UDC 514.74:515.142.4:519.65
Original scientific paper
Received: 01.06.2011.

Methods of obtaining smooth surface in 2D/3D surface reconstruction

Amod Tiwari(1), **Nitya Bajpai**(2), **Vijay Pal Singh Rana**(2) and **Vinay Kumar Pathak**(3)
(1)Department of Computer Science and Engineering, Pranveer Singh Institute of Technology, Kanpur, INDIA
e-mail: amodtiwari@gmail.com
(2)Singhania University Rajasthan, INDIA
e-mail: vartikasrivastava1@gmail.com; aics_ddun@yahoo.com
(3)University of Haldwani, INDIA
e-mail: vpathak@lycos.com

SUMMARY

Surface reconstruction is an emergent research area in the field of computer aided design and manufacturing. There are various methods / algorithms which are working considerably well for surface reconstruction problem but we cannot say to the best of our knowledge that we got all the solutions. Missing surface can be repaired either by surface patch or by extending boundary curves. However, in both cases, surface smoothening problem arises in form of flat surface. The present paper has been tried to offer a solution to above problem which makes the curve smoother.

***Key words**: stretching, flat surface, curve extension, curve angle, LINCE model, stl format1.*

1. INTRODUCTION

Due to recent technological development in scanning process (laser and optical), it has become easy to get point cloud data for a given artifact or an object. The artifacts scanned may be broken and may require the reconstruction work. Different algorithms are available for reconstruction of given object from its point cloud data.

An object surface can be viewed as a family of the curves. If the surface has a hole in it (Fig. 1(a)), the curves forming it will also be broken in between (Figures 1(b) and 1(c)). This paper presents an algorithm, which uses the polar geometry and concept of Polar Radial Angle Model [1] of curves. Using this geometry, a relationship for the curvature (k) and arc-length (s) can be obtained from which the rate of change of curvature with respect to arc length can be found. The proposed algorithm first finds out the k-s information of the near by region of the broken part and then it interpolates the k-s information's for the missing region. After estimating the k-s information for the missing region, intrinsic LINear Curvature Element (LINCE) Model of a curve is used to get the x-y values for the missing part [1].

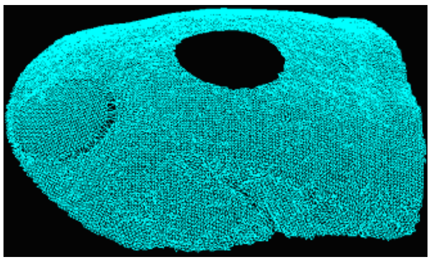


Fig. 1(a) Original triangulated 3-D data

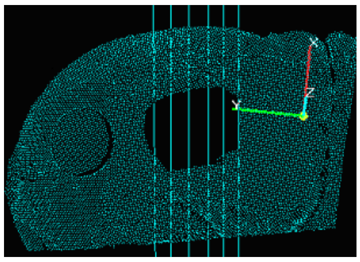


Fig. 1(b) Cross-sectional view data

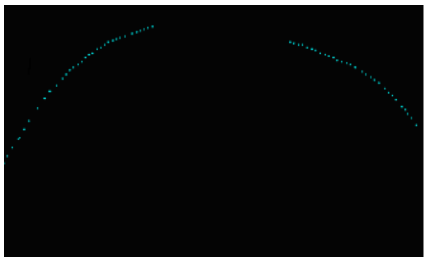


Fig. 1(c) Cross-sectional of point cloud data

2. RELATED PREVIOUS WORK

There exist several techniques to reconstruct surface from the point cloud data. This section deals with some of the well-known techniques that can be used in the field of surface reconstruction.

2.1 Hole filling using triangulation of each connected component

One technique is to triangulate each connected components of the surfaces boundary; thereby filling each hole with a patch that has the topology of a disc. This technique works well for simple holes in nearly flat surfaces. But on convoluted holes it is likely to result in self-intersecting geometry. In such case we would like mesh-based reconstruction.

2.2 Mesh-based reconstruction with hole filling

This method creates a surface that bounds the maximum region of space consistent with the scans, so it is guaranteed to produce a watertight surface. However the method may lead to surfaces that are less plausible than smoothly extending the observed surfaces. Moreover, the method requires knowledge of scanner lines of sight, and it performs poorly if these lines of sight do not adequately cover the volume outside the object. Additional lines of sight can be obtained by scanning backdrops placed behind the object but still within the scanner's working volume [2, 3], or by using a separate sensor to detect objects silhouettes [4]. However these solutions may be difficult to deploy outside the laboratory. This algorithm does not require any such information like type of scanner, line of sight, etc.

2.3 Volumetric diffusion

Volumetric diffusion algorithm is also a method used for the reconstruction of bad surfaces. It takes the time and memory proportional to n^2 where '*n*' is the number of points in the data. This uses the diffusion to complete a volumetric representation of the surface. It produces a closed, manifold triangle mesh without self-intersection. Although the algorithm is robust, it contains a number of free parameters, like: the distance at which we clamp source terms, the size of convolution filter, the distance from a hole boundary, etc. [5]. This boundary are curve base measure with the help of Serret Frenet Equations.

3. SERRET FRENET EQUATIONS

Consider a curve *AC* as shown in Figure 2. Let *r* be the position vector of a generic point *P* and let *s*, the arc-length of *P* from a reference point *A*, be the parameter describing the curve as *r*(*s*). The unit tangent vector, the curvature, the torsion, the normal and the binormal of the curve *AC* at the point *P* are given as follows:

$$t = \frac{dr}{ds} \tag{1}$$

$$\frac{dt}{ds} = kn \tag{2}$$

$$\frac{dn}{ds} = -kt + \tau b \tag{3}$$

$$\frac{db}{ds} = -\tau n \tag{4}$$

It is clear that *t*, *n* and *b* are mutually perpendicular on each other. If we consider that ***t***, ***b*** and ***n*** are vectors quantity there for:

$$\boldsymbol{b} = \boldsymbol{t} \times \boldsymbol{n} \tag{i}$$

$$\boldsymbol{t} = \boldsymbol{n} \times \boldsymbol{b} \tag{ii}$$

$$\boldsymbol{n} = \boldsymbol{b} \times \boldsymbol{t} \tag{iii}$$

This can be proved with the help of the vector cross product method.

