Computer-Aided Design and Applications

http://www.cadanda.com

On Reconstructing 3D Feature Boundaries

Vasiliki Stamati[1] and Ioannis Fudos[2]

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

ABSTRACT

Raw point data collected by 3D scanning techniques are usually rendered by building an interpolating robust polygonal mesh. This approach is accurate and fast but provides no means for large scale subsequent modifications. Only local interactive or non-interactive tools are provided that are usually targeted to correcting small imperfections and eliminating noise effects. CAD applications require robust and editable CAD models to support processes such as reproduction, design modification and redesign. In this paper we present a curve approximation method used in a feature-based approach to building feature-based CAD models from 3D point clouds. This approach is based on discovering features on the point cloud by detecting local changes in the morphology of the point cloud. This results in a number of regions that represent object features. The boundaries of the features are approximated by a collection of piecewise cubic rational Bezier curves that best fit the detected border point cloud and are G1 continuous.

Keywords: point cloud approximation, cubic rational Bezier curves, feature borders, curve fitting.
DOI: 10.3722/cadaps.2008.316-324

1. INTRODUCTION

The problem of constructing curves that approximate point cloud data has been approached using different type and degrees of curves depending on the nature of the application. For CAD data representing objects of freeform design an approach that yields reasonable results is the use of piecewise rational curves of low degree, because the weight factors allow the shape of the curve to be better adjusted to the point data by determining the effect the corresponding control point has on the curve. The work is this paper is focused on re-engineering point clouds of freeform objects to construct feature-based CAD models that can be used mainly in redesigning and customization of the reengineered object. We extract sets of points from the point cloud that correspond to boundaries of features that need to be approximated by piecewise curves. Our method is inspired by a rational B-spline curve approach proposed in [16] which we adapt by applying constraints on rational Bezier curves.

Many methods have been suggested for fitting curves to point data, depending on the parameters and the characteristics of the application. The most straightforward approach to curve fitting is to fix some curve parameters such as knots and weights (for B-splines), and then use a least squares fitting approach to compute the control points of the curve [8]. [4, 15] provide a thorough survey on curve fitting. We will briefly summarize the state of the art in reference to curve fitting for reverse engineering.

[16] presents a method for fitting rational B-spline curves to point data for reverse engineering applications. The authors suggest a linear least squares optimization process where the control points and the weights of a rational Bspline curve of degree *n* are iteratively refined until convergence is achieved. In [13] the authors focus on constructing curves and surfaces from point clouds obtained by 3D scanning techniques. The initial point cloud is triangulated and a method is introduced for selecting appropriate points from the data set based on the triangulation. Then a curve

refinement process is performed which fits the B-spline curve to the points under linear constraints related to the endpoints and tangent vectors. [9] proposes a method based on squared distance minimization, which starts with an initial Bspline curve that is iteratively fitted to the target curve. In [15] the authors suggest a method for computing a planar Bspline curve to fit an unorganized, possibly noisy, point cloud using an iterative squared distance minimization process based on [9] which converges from an initial curve to the desired target shape using a squared distance error objective quantity. [7] suggests a modified moving least squares method for thinning out a point cloud and approximating it with a smooth curve. In [2], the authors present an approximation method which constructs smooth curves from noisy unordered data.

[14] discusses the pros and cons of constrained and unconstrained fitting in reverse engineering applications. As noted by the authors, unconstrained fitting results in root-mean-square distance minimization of the points, however fairness of the curve is not taken into consideration, which is important in re-engineering applications. Also, with noisy point cloud data, unconstrained curve fitting may lead to unwanted results. The authors provide a general curve fitting approach which, depending on what type of constraints are integrated into the process, achieves different fitting results. Constraints to trim the search space of the curve fitting process are reported in [3], where the authors use constraints to ensure that the curves lie on smooth manifolds.

