A Feature-Based Approach to Re-engineering CAD Models from Cross Sections

Antonis I. Protopsaltis[1] and Ioannis Fudos[2]

[1]University of Ioannina, antonis@cs.uoi.gr
[2]University of Ioannina, fudos@cs.uoi.gr

ABSTRACT

We introduce a novel approach to reconstructing 3D objects from cross sections of point clouds acquired by 3D scanning. In this context cross sections are almost planar clusters of 3D points. We first thin each cluster to obtain an ordered one dimensional set of planar points. We then partition the point set to subsets that can be approximated adequately by piecewise quadratic rational Bezier curves using an optimal fitting method. For each curve we select a number of representative points that lie on the fitting curves which are then used for reconstructing the object surface. Inter-cross section and intra-cross section constraints are imposed to support parameterization and editing of the derived model. Shape and topological differences between adjacent object contours pose several issues for the 3D reconstruction process. By using the contour skeleton information we produce intermediate cross sections representing places where ramifications occur to achieve robust covering (meshing) of adjacent slices. Finally, we present a proof of concept implementation of our method and several examples that demonstrate its effectiveness and efficiency.

Keywords: feature-based, constraint-based reverse engineering.
DOI: 10.3722/cadaps.2010.739-757

1 INTRODUCTION

The creation of an appropriate computer representation of existing objects from vast sets of scanned data points has been an important necessity in many areas of engineering, medical sciences and arts. The process of capturing the geometry of existing physical objects and then using the data obtained as a basis for creating a new design is called Reverse Engineering of solids. Due to recent advances in laser scanning, the process of deriving accurate and topologically consistent models that are ready to use in CAD/CAM systems has become a realistic expectation in the geometric modeling community.

While conventional engineering transforms engineering concepts and models into real parts, in reverse engineering actual parts are transformed into computer models suitable for reproducing or redesigning these parts. In conventional computer-aided design the computer representation of objects is performed by means of operations typically defined interactively using advanced geometric and graphics primitives. The resulting representation is then used for further design, and finally for numerically controlled manufacturing, layered manufacturing or other manufacturing techniques. In

reverse engineering, engineering concepts are derived from actual parts when no drawings or documentation are available.

The process of reverse engineering is usually decomposed into the following steps: data acquisition, point cloud segmentation, surface fitting, and model creation (Fig. 1).

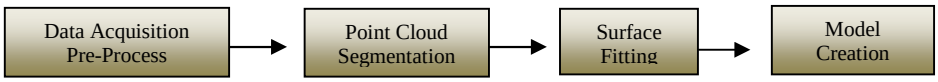

Fig. 1: Phases of reverse engineering.

Data acquisition is accomplished by means of 3D laser scanners or other less accurate techniques such as 3D reconstruction from 2D snapshots using correspondence and epipolar geometry. The data acquired is in the form of an unorganized 3D point cloud where each point corresponds to a point on the surface of the object. The measured data is pre-processed before further operations are performed. In many cases where the object is large or very complicated one point cloud is not enough to describe the entire object. In such cases we obtain multiple point clouds each one covering a different part of the object. These point clouds are either merged in one master point cloud or are considered as segments from the beginning.

The object may be anything from a combination of smaller objects to an open surface. Segmentation partitions the point cloud into disjoint subsets each represented by a boundary representation that consists of surfaces. Each derived subset may be classified for its surface types (planar, spherical, conical, etc) [2] or approximated in the fitting step with free-form surfaces.

The fitting surface step fits an appropriate surface to the point set. This is an open research field in CAGD (see e.g. [3]).

Finally, stitching together these surfaces (with appropriate continuity) creates a Boundary Representation that could be used in subsequent phases of CAD/CAM.

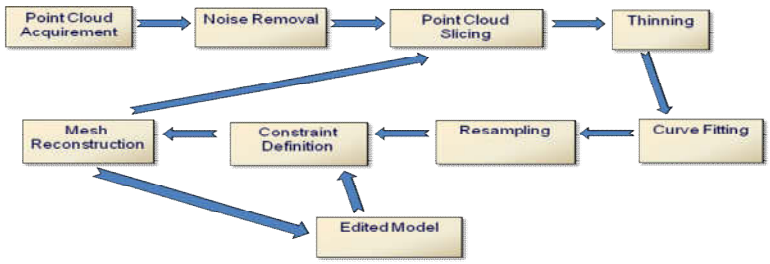

Fig. 2: Our reverse engineering framework.

Traditionally, the result of this process is a Brep model of the real object that is adequate to describe positional information and therefore it is suitable for reproduction but cannot capture any of the higher-level structure of the object or the designer's intent. Therefore, it is not suitable for redesign. Modification of a part is often a tedious task that requires experienced users and state of the art software and hardware. For instance, a Brep representation might be able to approximate the shape of a cylindrical hole, but the fact that the hole is actually cylindrical is not captured. As a result, it is difficult for a designer to perform a simple modification such as altering the diameter of the hole. Also, the initial model suffers from inaccuracies caused by sensing errors inherited from the data acquisition phase, approximation and numerical errors arising from successive transformations or other geometric manipulations, or possible wear of the actual part. All these errors introduce distortion and may act accumulatively. Redesign may be accomplished through geometric regularities and constraints that have been derived from the original cloud point.

We present a novel computer aided reengineering paradigm based on careful slicing of the 3D point cloud and advanced post processing of the resulting cross sections.

Post processing aims to eliminate noise and partition the point set to point sequences that correspond to low degree curve segments. The curve segments are then approximated using quadratic rational Bezier curves. We then subdivide the curve segments in equal length chord segments and use the corresponding points to perform 3D mesh reconstruction.

Fig. 2 illustrates the overall process.

2 RELATED WORK

Varady et al [19] compute a "feature skeleton" on the mesh that determines the primary regions of the object. The final surface structure comprises the optimally located boundaries of the connecting features and setback type vertex blends, which are faithfully aligned with the actual geometry of the object. This CAD-like surface structure is sufficient for high-quality surface approximations. Stamati [14] is using an advanced surface analysis technique to extract the morphology of the reconstructed point cloud. This technique is very powerful and accurate but is not suited for rapid reverse engineering since it requires an extensive analysis process.757

Researchers such as [17], [18] have focused on creating high accuracy models of manufactured mechanical parts. The REFAB project uses a feature-based and constraint-based method to reverse engineer mechanical parts. REFAB is a human interactive system where after the 3D point cloud is presented to the user, the user selects a feature from a predefined list of features, and specifies the approximate location of the feature in the point cloud. The system then fits the specified feature to the actual point cloud data using a least square means method iteratively. The authors give emphasis on the fitting of pockets, where the user draws a profile of the pocket on the point cloud and the system then fits the profile to the data and the profile is then extruded to create the pocket. This feature-fitting process is made more accurate by using constraints that are detected by the system, verified by the user and then exploited to achieve a better fitting of the features according to the data. The system supports constraints such as parallelism, concentricity, perpendicularity and symmetry. The constraints defined and used in REFAB seek to reduce the degrees of freedom associated with the object as much as possible, so as to achieve high precision models in less time.

A feature-based reverse engineering method was also used by Au et al [1] for reverse engineering a mannequin for garment design. Generic models of mannequin torsos are fit to 3D point clouds of human torsos for garment modelling applications. The basic concept in this method is to create a generic mannequin model of a human torso, which is appropriately aligned with the 3D point cloud of the desired human torso model, and the generic model is fit to the point cloud by matching up characteristic points of the models e.g. peaks. This method creates parameterized models by exploiting the features of the object and by using them to constrain the fitting process. It is an automated approach to reverse engineering human torsos that creates parameterized models with good accuracy.

