Regulating complex geometries using layered depth-normal images for rapid prototyping and manufacturing

Yong Chen
University of Southern California, Los Angeles, California, USA, and
Charlie C.L. Wang
The Chinese University of Hong Kong, Hong Kong, China

Abstract
Purpose – Most layer-based rapid prototyping systems use polygonal models as input. In addition, the input polygonal models need to be manifold and water-tight; otherwise the built objects may have defects or the building process may fail in some cases. This paper aims to present a regulation method of an arbitrarily complex polygonal model for rapid prototyping and manufacturing applications.

Design/methodology/approach – The method is based on a semi-implicit representation of a solid model named the layered depth-normal images (LDNI), which sparsely encodes the shape boundary of a polygonal model in three orthogonal directions. In the method, input polygonal models or parametric equations are first converted into LDNI models. A regulation operator based on the computed LDNI models is presented. A volume tiling technique is developed for very complex geometries and high accuracy requirements. From the processed LDNI model, an adaptive contouring method is presented to construct a cell representation that includes both uniform and octree cells. Finally, two-manifold and water-tight polygonal mesh surfaces are constructed from the cell representation.

Findings – The LDNI-based mesh regulation operation can be robust due to its simplicity. The accuracy of the generated regulated models can be controlled by setting LDNI pixel width. Parallel computing techniques can be employed to accelerate the computation in the LDNI-based method. Experimental results on various CAD models demonstrate the effectiveness and efficiency of our approach for complex geometries.

Research limitations/implications – The input polygonal model is assumed to be closed in our method. The regulated polygonal model based on our method may have a big file size.

Originality/value – A novel mesh regulation method is presented in this paper. The method is suitable for rapid prototyping and manufacturing applications by achieving a balance between simplicity, robustness, accuracy, speed and scalability. This research contributes to the additive manufacturing development by providing a digital data preparation method and related tools.

Keywords Mesh regulation, CAD preparation, Geometry processing, Ray representation, Self-intersection removing, Computer aided design, Rapid prototypes

Paper type Research paper

1. Introduction
Layer-based rapid prototyping and manufacturing (RP/RM) processes such as stereolithography apparatus (SLA) and selective laser sintering (SLS) can fabricate parts directly from computer-aided design (CAD) models without part-specific tooling or fixtures. A de facto standard for RP machines’ data input is the STL file format, in which, a set of triangular facets are used to approximate the shape of a part. The input STL files for RP systems are supposed to be manifold and water-tight without self-intersections. However, it is difficult for both RP users and system developers to enforce such a requirement since the syntax of a STL file only requires a set of triangles are defined with their normals. Figure 1 shows a simple 2D example, in which a set of meshes are shown in Figure 1(a). The STL model is obviously invalid since the input meshes have various self-intersections. Based on the defined orientations associated with the mesh normals, a STL model that is manifold and water-tight is shown in Figure 1(b) for defining the related 2D shape. In this paper, we call such a geometric operation mesh regulation, in which an arbitrary input polygonal model with defined surface orientation (i.e. normals) is converted into in a STL file that is manifold and water-tight without self-intersections.

The built objects may have undesired defects, or even worse, the building process may fail when using RP systems to build an invalid STL model. An illustration example based on the SLA process is shown in Figure 2. An input STL model
Figure 1 A 2D illustration of mesh regulation for RP applications

Before regulation: STL has complex self-intersections

After regulation: STL is manifold and water-tight

Notes: (a) Input model; (b) regulated model

(Figure 2(a)) contains both a scanned lower jaw and a set of teeth that intersect the jaw. As shown in the wireframe view, the input polygonal model is invalid since self-intersected triangles exist between the teeth and the jaw. To build a physical object using SLA machines, the SLA part preparation software, Lightyear 1.5 from 3D Systems Inc. (Rock Hill, SC) can be used in creating supports for the input model. One of the identified support regions and related supports are shown in Figure 2(a) (bottom). Note that Lightyear will search all the triangles in the input STL model. Hence the identified support region will include self-intersected triangles with related supports generated. However, such supports are actually not needed since the built object will be solid inside with the self-intersected triangles removed. In comparison, a regularized polygonal model based on our method is shown in Figure 2(b). As shown in the wireframe view, the newly constructed model is valid without self-intersections. Consequently, the support region identified by Lightyear for the same geometric feature is comparably smaller, which lead to less supports and consequently less undesired marks on the bottom surfaces.

Figure 2 An example of mesh regulation for RP applications

The test case will be used throughout the paper to illustrate the major steps of our regulation method.

An input STL model may define 3D geometry with self-intersections that are far more complex than the 2D example as shown in Figure 1. An illustration example of such STL models is shown in Figure 3. In the test, a structure configuration based on a given model of a Beethoven statue was generated using a microstructure taken from Chen (2007b). A sphere and a cylinder model were added at each joint and strut of the structure, respectively, (Figure 3(a)). Since all the spheres and cylinders were simply added without Boolean operations, the constructed STL model is invalid with complex self-intersections. A regularized polygonal model based on our method is shown in Figure 3(c). The regulated polygonal model is now valid with all the self-intersections removed. We tested the validity of the regulated model by building it in a SLA machine. The fabricated SLA part is shown in the middle of Figure 3(d). A split version of the statue with the designed complex internal structures is also shown in the figure. Note that a significant benefit of the layer-based RP processes is its capability of cost-effectively fabricating truly complex 3D shapes that were previously impossible. To fully utilize such a capability, a general and robust regulation method needs to be developed to ensure an arbitrarily complex polygonal model can be built by RP systems. More examples are shown in Section 6.

