1 Introduction

The process from measuring the geometry of existing objects to generating a new digital model is referred to as reverse engineering [1]. The conventional engineering converts the design concepts into real objects, while the reverse engineering transforms real objects into the concepts. The concepts are represented by design intent, which contains the intrinsic properties of the object’s shape. Typically, model reconstruction has three kinds of applications: (1) the reproduction application; (2) the quality control application, and (3) the redesign and modification application [2]. For the first two cases, the low-level design intent is sufficient for their purposes, where only the exact shape information is needed. For the third case, the high-level design intent, such as the geometric properties and relations, is required, which is the focus of this paper. More specifically, the high-level design intent refers to the information regarding features and constraints.

Over the last two decades, a wide diversity of model reconstruction methods has been studied for converting measured data into geometric models [3–10]. They are mainly classified into two types: (1) surface-based methods and (2) feature-based methods.

Comparatively, the surface-based methods are studied more profoundly. Most of the state-of-art commercial reverse engineering systems such as CATIA, CopyCAD, Geomagic series, Imageware Surfacer, Rapidform, and RE-SOFT take advantage of this surface-based strategy to reconstruct geometric models. The surface-based methods are especially suitable for the products of automobiles and aircrafts. However, they are not applicable to complicated industrial parts, which are designed and manufactured on the basis of geometric features and constraints.

Accordingly, the feature-based strategy is studied to carry out model reconstruction of industrial products [11–14]. Thompson et al. [11] initiated the REFAB project to reconstruct geometric models of mechanical parts based on features and constraints. Ke et al. [14] proposed feature-based reverse engineering strategies for modeling industrial components from point cloud to surfaces and developed a professional software: RE-SOFT. The main idea is to construct the surface features and thereby perform surface operations to generate the final solid model with boundary representation. However, it is nontrivial to impose the global constraints on surfaces to generate a highly accurate and topologically consistent model. Werghi et al. [15,16] and Fisher [17] studied the constrained reconstruction of 3D geometric models of objects from range data. They pioneered in giving such a large framework for the integration of geometric relationships in object...
reconstruction. This method uses the complex formulation of the constraint function, which heavily relies on the convexity of the constraint space. Benko et al. [18], Kos et al. [19], and Lukacs et al. [20] conducted the research of solid model reconstruction with boundary representation and constrained quadratic surface fitting in reverse engineering. Martin et al. [21], Marshall et al. [22], and Langbein et al. [23] studied the model beautification to improve the reconstructed models such that they have exact geometric regularities representing the original design intent. The reconstruction accuracy cannot be guaranteed because the original point cloud is not considered during adjustment.

To reconstruct 3D models of objects with high accuracy and editability, we present an effective solution for model reconstruction of complicated industrial parts. Specifically, we convert the model reconstruction problem into extracting feature parameters, and thereby propose comprehensive methods to extract the parameters of elementary features, including extrude, revolve, sweep, loft, and blend features, from input surface meshes. Having the feature parameters, the 3D geometric modeling engine (Open CASCADE [24]) is exploited to create features. Then, the final solid model is constructed by performing modeling operations on those features. Meanwhile, the complete history tree of the modeling operations is constructed. Note that the aforementioned features have the common characteristics that all of them are generated from 2D contours with some modeling operations. Accordingly, we emphasize on the reconstruction of the common parameter (2D contour) of those features based on geometric constraints, in which the approaches to curve segment partition, constraint detection, and constrained fitting are comprehensively studied. The main contributions of this paper are:

1. An efficient strategy is proposed for solid model reconstruction based on features, constraints and the history of modeling operations;
2. A novel algorithm of 2D point segmentation is designed on the basis of geometric properties and the curve fitting technique;
3. Robust approaches are developed to extract the feature parameters of extrude, revolve, sweep, loft and blend features.

2 Solid Model Reconstruction

The 2D contour is the fundamental parameter for all features; therefore, we introduce the detailed method for 2D contour reconstruction first.

2.1 Constraint-Based 2D Contour Reconstruction. In the reconstruction context, the input of the 2D contour is the planar point set, which is usually generated by slicing the input mesh with a sectional plane. How to determine the sectional plane is one of the key issues in the feature reconstruction, which will be addressed respectively in each feature reconstruction section.

Given a sectional plane, we intersect the plane with the input mesh to get the sectional point set, based on which we construct the 2D contour. Benko et al. [18] conducted the research on the curve and surface fitting based on constraints, and listed the comprehensive expressions of 2D and 3D geometric constraints. Ke et al. [14] developed a planar profile reconstruction method from the point cloud with geometric constraints. In those methods, how to automatically partition 2D sectional points into segments is not studied, and moreover, the conic section curves, commonly existing in the mechanical parts, like ellipse, parabola and hyperbola, are not taken into account. Accordingly, we present a new 2D contour reconstruction approach on the basis of geometric constraints, which covers line, circle (arc), ellipse, parabola, hyperbola, and B-Spline curve. The basic idea is to segment the point set into subsets, where all points have the same underlying curve, and the type of the underlying curve is recognized for each point subset, followed by fitting the subset with a curve. Then, according to the parameters of fitted curves, the potential constraints between curves are detected and verified. By applying those constraints, the fitting on the whole point set is performed to achieve the constrained 2D contour.