While the method proposed in [16] produces curves for the purpose of reconstructing objects of freeform design, its drawback is that convergence is somewhat slow, making it costly to use for re-engineering objects consisting of many complex features. The curve fitting methods proposed in [9] and [15] are very accurate and stable, however their performance depends on the initial curve specified interactively by the user. [7] constructs a bspline curve to approximate a thinned out point cloud, without providing an upper bound for the approximation error. In [13], even though constraints are imposed on the curve fitting process, the curves obtained are not sufficient for representing complex shapes often encountered in freeform objects. The work in [2] is based on the assumption that no connectivity information is available, however, in our case, the point cloud has been preprocessed and therefore we have acquired topological information regarding the point cloud.

Software tools have been developed that can be used for curve reconstruction in reverse engineering applications. An example of such software is SISL [11], a NURBS software library that provides functionality for building applications needing freeform geometry. SISL uses a global optimization approach to curve fitting that returns very good results. We have implemented a global optimization for our application problem and observed that is usually converging quite slowly, and depending on the initial condition we may encounter accuracy issues and convergence to local minima. Our application problem deals with a multitude of features resulting from processing the 3D point cloud. The boundary regions of the features should be approximated by piecewise curves for purposes of reverse engineering and therefore requires an effective and efficient method.

In this paper we present a fast curve approximation method that approximates raw data with cubic rational Bezier curves. Our approach combines least squares approximation with continuity constraints to ensure G^1 continuity between neighboring curves. We use the weights of the curve to adjust its shape and parametric structure so as to construct curves that pass as closely as possible between the data sets and join smoothly.

In Section 2 we provide a brief overview of our feature-based approach to reconstructing CAD models from point clouds acquired by 3D laser scanning techniques. From this process we obtain the point sets for which we create approximated curves. In Section 3 we describe our curve approximation method and in Section 4 we provide an application example. Finally in Section 5 offers conclusions.

2. PRELIMINARIES

In [12] we present a feature-based approach to re-engineering freeform objects from point clouds obtained by 3D laser scanners. This approach is based on discovering features on the point cloud by detecting local changes in the morphology of the point cloud. We employ region growing, detection of rapid variation of the surface normal and the concavity intensity, i.e. the distance from the convex hull. This results in a number of regions that represent object features.

More specifically, morphological features in the point cloud are detected by using a characteristic we define as the "concavity intensity" of a point which represents the smallest distance of a point from its convex hull. This characteristic

detects concave features in the cloud. Fig. 1(a) presents the point cloud of a screwdriver [1] and its convex hull facets, whereas fig. 1(b) displays a greyscale mapping of the concavity intensity value of each point (white color corresponds to the maximum distance whereas black corresponds to points located on the convex hull). Features are also detected by rapid variations of the surface normal. These two characteristics are combined in a region growing method that results in sets of points corresponding to individual features (fig. 1(c)). By obtaining the features of the object we can create an editable and parameterized CAD model described by its various connected components. This type of model provides the user-designer with the capability of editing, redesigning and reproducing the original object, depending on his preferences and needs, by editing the features of the model.

Fig. 1: (a) A point cloud of a screwdriver and its convex hull, (b) The screwdriver point cloud concavity intensity map (c) Different regions detected by the region growing method.

Our region growing method returns sets of points corresponding to the feature and sets of points corresponding to the boundaries of the features. The boundaries are used for curve approximation whereas the region points are used in surface approximation. In the following we will focus on our approach to constructing smooth approximate curves for the region boundaries.

For more details on our feature-based reconstruction scheme, please refer to [12].

3. CURVE APPROXIMATION USING CUBIC RATIONAL BEZIER CURVES

The region growing method mentioned in the previous section returns sets of points corresponding to borders of feature regions. Suppose we have a region border consisting of n 3D points. We would like to find the curve that best approximates this data set to represent the general morphology of the border.

Our approach is based on work by [16]. In this work the authors present an optimization process through which rational b-spline curves are fit to point data. We adopt this approach to fit cubic rational Bezier curves to sets of points corresponding to feature boundaries and extend it to ensure that the curves created conform to the conditions required for G^1 continuity. We use an equivalent instance of the general NURB, namely piecewise rational cubic Bezier curves because the constraints we apply decrease the degrees of freedom of our problem and our requirements are well met with this low degree simpler representation resulting in fast converging optimization algorithm. Using piecewise rational Bezier curves we basically follow an optimization approach which can inherently rule out noisy data without affecting the shape of the boundary as a whole.

