Proceedings of DETC'03
ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference
September 2-6, 2003 Chicago, Illinois, USA

DETC2003/DAC-48726

TOWARD AUTOMATIC PROCESS PLANNING OF A MULTI-AXIS HYBRID LASER AIDED MANUFACTURING SYSTEM: SKELETON-BASED OFFSET EDGE GENERATION

Kunnayut Eiamsa-ard
E-mail: kezp9@umr.edu

F.W. Liou[*]
E-mail: liou@umr.edu

Robert G. Landers
E-mail: landersr@umr.edu

Department of Mechanical and Aerospace Engineering and Engineering Mechanics
Intelligent Systems Center, University of Missouri, Rolla, MO 65409

Howie Choset
E-mail: choset@cs.cmu.edu
Department of Mechanical Engineering, Sensor-Based Path Planning Lab
Carnegie Mellon University, Pittsburgh, PA 15213

Keywords: Skeleton, Voronoi, Offset, and Path Planning

ABSTRACT

Even though the machining process has been integrated to the Multi-Axis Laser Aided Manufacturing Process (LAMP) System in order to get good surface finish functional parts [1], the quality of parts produced by the LAMP system is still very much dependent upon the choice of deposition paths. [2] Raster motion paths are replaced by offset spiral-like paths, which are discussed in this paper. Most commercial CAD/CAM packages are feature-based, and their use requires the effort and expertise of the user. The shape has to be decomposed into manufacturing features before the software packages can generate the paths. [3] Path planning has long been studied as discussed in this paper. There are still some problems associated with the previous algorithms and also assumptions are usually made. [6, 7, 27] An algorithm for directly generating offset edges, which can be developed to be the deposition paths, is presented in this paper. The skeleton of a layer or a slice of a 3-D CAD drawing is first generated. Based on that skeleton, the offset edges are incrementally constructed. This paper focuses on the characteristics of skeleton and offset edges as well as the construction algorithm for those edges. Simulations are used to verify this method.

1. INTRODUCTION

LAMP is a Layered Manufacturing Process (LM), which combines multi-axis deposition and milling processes shown in Figure 1. STL files generated from 3-D CAD models are used as the inputs to the LAMP System Processor. The use of a support structure is reduced by adding two rotation movements: 4th and 5th axis to the LAMP hybrid system. Thus, the hybrid system possesses the advantages of both the adding and subtracting processes as well as those of multi-axis movements. Complex parts can be built through the multi-axis deposition process and also the surface finish can be achieved by the 5-axis milling process.

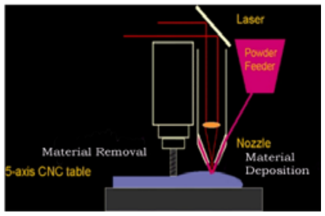

Figure 1: 5-axis LAMP System

STL files generated from 3-D CAD drawings are sliced by a multi-axis adaptive slicing algorithm described in the previous work done by Zhang and Liou. [4] Figure 2 shows slices generated by Zhang's adaptive slicing algorithm. (As can be seen in Figure 2, those slices are embedded in 3D-space. None of the available commercial software would be able to

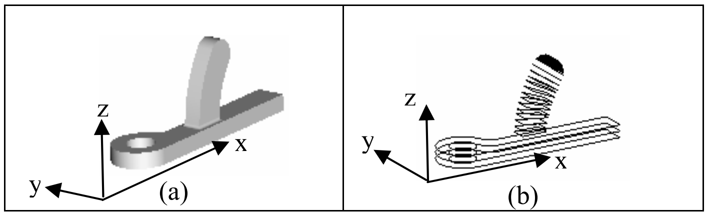

Figure 2: (a) 3-D Drawing (b) Slices of the STL File [1]

[*] Address all correspondence to this author

automatically generate the 3D deposition paths and plan the combined processes.) Each slice or layer is then fed back to the LAMP processor in order to generate paths for deposition as well as the milling paths. It is obvious that the problem to generate the deposition paths for those slices is not a 2D problem. However, the problem to generate the deposition paths for each slice can be a 2D problem once an auxiliary X-Y plane is embedded for each layer. This paper focuses only on generating paths for deposition for only one layer.

There are two main types of coverage path patterns: Zigzag and Contour patterns as shown in Figure 3. [5] In the Zigzag or Raster pattern, the tool or the tip of the nozzle (in this case) is moved back and forth parallel to referenced directions. In the Contour or Offset Pattern, offset segments of the boundaries are generated and used as guides for the nozzle to move along.

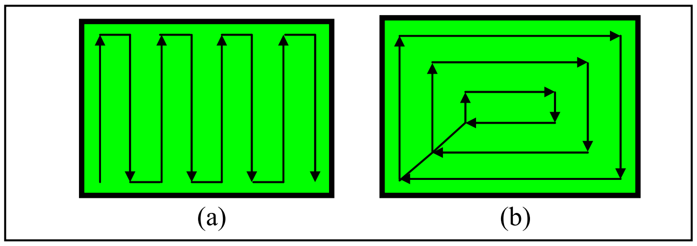

Figure 3: Coverage Path Patterns (a) Raster (b) Offset

Based on the work performed by Kao and Prinz [2], spiral offset paths are typically preferred for producing isotropic deposits. By tracing the offset paths, residual stress could be relieved before tracing the next adjacent edges as opposed to the raster paths. Therefore, stress-induced warping is reduced. A Contour or Offset Pattern is, thus, selected for the hybrid LAMP system processor to construct the deposition paths. This paper describes the characteristics of the skeleton and offset edges, and procedures to numerically generate skeletons as well as the coverage paths. The algorithm starts with generating the skeleton for a layer sliced from a 3-D CAD model. The skeleton is then used to construct the offset paths.

There are many algorithms for offsetting contours in 2-D Space. These algorithms can be categorized into three main groups: Pair-wise Offset, Pixel-based and Voronoi Approaches. The Pair-wise offset approaches [21, 22, 23] usually have some problems such as detecting the intersection of offset edges and removing invalid loops pointed out by Held et al. [24] The Pixel-based methods as [25, 26] are computationally intensive as pointed out by Park and Choi. [27] Voronoi-based methods are better in terms of robustness and efficiency. [27] However, these approaches generally have problems with numerical stability computations such as singularity problems. [28, 29]

A new approach to generate the Voronoi Diagram as well as the offset edges is introduced in this paper. The algorithm has been modified and generalized from the previous work done by Choset and Burdick [6] and Choset et al. [7] Our modified version is more robust and complete.

2. DISTANCE AND GRADIENT CALCULATIONS

Distance is a basic function of our work. This function is modified from the distance function in the previous work done by Choset and Burdick [6] and Choset et al. [7] There are some similarities as well as differences between the previous work and the current work, which will be stated later in the paper.