3 EXTRACTING AND RECONSTRUCTING CROSS SECTIONS

3.1 Slicing Direction

Our reconstruction process starts by slicing the point cloud data into a number of cross sections along a user-specified slicing direction. A single slicing direction may not be sufficient for complex objects. For such cases the object is decomposed into parts using advanced segmentation techniques. The cloud points in each slice are projected onto a plane perpendicular to the slicing direction. The slice thickness is controlled by a user defined thickness threshold value that specifies the maximum allowable width of a projected point set. The thickness threshold value is adapted iteratively until it falls under the user specified levels. Slice selection may be controlled by a user defined parameter called slicing distance. Slicing distance specifies the fixed distance between two adjacent slices. There may be cases where the slicing distance is too large for a certain object. As a consequence, the exact geometry of the object is not recorded accurately. The slicing distance parameter should be set according to the object particular features.

In many cases we obtain adjacent slices that are very similar. This might happen when the sliced object feature is symmetric such as a cylinder or parallelepiped part. Many of these slices may be eliminated from the entire process of reconstruction. If three adjacent slices are of similar shape, then the in between slice is eliminated. Similarity of slices may be detected using principal component analysis and skeleton extraction so as to achieve rotational and translational invariance.

3.2 Pre-prosessing and Thinning

Depending on the data acquisition and the slicing process a cross-section may contain points that form a shape with thick border. Thinning is the process that identifies the specific points from the data set that are essential to form the actual 2D shape of the cross-section. We call the outcome of this thinning process a thin data set.

The Medial Axis, is a well defined process for extracting a skeleton, but does not always produce a skeleton for the purposes of thinning due to the complexity of the result. Most thinning algorithms work iteratively. The edge pixels are examined against a set of criteria to decide whether they are essential skeleton pixels or not. A common disadvantage of many thinning algorithms is the deformation that is induced on the shape of the skeleton at regions where corners or boundary crossings are formed. Single pixel irregularities may yield unintuitive changes in an otherwise simple skeleton. Furthermore, the extraction of the skeleton does not often preserve the connectivity of the shape. Necking, tailing and spurious projection (line fuzz) are some common flows of many thinning methods [9].

The Force Based thinning algorithm [10] is based on the idea that the boundary should be used to locate the skeletal pixels by exerting a force towards the inner pixels. In that way, the skeleton of the shape lies at pixels where the forces imposed have opposite directions (Fig. 3(a)).

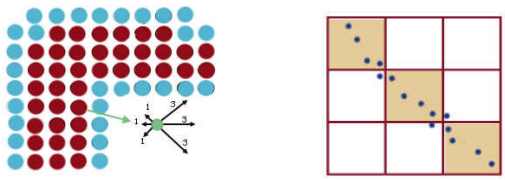

Fig. 3: (a) Force Based Thinning Strategy, (b) A virtual grid.

All thinning algorithms need as input a 2D array of pixels. In order to convert our unordered set of points to a 2D array of pixels we define a virtual grid (see Fig. 3(b)) of size G_x by G_y where G_x and G_y are the x and y resolution of the grid. Each grid cell G(i,j) corresponds to a certain area in the cross section specified by the two points of formula (3.1)

$$\left(x_{\min}+i\cdot\frac{x_{\max}-x_{\min}}{G_x},y_{\min}+j\cdot\frac{y_{\max}-y_{\min}}{G_y}\right)\quad\left(x_{\min}+(i+1)\cdot\frac{x_{\max}-x_{\min}}{G_x},y_{\min}+(j+1)\cdot\frac{y_{\max}-y_{\min}}{G_y}\right) \tag{3.1}$$

$$G\left(\left\lfloor\frac{p_x\cdot G_x}{x_{\max}-x_{\min}}\right\rfloor,\left\lfloor\frac{p_y\cdot G_y}{y_{\max}-y_{\min}}\right\rfloor\right) \tag{3.2}$$

where x_{min}, x_{max}, y_{min}, and y_{max} are the minimum and maximum coordinates in the original point set. Subsequently, for each point $P(p_x, p_y)$ we increase the intensity of the corresponding grid cell given by (3.2). Each grid cell will play the role of a pixel that is either *on* or *off*. We define a grid cell to be *on* if its total intensity is greater than the mean intensity of all cells in the grid. Fig. 3(b) shows an example grid and the pixels that are *on* and *off*.

Depending on grid resolution, the above process may produce a grid with a number of non connected pixels. For this kind of cases an anti-aliasing of the grid is performed to fill the gaps between disconnected pixels. Each neighbor cell will affect the anti-aliased intensity of the cell by its intensity multiplied by a coefficient. Eqn (3.3) computes the anti-aliased intensity of a cell:

$$AW_{i,j} = \sum_{m=i-1}^{i+1} \sum_{r=j-1}^{j+1} \frac{W_{m,r}}{F_{m,r}}, \qquad \Phi_{m,r} = \begin{cases} 16 & m \neq i \ \wedge \ r \neq j \\ 8 & m = i \ \oplus \ r = j \\ 4 & m = i \ \wedge \ r = j \end{cases} \tag{3.3}$$

Each "*on*" grid cell containing points from the original data set is mapped to the centroid of these points. In the case that the cell does not contain any points (characterized *on* by anti-aliasing) it may be mapped to the centroid of the points of the 8 neighbor cells. The result is then provided as input to the thinning process. We then use a graph traversing algorithm on the virtual grid [12] to order the point set.

3.3 Using Low-degree Bezier Patches for Approximation

The Bezier representation is one that is utilized most frequently in computer graphics and geometric modelling. Quadratic Bezier curves are often used by CAGD developers since they do not require complex computations as other higher degree curves do. However, in practice it is often desirable to approximate conic sections which cannot be represented in Bezier form. Conic sections such as parabolas hyperbolas and ellipses may be adequately represented by Rational Bezier curves.

3.3.1 Point Set Partitioning

First, the ordered point set should be partitioned in subsets of consecutive points that can be fitted by a single rational quadratic Bezier curve. To achieve this, for each poinr *Pi* we connect all neighbouring points with line segments and compute the average normal vector.

$$\overrightarrow{UR}_i = \frac{\vec{R}_i}{\left|\vec{R}_i\right|} = \frac{\vec{U}_i + \vec{U}_{i+1}}{\left|\vec{U}_i + \vec{U}_{i+1}\right|} \tag{3.4}$$

For a specific point P_i, the average normal vector UR_i is given Eqn (3.4) by averaging the normal vectors of the two adjacent line segments $P_{i-1}P_i$ and P_iP_{i+1} (Fig. 4(a)). In many cases where the data set contains a lot of noise, the average normal vector may be computed by averaging a larger number of neighbor line segments. We call this number smoothing neighbors.

Fig. 4: (a) Determination of normal vector for a point, (b) Inflection point detection.

Based on the fact that a single quadratic rational curve may approximate correctly a subset of points for which the induced curve exhibits a restricted concavity, we use the angle between the normal vectors of the end points to drive the partitioning of the point set. To determine the relative rotation of UR_{i+1} with respect to UR_i we use the cross product of the two vectors. An inflection point would be the point of the data set where there is a change in the sign of the z-coordinate of the cross product (Fig. 4(b)). In [12] the partitioning algorithm is presented in detail.

3.3.2 Middle Control Point

The partitioning process gives us a number of subsets of ordered points that may be approximated by a single rational quadratic Bezier curve. Therefore, the start and end points of each approximating curve are already known. Also, note that the end point of each partition coincides with the start point of the next partition. We will now present two methods for determining the middle control point of the fitting curve.

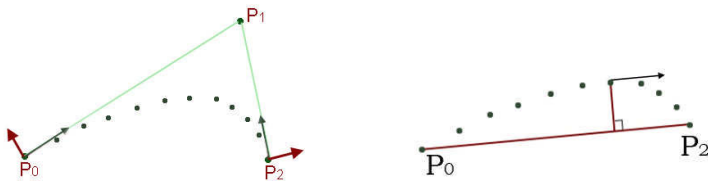

Fig. 5: Methods approximating the middle control point.