The mesh regulation considered in the paper is different from Boolean operations (Wang et al., 2010) in that intersections might not only happen between different objects, but also within one object. The approach of directly computing shell intersection to remove self-intersected triangles in an object is prone to computation error and hence not robust. In our previous work on uniform offsetting (Chen and Wang, 2011), a volumetric approach has been developed to compute offset boundary from an input polygonal model and an offset distance. The robustness of the developed offsetting approach motivated us to extend the idea into the mesh regulation operation for RP/RM applications. A general and robust mesh
regulation approach is presented, in which, a semi-implicit representation – layered depth-normal images (LDNI) is first used to represent the solid shape defined by an input STL file. The LDNI model can then be regulated based on a LDNI-based regulating operation (Figure 3(b)). Accordingly a valid STL model can be constructed from the regulated LDNI model (Figure 3(c)). One potential drawback of our method is that it is an approximation approach since the LDNI representation is a discrete representation of the exact geometry defined by the input polygonal model. However, as shown in the paper, the approximation error can be well controlled. For RP/RM applications, the LDNI sampling rate can be set to a level that is comparable to the fabrication capability of a RP machine. A volume tiling technique and a parallel implementation are also presented to address the needs of fine sampling resolutions and large STL files associated with complex geometries.

1.1 Related work

The boundary representation (B-rep) is the most popular representation of 3D geometry for computer-aided design and manufacturing (CAD/CAM) applications. Though accurate, the geometric operations based on the boundary representation lack simplicity and are prone to robustness problems (Chen and Wang, 2011). Instead of precisely defined geometry, volumetric representations to approximate geometry have also been studied (Hoffmann, 2001; Lorensen and Cline, 1987; Kobbelt et al., 2001; Ju et al., 2002). Various volumetric representations have been proposed such as voxel (Kim et al., 2003), distance field (Kaufman et al., 1993), surfel (Jones et al., 2006), and ray-rep (Pauly et al., 2003). Geometric operations based on such volumetric representations are robust and easy to implement. However, its use in CAD/CAM applications needs to address the general concern over its inaccuracy. Compared to computer graphics applications, most engineering applications have a much higher accuracy requirement. The computation framework introduced in this paper is also based on a volumetric representation; however, the LDNI representation is sparser (like a sparse matrix) so that it is efficient in memory cost. Based on the LDNI representation, the accuracy requirement for RP applications is also addressed in the paper.

Since many CAD/CAM systems adopt B-rep as the input to describe the shape of a solid, it is important to have an algorithm to generate two-manifold polygonal mesh surface (B-rep) from a volumetrically represented solid model. Extensive research has been done in this area. The marching cubes algorithm, proposed by Lorensen and Cline (1987), is a standard approach to extract an iso-surface from a volume raster of scalar values. Many extensions to the original marching cubes algorithm have been proposed to resolve ambiguities of certain cell configurations and generate topologically consistent iso-surfaces (Jones et al., 2006). The original marching cubes algorithm is unable to extract high quality triangle meshes with sharp features. Several extensions have also been proposed to reconstruct sharp features and reduce aliasing artifacts in the reconstructed model (Kobbelt et al., 2001; Ju et al., 2002). To overcome an excessively large number of triangles required to represent the iso-surface, many methods have also been developed for performing iso-surface extracting adaptively using hierarchies such as octrees and k-D trees (Ju et al., 2002; Varadhan et al., 2004; Frisken et al., 2000). Most algorithms do not provide guarantees on the topology of the reconstructed surface. Some recent work addresses topology-preserving reconstruction by an enhanced cell representation (Zhang et al., 2005) or by additional tests (Chen, 2007a).
Regulating complex geometries using layered depth-normal images

Yong Chen and Charlie C.L. Wang

The purpose of techniques presented in this paper is different from the point-sampled geometry approaches (Zhang et al., 2005; Gross and Pfister, 2007; Adams and Dutré, 2003), which focused on the interactive rendering. As aforementioned, the CAD/CAM applications such as CNC and RP planning need to have B-rep of solid models. Although we can generate B-rep from the surfels, the structural information of samples that can be used to speed up the solid modeling operations and the contouring process is missed. Moreover, the point-sample geometry does not give an efficient way to evaluate the inside/outside of a point related to the shape defined by the unstructured points. In a LDNI model, the surface information of solids is also encoded by a set of points coupled with normal vectors. However, the points in a LDNI are well organized in a data structure; hence the following solid modeling operations and contouring can be implemented easily and completed in an interactive speed.

1.2 Contributions

A novel solid regulation method is presented in this paper. It can be used by RP systems as a data pre-processor to handle invalid STL files that may come from different sources with various self-intersections. Our method has the following properties that are important for RP/RM applications:

- **Simplicity.** The LDNI-based regulation operation is straightforward and easy to implement.
- **Robustness.** Based on discrete sampling points, common robustness problems can be effectively addressed in the LDNIs-based regulation operation.
- **Accuracy.** The approximation error is bounded by the LDNI sampling resolution.
- **Scalability.** An input model can be sub-divided into multiple tiles and each tile is separately processed. This allows very complex geometry to be processed.
- **Speed.** Most steps in our method can be easily parallelized. Therefore, the computing time can be significantly reduced by using graphics hardware and a PC cluster.
- **Capability and efficiency in capturing sharp features.** Our contouring approach can capture sharp corners and edges in the geometry, which is important for engineering applications. In addition, adaptive sampling enables us to use a higher resolution to refine only the cells that have complex geometry inside. Therefore, a limited number of polygons can be produced according to a LDNI model in high resolution.

