Direct Slicing of a Point Set Model for Rapid Prototyping

Hayong Shin, Seyoun Park and Eonjin Park

Dept of Industrial Engineering, KAIST, hyshin@kaist.ac.kr

ABSTRACT

Recently point set model is getting increasing research attention in many geometric modeling application areas including computer graphics and CAD/CAM. This paper presents a novel approach to directly slicing point set model with the focus on making rapid prototyping part out of point set model without making any mesh or surface. Main challenge in handling point set model lies in how to interpret inter-point empty space and implicit quadric surfel is used in this research. This paper also explains how to utilize the quadric surfel for slicing the point set, so as to obtain contour curves for RP. Also described in this paper is how to extract smooth curve(s) out of the 2D point cloud obtained by slicing the 3D point set model.

Keywords: point set model, rapid prototyping, reverse engineering, surfel, slicing

1. INTRODUCTION

Thanks to the recent advancement in 3D shape scanning technology, 3D point set model can be easily obtained. Point set, also often called point-cloud, is one of the most fundamental shape representation methods, and is having increasing attention, mainly due to its simplicity – very simple geometry (only points) and no topology. The main challenge in dealing with point set model is how to fill in the gap between sampled points. Traditional ways are to construct triangular mesh connecting the points or to fit smooth surface passing the point set. However, it is by no means a simple task to construct such global models (either triangular mesh or surface fitting) with high fidelity.

In computer graphics area, recently, a number of research results have been published on rendering and hierarchically organizing point set for faster display of shape without hassling with complex geometry or connectivity [3][9][10]. Recent trend in handling point set model for graphics purpose is to locally interpret the gap in-between points, such as surfel [17] or moving least square [3].

As depicted in [13], RE (reverse engineering) and RP (rapid prototyping) are two important technologies to connect the physical world and the digital world bi-directionally. While RE process is to construct a computational geometric model from a physical object, RP process is to make a physical model from a digital model. Most of contemporary RP technologies use layered manufacturing, which is to accumulate thin slices in order to build a 3D object. For more detailed information on RP technology, the readers are referred to [20]. RE process needs to be combined directly with RP process in some application areas, for example, such as making a copy or a scaled model of a physical part. Fig.1 shows a typical RE-RP combined process. As one can see on the right side of the figure, many troublesome steps in the combined process can be avoided if the point set model can be directly sliced to create contours for RP. In this paper, we present a high fidelity direct point set slicing algorithm.

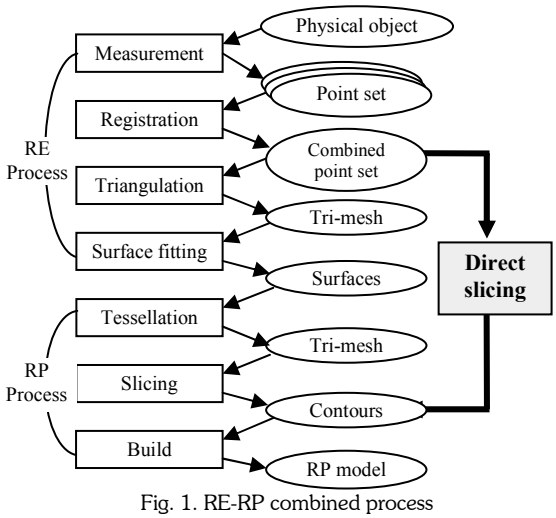


Fig. 1. RE-RP combined process

[10] proposed visualization and tool path generation algorithms for point-set based CAD/CAM. But, his

definition of point set is restricted to have regular grid type arrangement. [12] and more recently [16] presented methods for tool path generation from point set model. However, both papers assume that input point set is organized as a series of measured point sequence curves, which is a valid assumption when using contact-type CMM (coordinate measuring machine) or line type laser scanner. Though, if area type scanner is to be used, the assumption will not be satisfied. In this paper, we do not make any assumption on the underlying scanning method and the structure of point set.