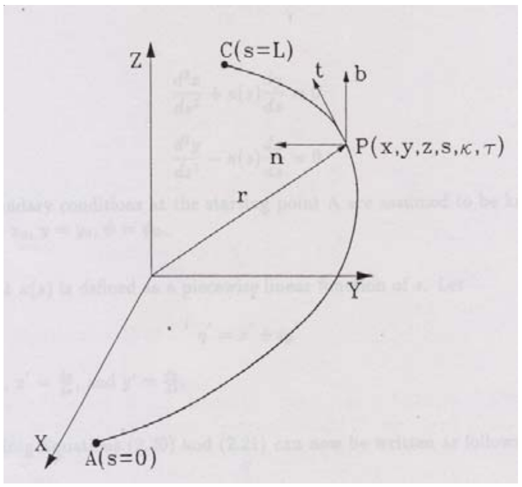


Fig. 2 Intrinsic geometry of 3-D curve AC

Here, ***t*** is the unit tangent vector, ***b*** is the unit binormal vector and ***n*** is the unit binormal vector, while *k* is the curvature and τ is the torsion. Point *P*(*x*, *y*, *z*, *s*, κ, τ) having six co-ordinates is known as frame of reference. Equations (2), (3) and (4) are known as Serret Frenet Equations. If the curve is planar then $\tau = 0$ and the above equations can be written as follows:

$$r = \frac{[x(s)]}{[y(s)]} \tag{5}$$

where *x*(*s*) and *y*(*s*) are the length of arc about the axes of *x* and *y* respectively, with:

$$t = \frac{dr}{ds} \quad \text{and:} \quad n \cdot t = 0 \tag{6}$$

The problem of finding the co-ordinates of *x* and *y* as a function of arc length parameter *s* can be solved using Serret Frenet Equations, Eqs. (2) through (4). Rewriting Eq. (6) follows:

$$t = \frac{dr}{ds} = \left[\frac{dx(s)}{ds} \frac{dy(s)}{ds} \right]^T \tag{7}$$

where *T* denotes transformation, and:

$$n = \begin{bmatrix} -\frac{dy(s)}{ds} \\ \frac{dx(s)}{ds} \end{bmatrix} \tag{8}$$

since $n \cdot t = 0$, and:

$$\frac{dt}{ds} = \begin{bmatrix} \frac{d^2x(s)}{ds^2} \\ \frac{d^2y(s)}{ds^2} \end{bmatrix} \tag{9}$$

Substituting Eqs. (8) and (9) in Eq. (2) yields:

$$\begin{bmatrix} \frac{d^2x(s)}{ds^2} \\ \frac{d^2y(s)}{ds^2} \end{bmatrix} = k \begin{bmatrix} \frac{-dy(s)}{ds} \\ \frac{dx(s)}{ds} \end{bmatrix} \tag{10}$$

Comparing Eq. (10) both sides as in matrices form yields:

$$\frac{d^2x(s)}{ds^2} = -k \frac{dy(s)}{ds} \tag{11}$$

$$\frac{d^2y(s)}{ds^2} = k \frac{dx(s)}{ds} \tag{12}$$

Now putting in Eq. (11):

$$\frac{dx(s)}{ds} = \cos\psi_{(\sigma)},$$

and putting in Eq. (12):

$$\frac{dy(s)}{ds} = \sin\psi_{(\sigma)},$$

where $\psi_{(\sigma)}$ is the radial angle, yields new equations which can be written as:

$$\frac{d}{ds}\left(\cos\psi_{(\sigma)}\right) = -k \frac{dy(s)}{ds} \tag{13}$$

$$\frac{d}{ds}\left(\sin\psi_{(\sigma)}\right) = k \frac{dx(s)}{ds} \tag{14}$$

Integrating both sides of Eqs. (13) and (14), where $[s_0, s]$ is interval of given boundary, one obtains:

$$x(s) = \int_{s0}^{s} \cos[\Psi(\sigma)]d\sigma + x_0 \tag{15}$$

$$y(s) = \int_{s0}^{s} \sin[\Psi(\sigma)]d\sigma + y_0 \tag{16}$$

$$\Psi(\sigma) = \int_{s0}^{\sigma} k(s)ds + \Psi_0 \tag{17}$$

4. LINCE MODEL

Let a curve pass through the points P_0 (x_0, y_0) and P_n (x_n, y_n) in a 2-D space (Figure 3). Let the directions of tangent at P_0 and P_n have been specified by ψ_0 and ψ_n and arc lengths from a reference point *O* as s_0 and s_n respectively. If *k* is the curvature at any point *P*, then it is assumed that the variation of *k* as a function of the arc length as the parameter $k=k(s)$ has been specified. It can be seen that $k(s)$ defines the shape of the curve.

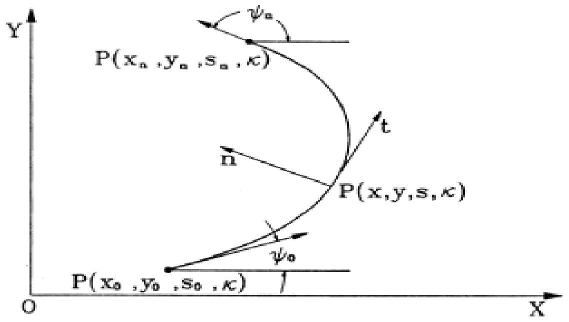


Fig. 3 Plane curve definition using Cartesian coordinates and tangent angles

The curvature $k(s)$ can be defined as a series of piecewise continuous linear functions with curvature k_0 and k_n at the initial and final points. The intrinsic equation of curve pass through two points and can be written as:

$$k(s) = \frac{k_1 - k_0}{s_1 - s_0}(s - s_0) + k_0 \quad \text{where } s_0 \le s \le s_1$$

$$k(s) = \frac{k_2 - k_1}{s_2 - s_1}(s - s_1) + k_1 \quad \text{where } s_1 \le s \le s_2 \tag{18}$$

$$k(s) = \frac{k_n - k_{n-1}}{s_n - s_{n-1}}(s - s_{n-1}) + k_{n-1} \quad \text{where } s_{n-1} \le s \le s_n$$

where $k(s)$, k_0, k_1, s_0, s_1, k_{n-1} and s_{n-1} are intrinsic coordinates.