Curve approximation is carried out with a least squares optimization procedure. Suppose $Q=\{Q_1,Q_2,\ldots Q_m\}$ is a set of ordered border points and C is an approximating rational Bezier curve given by the equation:

$$C(u_i)=\frac{\sum_{j=1}^{n} w_j P_j B_j(u_i)}{\sum_{j=1}^{n} w_j B_j(u_i)} \qquad (3.1)$$

where n=4 for a cubic rational Bezier curve, u_i is the parameter value associated with border point Q_i, P_j are the control points, w_j is the weight of each control point and B_j is the corresponding Bernstein polynomial.

Assuming that all points of Q should be approximated by the curve, we would like to minimize the error:

$$e_i = Q_i - C(u_i), \quad i=1..m. \tag{3.2}$$

We need to assign parameter values u_i to each point Q_i. We use chordal parameterization [5] in which we express the parameter value of each point Q_i in reference to its position in the point sequence:

$$u_i = \frac{\sum_{j=2}^{i} \Delta Q_j}{\sum_{j=2}^{m} \Delta Q_j} \tag{3.3}$$

The least squares problem is then to minimize the error:

$$E = \sum_{i=1}^{m} e_i^2 \quad (3.4) \overset{(3.1),(3.2)}{\longrightarrow} \quad E = \sum_{i=1}^{m} (Q_i - C(u_i))^2 \tag{3.5}$$

We consider the product w_jP_j as one variable and partially differentiate eqn. (3.5) by factor w_kP_k, k=1..4. This leads to equations:

$$\frac{\partial E}{\partial (w_k P_k)} = 0 \quad \Rightarrow \sum_{i=1}^{m} \left[2\left(\sum_{j=1}^{4} w_j P_j B_j(u_i) - Q_i \sum_{j=1}^{4} w_j B_j(u_i)\right) B_k(u_i) \right] = 0 \,, \quad k=1..4 \tag{3.6}$$

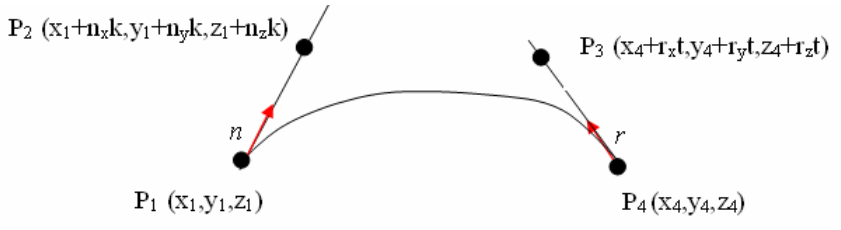
from which we obtain the following linear system of equations:

$$\begin{aligned} [B]^T [B][wP_x] &= [B]^T [Q_x B][w] \\ [B]^T [B][wP_y] &= [B]^T [Q_y B][w] \\ [B]^T [B][wP_z] &= [B]^T [Q_z B][w] \end{aligned} \tag{3.7}$$

where (i.e. for coordinate x):

$$B = \begin{bmatrix} B_1(u_1) & B_2(u_1) & B_3(u_1) & B_4(u_1) \\ \vdots & & & \vdots \\ B_1(u_m) & \dots & \dots & B_4(u_m) \end{bmatrix}, \quad wP_x = \begin{bmatrix} w_1 P_{x_1} \\ w_2 P_{x_2} \\ w_3 P_{x_3} \\ w_4 P_{x_4} \end{bmatrix}, \quad Q_x B = \begin{bmatrix} Q_1 B_1(u_1) & Q_1 B_2(u_1) & Q_1 B_3(u_1) & Q_1 B_4(u_1) \\ Q_2 B_1(u_2) & & & \\ \vdots & \vdots & \vdots & \vdots \\ Q_m B_1(u_m) & \dots & \dots & Q_m B_4(u_m) \end{bmatrix}$$

Without affecting the shape of the curve or the parametrization we can assume that one of the weights is 1. Therefore we assume that weight w_2=1. We could eliminate one more weight by reparametrizing u, but we need reparametrization to be a parameter in the optimization process. Also to ensure G^1 continuity between Bezier curves we make sure the starting point of one curve coincides with the end point of the previous curve and that the inner control points are located accordingly on the tangents of the end points.



