University of Cincinnati

Date: 10/3/2012

I, Nandkumar S Siraskar , hereby submit this original work as part of the requirements for the degree of Master of Science in Mechanical Engineering.

It is entitled:

Adaptive Slicing in Additive Manufacturing Process using a Modified Boundary Octree Data Structure

Student's name: **Nandkumar S Siraskar**

This work and its defense approved by:

Committee chair: Sundararaman Anand, PhD

Committee member: Sundaram Murali Meenakshi, PhD

Committee member: David Thompson, PhD

3031

Adaptive Slicing in Additive Manufacturing Process using a Modified Boundary Octree Data Structure

A thesis submitted to the

Graduate School of the University of Cincinnati

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in the Department of Mechanical Engineering
of the College of Engineering and Applied Science

by

Nandkumar Siraskar

Bachelor of Technology (B.Tech Mechanical)

Vellore Institute of Technology, India, 2009

Committee Chair: Dr. Sam Anand

ABSTRACT

In an Additive Manufacturing (AM) process, the layer-by-layer fabrication of a part leads to a staircase effect affecting its final quality. Parts manufactured using AM process may not satisfy the specified tolerance requirements due to this staircase effect. An improved surface quality (reduction in staircase effect) and a reduced build time are the two most important driving factors which led to the development of different slicing algorithms. The uniform slicing approach presents a tradeoff between the part's quality and the build time whereas current adaptive slicing methods are computationally expensive.

An innovative algorithm to compute adaptive slice thicknesses in an AM process is presented in this study. This method, termed as Modified Boundary Octree Data Structure (MBODS), is used to convert the STL file of an object to an octree data structure, by considering the part's geometry, the AM machine parameters, and user defined tolerance values. This algorithm ensures that the fabricated part satisfies the required geometric tolerances. The developed algorithm is characterized by its simple structure and its modest storage space and computation time requirements.

A subsequent algorithm computes the variable slice thicknesses using the MBODS representation of the part and virtually manufactures the part using these calculated slice thicknesses. Points sampled from the VM of the part are inspected to evaluate the part errors. In the present study, the volumetric, profile and cylindricity errors have been evaluated using the adaptive slice thicknesses. The MBODS algorithm is validated by comparing it with uniform slicing approaches using various constant slice thicknesses. The developed algorithm is observed to be more effective in improving the part-quality with lesser number of slices.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude to the people who contributed, in different ways, to the completion of this work.

First and foremost, I would like to thank my academic advisor Dr. Sam Anand and express my deep appreciation for his guidance and support for successful completion of this research. I also take this opportunity to thank Dr. David Thompson and Dr. Murali Sundaram for serving as members on my master's thesis committee.

I would like to thank Ratnadeep Paul and Neeraj Panhalkar for giving me invaluable insights and for sharing their knowledge to resolve difficult situations during the course of this research. I would also like to thank my past and present labmates who have been constant source of support and inspiration for me.

I would like to dedicate this work to my parents, brother, sister and other family members, for their unconditional support and encouragement in all of my ventures. I would like to thank my dear friends Kunal Sharma, Rupesh Bhatia, Shaleen Bhatia, Swati Tiwari, Minal Hundekari, Deepak Saagar, Bharathwaj Kumar and the entire Cincinnati family for some of the most memorable times.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Cost, quality, and velocity are three factors of major importance that are essential to excel under present global competitive pressures. Design is a crucial step in developing a new product consuming a substantial percentage of time and cost. Prototypes that do not qualify user defined criteria are rejected and rebuilt. Virtual manufacturing of parts plays an important role in minimizing these wastes. A part is physically built only if it qualifies all inspection tests in the virtual product development phase. This results in the desired quality of the part is achieved along with a noticeable reduction in time and cost. With the advent of rapid product development, additive manufacturing technologies have found their applications in almost all the industrial sectors. Enhanced visualization capabilities give the user a fair idea of looks and functionality of the final product by observing the working model in early design phase. Immediate feedback from users helps to find design flaws at early stages thus eliminating expensive errors.

In an Additive manufacturing (AM) process, a part is built by laying down successive layers of material. A distinct advantage of AM processes over traditional subtractive manufacturing processes is that parts with complex geometries can be easily fabricated. Generally, in an AM process a 3-D model of the part is created using a CAD package. This CAD model is then tessellated using a Stereolithography (STL) file that has become the standard data input format for the AM industry**.** The STL file is then sliced into thin layers before sending it to an AM machine for layer-by-layer fabrication yielding a part that does not require any additional machining or tooling.

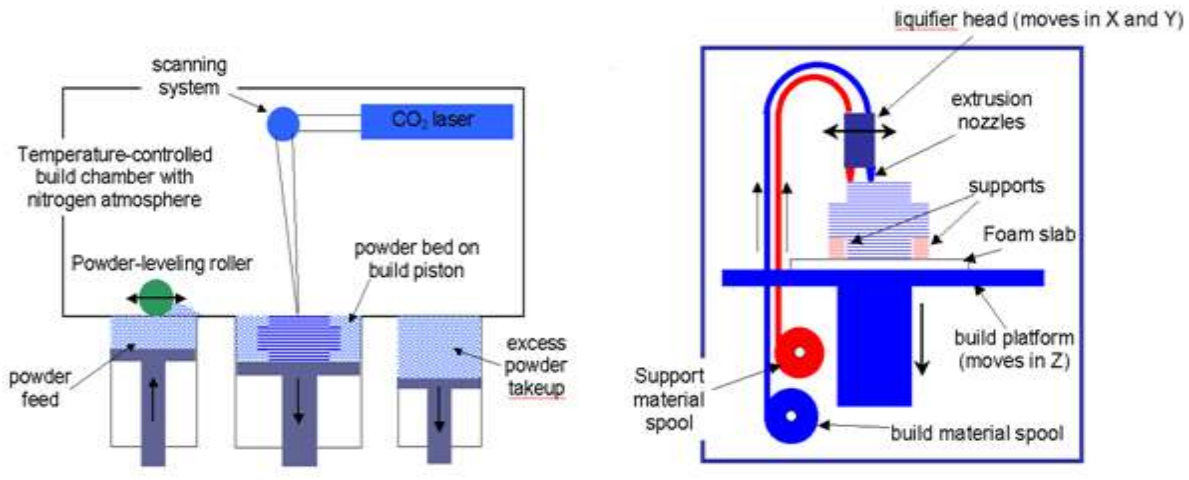
There are several AM technologies in use today and some of them are shown in Table 1

along with materials used for these processes.

Table 1: Types of AM technologies [1]

Additive manufacturing technologies	**Base materials**
Selective laser sintering (SLS)	Thermoplastics, metals powders, ceramic powders
Direct metal laser sintering (DMLS)	Almost any alloy metal
Fused deposition modeling (FDM)	Thermoplastics, eutectic metals
Stereolithography (SLA)	photopolymer
Laminated object manufacturing (LOM)	Paper, foil, plastic film
Electron beam melting (EBM)	Titanium alloys
3-D Printing	Plaster, Colored Plaster

Figure 1 shows schematics for two popular AM technologies, selective laser sintering and fused deposition modeling.