The remainder of the paper is organized as follows. The principle of ray-based regulating method is introduced in Section 2. The representation of LDNI is presented in Section 3. The construction of LDNI from a polygonal model related to its accuracy is also discussed in the section. The regulation operation based on the LDNI representation is discussed in Section 4. The adaptive sampling and contouring of a LDNI model are presented in Section 5. A parallel computing framework for the LDNI-based regulation method is also discussed in the section. The test results of the presented method are presented in Section 6. Finally, conclusions are drawn in Section 7.

2. Principle of the ray-casting-based regulating method

Ray-casting is a well established technique for finding whether a point lies inside or outside a polygon (Schneider and Eberly, 2003). Intersection points can be computed by following a ray from the test point through a polygonal model. If the given polygon is closed without self-intersection, the test on how many times the ray intersects the edges of the polygon can determine its inside/outside property. That is, if the test point is not on the boundary of the polygon, the number of intersections is an even number if the point is outside, and it is odd if inside. 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, then it alternately goes from the outside to inside, then from the inside to the outside, etc. However, if a given polygon has self-intersections, the number of intersections will not be able to determine its inside/outside property. For the purpose of regulation, we first define a normal index number of a ray as follows.

**Definition 1.** A normal index number \( I_{\text{Norm}} \) is an accumulated integer value along a ray with unit normal \( N_{\text{ray}} \) such that:

- \( I_{\text{Norm}} = 0 \) at the starting point if it is outside the model;
- \( I_{\text{Norm}} = 1 \) at the starting point if it is inside the model;
- from the starting point, for any intersection point \( P \) with unit normal \( N_p \) along the ray, increasing \( I_{\text{Norm}} \) by 1 if \( N_p \cdot N_{\text{ray}} < 0 \) and decreasing \( I_{\text{Norm}} \) by 1 if \( N_p \cdot N_{\text{ray}} > 0 \).

Figure 4 shows an illustration of the normal index numbers of an arbitrary ray in the X- and Y-axes in a 2D and 3D example. At each intersection point along the ray, the \( I_{\text{Norm}} \) value will change by 1 depending on whether its normal is the same or opposite the ray direction. The portions of the ray with \( I_{\text{Norm}} = 1 \) and \( I_{\text{Norm}} = 0 \) are shown in red and black, respectively, in the figure.

**Proposition 1.** For a ray sampled from a two-manifold solid model, the number of nodes on the ray should be even; in addition, \( I_{\text{Norm}} \) should be 0 at both \( d_{\text{min}} \) and \( d_{\text{max}} \) of a ray (i.e. outside the model).

**Proposition 2.** For a ray sampled from a two-manifold solid model, a point \( P \) on the ray is inside the model if \( I_{\text{Norm}}(P) > 0 \); otherwise, it is outside the model.

A two-manifold polygonal model may be invalid with self-intersections. An example is shown in Figure 5, in which, three common types of self-intersections are illustrated including:

1. loop twisting;
2. internal loop overlapping; and
3. external loop overlapping.

For an arbitrary ray, its \( I_{\text{Norm}} \) value along the ray can be computed based on Definition 1 (Figure 5(a)). Accordingly, as shown in Figure 5(b), the nodes whose two neighboring \( I_{\text{Norm}} \) values are 0 and 1, or 1 and 0 are shown in black dots. All the other sampling points are shown as red dots in the figure. After removing all the red dots, the \( I_{\text{Norm}} \) values along the ray are only 0 or 1. Figure 5(c) shows the portions of the ray with \( I_{\text{Norm}} = 1 \) and \( I_{\text{Norm}} = 0 \) in red and black, respectively. Accordingly the processed ray can be used in defining the boundary of...
Figure 4 An illustration of the normal index numbers of different rays

Notes: (a) A 2D example; (b) a 3D example

Proposition 3. A node $P$ along a ray is valid if and only if the values of $INorm$ before and after $P$ are 0 and 1, or 1 and 0.

Note that the input polygonal model is assumed to be closed in this paper. However, models especially the ones generated by 3D scanning technology may have gaps and holes. Nevertheless, it is easy to detect any gaps and holes in an arbitrary polygonal model based on the half-edge data structure (Schneider and Eberly, 2003). Accordingly the detected gaps and holes can be filled based on a sealing approach presented before (Barequet and Sharir, 1993; Campen and Kobbelt, 2010) to construct a closed polygonal model. Note that the process of filling holes may also lead to self-intersections in the constructed polygonal model.

A regulated model in which all the self-intersections have been removed (Figure 5(d)).

Figure 5 An illustration of removing self-intersections based on $INorm$ values along the ray

Notes: (a) Input boundary; (b) sampled boundary points; (c) processed boundary point; (d) reconstructed boundary

An example based on the method in Barequet and Sharir (1993) is shown in Figure 6. For an input model as shown in Figure 6(a), additional triangle facets can be automatically generated to fill the holes. However, as shown in Figure 6(b), the added triangles will introduce self-intersections in the sealed model. Our solid regulation method is beneficial for gap/hole filling methods by automatically removing defects in input polygonal models and constructing valid CAD models for RP systems (Figure 6(c)).

Based on the ray casting approach of removing invalid points along a ray, a systematic approach and related representation are required in order to process an arbitrary 3D solid with various self-intersections. For this purpose, a LDNI representation is first presented in Section 3. Based on such a representation, a solid regulation operation is then presented in Section 4.
3. Layered depth-normal images

Stimulated by the layered depth images (LDI) representation (Shade et al., 1998) that was originally introduced as an efficient image-based rendering technique, we have developed the sparse implicit representation, LDNI, for Boolean and offsetting operations (Wang et al., 2010; Chen and Wang, 2011). We extended the LDNI representation for the mesh regulation operation in which an accurate surface normal is recorded for each sampling point such that the normal index number as discussed in Section 2 can be exactly computed along a ray.