The unit vectors of the tangents at the end control points are estimated by expressing each tangent of each point as a linear combination of its 4 closest neighbors [10].

Vectors wP_x, wP_y, wP_z are modified by expressing the coordinates of inner control points P_2 and P_3 in relation to the end points. In eqn. 3.7 we now have:

$$\left[B^T\right]\left[B\right]\begin{pmatrix} w_1(x_1+0) \\ w_2(x_1+n_x k) \\ w_3(x_4+r_x t) \\ w_4(x_4+0) \end{pmatrix} = \left[B^T\right]\left[Q_x B\right]\left[w\right] \tag{3.8}$$

Since we assume that $w_2=1$, the system is transformed so that its final form is (e.g. for direction x):

$$\left[B^T\right]\left[B\right]\begin{pmatrix} w_1 x_1 \\ n_x k \\ w_3 r_x t \\ w_4 x_4 \end{pmatrix} = \left[B^T\right]\left[Q_x B\right]\left[w\right] - \left[B^T\right]\left[B\right]\left[A_x\right], \text{ where } \left[A_x\right] = \begin{bmatrix} 0 \\ x_1 \\ w_3 x_4 \\ 0 \end{bmatrix} \tag{3.9}$$

These systems of linear equation (for directions x, y and z) can be used to derive values for variables w_1, k, t, and w_4 (all weights in vector $[w]$ are initialized to 1), therefore essentially determining the inner control points' coordinates and approximate values for two of the three weights. To achieve better curve approximation, we proceed to a second step using the control points calculated above to compute more appropriate weight values. However, for each system solution we obtain a different set of variable values. Therefore the following weight optimization procedure is carried out once for every solution set and we accordingly keep the solution that best minimizes the least squares error.

Specifically, we express equation (3.4) as follows:

$$e_i = \sum_{i=1}^{m} \left(e_{x_i}^2 + e_{y_i}^2 + e_{z_i}^2\right) \tag{3.10}$$

We partially differentiate by weights w_k (k=1, 3, 4) and obtain a system of equations from which we can substitute the control points and the weights found in the previous step and optimize the weight vector. Specifically, for $\partial E/\partial w_k = 0$, k=1,3,4, the equations obtained are of the form:

$$\begin{aligned} &\sum_{i=1}^{m}\left(B_k(u_i)\left(Q_{x_i}Q_{x_i}\sum_{j=1}^{4} w_j B_j(u_i) + Q_{y_i}Q_{y_i}\sum_{j=1}^{4} w_j B_j(u_i) + Q_{z_i}Q_{z_i}\sum_{j=1}^{4} w_j B_j(u_i)\right)\right) = \\ &\sum_{i=1}^{m}\left(B_k(u_i)\left(Q_{x_i}\sum_{j=1}^{4} w_j P_{x_i} B_j(u_i) + Q_{y_i}\sum_{j=1}^{4} w_j P_{y_i} B_j(u_i) + Q_{z_i}\sum_{j=1}^{4} w_j P_{z_i} B_j(u_i)\right)\right) \end{aligned} \tag{3.11}$$