The area under the curvature equation represents the change in the tangent angles of the initial and final points of the 2D curve. So, from Eq. (17) follows:

$$\Psi(\sigma) - \Psi_0 = \int_{s0}^{\sigma} k(s)ds \tag{19}$$

Putting the value $k(s)$ in Eq. (19) from Eq. (18) one obtains:

$$\Psi_n - \Psi_0 = \frac{k_0 + k_1}{2}(s_1 - s_0) + \frac{k_1 + k_2}{2}(s_2 - s_1) + \ldots + \frac{k_{n-1} + k_n}{2}(s_n - s_{n-1}) \tag{20}$$

Similarly, from Eqs. (15) and (16) follows:

$$x(s) - x_0 = \int_{s0}^{s} cos[\Psi(\sigma)]d\sigma \tag{21}$$

$$y(s) - y_0 = \int_{s0}^{s} sin[\Psi(\sigma)]d\sigma \tag{22}$$

Substituting the value $k(s)$ from Eq. (18) into Eqs. (21) and (22) one obtains:

$$x_n - x_0 = \int_{s_0}^{s_1} cos[(\frac{k_1 - k_0}{s_1 - s_0})(\frac{\sigma^2}{2} - s_0\sigma) + k_0\sigma + C_1]d\sigma + \int_{s_1}^{s_2} cos[(\frac{k_2 - k_1}{s_2 - s_1})(\frac{\sigma^2}{2} - s_1\sigma) + k_1\sigma + C_2]d\sigma + \ldots$$
$$\ldots + \int_{s_{n-1}}^{s_n} cos[(\frac{k_n - k_{n-1}}{s_n - s_{n-1}})(\frac{\sigma^2}{2} - s_{n-1}\sigma) + k_{n-1}\sigma + C_n]d\sigma$$

$$y_n - y_0 = \int_{s_0}^{s_1} sin[(\frac{k_1 - k_0}{s_1 - s_0})(\frac{\sigma^2}{2} - s_0\sigma) + k_0\sigma + C_1]d\sigma + \int_{s_1}^{s_2} sin[(\frac{k_2 - k_1}{s_2 - s_1})(\frac{\sigma^2}{2} - s_1\sigma) + k_1\sigma + C_2]d\sigma + \ldots$$
$$\ldots + \int_{s_{n-1}}^{s_n} sin[(\frac{k_n - k_{n-1}}{s_n - s_{n-1}})(\frac{\sigma^2}{2} - s_{n-1}\sigma) + k_{n-1}\sigma + C_n]d\sigma$$

where C_1, C_2, …. C_n are constants. Note $C_1 = \Psi_0$. The other constants can be expressed in terms of k_i, s_i, $i=0,\ldots,n$ and Ψ_0 using following relation:

$$C_j = C_{j-1} + \frac{k_{j-1} - k_{j-2}}{s_{j-1} - s_{j-2}}(\frac{s_{j-1} * s_{j-1}}{2} - s_{j-1} * s_{j-2}) + k_{j-2} * s_{j-1} + \frac{k_j - k_{j-1}}{s_j - s_{j-1}}(\frac{s_{j-1} * s_{j-1}}{2}) - k_{j-1} * s_{j-1} \tag{23}$$

where $j = 2,3,\ldots n.$

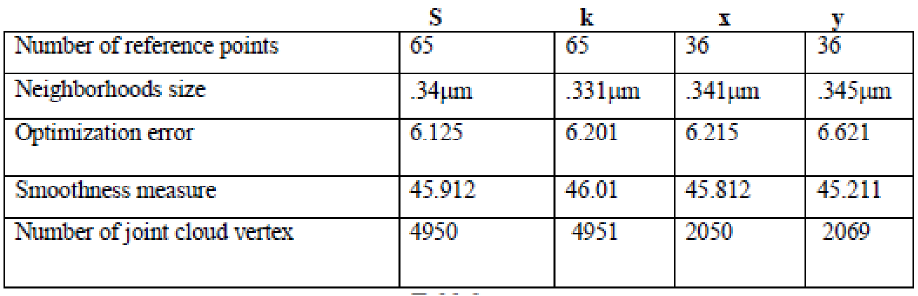
Table 1 k-s data (see plot in Figure 4)

s	0.00	2.80	5.10	6.92	8.27	9.17	9.75	10.32	11.23	12.57	14.39	16.70	19.49
k	0.009	0.015	0.029	0.063	0.179	0.710	2.000	0.707	0.179	0.063	0.029	0.015	0.009

Table 2 x-y data obtained using LINCE model for k-s values in Table 1 (see plot in Figure 5)

x	2.00	3.00	2.50	3.00	3.50	4.00	4.50	5.00	5.50	6.00	7.00	7.50	8.00
y	0.00	-2.75	-5.00	-6.75	-8.00	-8.75	-9.00	-8.75	-8.00	-6.75	-5.00	-2.75	0.00

Table 3 Characteristic data of the problem solved in Figures 4 and 5



	S	k	x	y
Number of reference points	65	65	36	36
Neighborhoods size	.34µm	.331µm	.341µm	.345µm
Optimization error	6.125	6.201	6.215	6.621
Smoothness measure	45.912	46.01	45.812	45.211
Number of joint cloud vertex	4950	4951	2050	2069

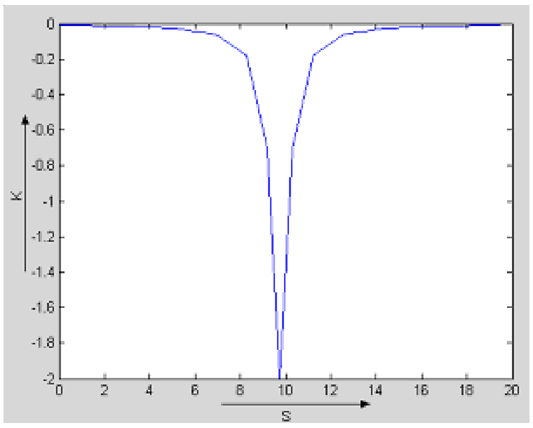


Fig. 4 Plot for k-s data shown in Table 1

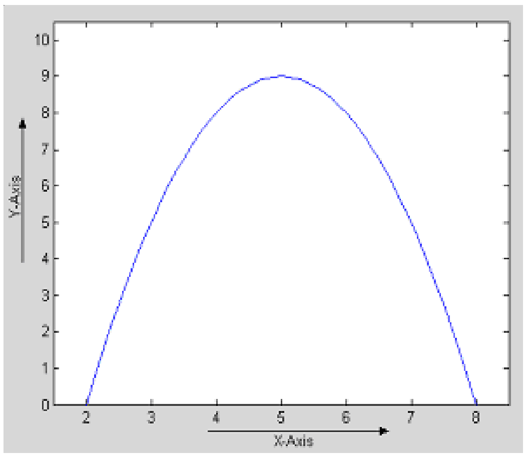


Fig. 5 Plot for x-y data obtained using LINCE model for k-s values shown in Table 2