The first method makes use of the fact that the middle control point is the intersection of the tangent lines to the Bezier curve on the two end points. These tangent lines may be approximated by the lines that are defined by the normal vectors *Uri* on the end points. Fig. 5(a) illustrates this.

The second method involves the fact that at *t=0.5* every quadratic Bezier curve has maximum distance from the line passing through its end points. Also, at *t=0.5* the tangent to the curve is parallel to the line passing through its end points. Approximating the point on the curve at *t=0.5* (*R(0.5)*) with the point from the data set that has maximum distance from the line segment *P0 P2* may give us the location of the middle control point *P1* (Eqn (3.5)).

$$P_1 = 2 \cdot R(0.5) - \frac{(P_0 + P_2)}{2} \tag{3.5}$$

3.3.3 Optimal Rational Bezier Curve

The Bezier representation is one that is utilized most frequently in computer graphics and geometric modelling. Quadratic Bezier curves are often used by CAGD scientists since they do not require complex computations as other higher degree curves do. However, in practice it is often desirable to approximate conic sections which cannot be represented in Bezier form. Conic sections such as parabolas hyperbolas and ellipses may be adequately represented by Rational Bezier curves. Non rational Bezier curves are a special case of rational Bezier curves. For these reasons, we will focus on constructing Rational Quadratic Bezier curves. In curve theory, a rational quadratic Bezier curve is defined by Eqn (3.6).

$$P(t) = \frac{\sum_{k=0}^{2} w_k p_k B_k^2(t)}{\sum_{k=0}^{2} w_k B_k^2(t)}, 0 \le t \le 1 \tag{3.6}$$

A 2nd degree Bezier curve requires 3 control points *pk*: a start point *p0*, an end point *p2*, and a 3rd control point *p1* which is obtained by the methods we described in the previous section.

The *Bk* terms in the above formula represent the 2nd degree Bernstein polynomials, while the terms w_k are the associated with each control point weights. Setting all weights equal to one to the above formula represents an ordinary non rational Bezier curve. Increasing the weight of a control point causes the curve to move towards the associated control point.

The curve fitting process fits equations of approximating curves to the raw field data. Nevertheless, for a given set of data, the fitting curves of a given type are generally not unique. Thus, a curve with a minimal deviation from all data points is desired (Fig. 6(a)). For cases where a rational Bezier curve is approximated the best-fitting curve can be obtained by varying the control point weights.

A rational Bezier curve *P(t)* that best approximates the given set of 2D points *Q* on a specific cross section is the one that minimizes the sum of the distances of the points from the curve:

$$\sum_{i=1}^{n} \left\| Q_i - P(t_i) \right\|^2 = \min \tag{3.7}$$

$$\forall Q_i, i = 0 \ldots n, P'(t_i) \bullet (Q_i - P(t_i)) = 0 \tag{3.8}$$

Also note that each vector $\overline{Q_i P(t_i)}$ is normal to the tangent of the curve at t_i (Fig. 6(b)). This means that their inner product is zero. To minimize the sum of square distances, Eqn (3.7) will serve as the objective function while Eqn (3.8) will provide n constraints. Without loss of generality we can set $w_0=w_2=1$. For the minimization of the objective function subject to the n constraints we have used Lagrange multipliers and Interior point [16] optimization methods.

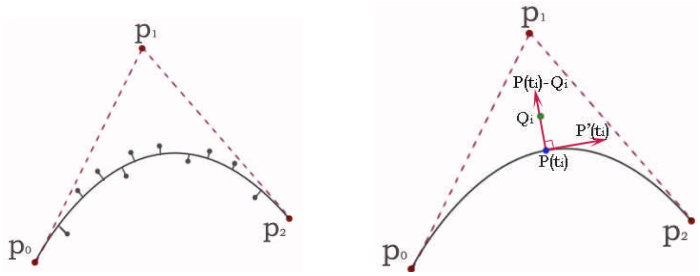

Fig. 6: (a) Point distances from curve P (b) Vector $Q_i P(t_i)$ perpendicular to tangent $P'(t_i)$.

Optimizing the objective function subject to these constraints may give the middle weight value of the rational Bezier curve that best fits the given set of points. Depending on the size of the data set that needs to be fitted, the optimization task could be a long and cumbersome effort. For this reason, we will perform an extra step of evaluating a starting value for the middle weight by making use the barycentric coordinates of each point with respect to the control triangle.

$$w_1 = \frac{\tau_1}{2\sqrt{\tau_0 \tau_2}} \tag{3.9}$$

Using Eqn (3.9) we may compute the value of the middle weight of the curve that passes through a certain point of the data set. τ_0, τ_1, τ_2 are the barycentric coordinates of Q_i with respect to the triangle formed by the three control points P_0, P_1, P_2 of each rational Bezier. Barycentric coordinates may be computed by Eqn (3.10).

$$\tau_0 = \frac{area(L_i, P_1, P_2)}{area(P_0, P_1, P_2)}, \quad \tau_1 = \frac{area(P_0, L_i, P_2)}{area(P_0, P_1, P_2)}, \quad \tau_2 = \frac{area(P_0, P_1, L_i)}{area(P_0, P_1, P_2)}$$
$$and \quad area(a,b,c) = \frac{1}{2}\begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{vmatrix} \tag{3.10}$$

Consequently, for each point Q_i in the data set, we may compute a value w_{1_i} for the middle weight of the curve. The value of the middle weight that will be selected is the one that minimizes the sum of square distances of the points in the data set from the curve. This value may be used as a starting value in the optimization process that was described above

4 CONSTRAINTS FOR EDITABILITY

A new generation of CAD systems has become available in which geometric constraints can be defined to determine properties of mechanical parts. The new design concept, often called constraint-based design or design by features offers users the capability of easily defining and modifying a design, but introduces the problem of solving complicated, not always well defined, constraint problems. In this chapter, we present the development of a user-friendly interactive system for imposing and solving geometric configurations inside cross-section (intra cross-section constraints) and among two or more cross-section (inter-cross-section) constraints. The system uses a powerful graph-constructive constraint solving method presented in [6], capable of efficiently analyzing certain classes of well-determined, over-determined and under-determined configurations. Minimal systems of geometric constraints that are not solvable by the core constructive method are detected and may either be handled by a numerical method and treated afterwards as rigid bodies, or edited by the user. A main

issue pertinent to geometric constraint solving is the solution selection problem. To this end, we have provided an interactive tool for navigating the constraint solver, to the intended solution. Consistent over-determined sub-configurations can be detected, interactively relaxed and solved appropriately. Under-determined subsystems are detected, isolated and subsequently presented to the user annotated with all possible constraint addition choices for interactive editing.

The objective of the entire method is to obtain an editable CAD model that would assist us in redesigning the original object. Editability in CAD is commonly achieved by using geometric constraints. When using the term constraint in CAD we usually refer to geometric dimensions and relations (lengths, angles, tangency, parallelism, perpendicularity, etc.) used to define accurately a particular solid geometry. Even though it is not necessary, object symmetry may provide additional auxiliary information in constraining an object. In this section we will categorize the geometric constraints and how they are adopted by our method to capture design intent and provide for redesign.

4.1 Intra-Cross Section Constraints

The first category of constraints is associated with the geometric and topological relationships among entities in a single cross sections which we will call intra – cross section constraints:
Point – line segment coincidence: special points (curve end points, curve control points, center of circle, etc) or line segments may coincide or be part of the same infinite line.
Tangency: an arc is tangent to a specific curve
Distance from a curve or point: an arc is located at some distance from a specific curve or point
Angle with a curve: an arc (its tangent) forms an angle with another curve or with a line segment at a specific point on the curve.
Parallel – Perpendicular line segments or tangents: a line segment is parallel or perpendicular with another line segment or tangent line.

