [27].

**Object Embedding.** The system also offers operations that embed objects in 3D space. With the operations, we can design multi-layered objects as shown in Figure 17 by designing the two objects separately and then embedding one inside the other.

### 4 Implementation and Results

Our prototype system is implemented on IRIS workstations using OpenGL as the graphics library and Motif for the user interface.

In our system, users specify the topological skeletons of an object shape and then design the geometry by modifying the flow curves that the system offers by default. Figure 14 shows the display examples of the design procedures of a torus. Firstly, we create the peak of the torus by an E2 icon (Figure 14(a)). Secondly, we create the upper pass of the torus by an E1N icon, where the initial shapes of the flow curves are also shown (Figure 14(b)). Thirdly, we create the lower pass of the torus by an E1S1 icon, which results in the hole of the torus (Figure 14(c)). Finally, we finish designing a torus by an E0 icon (Figure 14(d)). After this topological design, users can design the geometry of the torus by modifying the flow curves.

Figure 15(a) shows the shape of the torus designed in our system. In addition to the above top-down surface construction, we can also attach additional surfaces to the torus using macro operations described in Section 2.3. Figure 15(b) shows such an example, where two arms are added to the torus. We also divide the bottom pit into two to create the legs in the torus, which results in a monster-like object as shown in Figure 15(c).

Figure 16 presents examples designed in our system. Figure 16(a) shows a toy dog and Figure 16(b) shows the characters “SMST”. We can design the multi-layered surface as shown in Figure 17. As described previously, we design the two spiral objects separately and then embed one inside the other.

Note that these objects have smooth shapes around their critical points. In this way, we can design such complicated smooth surfaces systematically using the critical points as the shape features, which shows the capability of our system.

### 5 Conclusions and Future Work

This paper has presented a new feature-based modeling method for smooth surfaces that uses the critical points and CPGs as the shape features. In particular, we use the Reeb graph, which is one of the CPGs, to design the topological skeletons of an object shape.

The iconic representation called the embedded Reeb graph is also used to specify the embeddings of the object in 3D space. The geometry of the object shape is defined using the flow curves that run on the object surface. From the flow curves, the system automatically generates a control network that outlines the object shape. The whole surface is decomposed into local patches, which are determined using constrained variational optimization with regard to the surface deformation. The local patches are then blended smoothly over the whole surface of the object.

The goal of this research is to implement the bidirectional operations between smooth object shapes and their shape features, i.e., the design by features and feature recognition. While we have presented the model for the design by features, we have to develop a model for extracting critical points and CPGs from sampled data of existing object shapes. We are now developing algorithms for extracting the features from surfaces of arbitrary topological type [34, 33]. These algorithms can be used, for example, to change the height directions of the designed objects and to extract the topological skeletons from the objects generated by boolean operations.

We use the direction-dependent features in the proposed approach at the cost of uniqueness of the representations under rotational transformations. This is mainly because the proposed features are natural extensions of the entities in B-reps and the framework of B-reps can therefore be easily applied to our approach. This direction-dependent approach also enables us to describe object embeddings in 3D space. Furthermore, the above algorithms can be used to avoid the limitations caused by the direction dependency. Other extensions include incorporating our model with direction-independent features such as 3D medial axis transforms (MAT) [2]. A method of generating surfaces using a network of lines of curvature was also presented in [16]. While it is not enough intuitive...
and flexible to design surfaces of arbitrary topological type, it will serve as an efficient tool for incorporating our model with surface curvatures and can be used for artistic design of smooth surfaces.

It is efficient for both experienced and novice users to provide high-level operations because these operations hide the low-level implementations of the system. These operations will allow us to attach prototype feature parts such as the combinations of branches and tubes without time-consuming processes. Specifying constraints such as dimensions and tolerances is an important issue in feature-based modeling techniques. In our implementation, we can attach geometric constraints in order to control the shape of the features to meet the specified dimensions. Allowing variations in size such as tolerances is left as future work. Including polyhedral surfaces partially in an object shape will be needed for some engineering purposes, and is also a topic of our future research.

Since the local patches are represented by endpoint interpolating B-splines, they constitute the multiresolution representation with the spline wavelets [32]. We are now developing techniques for editing surfaces using multiresolution constraints [36]. Fair mesh generation from the parametric representation of an object surface is also an important issue for our research, where several techniques [26, 11] can be applied to our system. We would like to apply our model to CG animation and VR systems, too. In particular, modeling the time-varying object shapes is one of our future extensions.

Acknowledgments

The authors would like to thank Dr. Kenji Shimada and Dr. Hiroshi Masuda of IBM Tokyo Research Laboratory, Prof. Kenichi Kanatani and Dr. Naoya Ohta of Gunma University, and anonymous referees for their valuable comments on this research.

References


Figure 15: Designing a monster-like object from a torus: (a) a torus, (b) a torus with arms, and (c) a torus with arms and legs.
Figure 16. Design examples: (a) a toy dog and (b) characters


[27] Shinagawa, Y. A Study of a Surface Construction System Based on Morse Theory and Reeb Graph. Doctoral thesis, Department of Information Science, Graduate School of Science, the University of Tokyo, 1992.


Figure 17. A double-layered spiral


[33] TAKAHASHI, S. Critical-point-based Modeling for Smooth Surfaces, Dissertation submitted to the Department of Information Science, Graduate School of Science, the University of Tokyo, 1997.