[11], [13] and [19] are very closely related to this paper. In [11], they pre-processes the point set to obtain vertical feature curves first, then slice the feature curves to obtain slice contours. In their approach, the number of vertical feature curves depends on user's choice and so the accuracy of the resulting contour curve does. [13] takes a different approach. They first subdivide the point set into a series of slabs, and then select representative feature points by thinning the projected image. Then the feature points are connected together to form a net of feature curves. Finally, contour curves are obtained by cutting the curve net with slicing plane. [19], another paper by the same research group, explains a very similar approach, but with the focus on how to determine the slice thickness adaptively within given tolerance.

In this paper, we present a direct slicing algorithm to obtain contour curves with high fidelity. The next section explains the overall procedure, followed by details for each step, then examples and conclusion.

2. OVERALL PROCEDURE

As mentioned earlier, most of RP technologies utilize layered manufacturing, which is to make very thin layers and accumulate them [20]. The best building direction and layer thickness depend on the underlying technology, and hence here we assume that the building direction, which is perpendicular to the slicing plane, and the slice interval (layer thickness) is given as input. We further assume that the building direction is Z-axis, for the simplicity. Fig. 2 shows the overall flow of the algorithm.

3. PREPROCESSING

In this step, the points are first sorted along Z-axis for efficient proximity query processing. For even faster retrieval of neighboring point, voxel binning method (also known as bucketing) can be applied.

The main task of this step is to construct a local surface element (surfel) for each point, which approximates the neighborhood of the point. Surfels will be used for computing the cross section in the next section. Though the simplest type of surfel would be a plane (i.e. tangent plane), we use a quadric surface so as to achieve high fidelity for the given point set. For a given point $\mathbf{p} \in \mathbf{S}$ where $\mathbf{S}$ is the input point set, a quadric surface $\mathbf{Q}(\mathbf{p})$ is computed by weighted least square fitting of its local neighborhood $\mathbf{N}(\mathbf{p})$, which is a set of points around $\mathbf{p}$. The quality of resulting surfel heavily depends on how we choose $\mathbf{N}(\mathbf{p})$. If $\mathbf{N}(\mathbf{p})$ is too small, the surfels become sensitive to noise. On the other hand, if $\mathbf{N}(\mathbf{p})$ is too big, then small features can be eroded away. Hence it is difficult task to select appropriate neighbor size. In this paper, we simply set $\mathbf{N}(\mathbf{p})$ as the m-nearest neighbors, (and used m = 20 for the examples).

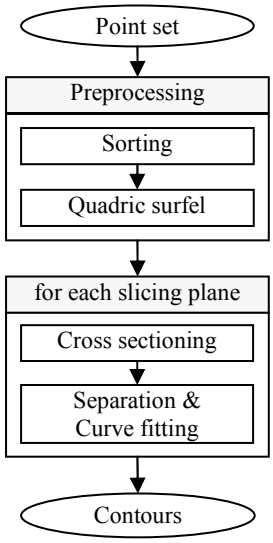


Fig. 2. Overall procedure

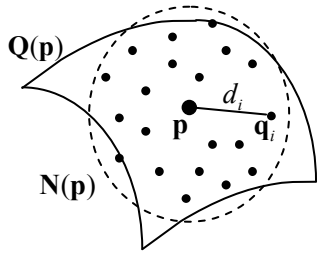


Fig. 3. Neighborhood $\mathbf{N}(\mathbf{p})$ and Surfel $\mathbf{Q}(\mathbf{p})$

Once $\mathbf{N}(\mathbf{p})$ is identified, computing $\mathbf{Q}(\mathbf{p})$ is more or less straightforward. A generic quadric surface can be represented in an implicit form as :

$$ax^2 + by^2 + cz^2 + 2(fxy + gyz + hzx + ux + vy + wz) + d = 0 \quad (1)$$

By letting d = -1, the above equation is normalized and can be rewritten in a matrix form as below :

$$e(x,y,z) = \mathbf{ac} - 1 = 0, \text{where}$$
$$\mathbf{a} = \begin{bmatrix} x^2 & y^2 & z^2 & 2xy & 2yz & 2zx & 2x & 2y & 2z \end{bmatrix}, \quad (2)$$
$$\mathbf{c} = \begin{bmatrix} a & b & c & f & g & h & u & v & w \end{bmatrix}^{\mathrm{T}}$$