4.2 Inter-Cross Section Constraints

The second category of constraints is associated with the geometric and topological relationships among the contours of different cross sections which we will call inter – cross section constraints:
Point co-linearity: a point from cross section C_A is on the same infinite line with a point from cross section C_B
Co planar line segments: a line segment from cross section C_A is on the same infinite plane with a point from cross section C_B
Equality or relation of distances: a specific distance in cross section C_A is equal or related with another distance from cross section C_B
Equality or relation of angles: an angle in cross section C_A is equal or related with an angle in cross section C_B
Curve translation: A curve in cross section C_A is translated by a specific distance and direction in cross section C_B. This constraint may fit cases of slanted or tori objects.
Curve scaling: A curve in cross section C_A is scaled by a certain scaling factor in cross section C_B. This constraint may fit cases of tapered objects.

4.3 Geometric Constraint Solving

We build a system of geometric constraints that captures user intent and at the same time guarantees solid model robustness and accuracy. Symmetry derived geometric constraints are considered to be *strict* with no tolerance allowed. User constraints fall under two categories: (i) *strict*, for which no tolerance is allowed and (ii) *flexible*, for which we wish to acquire the best approximation but we cannot guarantee their strict enforcement. For the purposes of usability we allow only for constraints that can be expressed as equation (e.g. distances, angles, relations of distances and angles, co-planarity, coincidence, tangency). Inequalities can also be handled but they tend to confuse the user with the multiplicity of solutions that they imply. Each flexible constraint has an associated weight which expresses its importance and is derived by two factors: explicitly by a user preference and implicitly by the rank in the user constraint enforcement. Finally the weighted sum of flexible constraint deviations properly normalized is used as the objective function to minimize and the strict

constraints are used as the set of constraints for this non-linear optimization problem. To solve this system we employ a local non-linear optimization algorithm from IpOpt [16]. The disadvantage of this method is that it may be trapped in local minima, which makes it depending heavily on the initial configuration. The user is thus advised to make incremental editing. Using global optimization methods or other constraint solving techniques is an interesting research problem [6]

5 RECONSTRUCTING SOLID PARTS

5.1 Point Re-sampling

The construction of the object's surface requires the generation of parts of the surface that lies in between two slices using triangulation. Triangulation may not be based on the thinned point set of each slice because its density would result in creating many small area triangles. To solve this issue, we must resample the point set to obtain a reduced set of points.

Point sampling is an important intermediate step for a variety of computer graphics applications. Specialized sampling strategies have been developed to satisfy the requirements of each problem. In this section, we present a sampling technique for 2D models. Our sampling domain is the set of points on a single cross section. Aim of the technique is to generate evenly spaced samples by subdividing the sampling domain into non overlapping parts.

Given a data set of points $Q=\{Q_i\}$ for which we have already determined the best fitted set of rational quadratic Bezier curves $P=\{P_k\}$, we suggest replacing the points Q with a reduced set of new points $R=\{R_j: R_j=P(t_j)\}$ that satisfy the curve equations.

In a previous chapter we fitted a rational Bezier curve on the points of each cross section. A Rational Bezier curve is usually defined over the interval *[0, 1]* but it may also be defined over any interval *[0, c]*. The part of the curve that corresponds to *[0, c]* may also be defined by a Bezier polygon. To subdivide the curve [13] to k equal length arcs we would first divide the interval *[0, 1]* into k subintervals of length $1/k$. The end points of each arc R_i are $P(t_{i-1})$ and $P(t_i)$ where $t_i= i/k$ and $i=0..k$. The length of each chord $\|P(t_i)\ P(t_{i+1})\|$ converges to the length of the arc R_i between t_i and t_{i+1} when k is a rather large value (Eqn (5.1)).

$$\Lambda = \sum_{i=0}^{k} \left\| P(t_i) - P(t_{i+1}) \right\| \tag{5.1}$$

Considering that the size of the sample set S of points is μ:

$$\forall s_i \in S, \quad i = 0...(\mu - 1), \qquad \left\| s_i s_{i+1} \right\| = \Lambda / \mu \tag{5.2}$$

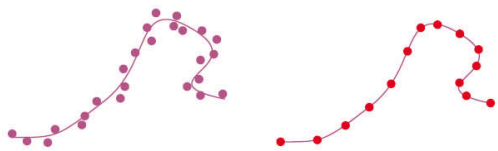

Fig. 7: Sample points (in red).

The last relation ensures that all points in the sample set S are evenly spaced by a distance of Λ/μ. All other points that do not satisfy the above relation are discarded and will not be used in the surface reconstruction process (Fig. 7).

5.2 Similar Adjacent Cross-Sectional Feature

The main design paradigm of CAD systems nowadays is feature-based design. Feature – based systems contain a vocabulary of design elements as long as object operations that are used to create the intended design. By performing operations such as extrusions, protrusions and cuts on the design elements (cylinders, cones, parallelepipeds, pyramids etc) we may generate the desirable design. Our

feature based CAD model provides modeling primitives that may be enforced low-level constraints, reducing the number of variables necessary to represent an object. Constraint based techniques apply high-level constraints over these features enforcing the hypothesized design intent.

In this section we will investigate ways for converting 3D point cloud to a set of features describing exactly the original object's geometry and satisfying all imposed constraints. Based on the fact that our method generates a set of planar consecutive curves for each cross section, we will be considering features that are based on a planar profile swept into a 3D shape by an extrusion operation. Consequently, our final CAD model will consist of a set of connected features. This makes our final CAD model easily modifiable since we only need to deal with modifying the geometry of the features.

As shown in Fig. 8, sweeping a planar profile creates a tubular surface that its bottom base is the planar profile and its top base is the same planar profile translated. Let two planar profiles P_1 and P_2 consisting of a set of quadratic rational Bezier curves. P_1 is said to be similar to P_2 if and only if all curves in P_1 are congruent to all curves in P_2 up to the same affine transformation. In other words, profile congruence requires curve congruence. Bezier curve congruence property implies control triangle congruence and middle weight equality. Therefore, two planar profiles are invariant if and only if all respective control triangles are congruent and all respective weights equal.

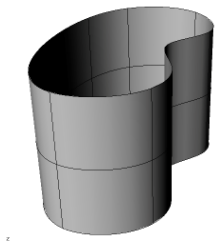

Fig. 8: Sweeping of a planar profile.

As it is defined in Euclidean geometry, triangles are congruent when all corresponding sides and interior angles are equal. These triangles will have the same shape and size. However, they can be in a different location, rotated or flipped over. Consequently, two triangles R and R' are congruent even if *R* is a mirror of *R'*. In contrast, in our method we are interested in triangles that may not be mirror images of each other because they generate different curves that cannot be interpolated.

Definition 1: We define as topologically congruent in 2D two polygons that are the same up to rotation and translation

Therefore, two profiles P_1 and P_2 are said to be congruent if and only if all control triangles are topologically congruent.

The extrusion direction vector may be either perpendicular or it may form any angle with the profile plane. As long as the starting and ending cross sections are invariant, computing the center of mass of both cross sections may derive the extrusion direction vector. This is true even for cases where the second polygon is scaled or rotated.

Detecting similarity between two or more polygons is performed based on two key ideas:

- normalizing a shape about its diameter and
- the notion of the ε-envelope.

Normalizing about the diameter. In order to detect whether two or more polygons are similar some kind of "normalization" is applied so that the matching is translation-, rotation-, and scaling-independent. In previous work researchers would normalize each shape about each of its edges: they translate, rotate, and scale the shape so that the edge is positioned at *((0, 0), (1, 0))*. Although this approach gives good results in many cases, it would fail to detect similarity between slightly distorted shapes.

In our method, instead of normalizing about the edges, we normalize about the diameter of the shape, i.e., by translating, rotating, and scaling so that the pair of shape vertices that are farthest apart are positioned at *(0, 0)* and *(1, 0)*. This ensures better results, because the diameter is less susceptible to local distortion which is very common in shapes extracted using thinning and other point-based techniques.