4.1 Computation of cross-section

The step of computation of cross-sections of each hole consists of taking cross-sectional planes perpendicular to the hole. Let ‘*p*’ be a set of 3D points defined as:

$$p = \{ x_i, y_i, z_i \mid i \in [0,n], n \in N \}$$

and a 3D cross-sectional plane *P* with the equation *Ax* + *By* + *Cz* + *D* = *0* where *A*, *B* and *C* are the direction ratios of a cross-sectional plane.

If *L*, *M* and *N* are the direction cosines of cross-section plane there for:

$$L = \cos\alpha \qquad (24)$$

$$M = \cos\beta \qquad (25)$$

$$N = \cos\chi \qquad (26)$$

Relation between direction cosine and direction ratios are:

$$L = \frac{A}{\sqrt{\left(A^2 + B^2 + C^2\right)}} \qquad (27)$$

$$M = \frac{B}{\sqrt{\left(A^2 + B^2 + C^2\right)}} \qquad (28)$$

$$N = \frac{C}{\sqrt{\left(A^2 + B^2 + C^2\right)}} \qquad (29)$$

Therefore, the relation between direction cosines is $L^2 + M^2 + N^2 = 1$.

Distance of point p_i from the plane *P* can be achieved as:

$$D_i(P, p_i) = \frac{A * x_i + B * y_i + C * z_i + D}{\sqrt{A^2 + B^2 + C^2}} \quad i \in [0,n] \qquad (30)$$

where points (x_i, y_i, z_i), $i \in [0,n]$ for which $D_i \approx 0$ are the cross-sectional points and *A*, *B* and *C* are coefficients of (x_i, y_i, z_i). To get more cross-sections other parallel planes are taken. Number of cross sections taken for a hole depends on the size of the hole. Figure 6 shows a triangulated point cloud data with a hole and some cross sectional planes for the hole while Figure 7 shows the cross sectional data obtained from one cross sectional plane.

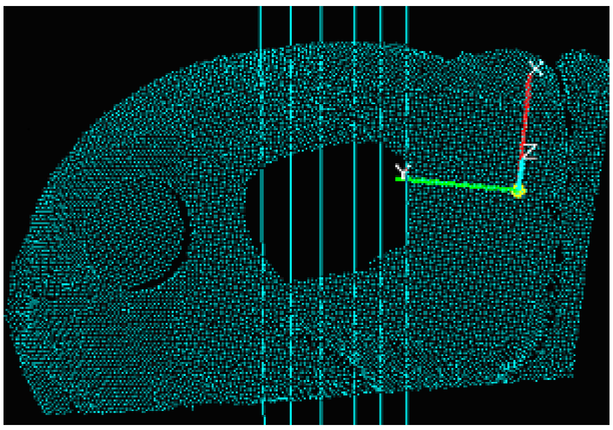


Fig. 6 Triangulated point cloud data with a hole and some cross sectional planes for the hole

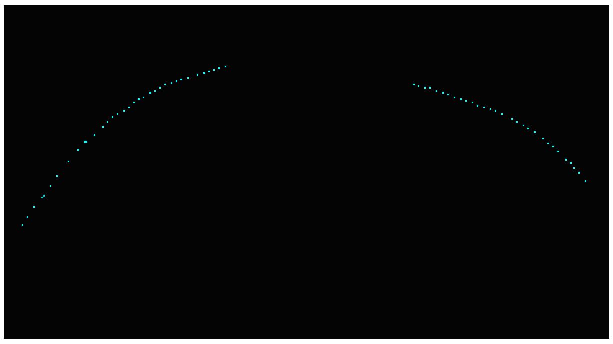


Fig. 7 Cross-sectional data with hole obtained from one cross sectional plane

4.2 Projection of 3D cross-sectional data on 2D plane

This step is to project cross sectional 3D point cloud data on 2D plane. Let (x_1, y_1, z_1) be a point on plane $z = \frac{-A}{C}x + \frac{-B}{C}y + \frac{D}{C}$ or $z = ax + by + c$ and after projecting it on *X-Y* plane (x_1', y_1') is obtained. Normal vector to the plane is $\overrightarrow{N_1} = -a\hat{i} - b\hat{j} + c\hat{k}$. By inspection a vector perpendicular to the plane $\overrightarrow{N_2} = -b\hat{i} + a\hat{j}$ can be obtained. Cross product of these two vectors gives another vector on the plane and is given by:

$$\overline{N_1 \times N_2} = -ac\hat{i} + bc\hat{j} - (a^2 + b^2)\hat{k}$$

$$\overrightarrow{e_1} = \frac{1}{\sqrt{a^2 + b^2}}(-b\hat{i} + a\hat{j})$$

and:

$$\overrightarrow{e_2} = \frac{(-ac\hat{i} + bc\hat{j} - (a^2 + b^2)\hat{k})}{\sqrt{(ac)^2 + (bc)^2 + (a^2 + b^2)^2}}$$

where $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$ are two orthogonal vectors in the fitted plane. Let the origin (o_i, o_j, o_k) be $(0,0,C)$ on the plane. So a vector from origin to point (x_1, y_1, z_1) is given by:

$$(x_1 - o\_i)\hat{i} + (y_1 - o\_j)\hat{j} + (z_1 - o\_k)\hat{k}.$$

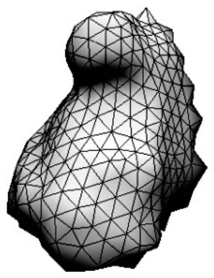
This vector lies in the plane of unit vectors $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$. So it can be written as linear combination of vectors $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$ as follows:

$$(x_1 - o\_i)\hat{i} + (y_1 - o\_j)\hat{j} + (z_1 - o\_k)\hat{k} = x_1'\overrightarrow{e_1} + y_1'\overrightarrow{e_2} \quad (31)$$

By equating the *i*, *j* and *k* component of the Eq. (31), three linear equations are obtained. Solving these linear equations (x_1', y_1') can be found.