In the previous work, the Voronoi Diagram (Skeleton) is generated for unknown "connected" Free Spaces, $\mathcal{FS}$. A robot is assumed to be a point operated in a Workspace, $\mathcal{W}$, which is populated by convex obstacles. In this work, on the other hand, the tip of the deposition nozzle is moved to cover "each point" on the Target Regions where there is material present. Note that the target regions do not have to be connected in this work. Workspace subtracted by Free Spaces is the same as Target Regions – $\mathcal{TR}$. ($\mathcal{W}$ - $\mathcal{FS}$ = $\mathcal{TR}$). The Target Regions are bounded by edges and vertices since the STL files contain only triangle faces, the boundaries on a cross section contain only straight edges and vertices as shown in Figure 4.

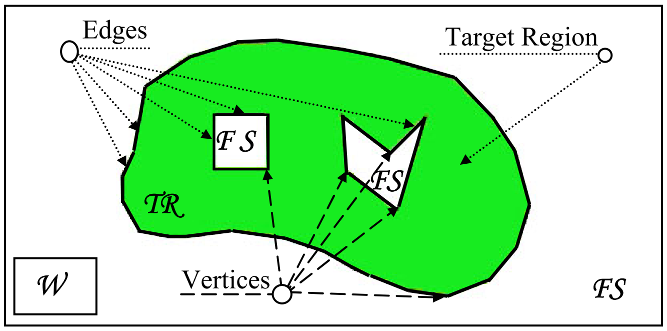

Figure 4: A Target Region on a Layer

As stated earlier, it is assumed that the Workspace, $\mathcal{W}$, contains edges $\{E_i\}$: $E_1, E_2, E_3, \ldots, E_n$. (n = numbers of edges) The distance from a point "$\vec{x}$" to edge i (bounded by two vertices – v_{i1} and v_{i2}) is:

$$d_i(\vec{x}) = \|\vec{x}-\vec{e}_0\|,\ \textbf{\textit{if}}\ \vec{x} \in \mathcal{TR}\ \&\ \exists\vec{e}_0 \in E_i^v(\vec{x}):\ \vec{x}-\vec{e}_0 \perp \vec{e}_0-\vec{v}_{i1} \tag{1a}$$

$$d_i(\vec{x}) = \min\|\vec{x}-\vec{v}_{ih}\|,\ \textbf{\textit{if}}\ \vec{x} \in \mathcal{TR}\ \&\ \vec{v}_{ih} \in E_i^v(\vec{x}),\ h \in \{1,2\}\ \&\ \nexists\vec{e}_0 \in E_i^v(\vec{x}): \vec{x}-\vec{e}_0 \perp \vec{e}_0-\vec{v}_{i1} \tag{1b}$$

$$d_i(\vec{x}) = \text{undefined},\ \textbf{\textit{if}}\ \vec{x} \in \mathcal{TR},\ e \in E_i,\ e \notin E_i^v(\vec{x}) \tag{1c}$$

$$d_i(\vec{x}) = \text{undefined},\ \textbf{if}\ \vec{x} \notin \mathcal{TR} \tag{1d}$$

Where $\vec{e}_0$ is the closest to "$\vec{x}$" in E_i^{th}, and line of sight $E_i^v(\vec{x})$ is defined as

$$E_i^v(\vec{x}) = \{e \in E_i \text{ s.t. } (1\text{-}t)\ \vec{x} + t.e \in \mathcal{TR},\ \forall t \in [0,1]\} \tag{2}$$

(An example is shown in Figure 5.) The gradient of $d_i(\vec{x})$ is defined as follows:

$$\nabla d_i(\vec{x}) = \frac{\vec{x}-\vec{e}_0}{\min\|\vec{x}-\vec{e}_0\|},\ \textbf{\textit{if}}\ \vec{x} \in \mathcal{TR}\ \&\ \exists\vec{e}_0 \in E_i^v(\vec{x}):\ \vec{x}-\vec{e}_0 \perp \vec{e}_0-\vec{v}_{i1} \tag{3a}$$

$$\nabla d_i(\vec{x}) = \frac{\vec{x}-\vec{v}_{ih}}{\min\|\vec{x}-\vec{v}_{ih}\|},\ \textbf{\textit{if}}\ \vec{x} \in \mathcal{TR}\ \&\ \vec{v}_{ih} \in E_i^v(\vec{x}),\ h \in \{1,2\}\ \&\ \nexists\vec{e}_0 \in E_i^v(\vec{x}): \vec{x}-\vec{e}_0 \perp \vec{e}_0-\vec{v}_{i1} \tag{3b}$$

$\nabla d_i(\vec{x}) = \text{undefined}, \quad \textbf{\textit{if}} \; \vec{x} \in \mathcal{TR}, e \in E_i, e \not\in E_i^v(\vec{x})$ (3c)

$\nabla d_i(\vec{x}) = \text{undefined}, \quad \textbf{\textit{if}} \; \vec{x} \not\in \mathcal{TR}$ (3d)

In case $d_i(\vec{x}) = d_j(\vec{x})$ and $\nabla d_i(\vec{x}) = \nabla d_j(\vec{x})$, one of them is occluded by the other.

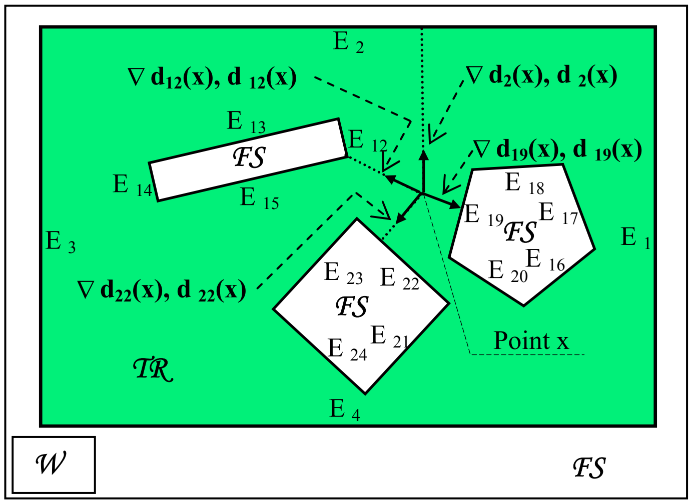

Figure 5: Distances and Gradients at Point "x"

3. VERTEX ANGLE CLASSIFICATION

Concave Angle (CCA): The solid angle between two adjacent edges is between 0 and 180 degrees. See Figure 6.

Convex Angle (CVA): The solid angle between two adjacent edges is between 180 and 360 degrees. See Figure 6.