3.1 LDNI representation

Following our work in Wang et al. (2010), a LDNI model is defined as follows to implicitly encode the shape of a solid as a structured collection of Hermite data.

**Definition 2.** A single LDNI with a specified viewing direction \( k \) is a 2D image with \( w_i \times w_j \) pixels, where axes \( i \), \( j \), \( k \) are orthogonal to each other. Each pixel of a LDNI contains a sequence of numbers that specify the distances from the intersections to the viewing plane and the unit normal vector of the sampled surface at the intersection point. Furthermore, all the depths of a pixel are sorted in the ascending order.

**Remark 1.** The intersections in a LDNI model exclude the case that a ray is parallel to the intersected faces (i.e. \( N_p \cdot N_{ray} = 0 \)).

**Remark 2.** An edge is defined as silhouette-edge if only one of its adjacent polygons faces along the current viewing direction. When a ray intersects an edge shared by two faces, no intersection will be counted if this edge is a silhouette-edge and one intersection will be sampled for the non-silhouette-edges. For a non-silhouette-edge, the normal vector at either of its two adjacent faces will be selected and encoded.

**Definition 3.** A structured set of LDNI consists of \( x \)-LDNI, \( y \)-LDNI and \( z \)-LDNI, where \( x/y/z \)-LDNI is a LDNI viewed along the inversed direction of \( x/y/z \)-axis. The three images are located to let the intersections of their rays to form \( w_x \times w_y \times w_z \) nodes of uniform grids in \( \mathbb{R}^3 \).

**Notes:** (a) The model to be processed with open surfaces; (b) self-intersection will be generated after filling the holes by the minimal area triangulation method proposed in Barequet and Sharir (1993); (c) self-intersection has been removed by our regularization method.

Figure 6 The method can also be used to regularize models with open surfaces.
When using the LDNI construction method based on graphics hardware, the guarantee of this property is based on the implementation of rasterization on the hardware. Based on our experimental tests, modern graphics cards will automatically exclude faces that are parallel to the viewing direction; they can also correctly handle the silhouette-edges cases. That is, when a ray intersects a non-silhouette-edge, only one intersection is counted in the stencil buffer. The value in the stencil buffer will be increased by two if the ray passing the center of a pixel intersects a silhouette edge. Hence the sorted Hermite samples at each pixel in LDNIs will correctly represent the sampled solid along the ray passing this pixel’s center.

Proposition 4. For a correctly sampled solid model represented by LDNI, the number of sampled depths on a pixel must be even.

Hence, by capturing the boundary information of an input solid model along pre-defined uniform grids, the solid can be implicitly defined by a LDNI model. Based on highly parallel structures, modern graphics hardware is very efficient at displaying polygonal models. Consequently, a LDNI model can be constructed quickly from a polygonal model even though the model may have complex geometries. For example, by using a graphics card (NVIDIA GeForce 8800 GT), a LDNI model (in a resolution of 492 \( \times \) 350 \( \times \) 398) can be computed from the model as shown in Figure 2 within 2.2 s (x-LDNI is constructed in 0.9 s with 271,508 nodes generated; y-LDNI is constructed in 0.5 s with 133,040 nodes generated; and z-LDNI is constructed in 0.8 s with 213,796 nodes generated). The computed LDNI nodes are shown in Figure 8(b), in which, red, blue and green points are used to represent the x/y/z-LDNI models, respectively.

3.2 Pixel width and accuracy of LDNI

The accuracy of a LDNI model depends on the pixel width \( \delta \) used in the computing process. Suppose the bounding box of an input model is given as \( \text{Ext}_{\text{min}} \) and \( \text{Ext}_{\text{max}} \). If we use the graphics hardware to construct the corresponding LDNIs model, the minimum pixel width \( \delta = (\text{Ext}_{\text{max}} - \text{Ext}_{\text{min}})/(w - 1) \), where \( w \) is the maximum image resolution available in the rendering (e.g. \( w = 1,024 \) for the graphics hardware with a resolution of 1,280 \( \times \) 1,024). Depending on the available memory, the maximum image resolution can be much larger if the rendering is performed by a simulated frame buffer. However, the rendering speed in this case would be slower.

Proposition 5. For a LDNI with pixel width \( \delta \) in a sampling orientation \( O \), a gap or a thin-shell in the solid model whose thickness is less than \( \delta \) may be missed in the computed LDNI model, if such a gap or thin-shell is parallel to \( O \).

A LDNI model has three LDNI in three orthogonal sampling orientations. Hence, if a gap or a thin-shell is parallel to one orientation, the can be captured by the other two LDNIs in the model.

Proposition 6. For the LDNI with pixel width \( \delta \), a feature whose overall size is less than \( \delta \) may be missed in all three orthogonal LDNIs.

Therefore, if the minimum feature size in a polygonal model is \( \theta \), pixel width needs to be set such that \( \delta \leq \theta \) to ensure the LDNIs model can capture all the features that are bigger than \( \theta \) in the given model. In RP applications, the setting of \( \delta \) is affected by the design requirement and the capability of a RP machine that will be used in fabricating the model. Suppose the minimum manufacturable feature size of a RP process is \( \tau \). We can accordingly set \( \delta \leq \tau \) to capture
all the features in the model that can be fabricated. For example, in the layer-based additive manufacturing processes, the smallest feature size in Z-direction is the layer thickness (i.e. $\tau_z = \text{Thickness}_{\text{layer}}$).