The ε-envelope [7]. Polygon matching works by considering a "fattened" version of the one polygon which is computed by taking lines parallel to the query shape edges at some distance on either side; we call this fattened shape the ε-envelope. The good matches are expected to fall inside or at least have most of their vertices inside the ε-envelope even for small ε. Therefore, if we start by using a small initial value of ε and keep increasing it, we expect to collect the good matches after a few iterations of this procedure.

The ε-envelope can be seen as a collection of trapezoids of height 2ε, one for each edge of the query shape. (For simplicity, we assume that ε is such that no two trapezoids are overlapping; the method can be extended to handle overlapping trapezoids.)

The center of mass of a planar profile may be approximated by the center of mass of its convex hull polygon. A better approximation could be the minimal control polygon's center of mass. To determine the minimal control polygon of a planar profile all quadratic Bezier middle control points are used as polygon vertices. The minimal control polygon must include all Bezier curves. While a Bezier curve is always inside its control triangle, the minimal control polygon may not always include a curve. The polygon in Fig. 9(a) depicts this case.

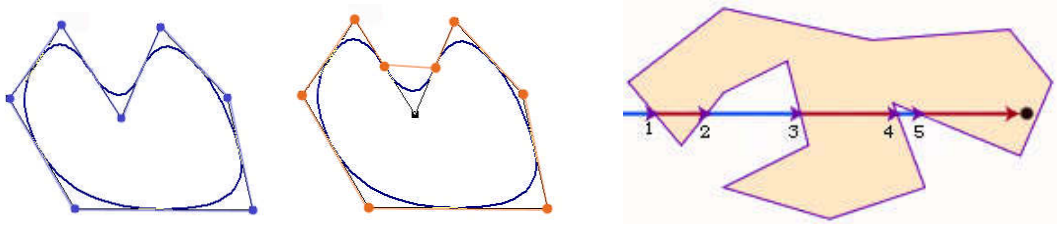

Fig. 9: (a,b) Derivation of minimal Control Polygon, (c) Ray casting algorithm.

In the case where a curve is excluded from the minimal control polygon we edit the list of polygon vertices by replacing the specific middle control point with the respective Bezier curve's end points. The polygon in Fig. 9(b) illustrates this procedure.

If one point of the curve is inside the polygon then the entire curve is inside also. Therefore, we only need to determine if a single point on the curve is inside the polygon. A ray casting algorithm may be used to determine whether a specific point is included in the control polygon. The algorithm is based on a simple observation that if a point moves along a ray from infinity to the probe point and if it crosses the boundary of a polygon, possibly several times, then it alternately goes from the outside to inside, then from the inside to the outside, etc. As a result, after every two "border crossings" the moving point goes outside. Therefore, the number of intersections is an even number if the point is outside the polygon, and it is odd if it is inside (Fig. 9(c)).

Both the convex hull and the minimal control polygon of a profile are n-polygons. Computing the center of mass of an n-polygon $\{A_1, A_2, ..., A_n\}$ is rather straightforward using Eqn (5.3).

$$CM = \frac{1}{n} \cdot \sum_{i=2}^{n} \overrightarrow{A_1 A_i} \tag{5.3}$$

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. The *orthogonal extrusion* of R is defined to be a solid obtained by sweeping profile *R* in a direction perpendicular to *C* up to the parallel cross section *C'* resulting to one or more tubular surfaces (Fig. 10(a)).

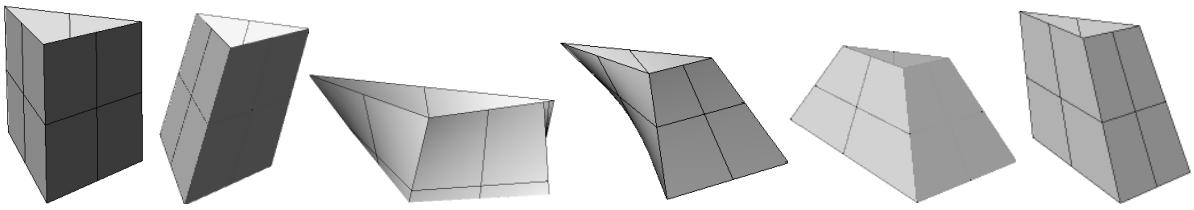

Fig. 10: (a) Orthogonal Extrusion, (b) Oblique Extrusion, (c) Orthogonal Rotated Sweeping, (d) Oblique Rotated Sweeping, (e) Orthogonal Linear Scaled Skinning, (f) Oblique Linear Scaled Skinning.

Let R and R' be two congruent closed profiles on the parallel cross sections C and C' respectively. Also let V be a vector on the line that connects the centers of mass of the profiles R and R'. The *oblique extrusion* of R is defined to be a solid obtained by sweeping profile R in a direction specified by vector V up to the parallel cross section R' resulting to one or more oblique tubular surfaces (Fig. 10(b)).

Let R and R' be two congruent closed profiles on the parallel cross sections C and C' respectively. Also let profile R' be determined by a θ angle rotation of profile R with the center of rotation being the center of rotation. Let's also denote as d the distance between the centers of mass of R and R'. The *orthogonal rotated sweeping* of R is defined to be a solid obtained by sweeping profile R in a direction perpendicular to C and the same time rotating the profile R with a rate of rotation θ / d up to the parallel cross section R' resulting to one or more rotated tubular surfaces (Fig. 10(c)).

Let R and R' be two congruent closed profiles on the parallel cross sections C and C' respectively. Also let profile R' be determined by a θ angle rotation of profile R with the center of mass being some point P. This rotation is equivalent to a θ angle rotation of profile R with the center of mass being center of rotation, followed by a translation in the same plane. Therefore, we may denote as V the vector on the line that connects the centers of mass of the profiles R and R'. Let's also denote as d the distance between the centers of mass of R and R'. The *oblique rotated sweeping* of R is defined to be a solid obtained by sweeping profile R in a direction specified by vector V and the same time rotating the profile R with a rate of rotation ϑ/d up to the parallel cross section R' resulting to one or more oblique rotated tubular surfaces (Fig. 10(d)).

Let R and R' be two congruent closed profiles on the parallel cross sections C and C' respectively. Also, let μ be the linear scaling factor of the two profiles and d the distance between the centers of mass of R and R'. The *orthogonal linear scaled skinning* of R is defined to be a solid obtained by skinning profile R in a direction perpendicular to C and the same time scaling the profile with a scale rate μ / d , up to the parallel cross section C' resulting to one or more frustrum surfaces (Fig. 10(e)).

Let R and R' be two congruent closed profiles on the parallel cross sections C and C' respectively. Also, let μ be the linear scaling factor of the two profiles, d the distance between the centers of mass of R and R', and V the vector on the line that connects the centers of mass of the profiles R and R'. The oblique linear scaled skinning of R is defined to be a solid obtained by skinning profile R in a direction specified by vector V and the same time scaling the profile with a scale rate μ/d, up to the parallel cross section C' resulting to one or more frustrum surfaces (Fig. 10(f)).

5.3 Non-similar Adjacent Cross-sectional Features

So far, we investigated ways for reconstruction by extrusion of solid parts that are between similar adjacent cross sections. In this section we will investigate ways to reconstruct solid parts that are between non-similar adjacent cross sections.

5.3.1 Curve-based Morphing and Interpolation

When we dealt with similar adjacent cross sections the same sweeping strategy was applied to the entire profile. For cases with non-similar adjacent profiles, we cannot apply the same sweeping strategy to the entire profile. Curve-based morphing is an advanced type of sweep where each curve is being applied a different sweeping strategy.

Two adjacent cross sections C_1 and C_2 are non-similar when there is at least one curve in C_1 that is non-similar (not congruent) to its respective curve in C_2. Fig. 11(a) depicts a case with two non-similar adjacent profiles.