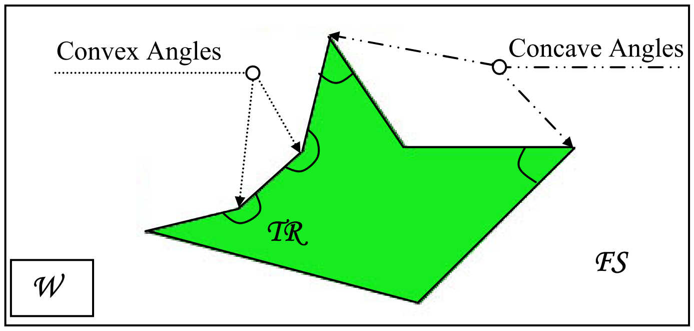

Figure 6: Vertex Angle Classification

4. SKELETON IN $\Re^2$

Skeleton is also known as *Medial Axis Transform* (MAT) or *Voronoi Diagram* ($\mathcal{VD}$). The Voronoi Diagram is used in a wide range of applications such as robot path planning [6, 7], tool path planning [2, 3, 5, 8], feature recognition [9], finite elements [10, 11], and shape analysis. [12] The Medial Axis Transform was first proposed by Blum [12] to describe shapes in biology. The medial axis is defined as loci of centers of locally maximal disks inside a region.

In order to make the Skeleton meaningful and useful for process and path planning, a Skeleton generated from a selected algorithm should yield the following properties [3]:

Accuracy: Manufacturability associated with a specific fabrication process is often needed. Thus, the Skeleton should be as accurate as possible to represent the geometry.

Connectivity: Since the Skeleton can be used to perform reasoning and shape analysis, the connectivity of the skeleton edges is very important.

Associativity: The Skeleton alone would not be sufficient to do reasoning or analysis. There should be other information such as the distances and direction to the closest edges. Also, information about a boundary point on the closest edge could be useful.

Complexity: The algorithm should not be too complicated since the overall processes of the LAMP system involves several build cycles and layers. Therefore, the Skeletonization technique is very important to the overall process efficiency.

Completeness: The Skeletonization technique should be able to handle any STL file generated from any 3-D CAD drawing. The algorithm should be able to generate the Skeleton directly and successfully from the slicing information given by the slicing processor.

There are several existing approaches to perform Medial Axis Transform Extraction. Thinning is the most widely used in image processing and pattern recognition. The output is a "thinned" representation of the original data image, which is an array of image pixels. A survey of thinning approaches can be found in [13]. Skeletons generated by thinning methods are usually not good representations of the original geometries due to the connectivity. [14] Discretization methods have been used to generate discrete Medial Axis Transform representation. Domain boundaries are sampled and the Skeleton is generated from the set of discrete boundary points. [15] Subdivision or the "divide-and-conquer" method is also used to construct the Skeleton. Usually, this category of Skeletonization method can be used for simple polygons. The algorithm recursively divides the polygon boundaries into two lists of polygon edges bounded by two convex vertices. The complexity of the algorithm grows as the boundary edges increase. [16] Pairing is a method to generate the approximated Skeleton. [17] Therefore, this method is appropriate to the applications that do not need exact representation of the Voronoi Diagram or Skeleton. [3] Tracing algorithms are the most suitable for smooth non-linear boundaries. [3] This type of method is also appropriate for our application since it can provide a precise and connected Skeleton. The associated distances with any points on the Skeleton can be included, if needed.

Our skeletonization algorithm is based on previous work in mobile robot path planning for connected unknown environments done by Choset and Burdick [6], and Choset et al. [7] Functions and procedures such as Distance Function, Gradient Function, and tracing procedure are adapted and modified so the LAMP processor software can deal with more complicated and unconnected known environments. There are quite a few changes that make the algorithm much more robust.

A Voronoi Edge ($\mathcal{VE}$) is defined as a set of points equidistant to the two closest edges. A Voronoi Diagram ($\mathcal{VD}$) is the superset of Voronoi Edges including Intersection Points ($\mathcal{IP}$), which will be defined later. (See Figure 7) Intersection – Points have more degrees of equidistant edges compared to

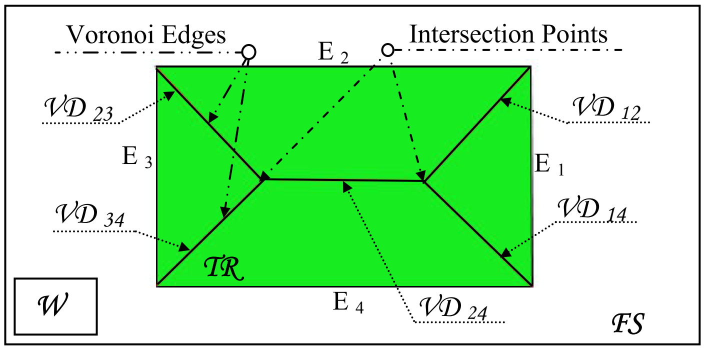

Figure 7: Voronoi Edges and Intersection Points

Voronoi Edges. Voronoi diagram is basically the set of Voronoi Edges combined with the Intersection Points. Therefore, the Voronoi Diagram is defined as follows:

$$\mathcal{VD} = \text{cl}\,\{\,\vec{x} \in \mathcal{TR} \text{ s.t. } d_p(\vec{x}) = d_q(\vec{x}) \le d_r(\vec{x}),\; p, q, r \in \{0, 1, 2, \ldots, n\},\, p \ne q,\, \forall\, r \ne p, q, \text{ where } n = \text{number of edges in } \mathcal{W}\,\} \quad (4)$$

4.1 Skeletonization

Our Skeleton extraction algorithm borrows some basic ideas and techniques from earlier work done by Choset et al. [7] This approach was used by Keller in 1987 as a technique for the numerical continuation method. [18] The previous technique has been described in the literature [7] and modified in order to handle more complicated environments. Our modified version is able to handle cases that are the assumptions in the previous technique such as Equidistant Edge, Transversality Assumption and Unconnected Target Regions.

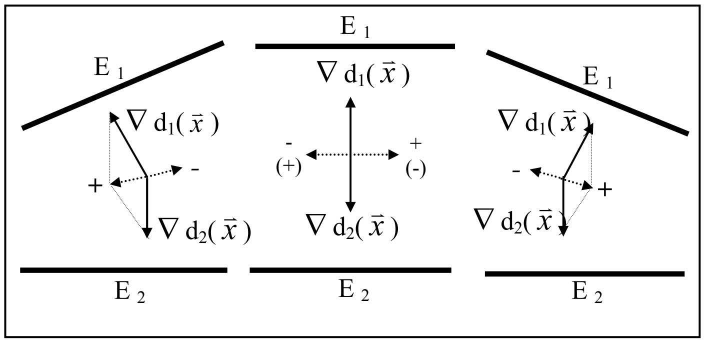

Figure 8: Pick a Direction to Move at a Point "x"

The algorithm starts at one of the seed points, which are actually the corners or vertices that are defined as concave vertices. The tool, which is assumed to be a point for now, is first moved along these Voronoi edges. In the prediction step, at any point x, the tool is moved with a small step size along the direction $\nabla d_1(\vec{x}) + \nabla d_2(\vec{x})$ or - $(\nabla d_1(\vec{x}) + \nabla d_2(\vec{x}))$. (See Figure 8) The direction is picked based on the previous direction in order not to backtrack the tracing. (Inner product $<dir_{prev},dir_{curr}>$ has to be greater than 0.)