(a) (b)

Figure 1: (a) Selective laser sintering (b) Fused deposition modeling [2]

1.1 Motivation for Research

Additive manufacturing technologies have recently emerged as a chosen approach for rapid fabrication of 3-D products directly from a CAD system. Unfortunately, the layered fabrication of a part leads to staircase effect affecting the final quality of the part. As a result, the Geometric Dimensioning & Tolerancing (GD&T) requirements of parts thus manufactured using an AM process may not be satisfied. The requirement for simultaneous improvement in the two important aspects of AM methods, namely, the surface quality of a part, and build time have led to the development of different slicing algorithms. Uniform slicing technique, where a fixed layer thickness is used to slice the entire STL file, results in a tradeoff between the build time and the resulting part surface quality. On the other hand, adaptive slicing procedures offer a combination of improved part surface quality and reduced build time. Several approaches at adaptive slicing have been discussed in the literature, where the researchers have considered parameters such as part geometry, orientation, and cusp height for slicing of the CAD model. However, the proposed methods involve tedious computations demanding large computing space and time. In this research, an attempt has been made to efficiently slice an STL file (Stereolithography file) obtained from a CAD model into horizontal layers using variable slice thicknesses to achieve the desired surface quality of the part. Octree data structures [3], with their superior time and space requirements and innate simplicity, have been used to successively accomplish the adaptive slicing of the object.

1.2 Objective and Impact of Research

The objective of this research is to compute variable slice thicknesses for a part in an AM process so that it satisfies specific tolerance requirements. A unique algorithm for calculating

adaptive slices based on object geometry and AM machine parameters is proposed in this research. Existing adaptive slicing methods involve complex and time consuming computations. The present research considers a simple octree data structure and modifies it to represent surfaces of a given part. A methodology to decompose an STL file into a Modified Boundary Octree Data Structure (MBODS) is presented and an algorithm to compute adaptive slices based on the octree representation has been developed. The cubes in the octree data structure cubes are recursively divided until each cube satisfies certain predetermined criteria. This allows user control over the decomposition while maintaining the volumetric error of the object under a threshold value. To validate these methods, parts are virtually fabricated using these variable slice thicknesses to satisfy the geometric tolerances.

1.3 Thesis Outline

An introduction to AM technologies and their applications was discussed in the previous section. The objectives and possible impact of this research were also explained above. Chapter 2 of this thesis focuses on a detailed review of the work that has been carried out in the fields of AM processes, files inputs for AM, slicing methods, decomposition of objects into octree data structure, virtual manufacturing and inspection. Chapter 3 explains the proposed methodology and steps involved in achieving the expected results. Chapter 4 contains several simulated test examples and results validating the proposed methodology. Chapter 5 includes the conclusions from the present research and possible avenues for future research in this field.

2. RELATED LITERATURE REVIEW

This section reviews the work that has been performed in the field of additive manufacturing (AM) technology and octree decomposition. Areas relevant to this research are AM processes, file inputs for AM, slicing methods, octree data structures, virtual manufacturing and inspection.

2.1 Additive Manufacturing (AM)

In AM processes, CAD models are used as a direct or indirect input, to facilitate a layer-by-layer fabrication of 3-D objects. Generally, a CAD model is tessellated before sending it to the AM process. The tessellated CAD model is then sliced into thin layers and the part is built one layer atop another.

Research issues in an AM process focus on areas such as part build orientation, slice layer thickness, support structure, hollowing, workspace optimization, path planning, and material properties [4]. The various process-planning tasks which typically arise in an AM process such as slicing, orientation determination, support generation, and path planning were comprehensively reviewed by Kulkarni et al. [5]. Byun and Lee [6] aimed at achieving an optimal build orientation to reduce the surface roughness of a part by considering the support structures and build time as factors. Puduhai and Dutta [7] investigated the fabrication of a part at the optimal build orientation. Such an orientation yields a reduced build time with improved stability and surface quality by minimizing the contact area with the support structures.

2.2 File input for AM

A CAD model can be used with or without tessellation. When a CAD model undergoes

tessellation the surface of the object is approximated into a series of triangles, yielding an STL file that exists in binary and ASCII formats. Binary files are compact whereas ASCII files are easy to read [8]. ASCII and binary file formats of an STL file are shown in Table 2.

Table 2: ASCII and binary file format of an STL file [9]

ASCII file format of an STL file:	**Binary file format of an STL file:**
solid <name>	*UINT8[80] – Header*
facet normal n_i n_j n_k	*UINT32 – Number of triangles*
outer loop	*For each triangle*
vertex $v1_x$ $v1_y$ $v1_z$	*REAL32[3] – Normal vector*
vertex $v2_x$ $v2_y$ $v2_z$	*REAL32[3] – Vertex 1*
vertex $v3_x$ $v3_y$ $v3_z$	*REAL32[3] – Vertex 2*
endloop	*REAL32[3] – Vertex 3*
endfacet	*UINT16 – Attribute byte count*
endsolid	end

As shown in Figure 2, each triangular facet has three vertices and a unit normal vector. The normal of every triangular facet always faces away from the object. Figure 3, shows a CAD model and the corresponding STL file.

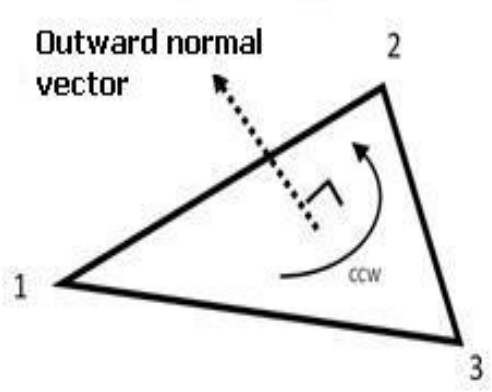
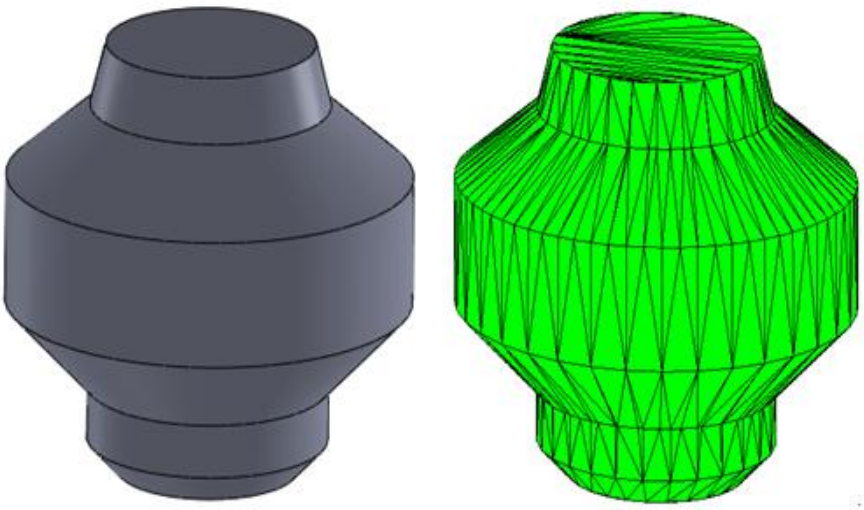


Figure 2: A triangular facet in an STL file



(a) (b)

Figure 3: (a) CAD model (b) STL file of the CAD model

As discussed earlier, STL file format has always been the industry standard for file input for AM processes. Recently, a new Additive manufacturing file (AMF) format has been developed by ASTM international [10]. AMF unlike STL file format supports color, texture, material, and constellations of the fabricated object. This file format can represent one or multiple objects and each of these objects are represented as a set of non-overlapping volumes. The color and the material of each volume can be specified using AMF file.

2.3 Slicing of an STL File

An STL file is sliced into thin layers before sending it to an AM process. Many approaches have been proposed and implemented to calculate the layer thicknesses used to slice an STL file. The surface quality and build time are the two important factors which led to development of different slicing algorithms [11]. Existing slicing algorithms can be classified

into two groups, slicing with uniform slicing procedure and adaptive slicing procedure. The different slicing procedures are classified as shown in Figure 4.