2.1.1 Segmentation of 2D Sectional Points. In this section, we partition the point set into subsets. In each subset, all points possess the same underlying curve representation. Based on this definition, we propose a new segmentation algorithm for sectional points, in which the line and circle (arc) are detected first based on the normal vector and curvature information, then a quadratic curve fitting strategy is adopted to recognize the conic curve segments, and the freeform curve segments are extracted finally.

Line segmentation. Given a sectional point set $P = \{p_1, \ldots, p_n\}$ on the plane, let $e = p_{i+1} - p_i$ be the edge of $p_i$ and $p_{i+1}$, we define the normal vector of $e$ as $\mathbf{n}_e = \mathbf{n}_e \times (p_{i+1} - p_i)$, where $\mathbf{n}_e$ is the normal vector of $\pi$. Then, the region growing process is performed to implement the line segmentation. By choosing an arbitrary, unsegmented edge as the first seed, the region is initialized with this seed edge. The region is growing by adding one of two neighboring edges of this seed. The added neighboring edge should satisfy two conditions: (1) the angle between the normal vector of this edge and the seed edge is comparatively smaller; (2) the angle is less than a small predefined value. The angle threshold is typically set to $5^\circ$, which usually produces good results in our implementation. The seed is updated with the latest added edge and the region growing is iterated until no more edges could be added in the current region. By repeating the process above until all edges are segmented, the initial segmentation is thereby completed. For each segment, we store the first seed edge, and calculate the normal vector angles between this seed edge and all other edges within the same segment. If all those normal vector angles are less than the angle threshold (e.g., $5^\circ$), we consider this segment as a line; otherwise, the further segmentation is needed.

Circle segmentation. The geometric property of a circle is that the curvature of each point on the circle is constant. For all points which are not within any line segments, we reset them and their associated edges with the un-segmented state. Similarly, an unsegmented edge is chosen as a seed edge and initially assigned to a region. For each edge in this region, we search its neighboring, un-segmented edges and compute the difference of curvatures of those edges. If the difference is less than a predefined threshold, the neighboring edge is added in the region; otherwise, it is skipped and set a flag to end the search at this edge. The procedure above is repeated until no more edges could be added in the current region. As a result, if this region has sufficient edges, the region is considered as a circle (arc) segment. Up to here, the line and circle segments are extracted. The remaining points and edges are reshuffled to an un-segmented state again, and need to be partitioned further.

Conic and freeform curve segmentation. In mechanical parts, conic section curves are widely used, which are special forms of quadratic curve. Accordingly, we exploit the quadratic curve fitting method to carry out further segmentation on the remaining un-segmented point set. We first choose a few un-segmented, connected edges as a seed region (named as an original seed region), and fit the inclusive points with a quadratic curve. Taking the fitted quadratic curve as an input, the region is growing by adding the points which are not included in but connected with the current region. Each added point should satisfy three compatible conditions with the region: (1) the point should be close to the fitted curve; (2) the normal vector of the projection point on the fitting curve should be close to that of the point; (3) the curvature of the projection point should also be close to those of the point. The region growing process terminates when no more points can be added into the region, followed by fitting a new quadratic curve on the region. Then, we clear all edges in the current region and update it with the original seed region. The region growing procedures above is performed again by taking the new fitting quadratic

011004-2 / Vol. 13, MARCH 2013 Transactions of the ASME
curve as an input. This procedure is repeated until the maximal region is achieved, and hence a final segment is obtained. Essentially, the region growing method attempts to maximize the number of topologically connected points that can be faithfully fitted by a single underlying quadratic curve. In our implementation, we always choose as an original seed region the connect edges having the lowest variance of curvatures. Applying this region growing process to all un-segmented points, all new segments generated here could be fitted by quadratic curves.

Then we can recognize the type of conic section curve from each segment using the intrinsic quantities defined in Ref. [25]. For each segment, if its associated quadratic curve is not a conic section curve, we consider the curve type of this segment as the freeform curve, which will be fitted with a cubic B-Spline curve.