In case $\nabla d_1(\vec{x}) + \nabla d_2(\vec{x}) = 0$, one of two unit vectors perpendicular to either $\nabla d_1(\vec{x})$ or $\nabla d_2(\vec{x})$ is selected as the moving direction instead. The direction is selected based on the previous direction in order not to backtrack the tracing. The inner product $<dir_{prev},dir_{curr}>$ has to be greater than 0.

Usually, the prediction steps take the tool off the Voronoi edges. However, if the step is small "enough", the tool should stay on track or close enough to approximate the Voronoi edges. In case the prediction step is quite large, an optimization technique is used to bring the tool back on track along the hyper-plane orthogonal to the prediction direction. This technique has been described in literature. [7] Therefore, the correction technique is omitted in the paper.

4.1.1 Terminate a Voronoi Edge Tracing (Intersection Point and Detection)

Finding the Intersection Points is very critical to the Skeleton construction. Comparing the distances to the closest edges is not a good approach to detect an Intersection Point. This is because it is likely that the tool could move past the exact Intersection Point. The new location after passing the exact Intersection Point could be out of the vicinity of that exact point.

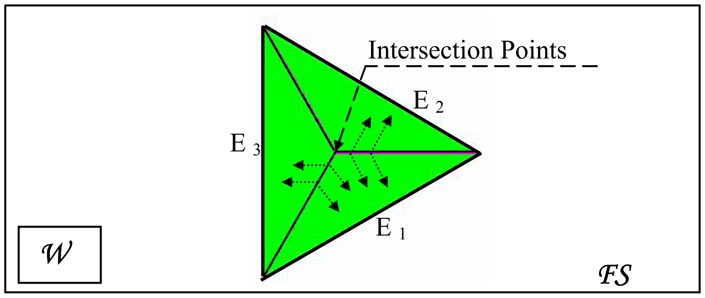

Figure 9: Intersection Point Detection

The Intersection Point can be robustly detected by watching for a sudden change in one of the two closest edges, not the change in gradient directions. For example, let a Voronoi Edge be traced and the closest edges are E_1 and E_2 as shown in Figure 9. Tracing is stopped once the closest edges E_1 and E_3 are detected.

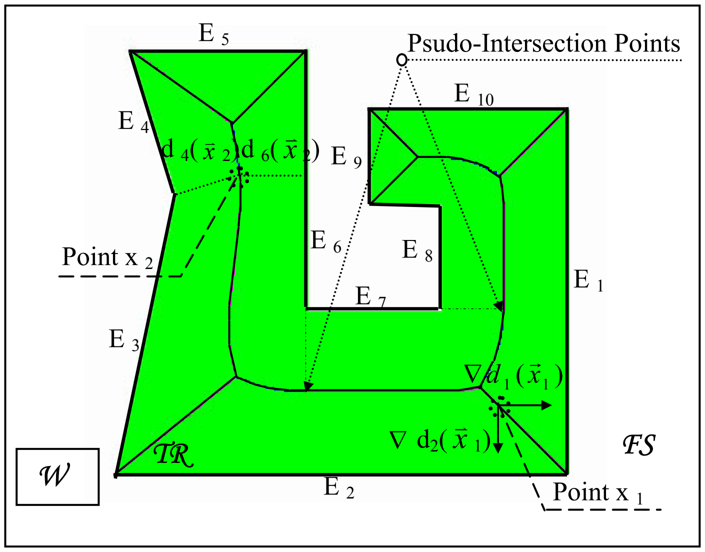

Figure 10: Distances and Gradients to Equidistant Edges

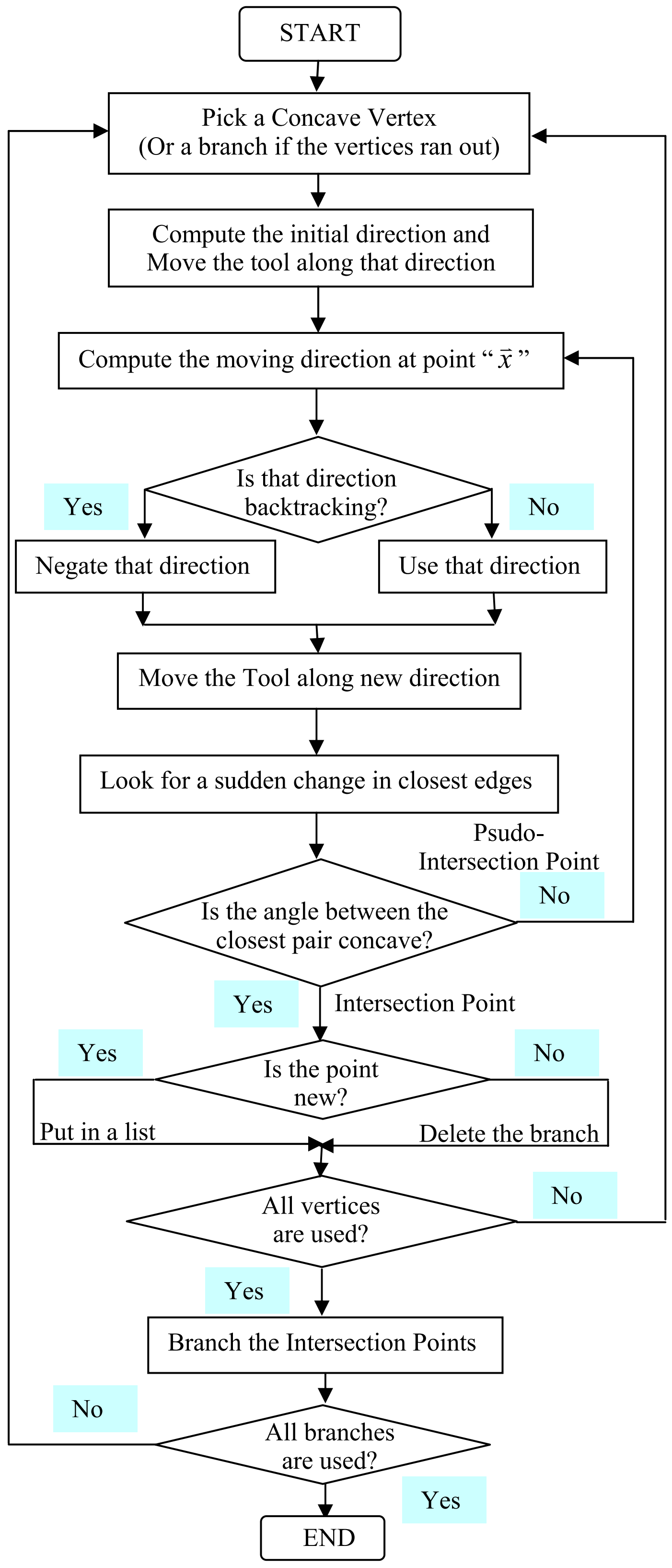
START
Pick a Concave Vertex
(Or a branch if the vertices ran out)
Compute the initial direction and
Move the tool along that direction
Compute the moving direction at point "$\vec{x}$"
Is that direction backtracking?
Yes
No
Negate that direction
Use that direction
Move the Tool along new direction
Look for a sudden change in closest edges
Psudo-Intersection Point
No
Is the angle between the closest pair concave?
Yes
Intersection Point
Yes
Is the point new?
No
Put in a list
Delete the branch
All vertices are used?
No
Yes
Branch the Intersection Points
No
All branches are used?
Yes
END