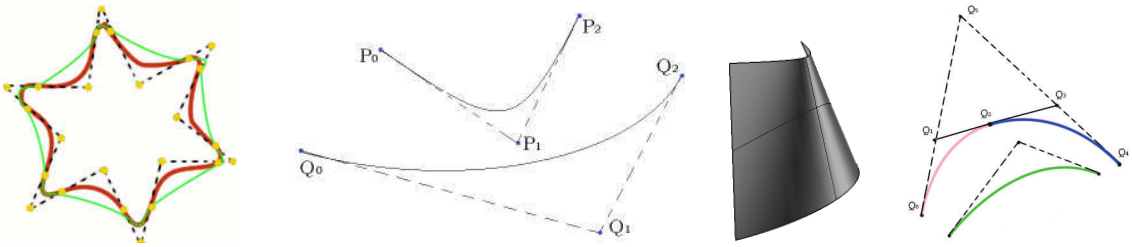

Fig. 11: (a) Non-Similar profiles, (b,c) Control triangle linear morphing, (d) Curve mapping.

Curve-based morphing breaks down the profile sweeping problem to a number of curve sweeping problems. Since a rational quadratic Bezier curve is fully specified by its control triangle and its middle weight, we only need to sweep the source control triangle to the destination control triangle and gradually change the value of the middle weight from the source to the destination value (Eqn (5.4)). Fig. 11(b,c) illustrates this type of sweep. Morphing curve P to Q is equivalent in morphing between their control triangles. The middle weight undergoes a gradual transition from value W_P to W_Q. Let R be a triangle in some ith state of morphing P to Q. Each control point R_i should be on the line segment P_iQ_i such that

$$\begin{aligned} R_o(t) &= P_0 + t(Q_0 - P_0) \\ R_1(t) &= P_1 + t(Q_1 - P_1) \\ R_2(t) &= P_2 + t(Q_2 - P_2) \\ W_R(t) &= W_P + t(W_Q - W_p) \end{aligned} \tag{5.4}$$

Morphing of two dimensional shapes can be divided into two sub problems that have to be solved. These problems deal with vertex correspondence and vertex path. Common morphing literature is usually concerned with the vertex path problem. However both problems are equally important. 2D morphing techniques pay special attention to the goal that all intermediate shapes are free of self-intersections because apart from some fancy special cases, a morphing sequence that contains self-intersections is considered to be unnatural transition from source to target.

The vertex correspondence problem deals with the creation of a bijective mapping between the vertices contained in the source S, and target T cross sections in a way that for each vertex in S there is exactly one vertex in T that is mapped to and vice versa. Obviously, such a mapping is not always possible. In the case that two cross sections S, T do not have the same number of curves an extra processing of the set of curves is required. This extra processing involves curve splitting or curve concatenation resulting in two cross sections with the same number of curves. Therefore, when dealing with one curve from each slice, the vertex correspondence problem between the two sets of control points is solved by least square distance minimization. Since the control point sets are ordered, vertex correspondence must maintain this order. Therefore, if S_i is mapped with T_j then S_{i+m} should be mapped with T_{j+m}.

The vertex path problem deals with the selection of a path that a point will travel from the source cross section to its mapped point in the target cross section.

A cross sectional profile consists of a set of rational quadratic Bezier curves. For cases where two non-similar adjacent profiles consist of a different number of curves, some curves might need to be concatenated or split, if possible, so the necessary matching of the two profiles can take place. Fig. 11(d) shows two adjacent non-similar profiles with different number of curves. To perform curve-based morphing on these profiles either profile Q needs to be concatenated to one curve or profile P needs to be split to two curves.

5.3.2 Polygon-based Morphing and Interpolation

The task of surface reconstruction deals with the creation of a ribbon between two adjacent cross sections. This may be accomplished by performing triangulation between the sampled sets of vertices that belong to a pair of adjacent cross sections. In most real cases the material of interest lies in the region that separates the adjacent contours.

A rather simple solution that forces a connection of each vertex of a section with some vertices of the adjacent sections was proposed by the literature in the past. However, as the distance between two cross sections may vary, the chance of missing important information of the places where ramifications occur is rather high. As a result, the reconstructed object does not have the correct shape. To overcome this problem, we propose a method that automatically creates intermediate sections.

The projection of the region, which separates the adjacent cross sections, on an intermediate parallel plane is the region that is not common to both contours. We will denote a cross section as a binary image where the two values represent the background and the object. This intermediate plane projection may be expressed as an exclusive *OR (XOR)* operation on the binary images of the two contours [4]. In the case where the contours of the adjacent sections intercept, it is required to include the pixels of the contour boundary where the interception occurs.

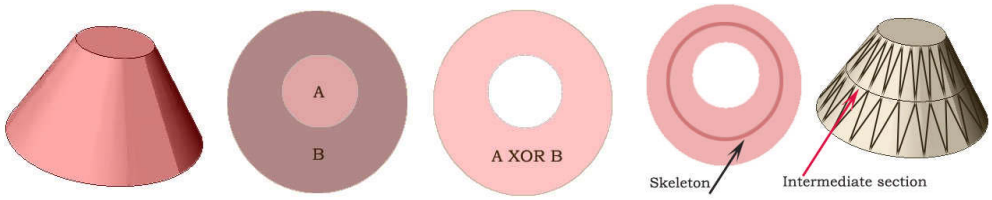

Fig. 12: XOR operation on sections A and B.

The result of the XOR operation is also a binary image whose boundary is formed by the contours of the contiguous sections. Fig. 12 shows that the outer border of the binary image is formed by the second contour while the inner border is formed by the first contour. Fig. 13 shows two slices that their boundaries intersect. The XOR operation result is shown in pink while the result of the region thinning is shown by the curves in the pink regions.

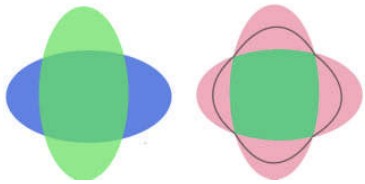

Fig. 13: Inner and outer boundaries intersecting. XOR region in pink.

In many cases we may see portions of the binary image to have both inner and outer borders formed by the same contour. This is an indication that in the particular portion of the material of interest there is a ramification. For these cases the skeleton of that portion of the binary image may be used to represent the place where the ramification occurs at an intermediate height of the analyzed sections.

Applying a thinning algorithm on the binary image we may obtain its skeleton (Fig. 12, Fig. 13). Using the shortest diagonal algorithm [5] we are able to create two ribbons (one with each slice).

6 EXPERIMENTAL EVALUATION

We have implemented and tested a prototype of the proposed method using the

- MS Visual C++ programming language,
- the OpenGL graphics libraries,
- the IpOpt optimization software [16]
- the ACIS solid modeling libraries by Spatial Corporation [15].

The entire system was built using the object oriented design framework. We have used extensive testing with several cloud point sets. For the internal representation of the contours (sequences of G^s rational Bezier patches) we have used NURBS.

To demonstrate how this method works we have used a 3D point cloud of a screwdriver object containing *27500* points which was then sliced to equidistant parallel cross sections (Fig. 14(a)).

Fig. 14: (a) Screwdriver point cloud slicing, (b,c) Slice thinning, (d) Concavity change detection.

Fig. 14(b) shows part from a cross section of the screwdriver's handle containing *437* points. Thinning and quantization of this cross section results in a point set with *323* points that form a *1*-point-thick curve boundary.

While partitioning the thin slice point set, the algorithm filters out all noisy points (Fig. 14(c)). The final result of the concavity detection process is a set of contours that have the same concavity direction and may be approximated by a low degree rational Bezier curve.

Fig. 15(a) shows the computation of the middle control point of all quadratic rational Bezier curves that we are going to construct. This is actually a slice from the screwdriver's handle.