Eventually we end up with the linear system:

$$\begin{aligned}&\left(\left[Q'_x B\right]^T\left[Q'_x B\right]+\left[Q'_y B\right]^T\left[Q'_y B\right]+\left[Q'_z B\right]^T\left[Q'_z B\right]\right)\left[w'\right]=\\&\left[Q'_x B\right]^T\left[B\right]\left[w'P'_x\right]+\left[Q'_y B\right]^T\left[B\right]\left[w'P'_y\right]+\left[Q'_z B\right]^T\left[B\right]\left[w'P'_z\right]+(-1)\left(\left(\left[Q'_x B\right]^T\left[C_x\right]\right)+\left(\left[Q'_y B\right]^T\left[C_y\right]\right)+\left(\left[Q'_z B\right]^T\left[C_z\right]\right)\right)\end{aligned} \tag{3.12}$$

where (e.g. for x coordinate): $Q'_x B=\begin{bmatrix} Q_1B_1(u_1) & \cdots & Q_1B_4(u_1)\\ \vdots & \ddots & \vdots \\ Q_mB_1(u_m) & \cdots & Q_mB_4(u_m)\end{bmatrix}$ and $Cx=\left[Q'_x B\right]\begin{bmatrix} P_{x_2}B_2(u_i)\\ \vdots \\ P_{x_2}B_2(u_m)\end{bmatrix}$.

This procedure is carried out iteratively until the error function is minimized.

Generally this is a fast curve approximation approach that produces smooth continuous curves that interpolate or pass close by the data points. A good approximation is reached within a few iterations. In some cases, the minimal error is reached after the first iteration, if the point cloud data is not noisy and the tangent estimations are good. The accuracy of the approximating curve basically depends on how well the tangents are estimated at the end points, since we restrict the inner control points to be located on them. The weights are used not only to determine the shape of the curve but also to adjust the parameterization of the curve. Chordal parameterization works well, assuming that the points are given as a sequence in 3D space. For this reason we transform all points that we want to approximate by translating P_1 to the origin, and then aligning vector (P_4-P_1) to the positive z axis. Then we simply sort the points Q_i according to their z coordinate by:

$$Q'_i = A(v)T(-P_{x_1},-P_{y_1},-P_{z_1})Q_i \tag{3.13}$$

where $v=\left[P_4-P_1\right]=\begin{bmatrix}a\\b\\c\end{bmatrix}$, $A=\begin{bmatrix} \frac{l}{|\vec{v}|} & \frac{-ab}{l|\vec{v}|} & \frac{-ac}{l|\vec{v}|} & 0\\ 0 & \frac{c}{l} & \frac{-b}{l} & 0\\ \frac{a}{|\vec{v}|} & \frac{b}{|\vec{v}|} & \frac{c}{|\vec{v}|} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$, and $l=\sqrt{b^2+c^2}$

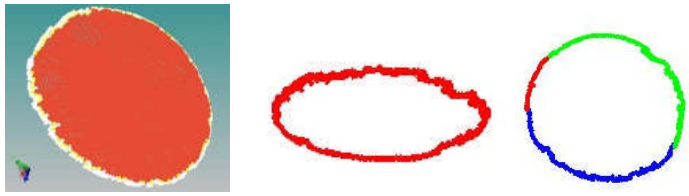
4. EXAMPLE OF CURVE FITTING USING RATIONAL BEZIER CURVES

In this section we will present examples of our curve fitting method which was implemented using Maple 10. The data sets used in the examples are areas corresponding to feature boundaries returned by our region growing algorithm. For each point in the border point cloud we have computed in a previous phase (feature detection phase) its corresponding surface normal vector estimate. The boundary point set is divided into subsets (curve segments) based on the progressive change in the surface normal of the points. The point cloud is divided into as many sets needed, so that the angle formed by the surface normals of the start and end point of each section is below a threshold and allows for alignment of the section's point data and sorting as described above (eqn. 3.13).

Example 1:

Fig. **2** shows one of the region borders detected by our region growing method. This border contains 450 3D points. We begin by determining the number of curves with which the border is to be approximated. In this example the border is approximated using three cubic rational Bezier curves. A small preprocessing is carried out on the border area points before we perform curve fitting. First, for each set of points corresponding to a curve we align the points with one of the coordinate axis to sort the segment points by this coordinate thus ensuring the correct layout of the points.