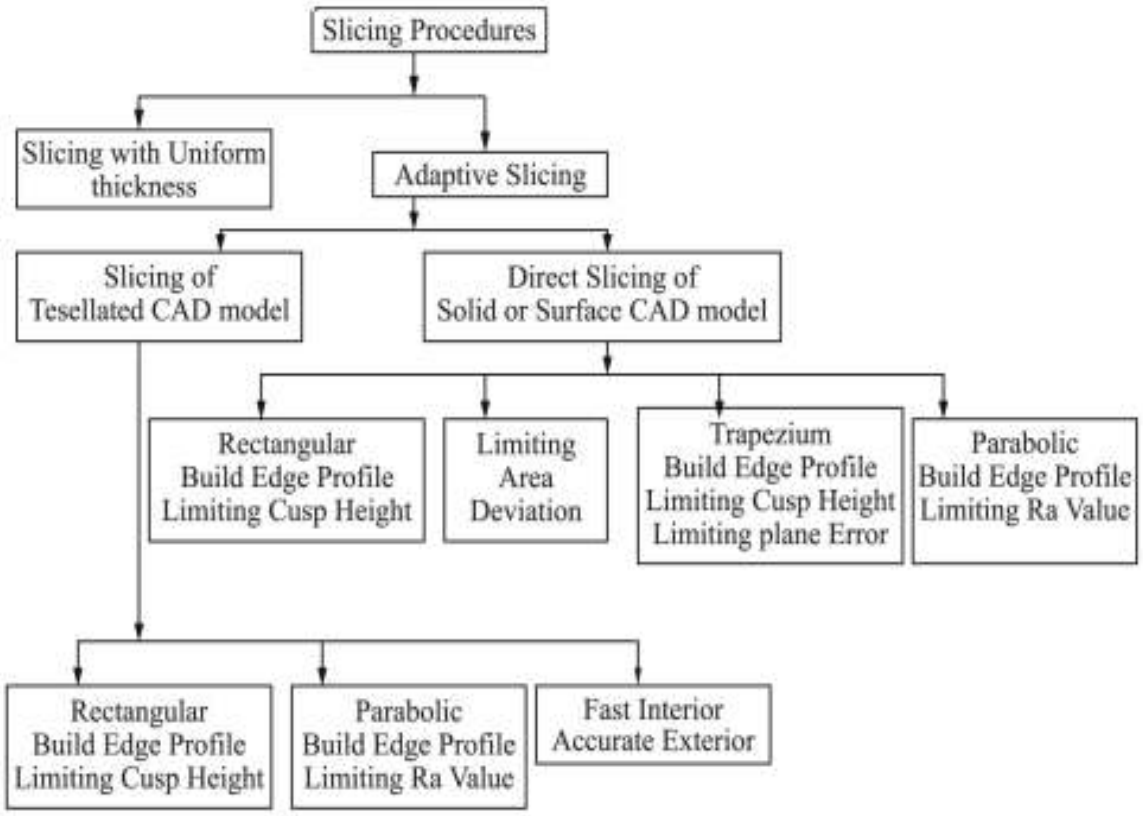
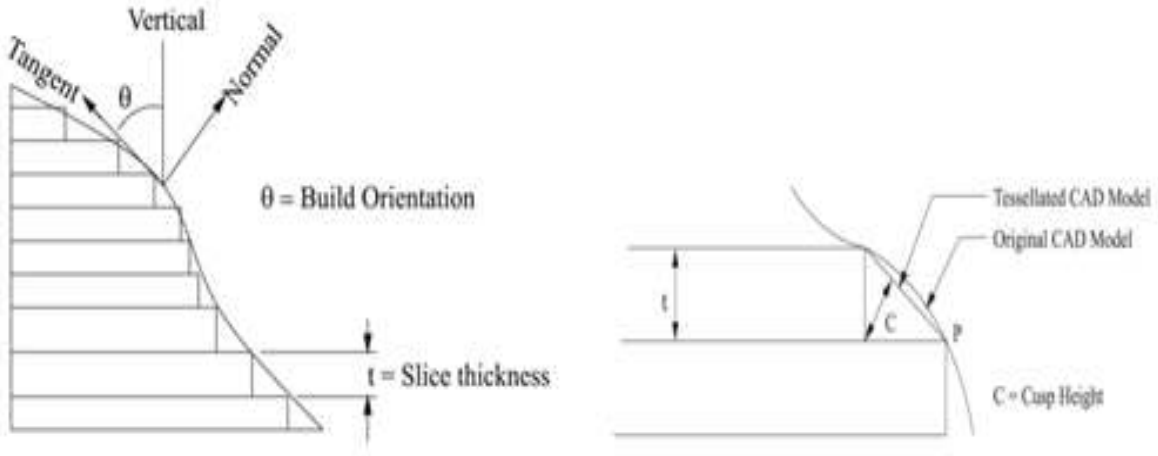


Figure 4: Classification of slicing procedures [11]

In uniform slicing, a fixed layer thickness is used to slice the entire STL file. This approach presents a tradeoff between the surface quality of the part and the build time. Smaller slice thickness will increase the build time whereas larger slice thickness will affect the desired surface quality.

In adaptive slicing, variable slice thicknesses are computed instead of a constant slice thickness. These variable slice thicknesses are governed by part geometry, orientation and AM machine parameters. A combination of improved part surface quality as well as reduced build time can be achieved using adaptive slicing algorithms.

Layered fabrication of a part leads to staircase effect as shown in Figure 5(a) affecting the final quality of the part. This staircase effect can never be completely eliminated in any AM process. Dolenc and Makela [12] introduced the concept of maximum allowable cusp height as a solution to reduce the staircase effect in AM processes. Local slice thicknesses are calculated by limiting the cusp height 'C' as shown in Figure 5(b).



(a) (b)

Figure 5: (a) Staircase effect due to layer thickness (b) Cusp height [12]

Stepwise uniform refinement procedure was proposed by Sabourin et al. [13]. This method is also based on limiting the cusp height. Initially, an STL file is sliced into uniform slices of maximum thickness allowed. Slices which do not satisfy the user defined cusp height are again divided into layers with smaller uniform thickness. In another approach as shown in Figure 6, Sabourin et al. [14] came up with a method in which exterior of the part was built with thin layers to maintain surface accuracy while thick layers are used for the interior of the part. They claimed that accurate exterior and faster interior method could reduce build time by 50-80 percent.

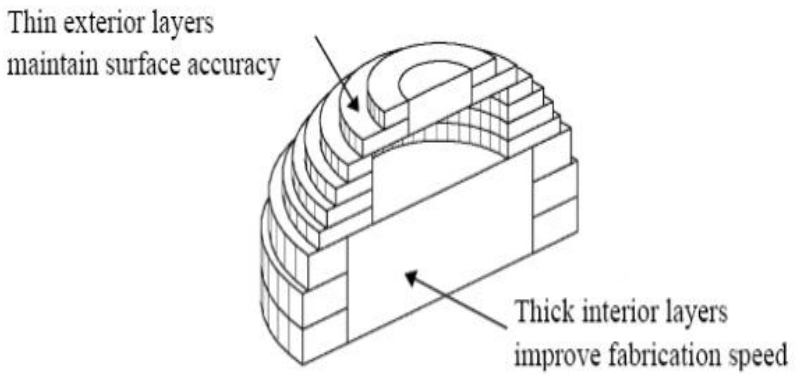


Figure 6: Slicing based on thick interior and accurate exterior [14]

According to Cormier et al. [15], non-uniform cusp height is preferred instead of assuming a maximum allowable cusp height, since a part may have different cusp height requirements at different faces. Grouping of facets in a tessellated CAD model based on edge finding algorithm has also been reported. Jamieson and Hacker [16] adopted a direct slicing approach of the CAD model with a uniform slice thickness, thus avoiding the intermediate step of tessellation of the CAD model. With a boundary representation (B-Rep) of the model, they used the areas of the successive layers as a criterion for implementing adaptive slicing.