2.1.2 Fitting of Conic Section Curves and B-Spline Curve. For each segment, we fit its associated points with the corresponding curve. The fitting methods for line and circle have been deeply studied [18]. We mainly introduce the fitting methods of the conic section curves and the cubic B-Spline curve using the squared distance minimization technique [26]. We give the squared distance of 2D curve briefly. Given a point set \( P = \{p_1, p_2, \ldots, p_n\} \subset R^2 \) in a plane \( \pi \), the fitting curve is represented with \( C(t) \). Let \( C(t_0) \) be the closest point of a point \( p \in P \) on \( C(t) \), if the shortest distance, and the curvature, curvature radius and curvature center of \( C(t_0) \) on \( C(t) \) are \( k, \rho = 1/|k| \), and \( e(t_0) \), respectively. According to curve theory, the Frenet-Serret frame of a point with the parameter of \( t \) on \( C(t) \) can be represented by the tangent vector \( z(t) \) and the normal vector \( \beta(t) \) of \( C(t) \). Consequently, a local Cartesian coordinate system \( xoY \) on \( C(t_0) \) in \( \Pi \) could be constructed, where the origin, \( x-, y- \) axis are \( C(t_0), z(t_0), \beta(t_0) \), respectively. Under \( xoY \), the local coordinates of \( p \) and \( e(t_0) \) are \((0, d)\) and \((0, -\rho)\).

Let \( q = (x, y) \) be in a small neighborhood of \( p \), the second-order Taylor approximant \( F \) of the squared function at \( q \) is

\[
F(x, y) = \frac{d}{d-p} \left[ (q - C(t_0))^2 \right] + [\beta(t_0) \cdot (q - C(t_0))]^2
\]

which is referred to as the squared distance function of 2D curve. Note that \( F(x, y) \) may have a negative value if \( 0 < d < \rho \), which may lead to the failure of the following optimization iterations. In this case, we remove all negative items in the squared distance function.

Let \( X \) be the parameter vector of the \( k \)-th-iteration fitting curve \( C(t) \). \( C^1(t) \) denotes the \((k+1)\)-th-iteration fitting curve with updated parameter vector \( X = X + D \), where \( D \) are incremental updates to \( C(t) \). The closest point of \( p_i \) on \( C^1(t) \) is \( c^1_{ij} \), which is different from the closest distance \( c_i \) of \( p_i \) to \( C(t) \). However, because the difference is quite small, we approximately substitute \( c^1_{ij} \) with \( c_i \). Suppose that \( p_i \) is the state vector of \( c_i \), Eq. (1) can be transformed to

\[
E(X) = \frac{d}{d-p} \left[ (q - C(t))^2 \right] + [\beta(t) \cdot (q - C(t))]^2
\]

By minimizing the sum of the squared distances of all points, the parameter vector \( X^* \) of \( C^1(t) \) is obtained, i.e., the fitting curve is updated to \( C^1(t) \) from \( C(t) \).

Based on this squared distance, we have the fitting methods for circle, ellipse, hyperbola, parabola and cubic B-Spline curve. The initial parameter vector of each curve is estimated by the least square minimization based on the algebraic distance [31]. We compute the squared distance for the whole data points in terms of the parametric representation of the curve, and hereby minimize the squared distance function. Using the initial parameter vector as an input, we solve a linear system of optimization equations to obtain the updated parameter vector of the curve. The process above is repeated iteratively until convergence. As a result, the final fitting curve is achieved.

2.1.3 Constraint-Based Fitting for 2D Sectional Points. After all segments are fitted individually, we can recognize the potential constraint relationship of fitted curves based on the geometric properties of constraints. For example, given two fitted lines: \( l_1x + l_2y + l_3 = 0 \), \( l_4x + l_5y + l_6 = 0 \), if \( l_2^2 + l_5^2 \geq \cos^2(\pi/2 - x) \), then there is a potential parallel constraint between these two lines; if \( l_2^2 + l_5^2 \geq \cos^2(\pi/2 - x) \), then these two lines are potentially perpendicular. We may consider a more complicated case. Based on this idea, we detect all potential constraints of fitted curves. Since the over-constrained or under-constrained cases may exist, the constraints still need to be verified.

Once the constraints are determined, we can apply those constraints to curve fitting so that a global optimization system is generated. Let \( S = \{S_1, \ldots, S_n\} \) be the set of individual curve segments in a sectional contour (\( m \) is the number of segments). Suppose \( X \) is the parameter vector of all curves in \( S \), and \( C_i(X) \) is the \( k \)-th constraint function, then the constraint-based fitting problem could be expressed as

\[
\min F(X) = \min \sum_{i=1}^{m} \left( \omega_i \sum_{j=1}^{N_i} d^2(C_j(p_i), S_i) \right) \quad (3) \quad \text{s.t.} C_i(X) = 0 \quad (k = 1, \ldots, t) \]

where \( p_j \) is the \( j \)-th point in the associated point set of the \( i \)-th curve \( S_i \), \( d(p_j, S_i) \) is the distance from \( p_j \) to \( S_i \), \( N_i \) is the number of associated points of \( S_i \), \( d(p_j, S_i) \) is represented by the algebraic distance. Using the penalty function method, this constrained optimization problem can be converted into the following nonlinear optimization model without constraints