Let $\mathbf{N}(\mathbf{p}) = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$. Then, the standard least square formulation for quadric surface fitting problem is:

$$\min E = \sum_{\mathbf{q}_i \in \mathbf{N}(\mathbf{p})} e(\mathbf{q}_i)^2 = \mathbf{e}^{\mathrm{T}}\mathbf{e}, \quad (3)$$

where

$\mathbf{q}_i = (x_i, y_i, z_i)$: neighbor point,

$\mathbf{e} = \begin{bmatrix} e(\mathbf{q}_1) & e(\mathbf{q}_2) & \cdots & e(\mathbf{q}_m) \end{bmatrix}^{\mathrm{T}} = \mathbf{Ac} - \mathbf{b}$: error vector

$$\mathbf{a}_i = \begin{bmatrix} x_i^2 & y_i^2 & z_i^2 & 2x_iy_i & 2y_iz_i & 2z_ix_i & 2x_i & 2y_i & 2z_i \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_m \end{bmatrix}, \text{ and } \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

And the solution can be easily obtained by differentiating the objective function in equation (3) with respect to the coefficient vector $\mathbf{c}$:

$$\mathbf{c} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b} \quad (4)$$

If we further consider that the impact of each point $\mathbf{q}_i$ should be different according to its distance from $\mathbf{p}$, we can introduce the weight factor w_i for each point $\mathbf{q}_i$:

$$w_i = \exp(-\frac{d_i^2}{\sigma^2}) \quad (5)$$

, where d_i = dist (p, $\mathbf{q}_i$) and σ is the attenuation factor. We can use the standard deviation of d_i's for σ. After taking the weights into account, the error vector $\mathbf{e}$ in equation (3) becomes :

$$\mathbf{e} = \begin{bmatrix} w_1 e(\mathbf{q}_1) & w_2 e(\mathbf{q}_2) & \cdots & w_m e(\mathbf{q}_m) \end{bmatrix}^{\mathrm{T}} = \mathbf{W}(\mathbf{Ac} - \mathbf{b}), \quad (6)$$

where $\mathbf{W} =$ diagonal $(w_1, w_2, \cdots, w_m)$.

And the solution to the weighted least square problem is similar to equation (4) with a little change :

$$\mathbf{c} = (\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{b} \quad (7)$$

Once the quadric surfel is computed for each point, the surfel (i.e. the coefficients $\mathbf{c}$) are stored along with the point, so that it can be used in the subsequent step.

4. CROSS SECTIONING

Since virtually no point in general is exactly located on a given slicing plane $\boldsymbol{\pi}$ ($z = k$), we need more information from the proximate points, and hence next step is to project the nearby points onto $\boldsymbol{\pi}$. Wu et al [19] used perpendicular projection, which may result in wide-band 2D image as shown in Fig. 4. The band width becomes larger as the slope of the object surface gets closer to slicing plane.

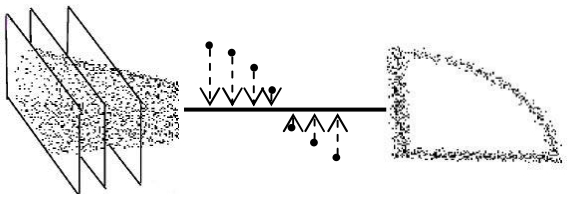


Fig. 4. Perpendicular projection in [19]

To overcome the above mentioned problem, we introduce '*voting along the surfel*' approach, which is illustrated in Fig. 5. For a point $\mathbf{p}$ closed to $\boldsymbol{\pi}$, the *voting plane* $\boldsymbol{\tau}$ is defined as the plane passing p and spanned by two vectors : surfel normal vector $\mathbf{n}_\mathbf{p}$ and z-axis vector. (Note that $\mathbf{n}_\mathbf{p}$ can easily obtained by the gradient of the quadric surfel $\mathbf{Q}(\mathbf{p})$. Also note that $\boldsymbol{\tau}$ is perpendicular to $\boldsymbol{\pi}$.) If $\mathbf{n}_\mathbf{p}$ is nearly parallel to z-axis (i.e. $\mathbf{p}$ has almost horizontal surfel), $\mathbf{p}$ is discarded from the projection. The intersection curve $\mathbf{C}(\mathbf{p})$ between $\boldsymbol{\tau}$ and $\mathbf{Q}(\mathbf{p})$ is called *voting curve*. In other words, the point $\mathbf{p}$ moves along $\mathbf{C}(\mathbf{p})$ toward $\boldsymbol{\pi}$ and makes a vote $\mathbf{v}_\mathbf{p}$ on $\boldsymbol{\pi}$. $\mathbf{C}(\mathbf{p})$ is a conic section curve and may have 0, 1, or 2 intersection(s) with $\boldsymbol{\pi}$. If no intersection is found, p does not participate in voting. If 2 intersections are found, the closer one from $\mathbf{p}$ is selected as $\mathbf{v}_\mathbf{p}$. To obtain $\mathbf{C}(\mathbf{p})$ efficiently, we can use coordinate transformation. By denoting a point in homogeneous coordinate, the equation (2) can be rewritten in a matrix form :