5. DETECTION OF HOLE BOUNDARY

Our algorithm (which is partly based on the research presented in Refs. [6-15]) takes the triangulated point cloud data in stl formatΨ. In a .stl file, triangles can be divided into two categories. First category can be named as boundary triangles and second one as inner triangles. In Figure 8, triangles with shaded colour are boundary triangles and others are inner triangles. In a correct .stl file, the sides of each inner triangle are shared by two triangles and in boundary triangles, there is at least one side, which is not shared by more than one triangle. These edges of the boundary triangles may be called the boundary edges. The property of boundary edges (i.e. it is shared by only one triangle) can be used to identify the holes in the triangulated 3D data. We can claim, the boundary edges are the edges, which form the hole boundary. In brief, all the edges in a triangulated file, which are shared by only one triangle form the hole boundary.



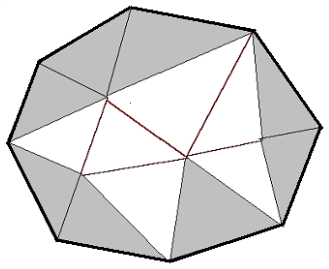


Fig. 8 Boundary triangles and inner triangles: thick line shows the boundary of the triangulated cloud and boundary triangles; inner triangles are shown with shaded colour

The algorithm is as follows:

```
with plane colour
Void GetHoleBoundry ()
   {
While (triangles left)
      {
SelectOne Triangle;
For (I=0; I<3; I++)
         {
If (Is BoundaryEdge (Edge))
Write To BoundaryEdge (Edge))
Edge = Edge -->next;
                    /*Select next edge of boundary*/
         }
      }
   }
```

5.1 Healing of cross sections

After getting the cross-sections projected on the 2D plane, these broken cross-sections are healed. Here we have presented the steps to heal a cross-section. These are repeated to heal all cross-sections.

Step1: First step takes a cross-section and separates it into two parts. Separation is done as follows. By looking into the point cloud data, maximum distance between two neighboring points can be found. So if in a cross-sectional point cloud data, distance between two consecutive points is found more than the maximum allowed distance between two data points, this is considered as a break in the cross sectional point cloud data (Figure 9). The data points before break are considered as Part 1 and remaining as Part 2.

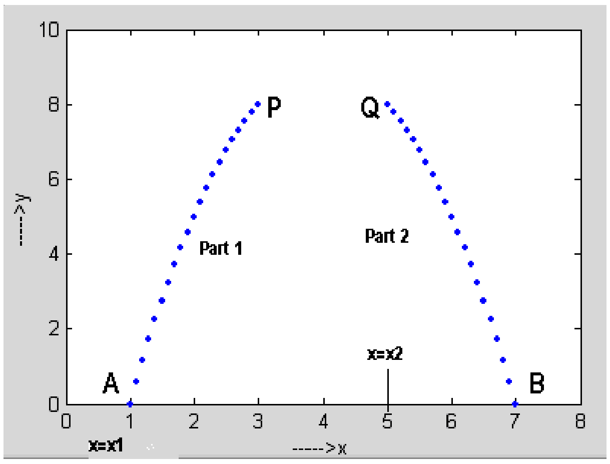


Fig. 9 Broken cross data - green color

Step 2: This step takes data points of the two parts of the cross-section and fits quadratic curve separately to these data sets using least square method. Let us say $y=f_1(x)$ and $y=f_2(x)$ are the two quadratic curves for the data sets of Part 1 and Part 2. Step 3 to step 9 heals the cross-section from left side taking point A as starting point.

Step 3: The curvature and arc-length (*k-s*) values are calculated for the two parts using equations:

$$k=\frac{\frac{d^2y}{dx^2}}{\sqrt[3]{1+(\frac{dy}{dx})^2}} \tag{32}$$

where k is reciprocal radius of curvature, and:

$$s=\int_{x=a}^{x=b}\sqrt{1+(\frac{dy}{dx})^2}\,dx \tag{33}$$

where $x=a$ is starting point and $x=b$ is last point, taking point *A* as starting point for Part 1 and *Q* as starting point for Part 2 (Figure 9). Plot of the *k-s* data for the above said two parts of the cross-sectional data is shown in Figure 10(a).

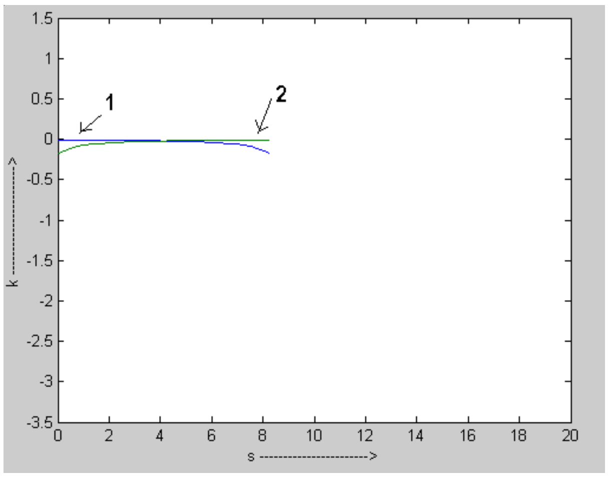


Fig. 10(a) k-s plot for sectional point cloud data . Part 1 (in blue color) and Part 2 (in green color)

Since the arc-length (*s*) for both the parts are calculated separately taking A and Q as starting points (points where $s=0$) for Part 1 and Part 2 respectively, *k-s* plot of the Part 1 and Part 2 is found overlapping.

Step 4: To compute arc-length for both parts taking *A* as reference point, a constant have to be added to each value of arc-length for Part 2. Let us say this constant is Δs (in further discussion Δs will be referred as the shift for Part 2). It includes the length of the curve Part 1 and the length of the missing part from *P* to *Q* (see Figure 10(b)).