\[
\min E(X, \lambda) = \min \left( F(X) + \lambda \sum_{i=1}^{t} [C_i(X)]^2 \right) = \min \left( X^T H X + h X + K + \lambda \sum_{i=1}^{t} [C_i(X)]^2 \right) \quad (4)
\]

where \( H, h, \) and \( K \) are the coefficients of each curve in terms of its associated points; \( \lambda \) is the penalty term. This nonlinear optimization problem can be further transformed to a linear equation system by Levenberg–Marquardt algorithm [16]. Specifically, we can start with the initial parameter vector \( X^{(0)} \) to minimize the least squares objective function so that the nearby parameter vector \( X^{(1)} \) is obtained given small penalty terms. Then, by slightly increasing the penalty terms, we compute the new parameter vector \( X^{(2)} \) based on \( X^{(1)} \). By repeating this procedure, we iteratively increase penalty terms and update the parameter vector until the parameter vector is stable for a given number of iterations. Consequently, the optimal parameter vector can be achieved. The initial parameter vector is from the individual fitting results from the Sec. 2.1.2. Note that the initial parameter vector is not far away from the optimal ones since they are obtained from the real point set, which is crucial to guarantee the convergence of the algorithm to the desired solution. We also notice that if there is a high level of noise contained in the point set, leading to the initial parameter vector with big errors, the constrained fitting may fail in some cases. After solving the equation system, the constrained sectional contour is obtained. To demonstrate the effectiveness of the 2D contour reconstruction method, the reconstruction results from two industrial parts are given in Figs. 1 and 2.

2.2 Extrude Feature Reconstruction. In the conventional modeling, an extrude feature is created by extruding a 2D contour along a constant direction with a certain distance, where the direction is perpendicular to the underlying plane of the 2D contour. As a result, the normal vector of each point on the side surface of the feature is orthogonal to the extrusion direction. Let \( P \) be the...
point set on the side surface of an extrude feature, \(N = \{n_1, \ldots, n_m\}\) (\(m\) is the number of points) the corresponding normal vector set of \(P\), then we construct the following optimization model to calculate the direction \(v_a\):

\[
\begin{align*}
\text{min} & \quad v_a^T \left( \sum_{i=1}^{m} (n_i \cdot n'_i) \right) v_a \\
\text{s.t.} & \quad \|v_a\|_2 = 1
\end{align*}
\] (5)

According to Kuhn-Tucker theorem [27], the optimal solution of this model is the corresponding unit eigenvector to the smallest eigenvalue of \(\sum_{i=1}^{m} (n_i \cdot n'_i)\). By solving the constrained optimization model, the direction of extrusion is obtained, followed by constructing the sectional plane determined by the extrusion direction and the centroid of \(P\). Therefore, we can reconstruct the 2D contour of the extrude feature using the method in section 2.1.

Next, we determine the distance of extrusion. The range of extrusion is usually given in two ways in CAD softwares: (1) the bound surfaces or (2) the specified distance. For the former case, we could simply choose the existing bound surfaces as the start, end points of the extrude feature. For the latter case, we project the points of the side surfaces onto the extrusion direction to determine the extrusion distance. Figure 3 gives the reconstruction result of the extrude feature in a mechanical part.

2.3 Revolve Feature Reconstruction. In the conventional modeling, a revolve feature is created by revolving a 2D contour around an axis within a certain angle range. The axis is coplanar with the lines along the normal vectors of all points on the side surface of the revolve feature, based on which we can extract the axis of revolution. Let \(P = \{p_1, \ldots, p_m\}\) be the point set on the side surface, \(N = \{n_1, \ldots, n_m\}\) the corresponding normal vector set of \(P\). In Fig. 4, \(p_a\) is a point on the axis, \(v_a\) is the unit vector of the axis, \(\varphi_i\) is the angle between \(n_i\) and \(v_a\), and \(\text{dist}_i\) is the distance between the axis and the line determined by \(p_i\) and \(n_i\), which is defined as

\[
\text{dist}_i = \frac{|(p_a - p_i) \cdot (v_a \times n_i)|}{|\sin \varphi_i|}
\] (6)
For each point, the line along its normal vector should theoretically be coplanar with the axis of revolution. Therefore, we have this following optimization model to obtain the axis

\[
\min \sum_{i=1}^{m} \text{dist}^2 = \min \sum_{i=1}^{m} \left[ (p_i \times v_u) \cdot n_i + (p_i \times n_i) \cdot v_u \right]^2
\]

\[\text{s.t.} \|v_u\|_2 = 1\]  

This constrained optimization problem can be solved by the Lagrange-Newton method [28], i.e., the axis of the revolve feature is obtained.