Figure 11: Proposed Algorithm to Construct Voronoi Edges

4.1.2 Pseudo-Intersection Point

Even though the Intersection Point can be robustly detected by watching for a sudden change in one of the two closest edges, this may mislead the algorithm to a stop at points that are not Intersection Points. Those points are termed Pseudo-Intersection Points. Basically, a Pseudo-Intersection Point is detected where there is an abrupt change in the closest pair but the vertex angle between the previous closest edge and the current edge is convex. Usually that case occurs at the points where the previous and current closest edges are adjacent to each other with the stated vertex angle condition. (See Figure 10)

4.1.3 Branching and Departing an Intersection Point

All of the closest edges are identified at an Intersection Point by moving around the vicinity of the Intersection Point. Once all of the closest edges are identified, adjacent edges will be paired up to compute the initial directions for moving. An edge is used in the direction computations only twice due to the adjacency. It is not always correct to move the tool away from the 3rd closest edge as described in previous literature. [7] It is more appropriate to move the tool in the direction that is between the gradients of the two edges defining the direction. (See Figure 12)

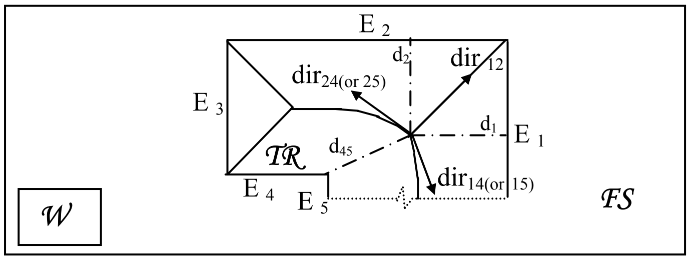

Figure 12: Branching and Departing an Intersection Point

The Voronoi Diagram construction procedure is shown in the flowchart Figure 11.

4.2 Voronoi Region

A Voronoi Region ($\mathcal{VR}$) is defined as a set of points closest to a closest edge than to any other edges. [6]

$$\mathcal{VR}_i = \text{cl}\,\{\,\vec{x}\ \in \mathcal{TR}\ \text{s.t.}\quad d_i(\vec{x}) \le d_h(\vec{x}),\ \forall h \ne i,\ \&\ i, h \in 1, 2, 3, \dots, n\,\} \qquad (5)$$

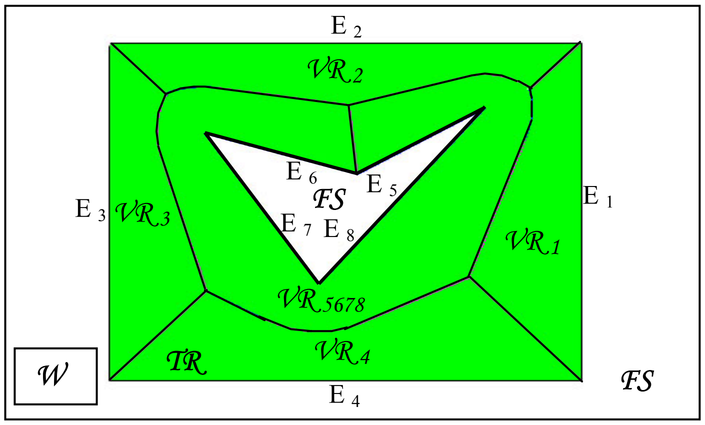

Figure 13: Voronoi Regions

In case there is no Voronoi edge between two adjacent Voronoi Regions, those adjacent regions can be combined. For example, $\mathcal{VR}_{5678} = \mathcal{VR}_5 \cup \mathcal{VR}_6 \cup \mathcal{VR}_7 \cup \mathcal{VR}_8$. (See Figure 13)

4.3 Voronoi Cycle

Basically, the Voronoi edges are the boundaries of Voronoi Regions. It is not true to conclude that there is always at least one Voronoi edge touching the closest edge in a Voronoi Region. The following example is a counter example to that misleading conclusion – A set of Adjacent Edges in which all of the angles between any pair of edges is convex. (See Figure 14) A Voronoi Cycle ($\mathcal{VC}$) is defined as:

$$\mathcal{VC} = \text{cl}\ \{x \in \mathcal{VD}_{gh}\ U\ \mathcal{VD}_{ghi}\ U\ \mathcal{VD}_{ghij}\ U\ldots,\\ \text{s.t. } \exists\ \mathcal{VR}_{cde\ldots l}\ \&\\ \nexists\ \mathcal{VR}_{uv},\ u \in \{c,d,e,\ldots,l\},\ v \in \{c,d,e,\ldots,l\}\ \} \qquad (6)$$

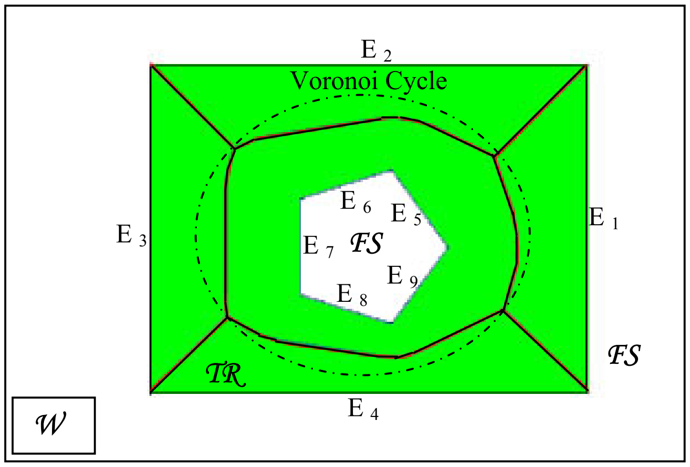

Figure 14: Voronoi Cycle

5. OFFSET EDGE

Our goal is to move the tool to cover all Target Regions. Instead of finding a way to cover all Target Regions at once, all Target Regions are subdivided to Voronoi Regions by generating a Voronoi Diagram as described earlier. For each