**Remark 3.** For a CAD model and a manufacturing process with given feature size $\theta$ and $\tau$, pixel width $\delta$ can be set as $\max(\theta, \tau)$ in constructing a LDNI model to ensure the sampled points can capture all the manufacturable features in the CAD model for the manufacturing process.

### 3.3 Volume tiling of LDNI

For a solid model with high accuracy requirement, a technique of volume tiling can be used in computing the related LDNI model. Based on the computed bounding box of the input model, the model can be split into multiple smaller tiles. Each tile can then be processed independently (either sequentially or in parallel) with the related LDNI models being constructed individually. To use volume tiling, the changes required in the LDNI representation include:

- $\text{Ext}_{\text{LDNI min}}$ and $\text{Ext}_{\text{LDNI max}}$ that are recorded in each LDNI model may be different from the minimum and maximum extent of a given polygonal model.
- In each pixel $(i, j)$ of a LDNI, $\text{Ext}_{\text{LDNI min}}$ is defined as the starting point of the tile, which is then used in setting the normal index number $I_{\text{Norm}}$ at the point.

The approach described in Section 3.1 can be used in constructing the LDNI model of each tile. The only difference in the OpenGL display is that the input polygonal model is zoomed into a target $\text{Ext}_{\text{LDNI min}}$ and $\text{Ext}_{\text{LDNI max}}$. Note that the viewing plane is still set by $\text{Ext}_{\text{min}}$ since we only know $I_{\text{Norm}} = 0$ at $\text{Ext}_{\text{min}}$. Hence all the sampling points are still calculated in the rendering process; however, only the nodes that are inside the extent $[\text{Ext}_{\text{LDNI min}}, \text{Ext}_{\text{LDNI max}}]$ will be recorded in the constructed LDNI model.

A volume tiling example is shown in Figure 9. In the test, a simple ring model was used as the design template (Figure 9(a)). Based on it, a sphere model is constructed at each edge of the ring model. The merged STL file with all the spheres and cylinders has 2,825,264 triangles, which is shown in Figure 9(a). Suppose the model is divided into $2 \times 2 \times 1 = 4$ tiles. The LDNI model of each tile can be computed individually. The LDNI models of tile $(0, 0, 0)$ and tile $(1, 1, 0)$ are shown in Figure 9(b), which have 805,426 and 855,906 nodes, respectively.

As analyzed above, the memory complexity of a LDNI is in the quadratic order to its resolution. For complex geometry with high accuracy requirement, the required memory of a LDNI model can easily exceed the memory limitation of a commodity personal computer (PC). The volume tiling technique can significantly reduce the amount of information that is required to be stored simultaneously. Hence it can effectively address the memory problem in constructing and processing a LDNI model for complex geometry with high accuracy requirement. In addition, each tile can be processed separately and in parallel by a PC cluster.

### 4. Regulating a LDNI-based solid

After a LDNI model has been constructed from an input polygonal model, the regulation operation based on the approach discussed in Section 2 can be performed on the well-structured LDNI points.

#### 4.1 LDNI-based regulation operation

For a solid defined in the LDNI representation, the regulation operation is straightforward and easy to implement. As discussed in Section 3, a LDNI model consists of a set of well-organized 1D volumes defined on uniform grids with accurate depth and normal information. Accordingly, a point filter called a ray casting filter is designed as follows.

For a given LDNI model, we process its $x$-LDNI, $y$-LDNI and $z$-LDNI separately (sequential or in parallel). For each $x/y/z$-LDNI, we go through each pixel $(i, j)$ and sort all the points $P_i - P_j$ based on their depths. We then calculate $I_{\text{Norm}}$ based on Definition 1 for each line segment along the ray. Finally, the calculated $I_{\text{Norm}}$ can be used to delete all the inner points whose two neighboring $I_{\text{Norm1}}$ or $I_{\text{Norm2}}$ are not $(0, 1)$ or $(1, 0)$ from the LDNI model. Hence all the nodes that correspond to the self-intersections will be deleted and only the nodes that correspond to the boundary of a regulated solid will be stored in the processed LDNI model.

**Proposition 7.** A ray casting filter will remove an even number of nodes along a ray.

This is because the difference of $I_{\text{Norm}}(P')$ and $I_{\text{Norm2}}(P')$ for any given point $P$ along the ray is $1$ or $-1$. Hence the points that are neighboring to the line segments with $I_{\text{Norm}} > 1$ or $I_{\text{Norm}} < 0$ must be even.

**Notes:** (a) An input invalid polygonal model; (b) a constructed LDNI model including $x/y/z$-LDNI.
The regulation operation of an input LDNI model can be performed quickly since the judgment on a ray related to any pixel is simple. An example based on the input solid as shown in Figure 8 is shown in Figure 10. It takes a commodity PC less than 1 s to calculate the regulated LDNI model from the input model. It takes even less time if the computation is performed by the highly parallel graphics processing unit (GPU). As shown in Figure 10(b), the sampling points related to the self-intersections have been removed in the regularized LDNI model.

### 4.2 Robustness enhancement of the LDNI-based regulation operation

Two common robustness problems that need to be considered are numerical error and degenerate data. Numerical error occurs due to the use of floating-point arithmetic in geometric computations. Degenerate data such as tangential contact between various geometric elements can be produced by geometric operators even if the input models are manifold. Such robustness problems have been extensively studied in computational geometry. Their handling can be challenging for geometric operations based on B-rep.