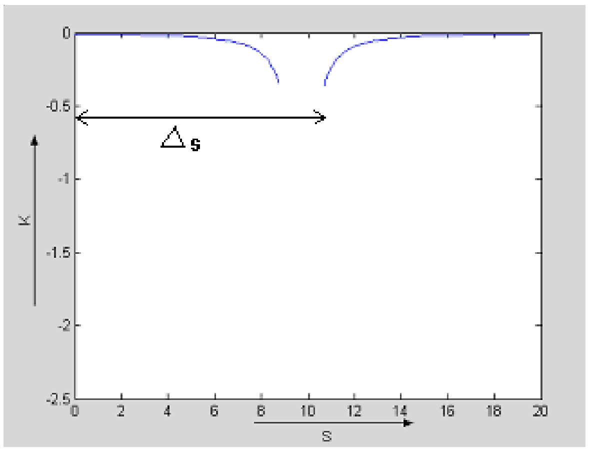


Fig. 10(b) k-s plot after shift for the data of Figure 9

Step 5: To calculate the shift Δs, a quadratic curve is fitted to both data sets of Part 1 and Part 2 (Figure 9) using least square method. The required shift is the arc length of the curve fitted between $x=x_1$ to $x=x_2$. Let this curve be $y=f_3(x)$. Now by integrating Eq. (33) from $x=x_1$ to $x=x_2$ for $y=f_3(x)$, the shift Δs can be found. After adding Δs to the *s* values of the Part 2 obtained in Step 3, new *k-s* values are obtained. Plot of these new *k-s* values are shown in Figure 10(c).

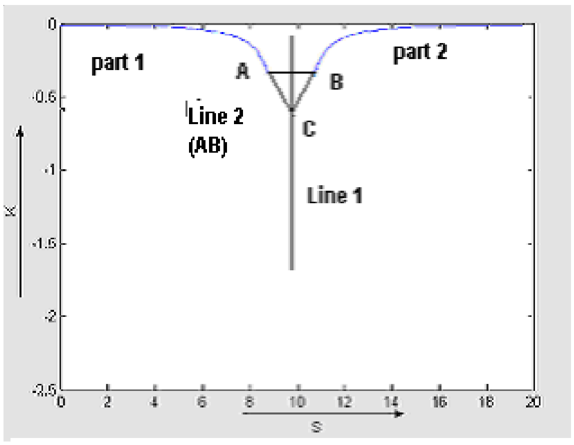


Fig. 10(c) New k-s values

Step 6: This step computes the values of *k* and *s* for the missing part of the cross-section shown in Figure 9 using the *k-s* estimated in step 5 (Figure 10(c)). Once the correct values of the *k* and *s* are known for missing part, using LINCE model missing *x-y* values can be estimated.

The steps involved in the process are as follows:

(a) Join the points *A* and *B* as shown in Figure 10(c) by a line and then find out a line (Line 1) perpendicular to line *AB*. Assume that there exist point *C* on the Line 1 for which line segments *AC* and *BC* give the required values of *k-s* for the missing part of the curve.

(b) To search correct point *C* on Line1, binary search method is used which takes intersection of Line 1, Line 2 and intersection of Line1 and the tangent of part 1 at point A as boundary of the binary search. So the point *C* is searched between these boundary points and to check whether the obtained point is correct or not, each time *x* and *y* values from the *k-s* values are find out using the LINCE model (Figure 10(d)).

(c) The values of *x* and *y* obtained in step 6(b) for point *B* are compared to the corresponding value of *x* and *y* for the original data of *x-y*.

(d) If the distance between these two points is nearly zero the algorithm terminates giving the final *x-y* for the missing part (Figure 11).

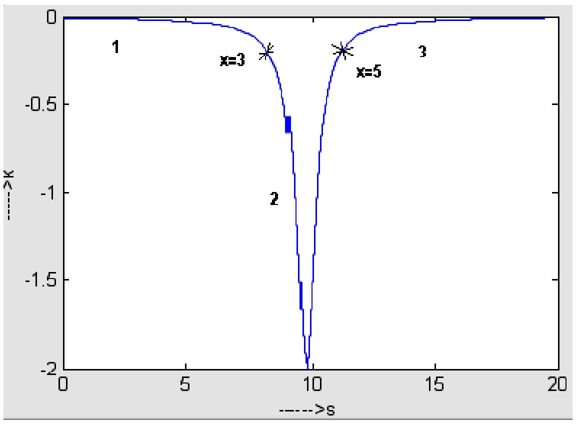


Fig. 10(d) Final k-s values

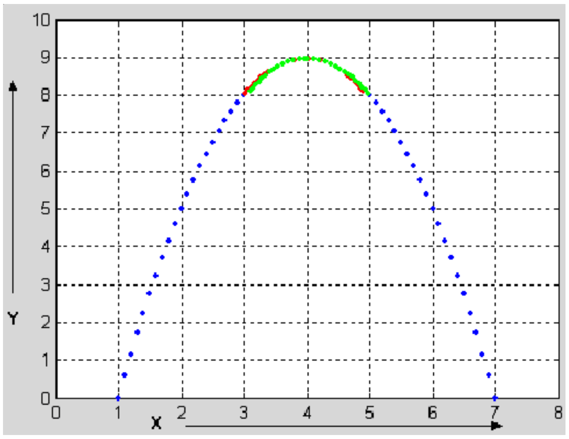


Fig. 11 Cross-section of Figure 9 after reconstruction from left (shown red) and right side (shown green)

Step 7: Repeat step 3 to 6 by taking point *B* (Figure 9) as starting point and estimate the missing point from the right side (Figure 10(d)).

Step 8: Calculate the area under the part PQ (Figure 9) using the missing data estimated from both left and right side. Also find out the change in angle from point *P* to *Q* (Figure 9).

Step 9: The data sets (estimated from left and right) for which the area under curve is close to the change in the angle is used as the final missing data of the curve (area under is equal to the change in angle).

5.2 Projection of healed 2D data points back to 3D

In this step 2D healed data obtained in the previous section are projected back to 3D. Consider the Eq. (31):

$$(x_1 - o\_i)\hat{i} + (y_1 - o\_j)\hat{j} + (z_1 - o\_k)\hat{k} = x_1'\overrightarrow{e_1} + y_1'\overrightarrow{e_2}$$

where (x_1, y_1, z_1) is the point on 3D and (x_1', y_1') is it's projection on 2D plane taking origin at (o_i, o_j, o_k).