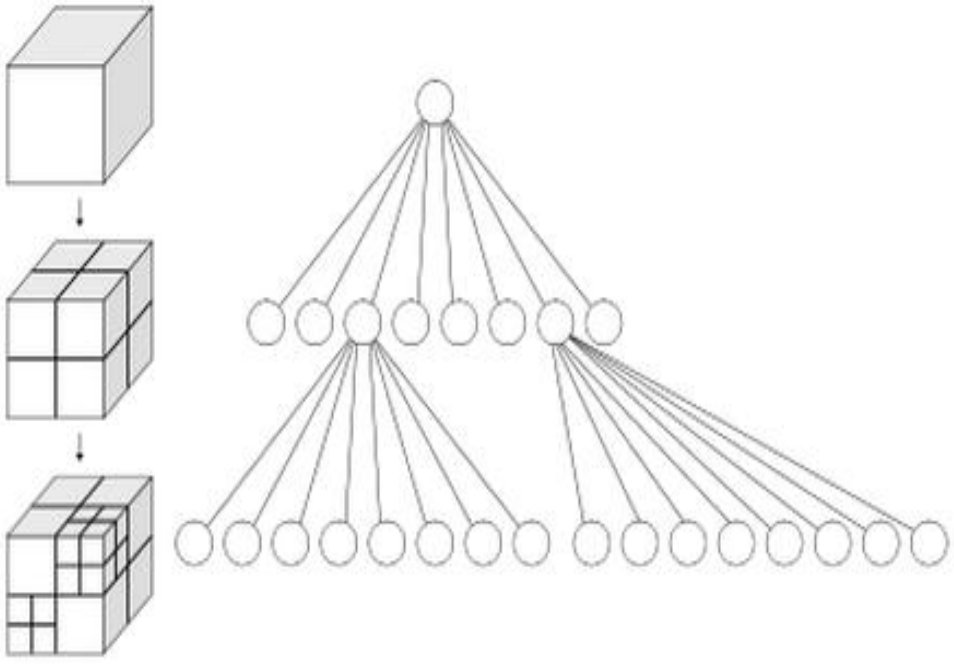
For accomplishing adaptive slicing, researchers have generally focused on the cusp height for direct or indirect slicing of the CAD model. These methods involve very complex and time consuming computations. In this research, the concept of decomposition of an object using an octree data structure will be explored to compute adaptive slices.

2.4 Octree Data Structure

In 1980, Jackins and Tanimoto [17] proposed an octree representation for the modeling of 3-dimensional objects and space planning. In octree based models, eight array trees are generated

through the repeated subdivisions of the object into eight octants [18]. Each internal node of the tree has exactly eight children [19].

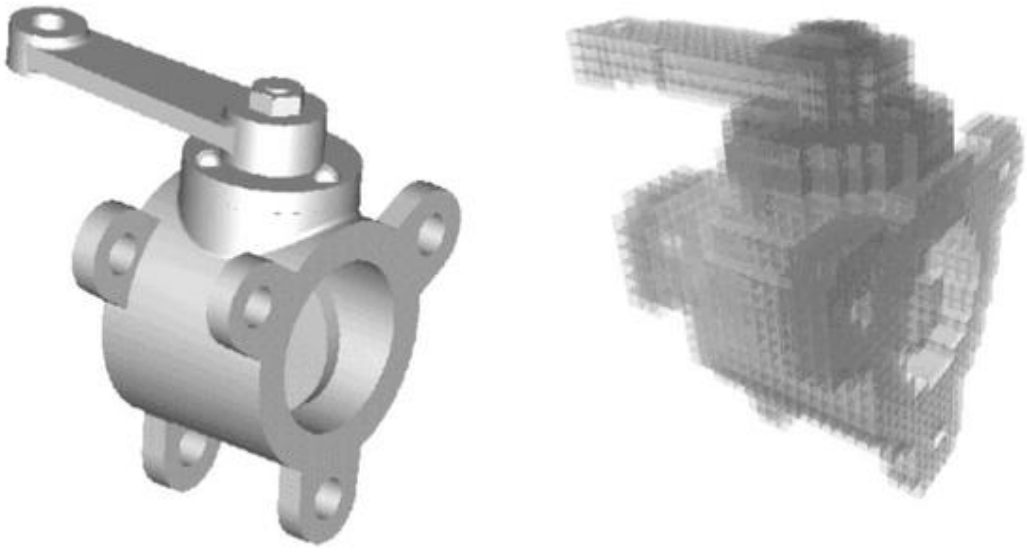
Figure 7 illustrates the recursive subdivision of a cube into eight children and the corresponding octree representation, and Figure 8 shows the CAD model of a valve with its corresponding octree representation.



(a) (b)

Figure 7: (a) Recursive subdivision of a cube into octants (b) the corresponding octree representation [20]

Meagher [18] demonstrated the applications of the octree representation in computer aided design and computer aided manufacturing by developing octree based algorithms for boolean and geometric transformations, interference detection, and the display of hidden surfaces.



(a) (b)

Figure 8: (a) Solid model of a valve; and (b) octree representation [21]

Samet [22] reported the use of hierarchical data structures such as quadtree and octree for representation of surfaces and space. Quadtree structures, where each internal node has four children, are mostly used to represent two dimensional spaces. Allada and Anand [23] had explored the applications of octree and quadtree in several areas such as robot task planning, tolerance and inspections, feature based designs, and assembly line.

Ayala et al. [24] introduced an octree model with a new type of node called Edge node that contains only one edge of an object instead of just being termed as black, white or gray, which helped in reducing storage requirements. The presented algorithm was used for exact backward conversion from octrees to boundary representation of polyhedral objects. Ayala [25] also looked into the representation of surfaces composed of biquadratic patches using an extended octree data structure with vertex, edge and face nodes. In an algorithm by Anand and Knott [26], the octree decomposition of an object is computed in four steps; intersection of

spheres enclosing the object and the cube, a bounding box check, point classification of cube vertices with the object, and the intersection of the object edges with the half planes of the cubes.

Octree data structures have the ability to store an object's information without any complexity, since each child cube inherits the information from its parent cube. The resultant octree structure helps in identifying any cube based on its origin, vertices, and path address information and also provides means of computing other relevant data such as cube length, volume, parent cube, and type of cube. The octree derived cube data structure, presented below, has been used in this study and is found to be very efficient as it has a pronounced effect on the simplicity and speed of the algorithm.

Cube Data Structure:

- Cube path address (or Cube ID)
- Cube Origin (vertex of a cube closest to universal origin (0, 0, 0))
- Cube Vertices (8 vertices)
- IDs of STL triangle facets intersecting with the cube

The following information can be easily deducted using the structure:

- Cube Length
- Cube Volume
- Cube Faces
- Cube Edges
- Intersecting triangle's Vertices
- Intersecting triangle's Edges
- Intersecting triangle's Normal
- Intersecting triangle's Area

Octree data structure properties such as volume decomposition capability, computational ease, simplicity, and less storage space requirements are exploited in this work to compute adaptive slice thicknesses used in an AM process.

2.5 Virtual manufacturing and Inspection

Virtual Manufacturing (VM) is defined as, "an integrated, synthetic manufacturing environment exercised to enhance all levels of decision and control" [27]. The part fabrication process is simulated in a computer considering all the variables involved in an actual manufacturing environment. The benefits of VM include shorter product development cycle, optimization of cost and quality, early consideration of affordability, and producibility. In [28], Marinov classified the virtual manufacturing system into two groups: virtual prototyping and virtual manufacturing. The author analyzed the complete manufacturing cycle for a product and performed an analysis of the virtual modeling technique to simulate the actual manufacturing process.