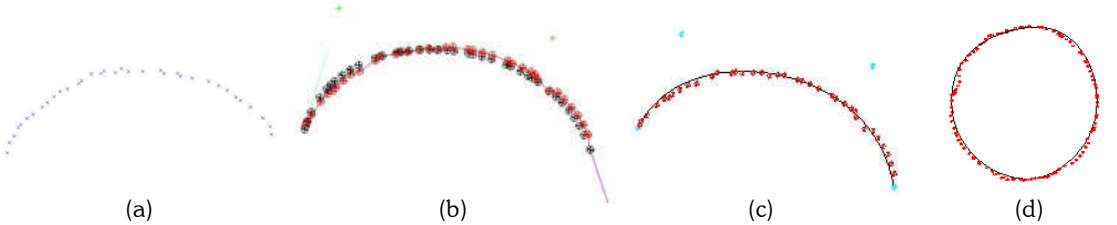
The border is then "thinned out" by representing groups of neighboring points with the border point that is closest to their center of mass. The size of the groups used depends on how dense we would like our final point set to be.



(a) (b) (c)

Fig. 2: (a) Region detected by region growing, (b) the border of the region to be approximated, (c) curve segments used for curve approximation.

After making our point cloud thinner, we estimate the tangents at the end points of each curve segment by considering the closest neighbors. We apply the curve fitting algorithm to each curve segment. For instance, Fig. 3 displays the first curve segment to be approximated. This method reaches the best solution after 10 iterations as shown in Fig. 3 (b),(c). The red points correspond to points of the curve segment whereas the black points are the respective points calculated on the rational curve. The final curve approximating the point cloud is shown in Fig. 3(d). The first set of points used for approximation consists of m=40 points, with the distance between the end points corresponding to 0.0279 and the average error is given by E/m=0.164·10^{-6}. For the second approximating curve, the set of points used consists of m=36 points with end points distanced at 0.024 and average error is 0.708·10^{-6} after one iteration. The final segment consists of m=34, the distance between the end points is 0.0229 and, after one iteration, average error is 0.825·10^{-6}



(a) (b) (c) (d)

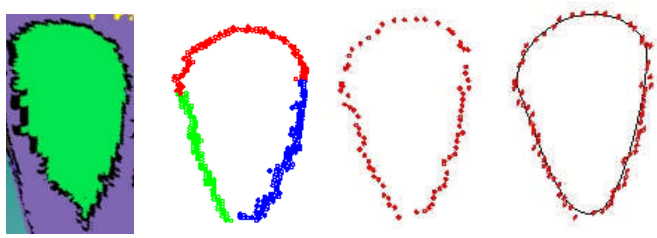
Fig. 3: (a) Points to be used for approximation, (b) points used for approximation (red) and corresponding points computed on curve (black), (c) the approximating curve (d) final curve spline approximation of point cloud.

Example 2: Fig. 4 shows another region detected with the region growing algorithm when applied to the screwdriver point cloud. The border of this region consists of 498 points which is divided into three segments as show in fig. 4(b). The points of each segment are sorted and we compute the thinner version of the point cloud (fig. 4(c)). The tangents at each end point are estimated and the curve approximation method is applied to each segment. The resulting curves are shown in fig. 4(d). The first segment consists of m=33 points, with the Euclidean distance between the end points corresponding approximately to 0.02. After 10 iterations the least squares error function is minimized and the ratio average error E/m is 0.64·10^{-6}. The second segment consists of m=19 points, the distance between the end points is 0.014 and the average error is 0.14·10^{-6} after 11 iterations (the sum of squared errors is 0.27·10^{-6}). Finally, the third segment consists of m=27 points, the distance between the end points is 0.017 and after 30 iterations the average error is 0.46·10^{-6}.

To evaluate the result of our approximation method we performed global optimization calculating the absolute optimal curve using the IpOpt software [6]. For instance, for the second curve segment we perform global optimization treating t_i, $i=1..n$, the weights and the control points as unknowns and we minimize the objective function of eqn. 3.5 under the following constraints:

$$(Q_i - C(t_i)) \cdot C'(t_i) = 0 \tag{3.14}$$

where $0 \le t_i \le 1$ and $i=1..n$. We use as initial state the one derived from our fitting method. The sum of the squared errors after convergence is $1.279 \cdot 10^{-6}$.