In the LDNI based method, polygonal meshes that define the continuous geometric boundary are first converted into a set of discrete sampling points. Hence a geometric operation will be performed only based on these points. Comparing to the geometric elements considered in B-rep (i.e. face, edge, vertex), the geometric elements in our method (i.e. points) are significantly less, which dramatically simplifies the robustness problem. Our method for enhancing the robustness of the LDNI-based geometric operations is given as follows:

- **Numerical error.** In the LDNI representation, we use a quantization resolution \( \delta \) to convert the depth of a sampling point into an integer representation. For a maximum extent \( G \), a \( N \)-bit integer is needed where \( N = \log_2(G / \delta) \) to represent any floating-point number within \( G \). Therefore, an input geometry is embedded inside a fine integer lattice with size \( 2^N \) in each dimension and each intersection point can be clamped to the nearest lattice point. By choosing \( \omega \) as \( m \times \delta \) where \( m \) is an integer, we can also clamp the uniform grid to their nearest lattice points. Therefore, most computations on 1D volume can be performed based on exact integers.

- **Degenerate data.** A point filter, small segment filter, has been developed to remove tangential contacts between geometric elements. With a limited manufacturing resolution \( \beta \), a RP system will not be able to fabricate small gaps or thin shells that are smaller than \( \beta \). Suppose a set of sampling points \( P_1 \sim P_n \) has been calculated for a pixel \((i, j)\). The points can be sorted based on their depth values. Each pair of points \( P_i \) and \( P_{i+1} \) with opposite normals can be processed to identify the 1D volumes whose thicknesses are less than \( \varepsilon \) (e.g. \( \varepsilon = 10^{-5} \) is chosen in our implementation). All such pairs of points will then be removed from the resultant LDNI model. Hence the tangential contacts (small gaps or shells) in the given polygonal model will be corrected. An illustrative example of removing a degenerated gap and shell is shown in Figure 11(a) and (b), respectively. Note that regular gaps and shells whose sizes are bigger than \( \varepsilon \) will not be affected in the regulation operation.
Figure 11 An illustration of the small segment filter

Notes: (a) A degenerated gap was removed; (b) a degenerated shell was removed

5. Adaptive sampling and contouring for boundary representation

A regulated LDNI model is a semi-implicit representation of the related solid. For a manufacturing system that requires the boundary representation as the input, the LDNI model needs to be converted into a polygonal model. Therefore, methods are required to convert a LDNI model into a polygonal model. Furthermore, the generated model needs to be watertight without defects such as gaps, holes, or self-intersections.

5.1 An adaptive cell representation and contouring

For complex geometry with high accuracy requirement, the pixel width $d$ of a LDNI model is typically small. Consequently, contouring based on the uniformly sampled points defined in the LDNI model will lead to a very large number of small triangles. To be more efficient in constructing a polygonal model from a LDNI model, it is desired to have an adaptive contouring method. Hence, more triangles can be allocated in regions that have complex topology or high curvatures. Similar to our previous work (Chen, 2007a), an adaptive cell representation including both uniform and octree cells is used in adaptive contouring. The uniform cells are used for rough sampling and are first constructed from a LDNI model. Based on them, octree cells are used to refine each cell that has complex topology or geometry inside. Suppose the pixel width of a LDNI model is $d$. The uniform cell size can be set as $g = 2^k d$, where $k$ is the maximum subdivision number of an octree cell. Based on $g$ and the extents $\text{Ext}_{\text{LDNI}_{\text{min}}}$ and $\text{Ext}_{\text{LDNI}_{\text{max}}}$, the uniform cells can be calculated. A 2D illustration of a LDNI model and the two types of cells are shown in Figure 12 ($k$ is set at 3).

In the algorithm, there are two major cell queries, Corner_Sign_Query and Get_Sampling_points. Corner_Sign_Query determines the inside/outside sign of a cell corner. Get_Sampling_Points returns a set of sampling points $\mathbf{v}_i$ (both position and normal) inside a cell. Both of them can be easily implemented based on the LDNI representation since a cell corner must lie on a ray (Figure 12).

Adaptive sampling test

1. Calculate an error-minimizing point $\mathbf{v}_e$ based on the quadric error function (QEF) of all the points $\mathbf{v}_i$ ($i = 1, \ldots, n$) in $C_k$ (Ju et al., 2002). If no $\mathbf{v}_e$ is found, return failed.

2. Topological test of $\mathbf{v}_e$ to ensure the contour within each cell is topologically equal to a simple disk:
   - if $\mathbf{v}_e$ is outside $C_{k_0}$, return failed; and
   - if the distance $d_i$ from $\mathbf{v}_e$ to a plane defined by the position and normal of $\mathbf{v}_i$ ($i = 1, \ldots, n$), if $d_i > \epsilon$, return failed.

3. Save $\mathbf{v}_e$ as an error-minimizing point for $C_k$ and return succeeded.

In the algorithm, there are two major cell queries, Corner_Sign_Query and Get_Sampling_points. Corner_Sign_Query determines the inside/outside sign of a cell corner. Get_Sampling_Points returns a set of sampling points $\mathbf{v}_i$ (both position and normal) inside a cell. Both of them can be easily implemented based on the LDNI representation since a cell corner must lie on a ray (Figure 12).

In the adaptive sampling test, an error-minimizing point of a cell is calculated from all the sampling points in the cell from a LDNI model. The approximation error is computed and
compared with $\varepsilon$. If the approximation error is smaller than $\varepsilon$, the error-minimizing point will be used in the contouring process; otherwise, the cell will be subdivided. When the subdivision reaches the finest level, a cell size is the same as the pixel width $\delta$. By using the cell center to approximate the features inside the cell, the approximation error during the contouring will be less than $\delta$. By setting tolerance $\varepsilon \ll \delta$, we know.