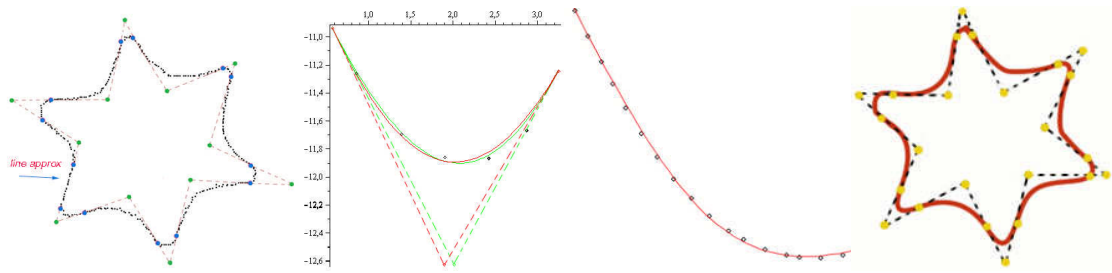

Fig. 15: (a) Deriving Control Points, (b) Control point method selection, (c) Minimized point distances from the curves by middle weight adjustment, (d) Fitted Rational Bezier Curves.

Depending on the point topology either method may be used to determine the middle control point. The method selection criterion should be the minimum squared distance of the point set from the fitted curve. Fig. 15(b) shows a data set with seven points and the curves that are fitted on them by both methods. The minimum squared distance for the first method (green color) is *0.004091* and *0.007578* for the second method (red color). It is clear that for the particular data set the first method produces a better fitted curve

Following, for each partition of points, the computation of the middle control point weight is performed by minimizing the sum of squared distances of all points from the fitted curve (Fig. 15b,c) using the IpOpt libraries. Fig. 15(d) shows the set of curves built by the algorithm.

The following diagrams evaluate the effectiveness of the fitting method. We compare *6* different point sets from different slices. Each point set has a different number of points to be fitted (curve1

24, curve2 *31*, curve3 *43*, curve4 *17*, curve5 *10*, curve6 *26*). The first diagram (Fig. 16(a)) shows the relation between the number of points that are to be fitted and the time that the fitting method needed to complete the task. Thus we observe that the time needed is linear on the number of points that are fitted. These experiments were conducted on an average computer system.

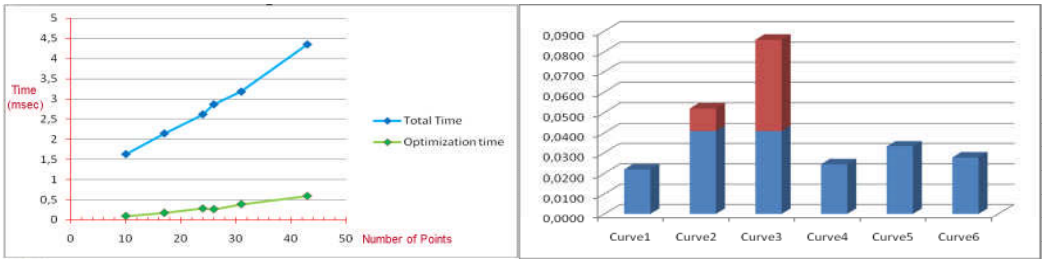

Fig. 16: (a) Time for fitting, (b) Average Error per point.

The diagram in Fig. 16(b) evaluates the effectiveness of the fitting method. It shows the average error of a curve point from the fitted curve. We notice that curve3 error is a lot above the average error. There are two factors that are responsible for this issue:

- The start point normal vector forms an angle greater than $\pi/2$ with the end point normal vector.
- The smoothing tolerance of the partition process was not used to split the set of points into two partitions.

Despite the above average error, the normalized error values are fairly low even though the original point cloud had a lot of noise.

We will now constrain the handle of the screwdriver by user defined constraints. We will consider the editing area to be the part of the screwdriver's handle between cross section *A* and *B* (Fig. 17(c)). Intra Cross Section Constraints will be defined on a user selected cross section *M* where maximum cavity depth occurs.

Fig. 17(a) shows the control polygon that results from the curve fitting process. We define a normal hexagon which is centered on the center of mass of the cross section. Each side of the hexagon is the base of an isosceles triangle (Fig. 17(b)). All peaks *(S1, S2,..,S6)* are equidistant from the center of mass *H=14*. All hexagon sides are equal *d=7* while all cavity peaks (*E1, E2, …,E6)* are equidistant from the center of mass *d=7*. The values *H* and *d* are completely independent of each other. In other words, the value of *H* controls the diameter of the handle while the value of *d* controls the radius of the normal hexagon, the height of the isosceles triangles and therefore, the depth of the handle cavities.

For inter cross section constraints we define that all slices between cross sections *A* and *B* must have their centers of mass on the same *z*-axis. The diameter of the handle changes linearly. Therefore, all corresponding slice peaks belong to the same line. Consequently the value *H* may be easily evaluated by the line equation. Furthermore, we determined a quadratic Bezier curve that best fits all corresponding cavity peaks among different slices. Therefore, the value *d* (radius of normal hexagon) may be easily evaluated using the Bezier curve (Fig. 17(c)).

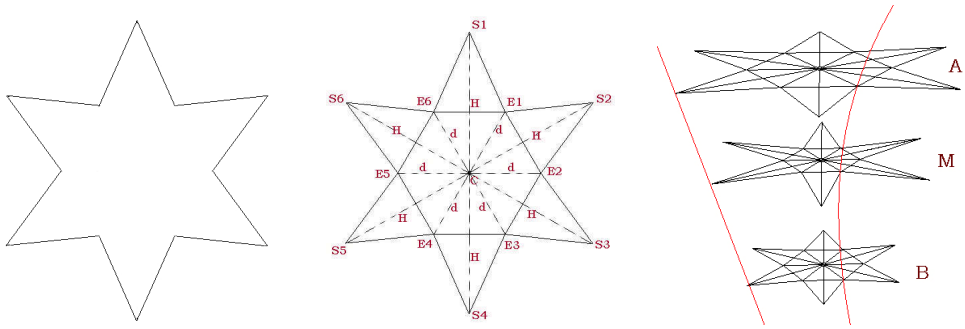

Fig. 17: (a) Control polygon, (b) Intra Constraints, (c) Inter Constraints.

The resampling step computes the length of each rational Bezier segment in the slice. The approximate length of the entire contour is Λ=*95.76*. Setting μ=*60*, we obtain the distance Λ/μ of each point from its neighbors to be around *1.596*. Fig. 18(a) shows the set of representative points that were selected.

Fig. 18(b) shows the intermediate slice generation using the *XOR* operation. Fig. 18(c) shows the result of the intermediate slice generation by curve morphing. The reconstructed part of the object between the two adjacent cross sections is shown in Fig. 18(d).

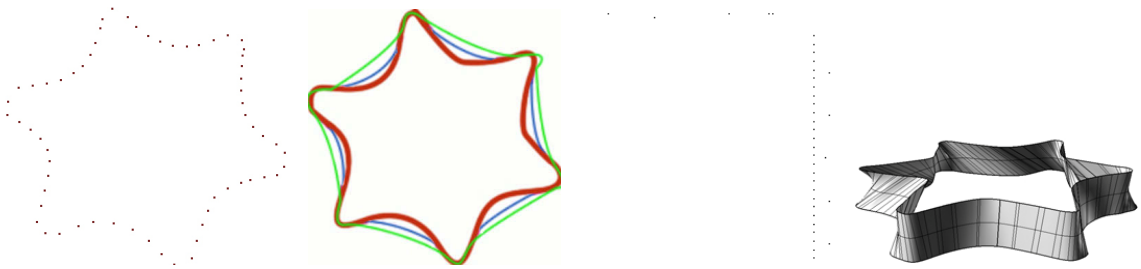

Fig. 18: (a) Resampling result, (b) Auto slice generation by *XOR*, (c) Auto slice generation by curve morphing, (d) Reconstruction of part using intermediate slice generation.