(a) (b) (c) (d)

Fig. 4: (a) The region created by region growing (b) the border of the region, (c) the sample of points used for approximation, (d) the final curve approximation of the border.

We observe that the method proposed is a fast curve fitting algorithm which constructs rational curves with G^1 continuity. If the tangent estimation is a good estimate and the point cloud is not "noisy" then the solution may be computed within a single iteration. Otherwise, a few iterations are needed to adjust the weights and to adapt the parameterization.

5. CONCLUSIONS

We have presented a fast curve fitting method which derives cubic rational Bezier curves conforming to G^1 continuity constraints. We impose continuity constraints into the least squares optimization process to ensure that the computed control points respect the estimated tangents at the end points. We also adjust and control points and weights until the error of the least squares is minimized. The weights are used not only to affect the shape of the curve but also to adjust the parametrization of the curve. We use this technique to construct curves which approximate the border regions that are derived from the feature detecting method that we use to create feature-based CAD models from 3D point clouds.

As compared to other curve approximation methods which perform curvature-based fairness our method is very effective and efficient with satisfactory accuracy. This is important given that the application in which this method is employed calls for the reconstruction of the borders of all features, which in the case of freeform objects, may be a few hundreds and quite complex.

6. REFERENCES

[1] Cyberware,Cyberware rapid 3d scanners - desktop 3d scanner samples, http://www.cyberware.com/products/scanners/desktopSamples.html

[2] Fang, L.; Gossard, D.: Multidimensional curve fitting to unorganized data points by nonlinear minimization, Computer Aided Design, 27(1), 1995, 48-58.

[3] Flory, S.; Hofer, M.: Constrained curve fitting on manifolds, Computer Aided Design, 40, 2008, 25-34.

[4] Galvez, A.; Iglesias, A.; Cobo, A.; Puig-Pey, J.; Espinola, J.: Bezier curve and surface fitting of 3d point cloud through genetic algorithms, functional networks and least-squares approximation, in Iccsa 2007, lncs 4706, O. Gervasi and M. Gavrilova, Editors, 2007, Springer-Verlag Berlin Heidelberg, p. 680-693.

[5] Hoschek, J.; Lasser, D.: Fundamentals of computer aided geometric design, ed. A. Peters, 1993.

[6] IpOpt - Interior Point Optimizer,http://projects.coin-or.org/Ipopt

[7] Lee, I.-K.: Curve reconstruction from unorganized points, Computer Aided Geometric Design, 17, 2000, 161-177.

[8] Piegl, L.; Tiller, W.: The NURBS Book, 2nd ed., Springer-Verlag, 1997.

[9] Pottmann, H.; Leopoldselder, S.; Hofer, M.: Approximation with active B-spline curves and surfaces, in Proc of the Pacific Graphics IEEE Press, 2002.

[10] Renner G.: A method of shape description for mechanical engineering practice, Computers in Industry, 3, 1982, 137-142.

[11] SINTEF, SISL - Sintef spline library, http://www.sintef.no/content/page1____5470.aspx

[12] Stamati V.; Fudos I.: A feature-based approach to re-engineering objects of freeform design by exploiting point cloud morphology, in SPM 2007: ACM Symposium on Solid and Physical Modeling, Beijing, China, 2007.

[13] Szobonya, L.; Renner, G.: Construction of curves and surfaces based on point clouds, in Proc. First Hungarian Conference on Computer Graphics and Geometry, Budapest Hungary, 2002.
[14] Ueng, W.-D.; Lai, J.-Y.; Tsai, Y.-C.: Unconstrained and constrained curve fitting for reverse engineering, International Journal of Advances Manufacturing Technology, 33, 2007, 1189-1203.
[15] Wang, W.; Pottmann, H.; Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization, ACM Transactions on Graphics, 25(2), 2006, 214-238.
[16] Yau, H.-T.; Chen, J.-S.: Reverse engineering of complex geometry using rational B-splines, International Journal of Advances Manufacturing Technology, 13, 1997, 548-555.