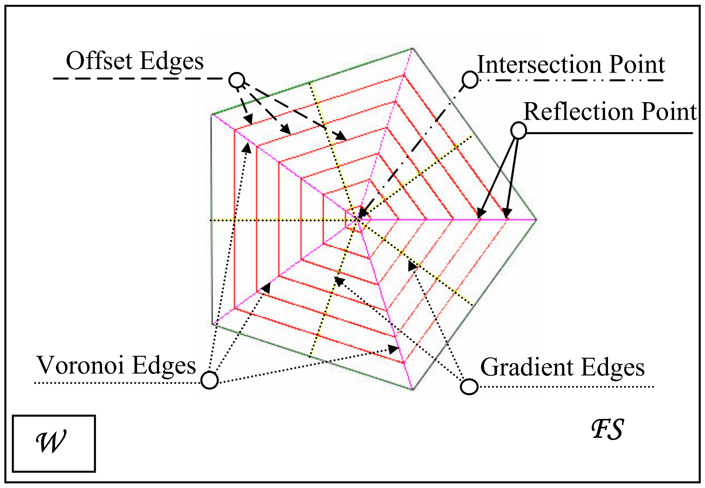

Figure 15: Offset Edges

Voronoi Region, the area can be covered with offset edges. [19] These edges are separated by a fixed distance D. D is the diameter of the deposition bead subtracted by the specified overlap, in this case. (See Figure 15 and Figure 16)

Offset Edge ($\mathcal{OE}$) is defined as follows:

$$\mathcal{OE}_p = \{\ x \in \mathcal{VR}_p \quad \text{s.t.} \quad d_p(x) = K * D, \text{ where}\\ K = \{1,2,3,\ldots\},\\ p = 1^{st} \text{ Closest edge, and}\\ D = \text{Fixed Distance from the closest edge}\} \qquad (9)$$

$$\mathcal{OE} = \cup_p\ \mathcal{OE}_p \qquad (10)$$

5.1 Gradient Edge and K-Folding Point

Recall that the tool is equidistant to more than two edges at an Intersection Point. Suppose that there are *n* equidistant edges at an Intersection Point. There are *n* Gradient Edges emanating from the Intersection Point ending at a point on corresponding equidistant edges. (See Figure 16) Basically, those edges are edges that connect the Intersection Point with the closest points on equidistant edges. [7, 19]

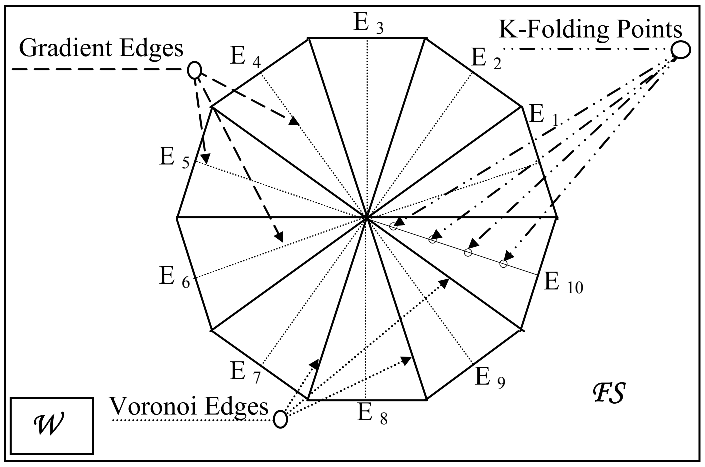

Figure 16: Gradient Edges and K-Folding Points

Let M be an Intersection Point, a K-Folding Point ($\mathcal{KFP}$) is defined as follows:

$$\mathcal{KFP}_p = \{\ x \in \nabla\ d_p(M),\ p \in \{c,d,e,\ldots,l\ \}\\ \text{s.t.}\\ d_p(M) < d_q(M),\ q \in \{c,d,e,\ldots,l\ \},\ p \neq q\\ d_p(x) = K * D,\ K = \{1,2,3,\ldots\}, \text{ and}\\ D = \text{Fixed Distance from the closest edge}\} \qquad (7)$$

$$\mathcal{KFP} = \cup_p\ \mathcal{KFP}_p \qquad (8)$$

5.2 Branching at K-Folding Point

Since there is no equidistant edge at a K-Folding Point, a perpendicular unit vector to $\nabla d_i(\vec{x})$ or the gradient of the 1st closest edge (edge ith) is computed to be the moving direction. The opposite of that moving direction is also calculated to be the other branch shown in Figure 17.

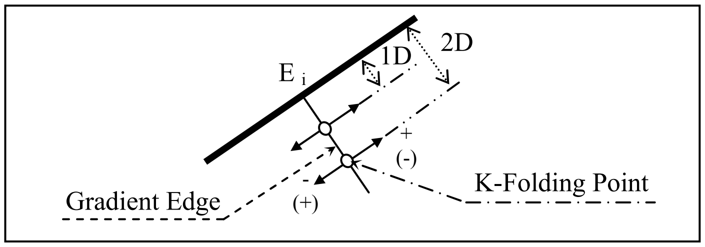

Figure 17: Branching at K-Folding Points

5.3 CONSTRUCTION OF OFFSET EDGES

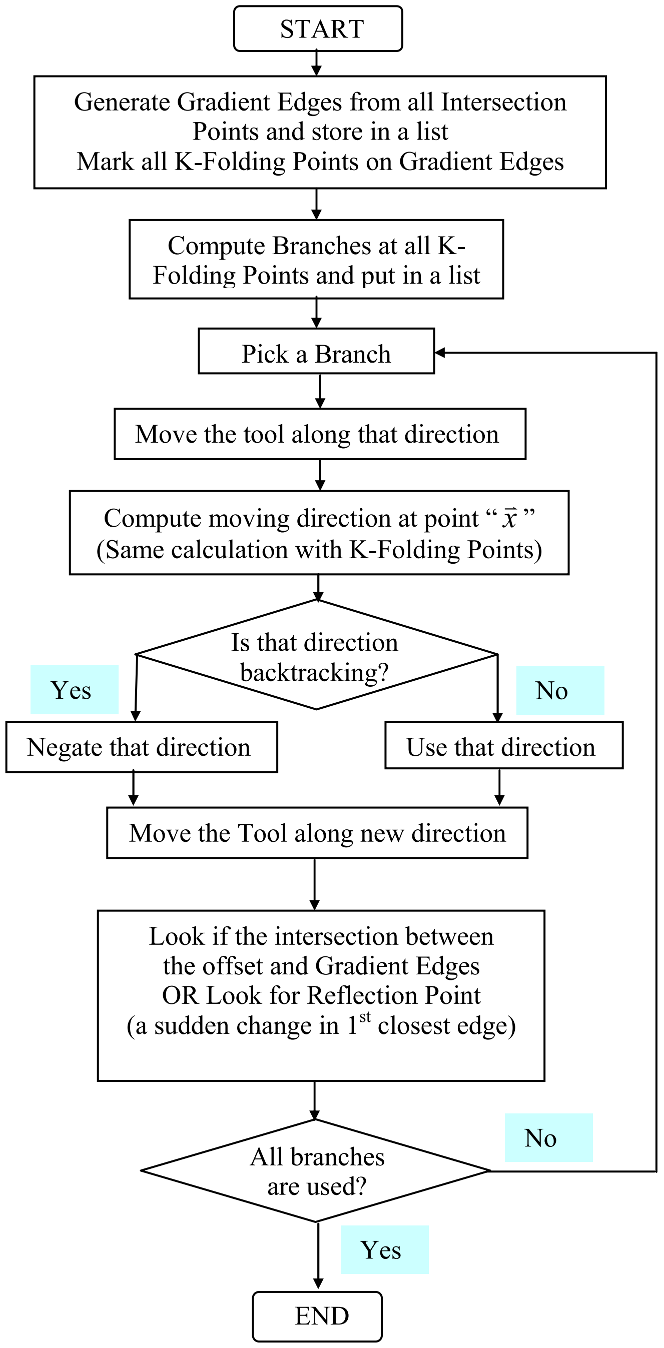
START
Generate Gradient Edges from all Intersection Points and store in a list
Mark all K-Folding Points on Gradient Edges
Compute Branches at all K-Folding Points and put in a list
Pick a Branch
Move the tool along that direction
Compute moving direction at point "$\bar{x}$" (Same calculation with K-Folding Points)
Is that direction backtracking?
Yes
No
Negate that direction
Use that direction
Move the Tool along new direction
Look if the intersection between the offset and Gradient Edges OR Look for Reflection Point (a sudden change in 1st closest edge)
All branches are used?
No
Yes
END