**Proposition 8.** The polygonal model generated by the aforementioned adaptive sampling and contouring process has an approximation error that is less than the pixel width $\delta$.

After an adaptive cell representation is constructed from a given LDNI model, a modified dual contouring method can be used in reconstructing mesh surfaces (Wang et al., 2010). Unlike the marching cube algorithm, the dual contouring algorithm will not generate cracks for an adaptive grid with different grid sizes (Ju et al., 2002). To generate manifold-preserved mesh surfaces, two additional strategies were also presented in Wang and Chen (2008) for overcoming the topology ambiguity that may occur inside the finest octree cells after the maximum subdivision. Figure 13 shows an example of the computed surfaces based on a regulated LDNI model as shown in Figure 10.

### 5.2 Contouring of multiple LDNIs for volume tiling

As shown in Figure 9, an input model can be split into multiple smaller tiles and a LDNI model related to each tile can be individually computed. In order to construct the contour surface of each LDNI model separately, it is critical to ensure that the independently generated contouring meshes can be properly merged into a valid polygonal model. We achieve this by adding a buffer region to the right, back and bottom sides of each tile. The width of the buffer region is one uniform cell size. Therefore, two neighboring tiles will overlap in the buffer region. Figure 14(a) shows a 2D illustration of a tile $(i, j)$ and its buffer region, as well as the two neighboring tiles, $(i + 1, j)$ and $(i, j + 1)$. Tiles $(i + 1, j)$ and $(i, j + 1)$ overlap tile $(i, j)$ by one layer of uniform cells in the right and bottom sides, respectively.

Accordingly the boundary of a tile is defined as $[\text{Ext}_\text{LDNI}_{\min}, \text{Ext}_\text{LDNI}_{\max}]$. In the modified dual contouring method, each active edge of a cell is tested before a quad is constructed for it. If an edge is inside $[\text{Ext}_\text{LDNI}_{\min}, \text{Ext}_\text{LDNI}_{\max}]$, it is a valid edge. A corresponding quad and related triangles will be generated for it. Otherwise, the active edge is invalid and no triangles will be generated. Consequently, the polygonal meshes of two neighboring tiles can be ensured to have no duplicate triangles.

After the polygonal meshes in all the tiles have been generated, a polygonal model can be computed by simply merging all the meshes together. Since the buffer regions overlap two neighboring tiles and the calculated error-minimizing points in the buffer regions are used in constructing polygons of both tiles, the meshes in the two neighboring tiles have a common boundary. Consequently, by zipping the entire boundary together, the merged polygonal model would be watertight. An example of contouring multiple LDNI models is shown in Figure 14(b). The constructed polygonal meshes related to the four LDNI models as shown in Figure 9 are simply merged into one model. Magnified views of portions of two neighboring meshes are also shown in Figure 14(b). The merged polygonal model is valid with no holes or duplicate triangles.

### 5.3 Implementation of parallel computing for the LDNI-based regulation method

A significant advantage of the LDNI-based solid regulation method is the simplicity of parallelizing it. Figure 15 shows a parallel computing framework for the presented method. The computation in each step and the analysis of their performance are discussed as follows:

- In order to construct a LDNI model from a polygonal model, the sampling process goes through each polygon and uses the scan conversion algorithm to generate intersection points in three axes. Therefore, the time complexity of this step is $O(N_{\text{pol}})$ where $N_{\text{pol}}$ is the number of polygons. Based on highly parallel structure, the graphics hardware used in the sampling process can construct a LDNI model very quickly. In our current

---

*Figure 13 A contouring example of a LDNI model*

Notes: (a) Input LDNI model; (b) computed valid polygonal model
implementation, the construction of LDNI models can be accomplished in seconds. All the generated LDNI models are then saved in a network-connected hard drive that can be accessed from a cluster of PCs.

- The regulation of a LDNI model in a tile does not require the information of other tiles. Hence all the tiles can be processed in parallel using a PC cluster. If a PC cluster has a sufficient number of machines, the total running time is determined by that of the most complex tile. In the LDNI-based regulation operation, the sampling points related to each pixel are judged to filter invalid nodes. Hence the time complexity of this step is \( O(w^2) \) where \( w \) is the pixel resolution. The operation is fast, typically within a second in our tests.

- The time complexity of constructing a cell representation from a LDNIs representation is \( O(N_{\text{Node}} \cdot k) \) where \( N_{\text{Node}} \) is the number of nodes in a LDNIs model and \( k \) is the maximum subdivision times. It is the most time-consuming step in our current implementation and takes more than 60 percent of the total running time. Since the adaptive sampling test in constructing octree cells only requires the information related to the cells in the test, a PC cluster can be used in parallelizing the cell generation. The contouring algorithm spends \( O(1) \) time on each cell of the grid. Therefore, the time complexity of the contouring is \( O(N_{\text{Cell}}) \) where \( N_{\text{Cell}} \) is the number of cells (including both uniform and octree cells).

- It is trivial to merge the separately generated polygonal meshes of each tile into a polygonal model for RP systems. The time complexity of this step is \( O(N_{\text{Tri}}) \) where \( N_{\text{Tri}} \) is the triangle number of all the meshes. The operation is fast (within seconds).

6. Results and discussions

We have implemented the proposed regulation operation in a program using C++ programming language. Various experimental tests have been performed to verify the presented method. In addition to the test cases as shown in Figures 2, 3 and 9, four additional tests are discussed as follows.