$$\mathbf{q}^{\mathrm{T}}\mathbf{A}\mathbf{q} = 0, \quad (8)$$

where $\mathbf{q} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^{\mathrm{T}}$

and $\mathbf{A} = \begin{bmatrix} a & f & h & u \\ f & b & g & v \\ h & g & c & w \\ u & v & w & -1 \end{bmatrix}$.

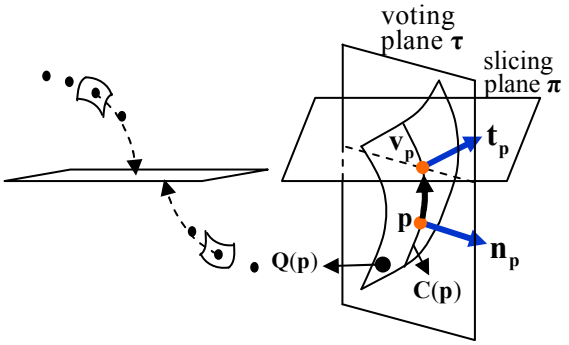


Fig. 5. Voting along the surfel

Now consider a local coordinate system L with the origin at p, z-axis vector $\mathbf{Z} = \nabla\mathbf{Q}(\mathbf{p}) / |\nabla\mathbf{Q}(\mathbf{p})|$, y-axis vector $\mathbf{Y}$ being the unit normal vector of the voting plane $\boldsymbol{\tau}$, and x-axis vector $\mathbf{X} = \mathbf{Y}\times\mathbf{Z}$. Then, the 4x4 coordinate transform matrix $\mathbf{T}$ is defined as :

$$\mathbf{T} = \begin{bmatrix} \mathbf{X} & \mathbf{Y} & \mathbf{Z} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (9)$$

where $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, and $\mathbf{p}$ are all 3x1 column vectors in global coordinate frame (See Fig. 6). Hence the conversion between $\mathbf{q}$ (in global coordinate frame) and $\mathbf{q}_L$ (in local coordinate frame L) is $\mathbf{q} = \mathbf{T}\mathbf{q}_L$. By substituting this into (8), we get :

$$\mathbf{q}_L^{\mathrm{T}}\mathbf{A}'\mathbf{q}_L = 0, \text{ where } \mathbf{A}' = \mathbf{T}^{\mathrm{T}}\mathbf{A}\mathbf{T}. \quad (10)$$

Now, we have the quadric surfel equation $\mathbf{Q}_L(\mathbf{p})$ in the local coordinate frame L, and $\mathbf{C}_L(\mathbf{p})$ is obtained by setting $y=0$ (the voting plane $\boldsymbol{\tau}_L$) in $\mathbf{Q}_L(\mathbf{p})$, which yields a quadratic equation of x and z. By applying the coordinate transformation to the slicing plane $\boldsymbol{\pi} : z = k$, $\boldsymbol{\pi}_L$ becomes the plane $\mathbf{m}^{\mathrm{T}}\mathbf{x}_L = k - \mathbf{p}_z$, where $\mathbf{m} = (\mathbf{X}_z, \mathbf{Y}_z, \mathbf{Z}_z)^{\mathrm{T}}$. is the normal vector of $\boldsymbol{\pi}_L$ and $\mathbf{x}_L$ is any point in L. Finally, the voting point $\mathbf{v}_\mathbf{p}$ is obtained by intersecting $\mathbf{Q}_L(\mathbf{p})$ with $\boldsymbol{\pi}_L$, which yields a simple quadratic equation.