Then, we construct sectional planes and extract the 2D contour of the revolve feature. A base plane \( \pi \) is created firstly, which passes the line determined by \( p_u \) and \( v_u \). We construct a coordinate system \( o-xyz \), in which \( p_u \) is the origin, \( v_u \) is the z-axis, and the x-, y- axis are randomly chosen from two unit orthogonal vectors having the same start point \( p_u \) in \( \pi \). The plane passing x-o-z is
constructed as the start sectional plane \( p_{l0} \), then a series of sectional planes \( p_{l1}, \ldots, p_{lk} \) are generated by rotating \( p_{l0} \) along the \( z \)-axis with a series of angle intervals \( \alpha, \ldots, \alpha_k \). By intersecting those sectional planes with the side surface, a series of sectional points are obtained and transformed to \( x-o-z \), see Fig. 5. After that, we use the method in Sec. 2.1 to reconstruct the 2D contour of the revolve feature.

Next, the angle range of the revolve feature needs to be extracted. Similarly, there are two methods to determine the range: (1) the bound surfaces or (2) the angle range. For the former case, we could choose the bound surfaces manually as the start, end points of the revolve feature. For the latter case, we project all points on the side surface of the revolve feature onto \( \pi \) and calculate the angle range of the projection points around \( p_a \) on \( \pi \) as the angle range of the revolve feature. Figure 6 shows the reconstruction result of the revolve feature.

2.4 Sweep Feature Reconstruction. In the conventional modeling, a sweep feature is constructed by sweeping a profile section (2D contour) along a spatial path. Basically, the sweeping operation has two types: (1) the profile section remains perpendicular to the path at all times; (2) the profile section remains parallel to the beginning section at all times. The feature from the latter case could be considered as a loft feature, which will be discussed in Section 2.5. Here, we only consider the former case. From the construction of the sweep feature, the profile section at each point on the path should be orthogonal to the tangent vector in the Frenet–Serret frame of the point on the path. Analogously, the profile section at each point on the surface of the sweep feature should be orthogonal to the tangent vector in the Darboux frame of the point on the surface. Based on this rule, we extract the path of the sweep feature.

Let \( P = \{p_1, \ldots, p_i, \ldots, p_m\} \) be the point set on the surface of a sweep feature, \( N = \{n_1, \ldots, n_i, \ldots, n_m\} \) the corresponding normal vector set of \( P \). For each point \( p_i \), we search its 2-ring neighboring points and fit them with a quadratic surface, followed by computing the principal directions of \( p_i \). The Darboux frame of \( p_i \) is constructed and the tangent vector \( T_i \) is obtained in the frame. Then the section plane \( \pi_i \) is created at \( p_i \) with the normal vector of \( T_i \). After that, we intersect \( \pi_i \) with \( P \) to get the intersection points \( IP_i = \{ip_i1, \ldots, ip_ij, \ldots, ip_il\} \), the normal vectors of which are \( N_i = \{n_i1, \ldots, n_ij, \ldots, n_il\} \). Note that the section plane may encompass multiple shape parts and thus the intersection points may include some “fake” points, which are not geometrically coherent with \( p_i \). Since we already have the connectivity information, the “fake” intersection points could be easily detected and removed. Having the intersection points \( IP_i \) of \( p_i \), we can calculate the associated point \( ap_i \) of \( p_i \) on the path, i.e., the intersection point between the profile section at \( p_i \) and the path of the sweep feature. According to the geometry characteristics of the sweep feature,
proposed an effective method to extract curve skeleton from plane of each point can be found iteratively. Tagliasacchi et al. with the surface of the feature. Accordingly, the optimal section vectors at a set of relevant points intersected by the section plane normal vector minimizes the variance of angles with the normal symmetry, the optimal section plane of each point features that its feature. According to the characteristics of the profile sections. Specifically, we need to obtain the optimal rotationally symmetric. Based on this observation, we can extract with a sufficiently small distance, could be considered generalized tours. The shape, cut from the loft feature by two section planes transformed into skinning a series of profile sections (2D con-sections via some constraints. Basically, this operation can be solved via an iterative method with the following programming problem, which is quite complicated. For simplicity, it can be solved via an iterative method with the following model:

\[
\mathbf{n}^{(t+1)} = \arg\min_{\mathbf{n} \in \mathbb{S}^2} \text{var}\{ \langle \mathbf{n}, \mathbf{n}_p \rangle : p \in H_i^{(t)} \}, \quad t \geq 0
\]

where \( H_i^{(t)} \) is the point set intersected by the section plane at the \( t \)-th iteration with the surface of the feature, and \( \mathbf{n}_p \) is the unit normal vector of \( p \) in \( H_i^{(t)} \). This variational problem has a closed form solution. After solving this problem, the optimal section plane of each point is obtained. Then, we intersect the section plane \( \pi_p^t \) of each point \( p \) with the surface, and get the intersection points \( I_p^t \) of \( p \).