Volumetric error in an AM process can be defined as the difference between the volume of the final part and the volume of the CAD model. Masood and Rattanawong [29] considered the volumetric errors encountered in AM processes and presented an algorithm to develop a generic part orientation system of solid parts to determine the best part orientation for any given complex part. Such a part orientation can effectively control the staircase effect by minimizing the volumetric error.

The profile error of a free form surface is an important metrology metric defining the quality of a part. The objective of evaluating the profile error of a surface is to compute the minimum distance between the manufactured points and the design surface [30]. Besl et al. [31]

presented the Iterative Closest Point (ICP) method to evaluate the profile error for any given free form surface. In their paper Gunnarson et al. [32], computed the closest points to establish a relation between design surface and manufacturing points. To calculate the transformation parameters, the sum of squared errors between the two surfaces was minimized. The algorithm for the design surface creates a dataset of points that are closest to measurement point dataset followed by calculations for optimal rigid body transformations. This minimizes the distance between the manufacturing points and related design surface dataset from the first step.

Least square (LS) and Minimum zone (MZ) formulations are typically used to evaluate form tolerances such as circularity, flatness, and cylindricity. LS method deals with minimizing the sum of the square of the deviations between the observed and expected values. LS form tolerance is the difference between the furthest point locations on both sides with respect to the least square feature. The MZ algorithm minimizes the tolerance zone enclosed by two perfect profiles within which all the inspection data points should lie. Murthy et al. [33] developed linear and non-linear models using least square methods to evaluate cylindricity such that the square of perpendicular distances between the points in the dataset and the ideal cylinder axis is minimized. Shanmugam [34] also used the least square method for cylindricity evaluation through simplex search and presented a closed form solution to optimize the average cylinder deviation. Dhanish and Shanmugam [35] provided a quick but approximate solution to measure cylindricity, assuming that the axis of the cylindrical feature and that of the measuring device are aligned.

Carr and Ferreria [36] formulated a sequential linear optimization problem where they transformed the least square method and minimum zone algorithm into a sequence of linear programs to evaluate cylindricity. Lai and Chen [37] converted the measured dataset of a

cylinder into a planar dataset using non-linear transformation. This is then used in the evaluation of the cylindricity error using the Control Plane Rotation Scheme (CPRS) technique developed by Huang et al. [38] for flatness. A series of inverse transformations is then used to compute cylindricity. Kovvur et al. [39] used minimax formulations to develop a particle swarm optimization approach which was then employed to evaluate form tolerances. Ramaswami et al. [40, 41] presented an algorithm to accurately evaluate size of cylindrical features using particle swarm optimization.

3. METHODOLOGY

The present study focuses on a new approach of adaptive slicing to reduce the staircase effect encountered in an AM process, thus increasing the surface quality of the part. This is achieved by converting the STL file into a Modified Boundary Octree Data Structure (MBODS). The octree data structure explained earlier is modified to represent STL file facets through a recursive subdivision of the STL boundaries into eight cubes. The depth of division of the MBODS depends on three criteria as explained in the following sections. The MBODS of an STL file is then used to compute the adaptive slice thicknesses utilized in an AM process. The local slice thicknesses are computed based on the smallest cube size at each level and the AM machine parameters. This chapter explains the algorithms involved in the computation of variable slice thicknesses using MBODS, and the subsequent virtual manufacturing of a part. Once the part is virtually manufactured it is inspected for volumetric, profile, and cylindricity errors.

3.1 Decomposition of STL file into Modified Boundary Octree Data Structure (MBODS)

The decomposition methodology of an STL file into MBODS is presented in this section. The general steps involved in the conversion algorithm are explained briefly and are discussed in detail later in this section.

a) Universal Cube: A cube is created which encompasses entire object. This universal cube serves as the root node of the octree. The universal cube undergoes subsequent division into eight child cubes, and each of the child cubes undergo further division as seen necessary.

b) Intersection of a STL File Triangular Facet with a Cube: The cube and the triangular

facets of the feature STL file are checked for intersections. The following three tests are performed to determine the cube and triangle intersection: point in a cube test, cube edges-triangle facets intersection test, and cube facets-triangle edges intersection test.

If a cube does not intersect with any triangle then it is termed as a white node. White nodes are terminal nodes that are ignored since they do not have any significance in the surface representation of the object. If a cube intersects with one or more triangles then it is termed as a gray node and the information of all the intersecting triangles is stored in the gray node.

c) Conditions for Further Division of Gray Nodes: Gray nodes are passed on to this stage to ascertain their eligibility for further division. The following three tests are performed for this purpose: (1) smallest allowable cube test, (2) geometry of the object inside a gray node test, and (3) desired level of accuracy (user defined volumetric tolerance) test. A gray node which satisfies any of the above three tests is not eligible for further division and is termed as a terminal black node. Gray nodes which do not satisfy these three conditions undergo sub-division resulting in eight child cubes.

d) All gray child nodes are processed through steps b and c.

At the end of this MBODS algorithm, the STL file is converted to terminal black nodes of different sizes. Figure 9 shows the process overview of an STL file to MBODS conversion algorithm.

3.1.1 Universal Cube

Universal cube is the largest cube of the MBODS with its origin coinciding with the origin of the Cartesian coordinate system (0, 0, 0). This serves as the root node of the MBODS. It encloses entire STL file. Figure 10 gives a schematic representation of a universal cube. The

size of the universal cube depends upon the largest bounding box of the STL file and the minimum slice thickness of AM process. The creation of a universal cube is restricted to a set of conditions as discussed below.