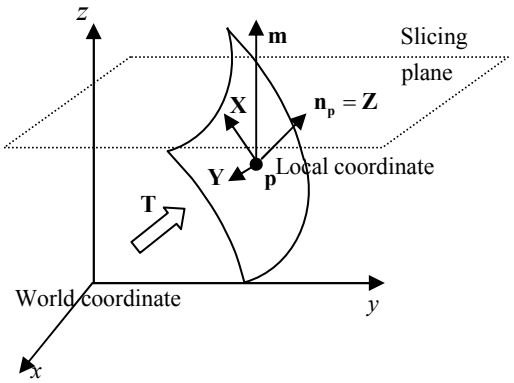
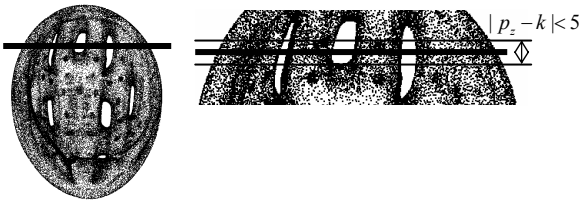


Fig. 6. Local coordinate transformation

For each vote $\mathbf{v}_\mathbf{p}$ on $\boldsymbol{\pi}$, we store additional information : the weight $\alpha_\mathbf{p}$ and the tangent vector $\mathbf{t}_\mathbf{p}$. The weight $\alpha_\mathbf{p}$ represents the importance of the vote according to the distance between $\mathbf{p}$ and $\mathbf{v}_\mathbf{p}$, and is computed by exponential function, which is similar to equation (5) :

$$\alpha_\mathbf{p} = \exp\left(-\frac{d_\mathbf{p}^2}{\sigma^2}\right), \quad (11)$$

where $d_\mathbf{p}$ = dist ($\mathbf{p}$, $\mathbf{v}_\mathbf{p}$) and σ is the standard deviation of $d_\mathbf{p}$'s. The tangent vector $\mathbf{t}_\mathbf{p}$ is obtained by cross product of $\nabla\mathbf{Q}(\mathbf{v}_\mathbf{p})$ with z-axis vector (the normal of slicing plane). The 3-tuple $V_\mathbf{p} = (\mathbf{v}_\mathbf{p}, \mathbf{t}_\mathbf{p}, \alpha_\mathbf{p})$ is called the ***vote*** by $\mathbf{p}$, and each element in the 3-tuple plays an important role in the contour curve fitting to be explained in the next section.