Next, we determine the profile sections. Since the intersection points of all points in \( P \) are gotten, the profile sections associated with all those points could be reconstructed with the method in Sec. \( 2.1 \). As a result, there are a huge number of profile sections. However, it is not realistic to use all of those profile sections to generate the loft feature. Note that there are many approximately congruent profile sections. Now, we need to detect those similar profile sections and remove them. To this end, the dilation operation in mathematical morphology is exploited.

Let \( I_{\mathcal{P}_0} = \{ p_0, \ldots, p_m \} \) be the set of intersection points between the optimal section plane \( \pi_p^* \) of \( p \) and the loft feature mesh \( M \), \( I_{\mathcal{T}_i} = \{ \mathcal{T}_0, \ldots, \mathcal{T}_l \} \) be the set of triangular faces intersected by \( \pi_p^* \) with \( M \). All points in \( P \) are set with the “unvisited” state. Suppose \( G_p \) represents a set of points whose profile sections are similar to that of \( p \), then we have the following pseudocode to

\[
\mathbf{n}^{(t+1)} = \arg\min_{\mathbf{n} \in \mathbb{S}^2} \text{var}\{ \langle \mathbf{n}, \mathbf{n}_p \rangle : p \in H_i^{(t)} \}, \quad t \geq 0
\]

Fig. 7 Reconstruction of a sweep feature. (a) The triangular mesh of the feature; (b) the reconstructed sweep path and profile; (c) the reconstructed sweep feature; (d) the reconstruction error graph. The diameter of the bounding sphere of the feature is 2.960 and the average edge length of the input mesh is 0.024, while the maximum error is 0.010. Again, the reconstruction result is favorable.

Fig. 8 2D illustration of iterative construction of an optimal section plane of a point on the loft surface. (a)-(c) The section plane anchored at \( p_1 \) from \( \pi_1^0 \) to \( \pi_1^1 \) and converges. During iteration, the normal vector of the next section plane makes the same angle with those normal vectors at the boundary, corresponding to the current section plane.

2.5 Loft Feature Reconstruction. In the conventional modeling, a loft feature is created by blending multiple profile sections and transitioning them into smooth shapes between the profile sections via some constraints. Basically, this operation can be transformed into skinning a series of profile sections (2D contours). The shape, cut from the loft feature by two section planes with a sufficiently small distance, could be considered generalized rotationally symmetric. Based on this observation, we can extract the profile sections. Specifically, we need to obtain the optimal section planes associated with all points on the surface of the loft feature. According to the characteristics of generalized rotational symmetry, the optimal section plane of each point features that its normal vector minimizes the variance of angles with the normal vectors at a set of relevant points intersected by the section plane with the surface of the feature. Accordingly, the optimal section plane of each point can be found iteratively. Tagliasacchi et al. [29] proposed an effective method to extract curve skeleton from incomplete point cloud, in which the similar cutting planes of points are computed. Here, we use this method to extract the optimal section plane. Figure 8 illustrates the iterative construction of the optimal section plane of a point on the surface of the feature.

Let \( P = \{ p_1, \ldots, p_m \} \) be the point set on the surface of a loft feature. \( \mathbf{N} = \{ \mathbf{n}_1, \ldots, \mathbf{n}_m \} \) the corresponding normal vector set of \( P \). For each point \( p_i \), we need to find an optimal section plane \( \pi_p^i \) such that the normal vector \( \mathbf{n}_p \) of \( \pi_p^i \) is most rotationally symmetric about the normal vectors of the intersection points between \( \pi_p^i \) and the surface of the loft feature. It is a nonlinear programming problem, which is quite complicated. For simplicity, it can be solved via an iterative method with the following model:
construct $G_p$ (initially, $G_p = \{p\}$), where there are a few variables: $F$ is a set of triangular faces, initialized with $I_Tri$; $V$ is the set of vertices of $F$; $T_Tri$ is another set of triangular faces, initialized with $\emptyset$; $E$ is a set of edges, initialized with $\emptyset$.

**Algorithm 2.1: GATHER POINTS($p$)**

while $I_Tri \neq \emptyset$

1. $F \leftarrow I_Tri$
2. $V \leftarrow$ all "unvisited" vertices of $I_Tri$
3. for each $v \in V$
   - do calculating the difference from $v$ to $\pi^*$ of $p$
   - sort $V$ increasingly in terms of the distance
   - do projecting the intersection points of $v$ onto $\pi^*$
   - calculating the Hausdorff distance $h_{d,v}$
   - between the projection points and $IP_t$
   - if $h_{d,v} < \xi$ then return $(G_p)$
   - else set a "visited" flag to $v$
   - add $v$ into $G_p$
4. $I_Tri \leftarrow$ DILATION OF FACES ($I_Tri$)

Consequently, all points in $G_p$ constitutes a segment. Using the point gathering method, the entire point set of the surface of the loft feature could be partitioned into different segments. In each segment, the associated profile sections of all points are approximately congruent. Therefore, we can adopt the method in the Sec. 2.1 to fit a 2D contour on the basis of the associated intersection points of points in each segment. By reconstructing 2D contours of all segments, the profile sections of the loft feature are obtained. Figure 9 gives the reconstruction results of a screwdriver.