Figure 18: Offset Edge Construction

Offset Edges are incrementally constructed using the same numerical method as used for the Voronoi Diagram. The construction procedure can is shown in the flowchart in Figure 18. Problems due to the numerical stability computations of singular points has been solved.

5.4 Terminate Offset Edge Tracing (Reflection Point and Detection)

The Reflection Point can be robustly detected by watching for a sudden change in the 1st closest edge. (See Figure 15) The Reflection Point ($\mathcal{RP}$) is defined as follows:

$$\mathcal{RP} = \{ \bar{x} \in \mathcal{VD} \text{ s.t. } d_p(\bar{x}) = K * D, \text{ where } K = \{1,2,3,\ldots\}, p = 1^{st} \text{ closest edge, and } D = \text{Fixed Distance from the closest edge}\} \quad (11)$$

6. SIMULATIONS

A $\Re^2$ simulator has been written to validate this algorithm. This 2-D simulator is written in C++ using some Graphics and OpenGL libraries. All of the simulations have been done in the scale of millimeters.

Figure 19 contains an example in which the algorithm was tested to handle unconnected Target Regions. Because of holes and slots, sometimes the path processor may have to deal with the case of unconnected regions on a slice. After the 3-D Solid model has been sliced, one solid object could be cut into two unconnected regions on a slicing plane.

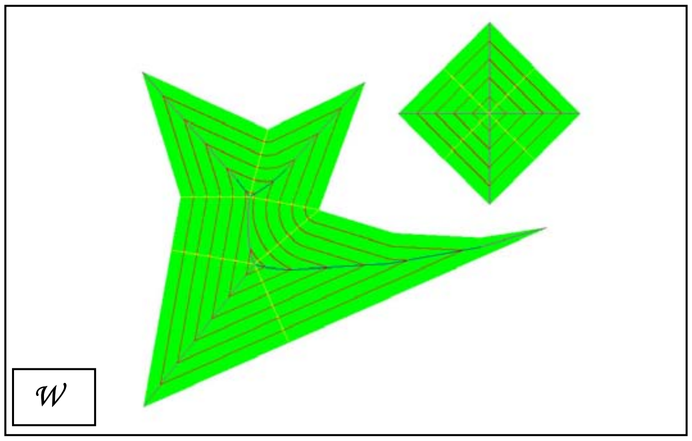

Figure 19: Unconnected Regions

Almost all engineering parts, if not all, consist of geometry elements. Usually the geometry elements tend to have symmetry for engineering applications. A Transversality assumption is usually made in the Voronoi literatures. [20] This assumption is equivalent to the "no four points are co-circular" assumption. Therefore, the effort has been put into this work so that the algorithm can deal with this Transversality. Basically, this is the case that the degrees of equidistant edges are higher than three. One of the cases occurs (but not limited to) with the approximation of a circle as shown in Figure 20.

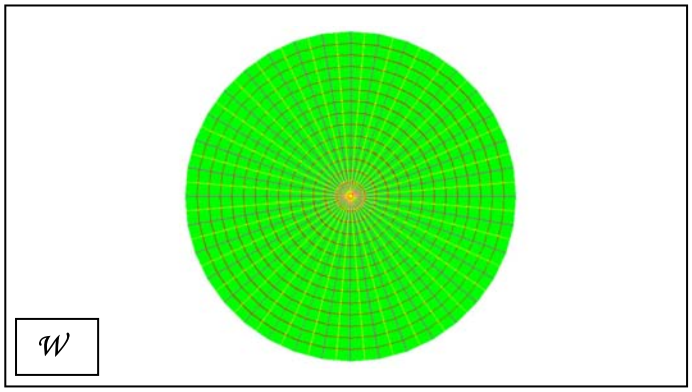

Figure 20: Tranversality Assumption in Some Algorithms

Figure 21 contains a case of a Voronoi Cycle. There is no Voronoi edge residing in the center Voronoi Region.

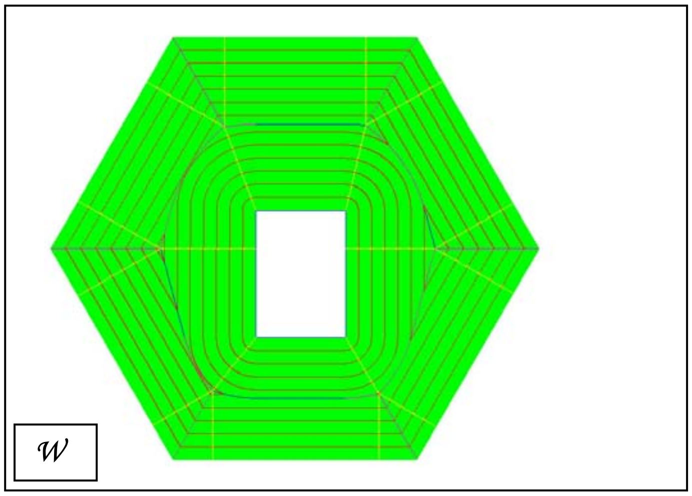

Figure 21: Voronoi Cycle.

Figure 22 has very a complicated geometry. The figure contains unconnected regions and Voronoi cycles as well.