If (e_1_i, e_1_j, e_1_k) and (e_2_i, e_2_j, e_2_k) are the *i*, *j* and *k* components of the vectors $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$ then (x_1, y_1, z_1) can be find as follows:

$$x_1 = o\_i + x * e_1\_i + y * e_2\_i\ ,$$

$$y_1 = o\_j + x * e_{1\_j} + y * e_2\_j\ ,$$

$$z_1 = o\_k + x * e_1 k + y * e_{2\_}k\ .$$

5.3 Getting the missed data in cross-sections

Algorithm for obtaining missed data in cross-sections:

```
FindMissed_xy( )
{
while (there is cross section left)
  {
ProjectTo2D (CrossSection (i);
ReadXYDataOfBrokenCrossSectionalCurve( i); /* x, y */
SeparateTwoParts (CrossSection(i) ); /* $x_1, y_1, x_2, y_2$ */
/* $x_1, y_1, x_2, y_2$ are known */
Get_ksForParts (CrossSection(i)); /* $k_1, s_1, k_1', s_2'$ */
FindShift & New_ksFor2ndPart( ); /* $k_2, s_2$ */
/* $S_{mid}, K_{mid}, S_{low}, K_{low}, S_{high}, K_{high}$ are known.
get xy_for_missed_part ($k_1, s_1, k_2, s_2, K_{mid}, S_{mid}$);
     while ( $y\_f(x_2(1)) \neq y_2$ )
     {  if ($y\_f(x_2(1)) > y_2$ )
     $K = (K_{mid} + K_{low})/2;$
     $S = (S_{mid} + S_{low})/2;$
     $K_{high} = K_{mid};$
     $S_{high} = S_{mid};$
     $K_{mid} = k;$
     $S_{mid} = s;$
```

```
    else
        $k = (K_{mid} + K_{high})/2;$
        $s = (S_{mid} + S_{high})/2;$
        $K_{low} = K_{mid};$
        $S_{low} = S_{mid};$
        $K_{mid} = k;$
        $S_{mid} = s;$
      end;
    }/*end-while*/
  get_xy_for_missed_part($k_1, s_1, k_2, s_2, K_{mid}, S_{mid}$);
  i++;
    }
  }
```

Let k and s values for the two parts are k_1, s_1, k_2 and s_2 and the calculated values for the missed part are k' and s'.

For a given curve in x-y, its arc length (s) and curvature (k) (i.e. k-s) plot can be obtained and vice-versa. In our approach the k-s values for a given broken curve are calculated and using these k-s values the k-s values for the missing part are calculated. Finally, the missing x-y is calculated using the obtained k-s values of the missing portion (Figure 12).

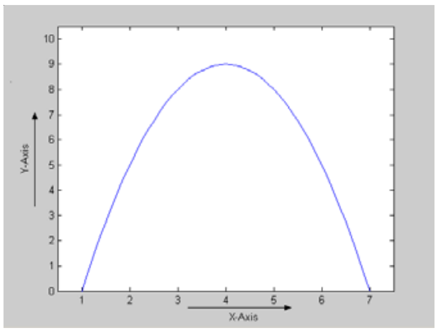


Fig. 12 Proposed curve after reconstruction

6. CASE STUDY

Step1: Step 1 takes the triangulated point cloud data file of an object and processes it for finding the holes. After finishing this step, all holes in the data are identified. Figure 13 shows an object and detected hole in it.

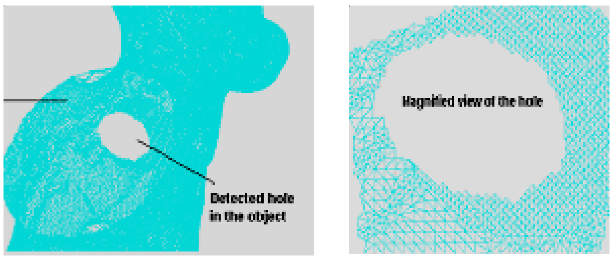


Fig. 13 Detected hole in the object (left) and the magnified view of the hole and neighboring data of the hole taken into consideration (right)

Step2: After identifying the hole, each hole is processed separately for computing the cross-sections.

Step3: In this step 3D cross-sections are projected on the 2D plane. Figure 14(a) shows the projection of 3D cross-sections on x-y plane.

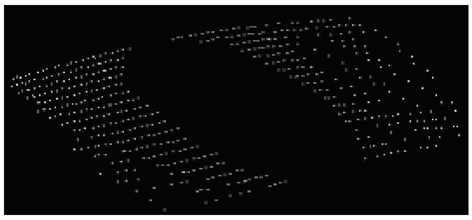


Fig. 14(a) x-y projected view of cross-sections

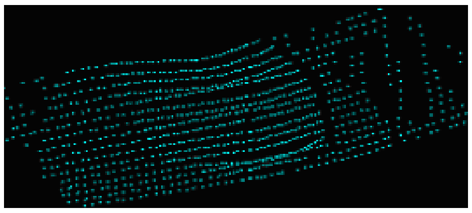


Fig. 14(b) Cross-sections after healing

Step4: Step 4 heals the individual 2D cross sections. The healed view of the cross-sections of Figure 14(a) is shown in Figure 14(b).

Step5: Healed data is projected back to 3D and then it is triangulated using Delaunay triangulation. Figure 15 shows triangulated point data of the hole (Figure 15(a)) and of the object surface (Figure 15(b)).

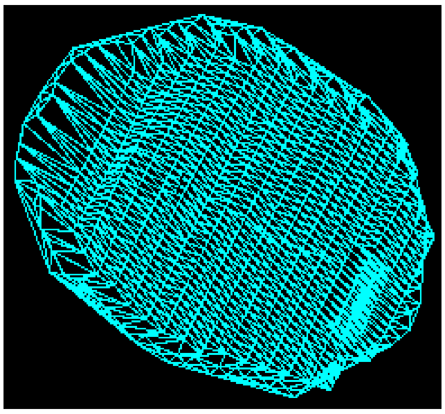


Fig. 15(a) Triangulated point cloud data of the hole

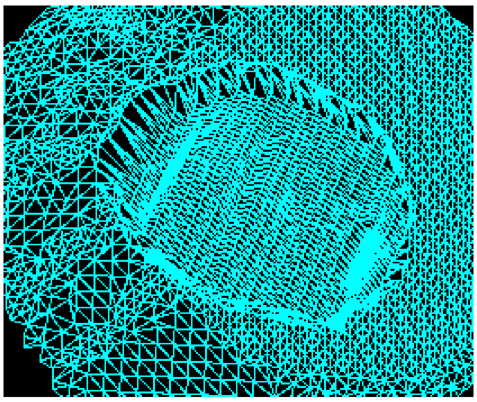


Fig. 15(b) Triangulated object surfaces

7. RESULTS

Figure 16 shows the result obtained using our algorithm. In Figure 16(a) a problem point cloud is shown, i.e. the object with a hole, and Figure 16(b) shows the point cloud after repairing it, i.e. after reconstruction.