**2.6 Blend Feature Reconstruction.** Blend features often exist in mechanical engineering parts for reasons of...
manufacturability, aesthetics, stress reduction and so on [19]. From the design point of view, the blend features are generated by rolling a sphere ball with constant or variable radius moving in contact with two adjacent primary surfaces, see Fig. 10. Consequently, the center of the rolling ball forms a spine curve. The main parameters consist of the spine curve and the radius of rolling ball. More specifically, we may convert the radius parameter to the profile curve, i.e., the intersection of the blend surface and the normal plane determined by the tangent vector of the point on the spine curve. For the rolling-ball blend feature, the profile curve is apparently a circular arc. Therefore, we reconstruct the blend feature by extracting the spine curve and the profile curve.

Fig. 11 The illustration of determining the section plane of blend feature

![Fig. 11](image1)

Fig. 12 The distribution of radius for three cases

![Fig. 12](image2)

Fig. 13 Reconstruction of mechanical parts with blend features. (a) The triangular mesh of a mechanical part containing the blend feature with the linear variable radius profile; (b) the series of profile curves (circles); (c) the reconstructed feature; (d) the feature in the original model; (e)--(h) The reconstruction of the blend feature with the nonlinear variable radius profile; (i), (j) The radius distribution of the blend features in (a) and (l). From (i), we can see that the fitting error is quite small, indicating that the radius varies linearly, while the radius distribution in (j) shows that the radius varies nonlinearly.

![Fig. 13](image3)
According to the modeling principle of blend feature, we make the reasonable approximation: given two sufficiently close points on the spine curve, the blend feature bounded within two normal planes of these points is considered as a cylinder. Consequently, the axis vector is the same as the normal of the normal plane. For simplification, Fig. 11 gives the 2D illustration of blend feature, where \( \alpha \) is an initial section plane associated with a point \( p_i \) on the blend surface and \( n_i \) is the normal of \( \alpha \). \( p_i \) is the closest point to \( p_j \) on the blend surface and \( \beta \) is the initial associated plane of \( p_j \), which is set parallel to \( \alpha \). \( c_{ti} \), \( c_{tj} \) are the centroids of the intersection points between \( \alpha \) and \( \beta \) and the original blend surface point data. The two centroids form a vector, i.e., \( n_{ij} = c_{tj} - c_{ti} \). According to the approximation, \( n_i \) should be equal to \( n_{ij} \). Based on this characteristic, we can iteratively adjust the section plane such that the normal of the section plane is sufficiently close to the vector determined by two intersection centroids, see Fig. 11. As a result, an optimal section plane is achieved for each point on the blend surface; meanwhile, the intersection points associated with each section plane are also obtained.

Since the profile curve is a circular arc, we fit with a circle the intersection points of the section plane of each point. After fitting all sequential intersections with a series of circles, the centers of those circles discretely constitute the spine curve. Thus, we fit those centers with a B-spline curve as the spine curve. Usually, the rolling ball is fixed, i.e. the radius of the ball is constant, while the radius also varies sometimes. For the latter case, the variation could be linear or nonlinear. There are three different cases in terms of the radius variation, and Fig. 12 shows the radius distribution of three cases.

![Figure 14: Model reconstruction of a mechanical part.](image)

- **(a)** Constant radius profile. All radii are same so the distribution is a line parallel to the x-axis in Fig. 12(a).
- **(b)** Linear variable radius profile. The radius varies linearly so the distribution is an oblique line in Fig. 12(b).
- **(c)** Nonlinear variable radius profile. The radius varies nonlinearly so the distribution is a nonregular curve in Fig. 12(c).

To determine the type of the profile curve, we collect all radii of the series of circles and fit them with a line. If the fitting error is greater than a threshold, we consider it as the nonlinear variable radius profile. Otherwise, we further check if the line is parallel to the x-axis within some tolerance; if so, it is the constant radius profile; otherwise it is the linear variable radius profile.

The blend feature with constant radius profile can be constructed by sweeping the constant arc along the spine curve; the blend feature with linear variable radius profile is constructed by lofting the starting arc to the ending arc along the spine curve. For the blend feature with nonlinear variable radius profile, we sample all profile arcs uniformly and loft those sampling profile arcs along the spine curve to obtain the blend feature. Figure 13 shows the reconstruction results of blend features.