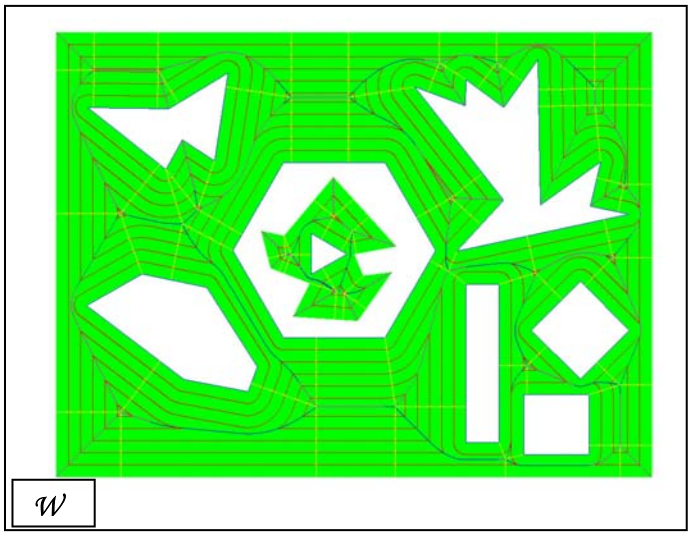

Figure 22: Complicated and Unconnected Regions

7. CONCLUSION AND FUTURE WORK

An algorithm to construct offset edges based on the Skeleton is introduced in this paper. The Skeleton generated by this algorithm is complete and satisfies other required properties. Problems in other algorithms discussed in the Introduction section are solved. Assumptions made in other algorithms namely Unconnected Regions and Transversality Assumption are removed as well. However, the offset edges generated by this method are not connected at the moment. The next step to get an automatic process planning for the LAMP system is to connect the edges so that the connected tool paths can be obtained.

We are in the process of integrating this algorithm into our 3D software. There are still several other issues and limitations that have to be addressed and continued in order to have an automatic planning processor for the LAMP system. We are working towards solving those issues and relaxing these limitations.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the National Science Foundation (Grant DMI-9871185), Missouri Research Board and the Missouri Department of Economic Development (MRTC grant) for supporting this work. Finally, the authors would like to thank the members of the Laser Aided Manufacturing Process Lab and the Intelligent System Center at the University of Missouri - Rolla.

REFERENCES

[1] Ruan, J., Eiamsa-ard, K., Zhang, J., and Liou, F.W., “Automatic Process Planning of a Multi-Axis Hybrid Manufacturing System”, Proceedings of DETC’02, September 29 – October 2, 2002, Montreal, CANADA.

[2] Kao, J., and Prinz, F.B., “Optimal Motion Planning for Deposition in Layered Manufacturing”, Proceedings of DETC’98, September 13 - 16, 1998, Atlanta, GA.

[3] Kao, J., 1999, “Process Planning for Additive/Subtractive Solid Freeform Fabrication Using Medial Axis Transform”, Ph.D. Thesis, Stanford University, CA.

[4] Zhang, J., and Liou, F.W., “Adaptive Slicing for a Five-Axis Laser Aided Manufacturing Process”, Proceedings of DETC’01, September 9 – 12, 2001, Pittsburgh, PA.

[5] Arkin, E.M., Held, M., and Smith, C.L., 1996, “Optimization Problems Related to Zigzag Pocket Machining”, 7th Annual ACM-SIAM Symp., Discrete Algorithms (SODA’96), Atlanta, GA.

[6] Choset, H., and Burdick, J., “Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph”, The International Journal of Robotics Research, Vol. 19, No. 2, February, 2000, pp. 96-125.

[7] Choset, H., Walker, S., Eiamsa-Ard, K., and Burdick, J., “Sensor-Based Exploration: Incremental Construction of the Hierarchical Generalized Voronoi Graph”, The International Journal of Robotics Research, Vol. 19, No. 2, February 2000, pp. 126-148.

[8] Held, M., “On the Computational Geometry of Pocket Machining”, Springer-Verlag, Berlin, Heidelberg, 1991.

[9] Gadh, R., Gursoy, H.N., Hall, M.A., and Prinz, F.B., “Feature Abstraction in a Knowledge-Based Critique of

Designs", Manufacturing Review, 4(2):115-125, June 1991.
[10] Gursoy, H.N., 1989, "Shape Interogation by Medial Axis Transform for Automated Analysis", PhD. Thesis, Massachusetts Institute of Technology, MA.
[11] Patrikalakis, N.M., and Gursoy, H.N., "Shape Interogation by Medial Axis Transform", Advances in Design Automation, 23:77-88, 1990.
[12] Blum, H., "Transformation for extracting new descriptors of Shape", Wathen-Dunn, W (Ed.) Models for Perception of speech and Visual Form, MIT Press, Cambridge, MA, 1967, pp362-380.
[13] Lam, L., Lee, S., and Suan, C.Y., "Thinning Methodologies – A Comprehensive Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(9):869-885, September 1992.
[14] Davies, E.R., and Plummer, A.P.N., "Thinning Algorithms: A Critique and a new Methodology", Pattern Recognition, 14(1-6):53-63, 1981.
[15] Ogniewicz, R.L., "Skeleton-space: a Multiscale Shape Description Combining Region and Boundary Information", Proceedings of 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp746-751, 1994.
[16] Lee, D.T., "Medial Axis Transformation of a planar Shape", IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-4(4):363-369, July 1982.
[17] Rezayat, M., "Midsurface Abstraction from 3D Solid Models: General Theory and Applications", Computer-Aided Design, 28(11):905-915, 1996.
[18] Keller, H.B. Lectures on Numerical Methods in Bifurcation Problems, Tata Institute of Fundamental Research, Bombay, India, 1987.
[19] Eiamsa-ard, K., and Choset, H., "Sensor-Based Path Planning: Three-Dimensional Exploration and Coverage", 1999 Mechanical Engineering Graduate Technical Conference, Carnegie Mellon University, April 16, 1999, Pittsburgh, PA.
[20] Aurenhammer, F., "Voronoi Diagram – A Survey of a Fundamental Geometric Structure", ACM Computing Surveys, 1991, 23:345-405.
[21] Hansen, A., and Arbab, F., "An Algorithm for Generating NC Tool Path for Arbitrary Shaped Pockets with Islands", ACM Transactions on Graphics, 1992, 11(2):152-182.
[22] Tiller, W., and Hansen, E.G., "Offset of Two-Dimensional Profiles", IEEE Computer Graphics and Applications, 1984:36-46.
[23] Remfield, R.F., "IGB-Offset for Plane Curves-loop Removal by Scanning of Interval Sequences", Computer Aided Geometric Design 1998:339-375.
[24] Held, M., Lukas, G., and Andor, L., "Pocket Machining Based on Contour Parallel Tool Path Generation by Means of Proximity Maps", Computer Aided Design 1994:189-203.
[25] Choi, B.K., and Kim, B.H., "Die-cavity Pocketing Via Cutting Simulation", Computer Aided Design 1997:29(12):837-846.
[26] Chiang, C.H., Hoffman, C.M., and Lynch, R.E., "How to compute offsets without self-intersection", SPIE Conference Proceedings on Curves and Surfaces in Computer Vision and Graphics II, Boston, MA, 1991, pp. 76-87.
[27] Park, S.C., and Choi, B.K., "Uncut Free Pocketing Tool-Paths Generation Using Pair-Wise Offset Algorithm", Computer –Aided Design. 2001:33:739-746.
[28] Choi, B.K., and Park, S.C., "A Pair-Wise Offset Algorithm for 2D Point-Sequence Curve", Computer –Aided Design. 1999:31(12):735-745.
[29] Kokichi, S., "Degeneracy and Instability in Geometric Computation", Proceedings of IFIP WG5.2 GEO-6 Conference in Tokyu University, December 7-9, 1998, pp5-15.