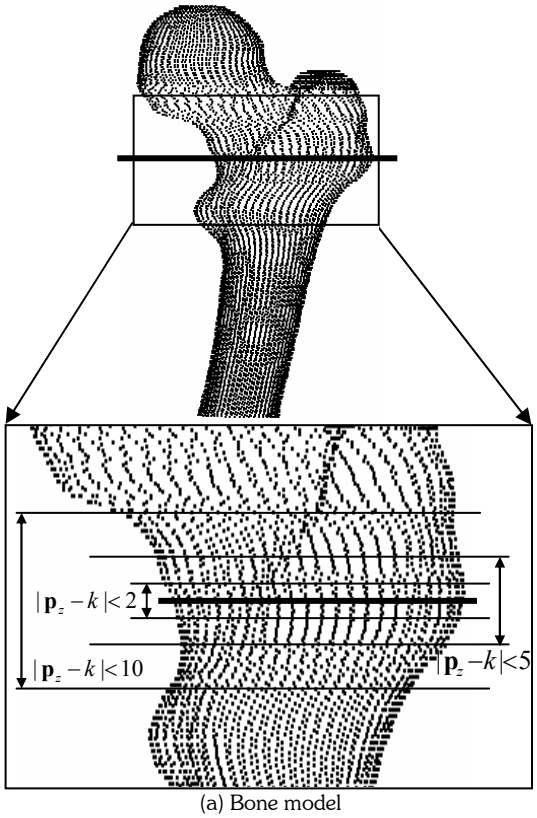


(a) Helmet model

(b) Perpendicular projection

(c) Voting along the surfel

Fig. 7. Comparison of two projection methods



(a) Bone model

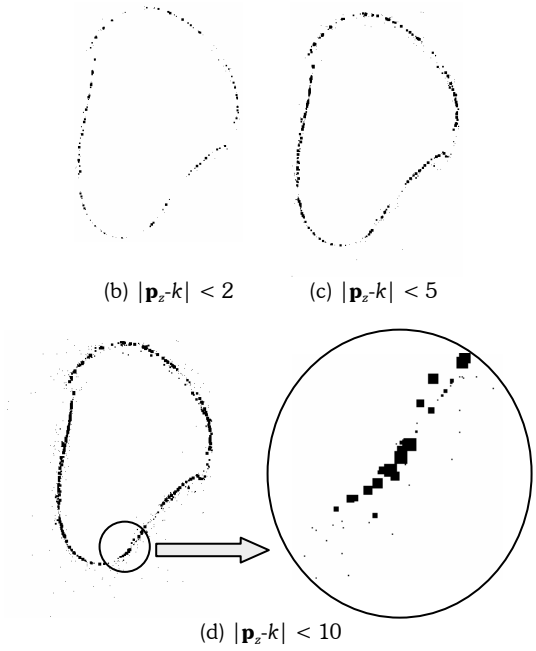


Fig. 8. Effect of voting participants range

As $\alpha_{\mathbf{p}}$ decreases exponentially, we can restrict the voting participants for a given slicing plane by considering the distance between the point and the plane. For example, 3 times of layer thickness works well. If we consider too many points for a slice plane, not only it takes much computation time, but also the resulting votes have undesired noise since our voting model is based on the surfel obtained by local approximation of neighborhood. Fig.7 shows the comparison of the two projection methods : Fig.7(b) is the resulting 2D image obtained by perpendicular projection approach taken by [19], while Fig.7(c) is the result of our approach, which is much more concentrated than (b). Fig.8 exemplifies the effect of enlarging the range of voting participants. (Note that the point set model used in Fig.8 is very sparse.) Though we can see many outliers in Fig.8(c) and (d), they have very small weights due to the distance from the original points to their votes. (Note that points are displayed with relative size proportional to their weights.)

5. SEPARATION AND CURVE FITTING

Once the voting is done for a given slicing plane, the next step is to construct contour curves out of the voting results. Since the resulting 2D image may consists of more than one contours as shown in Fig. 9, we should first separate the votes into groups, each of which belongs to a contour curve. [2] proposed a powerful algorithm called *crust* for separating and curve reconstruction. However, their method cannot be directly applied to this problem since their model does not take into account the weight factor, which is very important information in our application in that the resulting contour curve should be more influenced by the vote from closer points.

For the separation purpose in this research, we assume that the point set is dense and so is the vote set on a given slicing plane. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be vote points that should belong to different contour curves. Then, it is assumed that dist ($\mathbf{v}_1$, $\mathbf{v}_2$) > r, where r is the user specified separation tolerance value.

We first construct Euclidean minimum spanning tree (EMST for short) of votes on the plane. This can be done in O(n *log* n) time, where n is the number of vote points [15]. Or approximate EMST can be constructed with sub-linear time complexity [6]. Then we examine each edges of the EMST, and cut all edges longer than r, resulting in possibly several connected components, $\{G_i, i=1..g\}$ where G_i is a set of connected votes with edge length smaller than r. Each G_i serves as a candidate group of votes for a contour curve. G_i is discarded if G_i is too small either in its geometric size or in its importance. Geometric size of G_i can be measured by the radius of the bounding circle compared to the user specified minimum feature size. The importance of G_i is determined by the sum of weight $\alpha_{\mathbf{p}}$ of all votes in G_i, since G_i with small weight sum means that G_i consists of less important votes.