Fig. 19(a) shows the fully reconstructed object (exact copy). Fig. 19(b) shows the modified reconstructed object with the cylindrical part of the steel shaft longer. To accomplish this modification we increased the distance between the cross sections on the steel shaft by a factor of *1.4*. Fig. 19(c) shows the modified reconstructed object with the lower part of its handle wider. The modification actually made was an increase of the diameter of the lower handle by a factor of *1.3*.

Fig. 20 illustrates a modification in the cavities of the screwdriver handle. The designer decreased the value of *d* by *25%* in the revision slice. This decrease propagated to all slices in the editing area automatically by the reevaluation of the Bezier curve that constrains the value of d in the editing area. As a result, the depth of all cavities in the screwdriver handle were increased accordingly. Notice that the other constraint value *H* remained constant (line equation did not change) and therefore, the diameter of the handle does not change. Fig. 20(a,b) illustrate the original and the modified slice. The difference in the cavity depths is clearly seen in Fig. 20(c,d). The original and the modified object may be seen in Fig. 21(a,b).

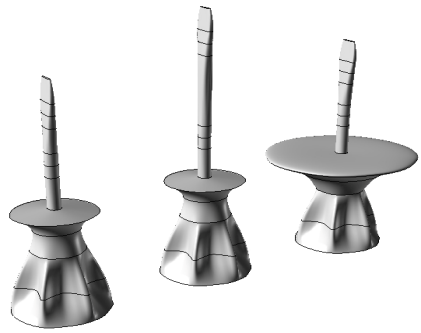

Fig. 19: (a) Result of Reconstruction, (b,c) Result of Editing after Reconstruction.

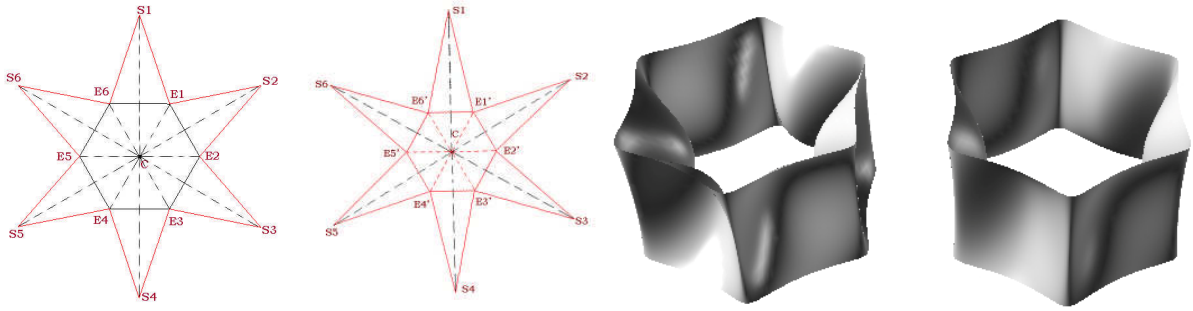

Fig. 20: (a) Original revision slice constraints, (b) Modified revision slice constraints, (c) Original reconstructed object part, (d) Modified object part.

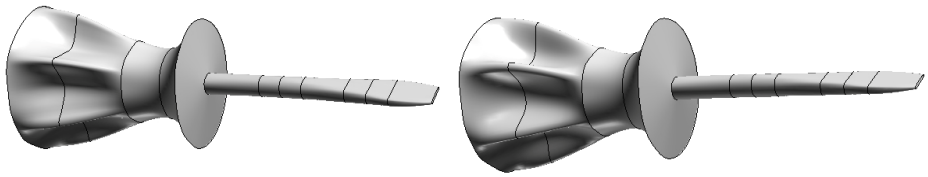

Fig. 21: (a) original screwdriver, (b) modified screwdriver (deeper cavities).

7 CONCLUSIONS

We have presented an effective and efficient method to build a 3D CAD model from a given point cloud representing the surface of an object.
Our approach to re-engineering uses point cloud slices along a principal axis. These slices are then processed to obtain a thinned, ordered set of planar points. Subsequently, this set is used to obtain a fully functional cross section represented by a number of constrained rational Bezier curves.

We have introduced inter-cross-section and intra-cross-section geometric constraints for supporting editability.

3D contour-based reconstruction has been extensively studied, and we have employed and tested several slice morphing and slice insertion techniques for covering between non similar adjacent cross sections. Model editability is also supported at this level by defining parameters for the 3D reconstruction of user defined slice groups.

We have performed a preliminary evaluation of the usability of our method with very good results even for users with no former CAD software experience. Our method provides the tools for robust and accurate editing of the produced CAD model prior to remanufacturing.

Automated detection of an optimal slicing direction is an addition that can save users a lot of effort. Finally, the effectiveness of the reconstruction process could be improved for complicated objects by first decomposing the object by employing sophisticated decomposition methods such as the one presented in [8].

REFERENCES

[1] Au, C. K.; Yuen, M. M. F.: Feature-Based Reverse Engineering of Mannequin for Garment Design, Computer-Aided Design, 31, 1999, 751-759.

[2] Benko, P.; Martin, R. R.; Varady, T.: Algorithms for reverse engineering boundary representation models, Computer-Aided Design, 33(11), 2001, 839-851.

[3] Benko, P.; Kos, G.; Varady, T.; Andor, L.; Martin, R. R.: Constrained fitting in reverse engineering, Computer-Aided Design, 19(3), 2002, 173-205.

[4] Christiansen, H.; Sederberg, T.: Conversion of complex contour line definitions into polygonal element mosaics, Computer Graphics, 13, 1978, 187–192.

[5] Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Academic Press, Boston, 1997.

[6] Fudos, I.; Hoffmann, C. M.: A Graph-constructive Method to Solving systems of Geometric Constraints, ACM Transactions of Graphics, 16(2), 179-216.

[7] Fudos, I.; Palios, L.: An Efficient Shaped-based Approach to Image Retrieval, Discrete and Applied Mathematics, 2000.

[8] Lien, J. M.; Keyser, J.; Amato, N. M.: Simultaneous Shape Decomposition and Skeletonization, ACM Solid and Physical Modeling Symp., 2006, 219-228.

[9] Parker, J. R.; Jennings, C.: Defining the Digital Skeleton, SPIE Vision Geometry, 1832, 1992.

[10] Parker, J. R.; Jennings, C.; Molaro, D.: A Force Based Thinning Strategy with Sub-Pixel Precision, Vision Interface, AB, 1994, 18-20.

[11] Protopsaltou, A.; Fudos, I.: Creating Editable 3D CAD Models from Point cloud slices, GraVisMa, 2009, 118-125.

[12] Protopsaltis, A.: Reconstructing 3D CAD Models based on geometrically constrained cross sections, Ph.D. Thesis, University of Ioannina, 2009.

[13] Randrianarivony, M.: Arc Length of Rational Bezier Curves and Use for CAD Reparametrization, World Academy of Science Engineering Technology, 34, 2008.

[14] Stamati, V.: Reconstructing feature-based CAD models based on point cloud morphology, Ph.D. Thesis, University of Ioannina, 2008.

[15] The 3D ACIS Modeler, http://www.spatial.com, ACIS Corporation

[16] The Ipopt - Interior Point Optimizer Project, https://projects.coin-or.org/Ipopt

[17] Thompson, W. B.; Germain, H. D. S.; Henderson, T. C.; Owen, J. C.: Constructing High-Precision Geometric Models from Sensed Position Data, ARPA Image Understanding Workshop, 1996.

[18] Thompson, W. B.; Owen, J. C.; Germain, H. D. S.; Stark, S. R.; Henderson, T. C.: Feature-Based Reverse Engineering of Mechanical Parts, IEEE Transactions on Robotics and Automation, 15(1), 1999, 57-66.

[19] Varady, T.; Facello, M.; Terek, Z.: Automatic Extraction of Surface Structures in Digital Shape Reconstruction, Computer Aided Design, 39, 2007, 379-388.