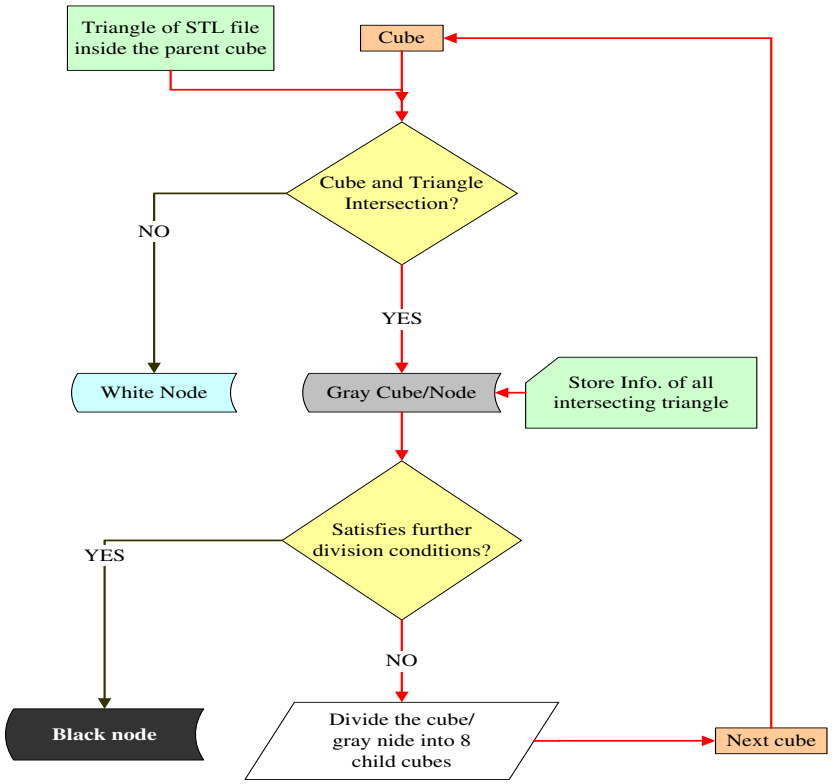


Figure 9: Process overview of STL file to MBODS conversion

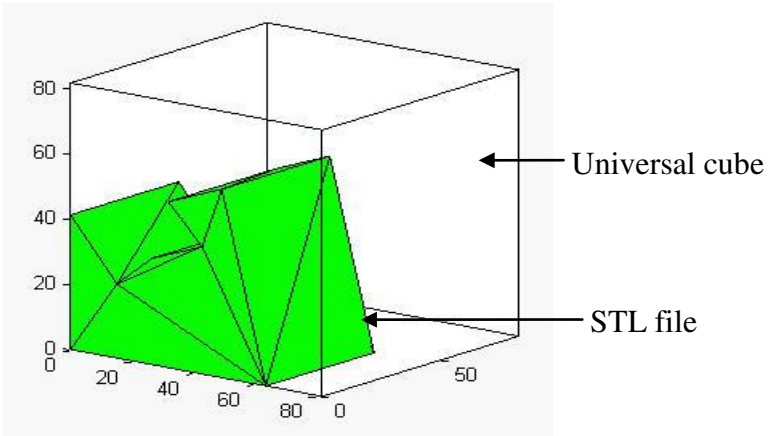


Figure 10: Universal cube encompassing STL file

3.1.1.1 Universal Cube Creation Conditions

Condition 1: The vertex-coordinates of all the triangular facets of an STL file are known and have to be aligned with the origin (0, 0, 0) of the universal cube such that the STL file lies entirely in the first quadrant of the Cartesian coordinate system. To accomplish this, the smallest values for the X, Y and Z coordinates of all the vertices in the object are determined and organized into an array of minimum X, Y and Z values, [Min X, Min Y, Min Z]. This array is then subtracted from every vertex coordinate to nullify the offset value, such that the new vertex coordinates would be [X-Min X, Y-MinY, Z-Min Z].

After nullifying the offset values, a single maximum value from the new coordinates is determined. For example, if the new vertex coordinates are represented as [X-Min X, Y-MinY, Z-Min Z], then the maximum value will be Max [Max(X-Min X), Max (Y-MinY), Max (Z-Min Z)]. The universal cube side will always be equal to or greater than this maximum value.

Condition 2: Each AM process has a minimum and maximum allowable slice thickness that is machine dependent. The minimum slice thickness of the given AM process is then used to finalize the dimension of the universal cube. In this algorithm, the side of the universal cube will always be a multiple of the minimum allowable slice thickness.

If T_{min} is the minimum allowable slice thickness, then L, the side of the universal cube is given by,

$$L = T_{min} \times \mathbf{2^N} \quad (1)$$

where, N is the depth of division.

For example, if 9.5 mm is the maximum value obtained from condition 1 and 0.01mm is the minimum available slice thickness, then the dimension of the universal cube will be 10.24 mm (i.e. 0.01×2^{10}).

A universal cube is always considered to be a gray node and is divided into 8 child nodes/cubes. These child nodes are then sent through the three intersection tests, discussed above, to determine whether a cube is white or black.

3.1.2 Intersection of a Triangle with a Cube

If a cube does not intersect with any triangle then it is termed as a terminal white node. White nodes are ignored since they do not have any significance in the surface representation of the object. If a cube intersects with one or more triangles then it is termed as a gray node and the information of the intersecting triangles is stored in the gray node data structure.

The following tests are carried out to decide whether a cube is a gray node or a white node. It is important to note that a triangle is considered to be intersecting with a cube even under the case where it lies entirely inside the cube.

3.1.2.1 Point in a Cube Test

In this test, a cube is checked against the vertices of the triangles of the STL file. A child node is checked against only those triangles which intersect with its parent node. A cube is termed as a gray node if at least one of the vertices of the object lies inside or on the cube. Triangles which have at least one vertex inside or on the cube are considered to be intersecting triangles. Relevant data of all such intersecting triangles are stored in the cube structure data. The following algorithm is used to determine the location of a given point with respect to a cube.

A point is inside a cube if it lies within the cube bound i.e. if all coordinates of that point lie within respective coordinates of vertices of that cube. For example as shown in Figure 11, consider a cube with origin at (X, Y, Z) and of sides L. A given point P(x, y, z) lies inside the cube if, $X \leq x \leq X+L$, $Y \leq y \leq Y+L$ and $Z \leq z \leq Z+L$.

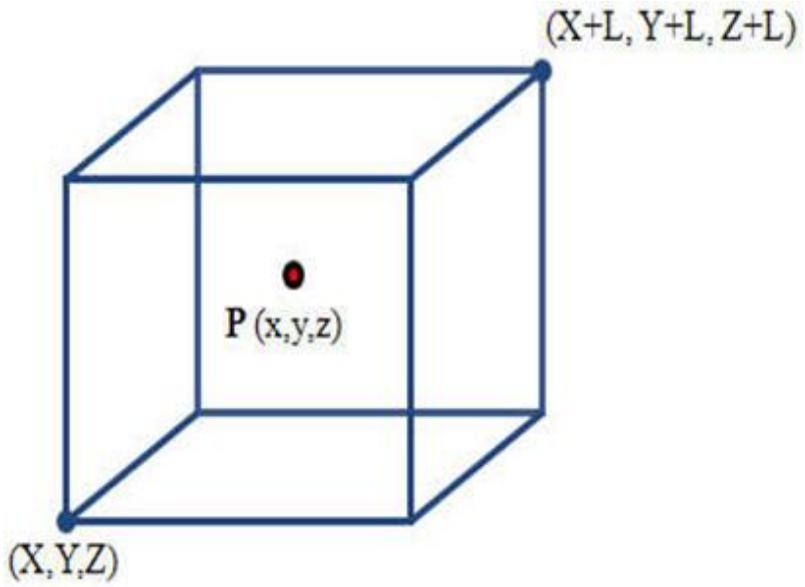


Figure 11: Point P inside a cube

If no vertex lies inside or on the cube then the cube is termed as "non-terminal white node". This node will undergo the next test before it can be considered as terminal white node. A gray node will also undergo the next test to check for intersections with the remaining triangular facets from its parent cube.

3.1.2.2 Cube Edges and Triangle Facets Intersection Test

This test is performed after performing the point in a cube test. A white node from the previous test does not have any triangle vertex inside or on the cube. However, there may be a case where a triangular facet intersects with cube's edges while all the vertices and edges of the triangle lie outside the cube.

A simple line and plane intersection algorithm is used to find whether a triangular facet intersects with the cube edge. The cube edge and triangle facet intersection point should lie within the bounds of the triangle edge as well as the cube face.

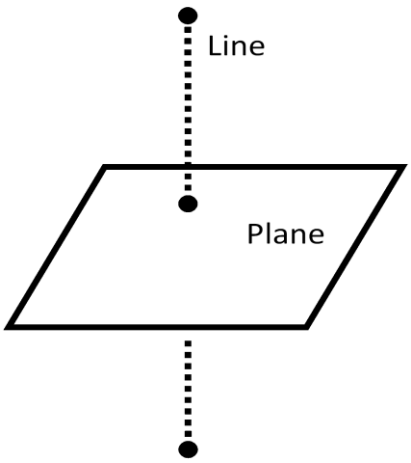


Figure 12: Line and plane intersection

For example, as shown in the Figure 13, the triangle $T_1T_2T_3$ intersects the cube at four different points with all the triangle vertices and edges lying outside the cube. The intersection point 'P' lies on the cube edge AB, and within the bounds of the triangle facet ($T_1T_2T_3$). The intersection point is valid if it satisfies both the following conditions.