### 2.7 Other Feature Reconstruction

We have presented corresponding reconstruction algorithms for comprehensive elementary features thus far. In CAD systems, there are a few more basic features, such as draft and taper features; hole feature; shell, hollow, and thicken features. For the draft or taper feature, we consider it as a loft feature, suggesting that we can extract the profile section curves and loft those profile curves along a certain

![Figure 15: Model reconstruction of another mechanical part.](image)

(a) The original triangular mesh of the part; (b) a reconstructed 2D contour and the corresponding extrude feature; (c) a series of sampled profile sections and the reconstructed loft feature; (d) a reconstructed 2D contour and the corresponding revolve feature; (e) the final reconstructed model; (f) the reconstruction error graph, where the average edge length of the input mesh is 0.099, while the maximum error is 0.011. (g) A redesigned model from (e).
Fig. 16 Model reconstruction of a cup. (a) The original triangular mesh of the teapot; (b) the reconstructed sweep profile and path; (c) the reconstructed contour and axis of a revolve feature; (d) the final reconstructed model; (e) the reconstruction error graph, where the average edge length of the input mesh is 0.877, while the maximum error is 0.324. (f) The sweep profile modified by changing the radius, and the sweep path modified by adjusting control points from (b); (g) the modified revolution contour from (d); (h) the new model.

Fig. 17 Solid models reconstructed from our method
direction (e.g., perpendicular direction to the profile section) to generate a draft or taper feature. For the hole feature, we treat the hole part as an extrude feature. We can construct it from the point set on the hole surface and execute the “subtraction” Boolean operation from the base feature to obtain the hole feature. For the shell or hollow feature, we consider the internal removed part as an extrude feature, and reconstruct the extrude feature from the point set on shell or hollow surfaces. After constructing the external bounding feature and the internal feature, we perform the “subtraction” Boolean operation to get the shell or hollow feature. For the thicken feature, it generally works towards surfaces, which is not handled in this paper.

3 Results and Discussions

All algorithms described have been implemented with Visual C++ and OpenGL, and run on a PC with 1.8 GHz CPU and 2 GB RAM. We have tested our method on a number of industrial models with triangular meshes, and presented a couple of results in this section.

Figure 14 gives the model reconstruction result of a mechanical part, which consists of a few extrude features. From the reverse engineering point of view, the model can be disassembled into many basic features. For each basic feature, we adopt the proposed method to extract the feature parameters and thus reconstruct the corresponding model. Then, we can perform Boolean operations on those features to generate a final model. Figure 15 shows the model reconstruction result of another mechanical part, which includes an extrude feature, a revolve feature and a loft feature. Figure 16 is the model reconstruction result of a cup model, including a sweep feature and a revolve feature. Figure 17 shows a few more examples with our method. To calculate the reconstruction error, the triangulation of the model is implemented. Then we measure as the error the distance from the reconstructed model to the original model. From the error graphs, the reconstruction accuracy is significantly high.

To demonstrate the editability of the results from our method, the redesigned models are also given in Fig. 14, Figs. 15 and 16. Since the history tree of modeling operations on basic features has been constructed and recorded, it is quite convenient to edit the leaves of the history tree (e.g., change the dimensions of elements in a 2D contour or imposing new constraints on elements in a 2D contour) so that all associated features will be notified to update recursively to generate a new model.

4 Conclusions

We have developed an effective solution to reconstruct geometric models of existing objects based on the features, constraints and the history of modeling operations. The high-level structures of objects and the design intent are retrieved and exhibited in the reconstructed result. Consequently, the model from our method is highly accurate, which is suitable for the reproduction and quality control applications, and meanwhile it is also flexibly editable, which is quite applicable for the model redesign and modification purposes in the innovation applications. Specifically, we convert the feature reconstruction into extracting feature parameters, and hereby develop the comprehensive methods to acquire the feature parameters of basic features. In addition, special efforts have been put on the reconstruction of the 2D contour based on constraints from a 2D point set. A number of industrial parts have been tested with our method and the favorable results are achieved.

For a complicated 3D model, from the design point of view, it is assembled based on many elementary features. Therefore, prior to reconstructing features from points, we need to disassemble the entire model into unit features which can be reconstructed with the aforementioned methods. In our system, we first partition the input point data into different segments each of which has the same geometric and topological properties, such as curvature, normal, and so on [30]. Then we need to interactively pick those surfaces belonging to the same feature to complete reconstruction. Thus, how to automatically recognize the segmented surfaces belonging to the same elementary feature is worth studying in the future.

In some cases, features are symmetric within the models. For example, the rectangle holes in Fig. 14 are centrally symmetric and the circular holes in Fig. 15 are reflectively symmetric. Applying the symmetric constraints gives rise to more accurate results. Even though a number of reconstruction algorithms for fundamental features have been proposed in this paper, it still may not be sufficient to reconstruct complicated 3D models, especially for the models containing freeform surfaces. In those cases, the surfaces are represented with complex geometries (e.g., NURBS surfaces), and the models are designed partially by surface stitching or trimming. Consequently, our method is incapable of handling those cases. All these aspects are worthy to be studied in our future work.

References