As for extracting a smooth curve approximating G_i, we first make a polygonal approximation using EMST information and tangent vectors of the votes in G_i. The initial polygon is constructed by connecting the votes with higher weights in the order suggested by the tangent vector $\mathbf{t}_{\mathbf{p}}$ of the votes in G_i. Then the current polygon is checked against the remaining votes to see if any important votes are far from the polygon by the combined measure of distance to the polygon and the weight. The polygon is refined to incorporate those important leftover votes until all votes are within tolerance in terms of the combined measure. If a smooth curve model is required, then the final polygon is used for assigning parameters for the votes in G_i, the weighted least square approximation method is applied to G_i so as to get a B-spline curve [18].

6. EXAMPLE

Fig.9 shows a bike helmet model consisting of 54000 points. In order to compute quadric surfel for a point $\mathbf{p}$, we chose 20-closest points as $\mathbf{N}(\mathbf{p})$. The helmet model was sliced at 100 layers. For each layer, points within ± 3 mm fro the slicing plane participated in voting. It took about 10 minutes with Pentium P4 computer, about 80 seconds of which was consumed in surfel computation requiring a lot of proximity query. With more sophisticated hierarchical structure, we believe the surfel

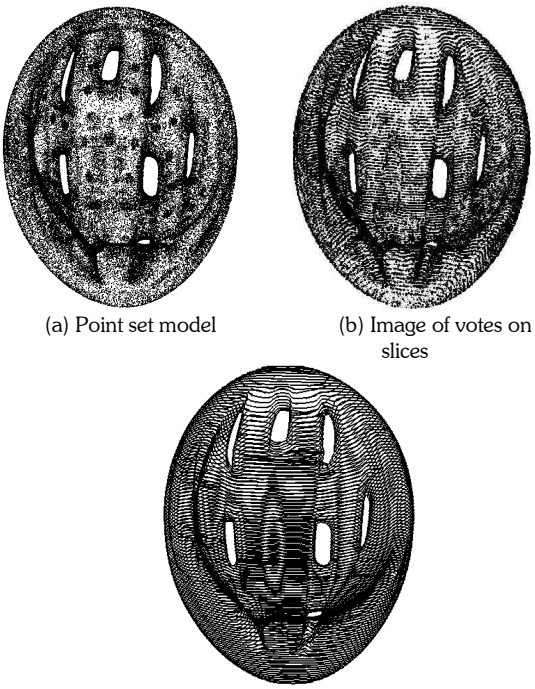
computation can be much improved. In the current implementation, most of the computation time is spent on curve separation step, which has a lot of room to be improved in terms of computational efficiency.

7. CONCLUDING REMARKS

In this paper, we have focused on slicing point set model directly for RP application. Creating contour curves for RP from point set allows us to bypass unnecessary intermediate models when RP is to be combined with RE. Contour curves with high fidelity makes more sense when thick-layer RP is to be used as in [1] and [8]. While thin layer RP technologies makes and accumulates vertically-sided layers as shown in Fig.10 (a), it is critical for thick layer RPs to make either slanted or curved sides for each layer in order to reduce the shape deviation (see Fig.10 (b) and (c)). The algorithm proposed in this paper generates quality contour curves efficiently. Nevertheless, the algorithm still needs to be further refined in that there are some parameter values to be determined in ad-hoc manner, such as neighborhood size for surfel computation, weight attenuation factor for voting, and curve separation tolerance. This paper assumes smooth (G^1-continuous) objects. In order to apply the approach presented in this paper to the objects with sharp edges, the point set should be segmented first so that the surfel can be constructed from the points in the same segment.

Finding a smooth curve approximating unorganized point data is not a trivial task. Main difficulty in using B-spline curve model for approximation is to determine parameterization for each data point. Well-known methods such as chord length parameterization or centripetal parameterization can be applied only after the point sequence is determined, which is the hardest part to be implemented in a robust manner. In the section 5, we only explained the curve fitting method implemented in this research, however, there are many other alternative avenues. Using radial basis function [4] is one of the strong candidates, because parameterization is not required. Another alternative is to use tensor voting framework [14].

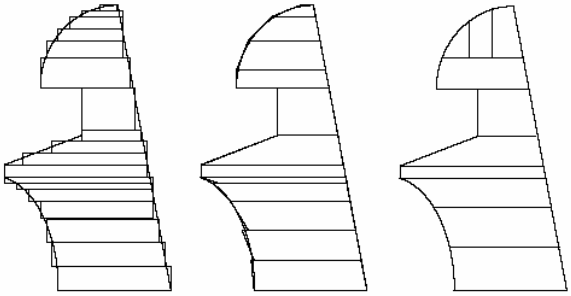
The application area of point based technique is ever expanding not only in computer graphics area but also in manufacturing industry. For example, reverse engineering for car design could also take benefit from point based technique in computing cross section curves and characteristic curves directly from the point data scanned from the clay model.