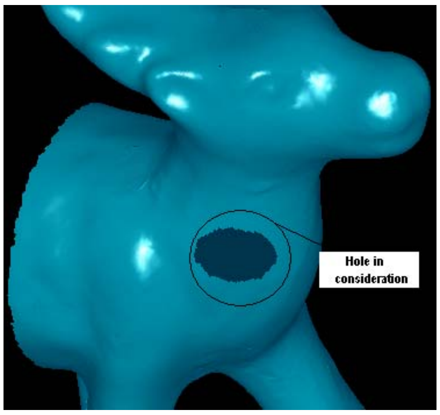


Fig. 16(a) Object with a hole (before reconstruction)

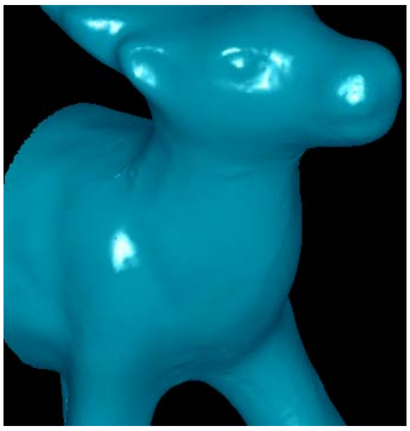


Fig. 16(b) After reconstruction

8. REFERENCES

[1] S. Tavakkoli and S.G. Dhande, Shape synthesis and optimization using intrinsic geometry, *Journal of Mechanical Design*, Vol. 113, pp. 379-386, 1991.

[2] B. Curless and M. Levoy, A volumetric method for building complex models from range images, Proc. 23rd Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH' 96), August 1996., New Orleans, ACM, pp. 303-312, 1996.

[3] B. Curless, New method for surface reconstruction from range images, Ph.D. Thesis, Computer Science Dept., Stanford University, 1997.

[4] W. Matusik, Ch. Buehler, R. Raskar, S.J. Gortler and L. McMillan, Image based visual-hulls, Proc. 27th Int. Conf. on Computer Graphics (SIGGRAPH 2000), July 2000., New Orleans, ACM, pp. 369-374, 2000.

[5] J. Davis, S.R. Marschner, M. Garr and M. Levoy, Filling holes in complex surfaces using volumetric diffusion, Proc. 1st Int. Symposium on 3D Data Processing, Visualization and Transmission, June 2002., Padua, IEEE Computer Society, pp. 428-438, 2002.

[6] J.A. Adams, The intrinsic method for curve definition, *Computer-Aided Design*, Vol. 7, No. 4, pp. 243-249, 1975.

[7] J. Venkatesh, Shape optimization using intrinsic geometry and boundary elements method, Ph.D. Thesis, Department of Mechanical Engng., IIT-Kanpur, 1994.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, Surface reconstruction from unorganized points, Proc. 19th Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH' 92), Ed. J.J. Thomas, ACM, pp. 71-78, 1992.

[9] H.Q. Dinh, G. Slabaugh, and G. Turk, Reconstructing surfaces by antistropic basic functions, Proc. Int. Conf. on Computer Vision (ICCV 2001), July 2001., Vancouver, pp. 606-613, 2001.

[10] N. Amenta, M.W. Bern and M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, Proc. 25th Int. Conf. on Computer Graphics (SIGGRAPH' 98), July 1998., Orlando, ACM, pp. 415-421, 1998.

[11] L. Fang and D.C. Gossard, Reconstruction of smoth parametric surfaces from unorganized data points, Proc. SPIE, Vol. 1830, Curves and Surfaces in Computer Vision and Graphics III, Ed. J.D. Warren, pp. 226-236, 1992.

[12] J. Wu and L.P. Kobbelt, Fast mesh decimation by multiple-choice techniques, Proc. Int. Conf. on Vision, Modeling and Visulization (VMV 2002), November 2002., Erlangen, Ed. G. Greiner, Aka GmbH, pp. 241-248, 2002.

[13] J.C. Carr, R.K. Beatson, B.C. McCallum, W.R. Fright, T.J. McLennan and T.J. Michell, Smooth surface reconstruction from noisy range data, Proc. 1st Int. Conf. on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE 2003), February 2003., Melbourne, Eds. M. Adcock, I. Gwilt and Y.T. Lee, ACM, pp. 119-126, 2003.

[14] H.K. Zhao and S. Osher, Visualization analysis and shape reconstruction of unorganized data set, In: *Geometric Level Set Method in Imaging, Vision and Graphics*, Eds. S. Osher and N. Paragios, Springer, Ch. 19, pp. 361-380, 2002.

[15] S. Muraki, Volumetric shape description of range data using "Blobby Model", Proc. 18th Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH' 91), Ed. J.J. Thomas, ACM, pp. 227-235, 1991.

METODE ZA DOBIVANJE GLATKIH POVRŠINA U 2D/3D REKONSTRUKCIJI POVRŠINA

SAŽETAK

Rekonstrukcija površina proizlazi iz istraživanja površina iz područja računalno potpomognutog projektiranja i proizvodnje. Postoje različite metode / algoritmi koje sasvim dobro razmatraju ovaj problem, ali se ne može reći da daju rješenje u svim situacijama. Nedostajajuća površina se može popraviti ili krpanjem površine ili pak produljenjem rubne krivulje. Međutim, u oba slučaja nameće se problem zaglađivanja tih površina. Ovaj članak će pokušati ponuditi rješenje gornjeg problema odnosno kako zagladiti krivulju / plohu.

***Ključne riječi**:* *razvlačenje, glatka površina, produljenje krivulje, kut zakrivljenosti, LINCE model, stl format1.*