In a test on complex surface textures, we used the same structural configuration and the ring model as shown in Figure 9. However, we flipped the normals of all the triangles related to the constructed spheres and cylinders. We then
merged all the triangles of the inverted spheres, cylinders and the original ring model. The merged polygonal model is shown in Figure 16(a). Obviously the model is invalid with various types of self-intersections. Based on the developed operation, the regularized model has 2,825,264 triangles, which is shown in Figure 16(b). The computed polygonal model is two-manifold without self-intersections. Similar to the test in Figure 9, our algorithm divided the input model into $2 \times 2 \times 1 = 4$ tiles in the test. The running time of each tile is given in Table I.

In a test on design with surface structures, we designed two spherical balls with different meso-structures on their surfaces. One meso-structure has zigzag beams that are similar to springs to make the related ball flexible. Another meso-structure has straight beams to make the related ball rigid. Based on such structure configurations, a cylinder model was added at each joint; a cube model was added at each strut. We then merged all the cylinders and cubes together. The merged polygonal models are shown in Figure 17(a) and (c), respectively. Note that multiple faces from the cylinders and cubes may overlap with each other. Such test cases would be challenging for the geometric operations based on the B-rep since the exact surface intersections are prone to numerical errors. Instead, the aforementioned LDNI-based operation was used in regularizing the invalid polygonal models. Such overlapping surfaces can be handled by the small segment filter. The resulted polygonal models are shown in Figure 17(b) and (c), respectively. The validity of the regulated models was verified by successfully building the related STL files using a SLS machine. Figure 17(c) shows the built physical models with different meso-structures. The computation time of two test cases is given in Table I.

In a test on regulating input models with small gaps, a gear model as shown in Figure 18(a) was used. As a well-known model in the RP community, the model can be directly fabricated without additional assembly operations. Considering the capability of a RP process, a sufficient tolerance is required between the gears and the frame such that the fabricated features will not be fused together (e.g. a gap of 0.33 mm for the SLS process). A LDNI pixel width $\delta$ can be set based on the fabrication capability of the selected RP process (e.g. 0.1 mm used in the test). As shown in Figure 18(b), the desired gaps remain intact in the regulated model. In addition, the gap size will remain the same in our method since the reconstructed surface is computed based on the LDNI points that are sampled on the input boundary surface with the floating point accuracy ($< 10^{-5}$). This is different from the voxel-based approaches, in which the input boundary surfaces are converted into a set of voxels with an approximation error of a voxel size.

The computational statistics of the presented test cases is given in Table I. All the tests are based on commodity PCs with an Intel Core2 Quad CPU Q9300 2.5 GHz and 4 GB DRAM running Windows 7. The computational statistics of the four major steps for multiple-tile cases is given in Table II.

A test on the approximation error made by the LDNI-based regulation method has been performed. Tow input models as shown in Figure 19(a) and (b) are generated by using the commercial software package ACIS R15 (www.spatial.com),
Figure 17 Test result of a flexible and rigid ball

Notes: (a) An input model that is designed to be flexible; (b) the regularized polygonal model; (c) an input model that is designed to be rigid; (d) the regularized polygonal model; (e) the physical models fabricated by the SLS process

Figure 18 Test result of a gear model with small gaps

Notes: (a) An input model with three magnified views (i), (ii) and (iii); (b) the regularized polygonal model with the same magnified views; note the rotation gaps remain in the regulated model
respectively. In comparison, the regulated models generated by our method based on a pixel width of 0.005 are also shown in the figure. A publicly available mesh comparison tool, Metro (Cignoni et al., 1998), was used to compare the two pairs of models. Compared to the input polygonal models, the surface errors (Hausdorff distance) of the regulated models are shown as follows; both the maximum surface distance $E_{\text{max}}$ and the mean surface distance $E_{\text{mean}}$ are reported. It can be seen from the mesh comparison results that the regulated models are close approximations of the input models with an error that is much smaller than the LDNI pixel width. Note that although different triangular facets have been used in defining the same shape, identical physical objects will be fabricated as long as the approximation errors are smaller than the related manufacturing variations:

- chair – sphere: $E_{\text{max}} = 2.1 \times 10^{-3}$, $E_{\text{mean}} = 1.2 \times 10^{-4}$; and
- chair $\cup$ sphere: $E_{\text{max}} = 1.6 \times 10^{-3}$, $E_{\text{mean}} = 1.3 \times 10^{-4}$.

7. Conclusion and future work

We have presented a novel LDNI-based solid regulation method that can handle a polygonal model with complex geometries. Our approach is volumetric and hierarchical. It achieves a balance between various requirements of CAD/CAM applications. Compared to other existing implicit representations of solid models, the encoded data of a LDNI model is well structured and sparse. A volume tiling technique can significantly reduce the memory requirement for processing the input model with a high accuracy requirement. Many steps of the presented method can run in parallel. Hence the computing time can be significantly reduced by using a PC cluster and graphics hardware. For CAD/CAM features that are bigger than the sampling resolution, the polygonal model generated by our approach is topologically equivalent to the exact surface and has a bounded error specified by an error tolerance. Our contouring algorithm can capture sharp features in the constructed model. The experimental results on a variety of test cases have verified the effectiveness and efficiency of our method.

Some future work includes:
- investigating the use of GPU in implementing all the steps of the method such that the computation can be finished in an interactive speed;
- exploring the use of the presented method in more CAD/CAM applications; and
• testing the direct fabrication of a LDNI model using a RP system without constructing a polygonal model as the input.

References