(a) Point set model (b) Image of votes on slices

(c) Curves obtained from (b)

Fig.9. Cycle Helmet example



(a) vertically-sided layers (b) layers with slanted sides (c) layers with curved sides

Fig.10. Comparison of layer side shapes [7]

ACKNOWLEDGEMENT

The research was supported in part by Korean Government through a NRL Grant.

REFERENCES

[1] DG Ahn, SH Lee, DY Yang, "Development of transfer type variable lamination manufacturing" International Journal of Machine Tools & Manufacture, 42(14), pp.1577-1587, 2002.

[2] N Amenta, M Bern, D Eppstein, “The crust and the β-skeleton : combinatorial curve reconstruction”,

Graphical Models and Image Processing, 60(2), pp.125-135, 1998.

[3] M Alexa, J Behr, D Cohen-Or, S Fleishman, D Levin, C Silva, "Computing and rendering point set surfaces", Proceedings of IEEE Visualization 2001, pp.21-28, 2001.

[4] JC Carr, RK Beatson, JB Cherrie, TJ Mitchell, WR Fright, BC McCallum, TR Evans, "Reconstruction and representation of 3D objects with radial basis functions", Proceedings of ACM SIGGRAPH 2001, pp.67-76, 2001.

[5] RJ Cripps, "Algorithms to support point-based CADCAM", International Journal of Machine Tools and Manufacture, to appear in 2003 (Article in press)

[6] A Czumaj, F Ergun, L Fortnow, A Magen, I Newman, R Rubinfeld, C Sohler, "Sublinear-time approximation of Euclidean minimum spanning tree", Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms, pp.813-822, 2003.

[7] I Horvath, J Vergeest, Z Rusak, Z Kovacs, G Kuczogi, "Building very large complex shapes using a flexible blade cutter", Machining Impossible Shapes (Proceedings of SSM'98 conference), USA, 1998.

[8] I Horvath, J Vergeest, JJ Broek, Z Rusak, B Smit, "Tool profile and tool path calculation for free-form thick-layered fabrication", Computer-Aided Design, 30(14), pp.1097-1110, 1998.

[9] O Kreylos, KL Ma, B Hamann, "A multi-resoultion interactive previewer for volumetric data on arbitrary meshes".

[10] J Krivanek, "Representing and rendering surfaces with points"

[11] KH Lee, H Woo, "Use of reverse engineering method for rapid product development", Computers in Industrial Engineering, 35(1), pp.21-24, 1998

[12] A Lin, HT Liu, "Automatic generation of NC cutter path from massive data points", Computer-Aided Design, 30(1), pp.77-90, 1998

[13] GH Liu, YS Wong, YF Zhang, HT Loh, "Error-based segmentation of cloud data for direct rapid prototyping", Computer-Aided Design, 35(7), pp.633-645, 2003

[14] G Medioni, MS Lee, CK Tang, A Computational Framework for Segmentation and Grouping, Elsevier, 2000.

[15] J O'Rourke, Computational Geometry in C, pp.188-190, Cambridge Univ. Press, 1993.

[16] S Park, YC Chung, "Tool path generation from measured data", Computer-Aided Design, 35(5), pp.467-475, 2003.

[17] H Pfister, M Zwicker, J Baar, M Cross, "Surfels : Surface elements as rendering primitives", Proceedings of SIGGRAPH 2000, pp.335-342, 2000.

[18] L Piegl, W Tiller, The NURBS Book, pp.410-419, Springer, 1995.

[19] YF Wu, YS Wong, HT Loh, YF Zhang, "Modeling cloud data using an adaptive slicing approach", Computer-Aided Design, 36(3), pp.231-240, 2004

[20] X Yan, P Gu, "A review of rapid prototyping technologies and systems", Computer-Aided Design, 28(4), pp.307-318, 1996.