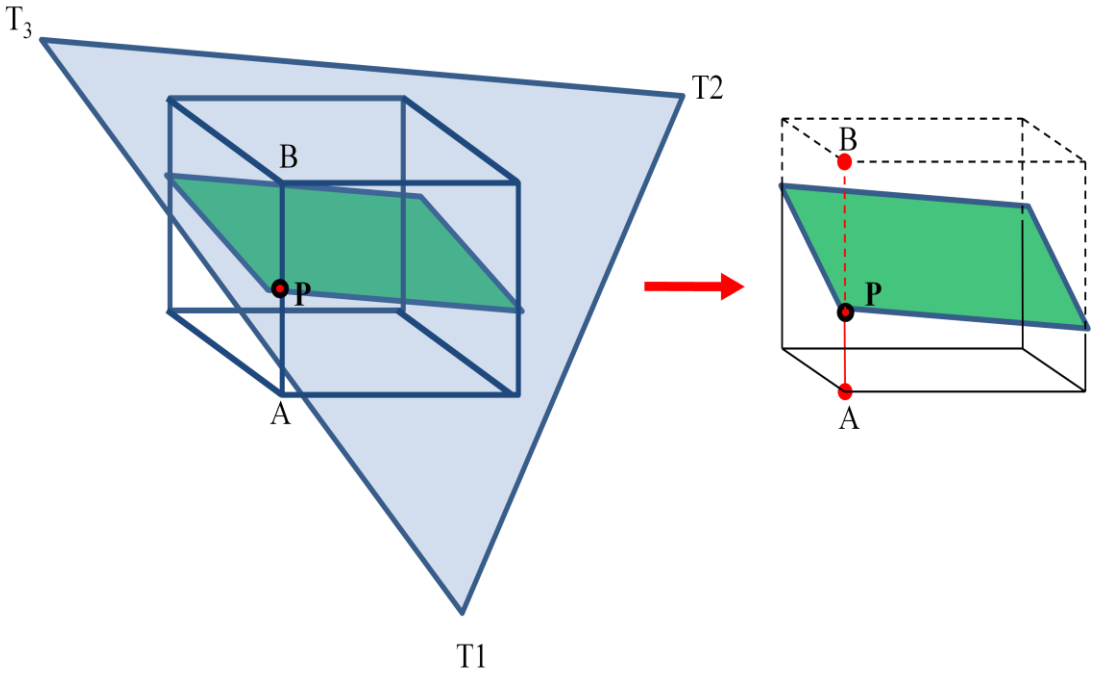


Figure 13: Cube edge and triangle face intersection

Point ‘P’ is intersection of the triangle facet $T_1T_2T_3$ and the cube edge AB if,

(a) As shown in Figure 14, the point P lies within the edge bound (AB) that is verified by,

$$\text{Distance (AB)} = \text{Distance (AP)} + \text{Distance (PB)} \quad (2)$$

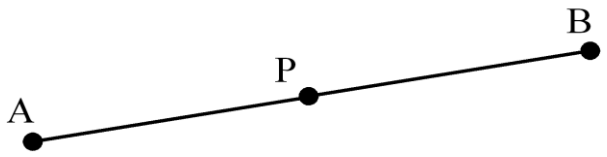


Figure 14: Line bound AB for a point P

(b) As can be seen in Figure 15, P lies within triangle facet bound ($T_1T_2\ T_3$) of the cube if,

$$\text{Mod [Area } (\Delta T_1T_2T_3)] = \text{Mod [Area } (\Delta PT_2T_3)] + \text{Mod [Area } (\Delta T_1PT_3)] + \text{Mod [Area } (\Delta T_1T_2P)] \quad (3)$$

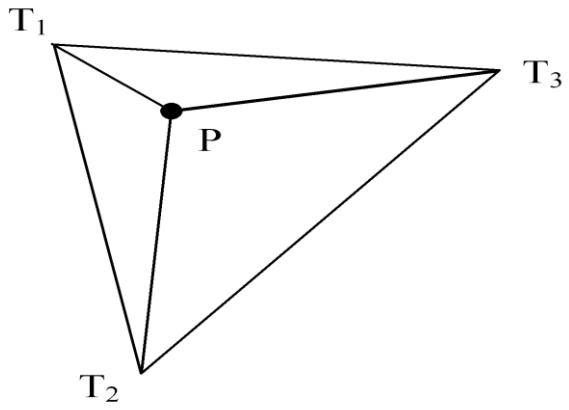


Figure 15: Point P in a triangular facet bound

A “non-terminal white node” from the previous point in a cube test is reclassified as a gray node if even one of the triangular facets of an STL file intersects with any of the cube faces. The corresponding intersecting triangle information such as the triangle’s vertices and normal to the triangle is stored in the cube structure. All the gray nodes, irrespective of their classification

following the first two tests, will undergo the next test to check for the existence of triangle-edge cube-facet intersections. All the "non-terminal white nodes" are also passed onto the next test.

3.1.2.3 Cube Facets and Triangle Edges Intersection Test

This test follows the Cube Edges and Triangle Facets Intersection Test. There is a possibility that there are no triangle vertices inside a cube and no triangle facet-cube edge intersections. In such a situation, the previous tests will fail to detect any existing triangle-cube intersection.

As shown in the Figure 16, the triangle $T_1T_2T_3$ intersects with cube at four different points. The triangle passes through the cube faces without touching any of the cube edges such that all the triangle vertices lie outside the cube. Hence, the cube face and triangle edge intersection test is necessary to detect an intersection and to decide whether a cube is a gray node or a terminal white node.

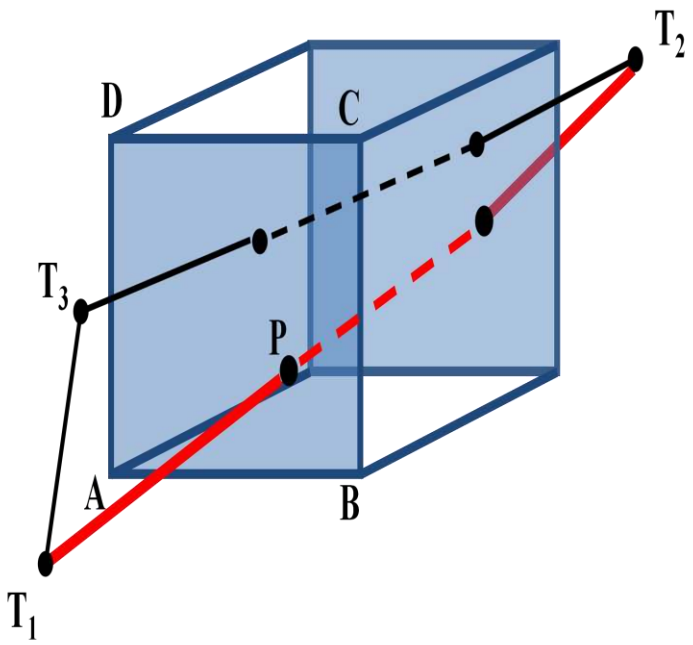


Figure 16: Cube face and triangle edge intersection

A line and plane intersection algorithm, as discussed in the previous section, is used to find out the intersection point of a cube face and a triangle edge. If the intersection point satisfies the cube facet and triangle edge bounds then it is a valid intersection as discussed below.

Point 'P' is an intersection of cube face ABCD and triangle edge T_1T_2 if,

(a) P lies within bounds of T_1 and T_2 that is verified by,

$$\text{Distance } (T_1T_2) = \text{distance } (T_1P) + \text{distance } (PT_2) \quad (4)$$

(b) As shown in Figure 17, P lies within bounds of the face(ABCD) of the cube and is verified by,

$$\text{Mod [Area } (\square ABCD)] = (\text{Mod [Area } (\Delta APD)] + \text{Mod [Area } (\Delta DPC)] + \text{Mod [Area } (\Delta BPC)] + \text{Mod [Area } (\Delta ABP)]) \quad (5)$$

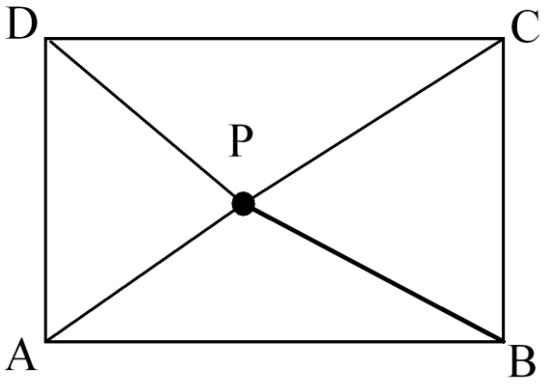


Figure 17: Point P in a square facet bound

If a cube has at least one such intersection point then the cube is a gray node and if it does not have a single valid intersection point then the cube is termed as a terminal white node. At the end of these three intersection tests, all cubes can be differentiated into gray and terminal white nodes. Gray nodes will hold the information of the respective intersecting triangles in their structure. Terminal white nodes are insignificant to represent the surface of an object and will be

ignored. The gray nodes will then be processed through three new tests to check for their eligibility for further division.

3.1.3 Conditions for Further Division of a Gray Node

Gray nodes thus obtained from the cube-triangle intersection detection tests may be a coarse surface representation of an object. These gray nodes are checked against three other conditions to decide their eligibility for further division. These conditions will help in decomposing an object up to the desired level of accuracy.

3.1.3.1 Smallest Allowable Cube Test

In an AM process, the layer thickness at a particular level cannot be less than the minimum allowable slice thickness for the AM machine and consequently the size of the smallest possible cube in the octree representation is equal to minimum allowable slice thickness.

Min allowable slice thickness = Size of smallest possible cube in the octree representation

The division of a gray node is terminated if its size matches the smallest cube size calculated above, and such gray nodes are termed as terminal black nodes. AM machine parameters and the smallest possible cube are the factors used to predetermine the depth of cube division in an octree representation.

3.1.3.2 Geometry of the Object inside a Gray Node Test

Gray nodes larger than the smallest possible cubes will undergo a second test. In this test, the geometry of the object (i.e. geometry of intersecting triangular facets of the STL file) inside a gray node is checked. As mentioned earlier, every cube has information about the list of triangles intersecting that cube. This information is used to describe the geometry of the object inside the

cube. Surfaces that are parallel or perpendicular to the build direction (positive Z axis) do not contribute to staircase effect. If all the intersecting triangles are parallel and/or perpendicular to Z-axis then the gray node is not divided further and is termed as a terminal black node. In Figure 18, all the intersecting triangles are either parallel or perpendicular to cube faces, and hence the cube represented becomes a terminal black node.

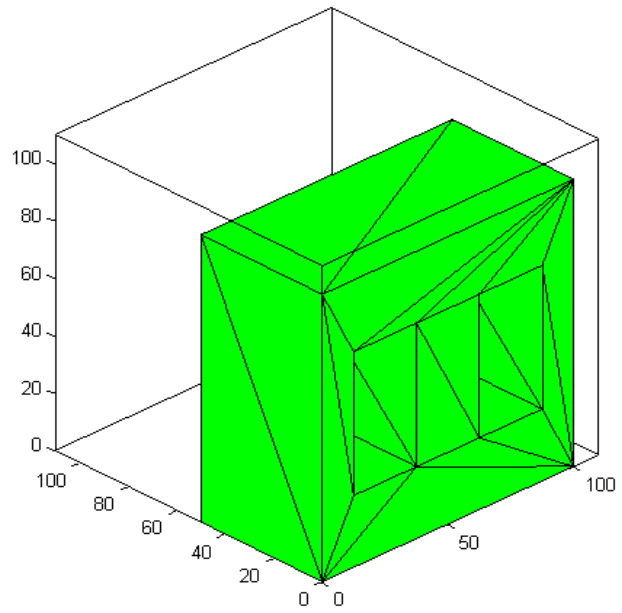


Figure 18: Cube and STL file of a cuboid with two slots

The scalar product of the triangle surface normal stored in the STL file, and the unit positive Z-axis is used to determine its orientation inside a cube. If the value of the scalar product is 0 then the triangle is parallel to the Z-axis and if the absolute value of the scalar product is 1 then the triangle is perpendicular to the Z-axis. In both these cases, the cubes will be deemed as terminal black nodes.

All the other remaining gray nodes are larger than the smallest possible cube and have intersecting triangles at an inclined angle to the Z-axis therefore necessitating the performance of a third test to determine whether these gray nodes are to be divided further.

3.1.3.3 Desired Level of Accuracy (User Defined Volumetric Tolerance) Test

A gray node intersects the STL file such that a portion of the cube is inside the STL file and the remaining portion is outside. In this test, the volume of a cube outside the object is computed. The user can control the volume of the cube outside the object, termed as the volumetric tolerance for the cube. If a gray node satisfies this user defined volumetric tolerance, it is stored as a terminal black node, else it is divided into 8 child cubes. This provides a mean of controlling the resolution of the octree based on volumetric error at each node. The algorithm to calculate volume of a cube outside an object is explained in detail below.

3.1.4 Volume of a Cube outside an Object

A ray casting algorithm [42] is used to calculate the volume of a gray node outside the object. For the purpose of the algorithm, the top face of the cube is divided into 'nxn' number of squares. The center points of these squares are termed as grid-points. Rays are extended from these grid points to the opposite face of the cube, parallel to the negative Z axis, as shown in Figure 19.

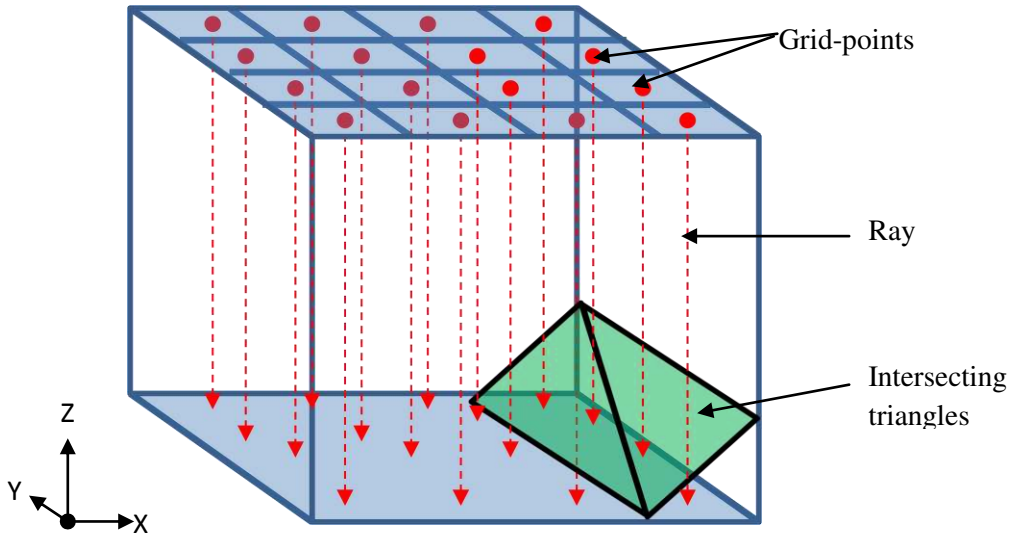


Figure 19: Grid-points and casted rays in a cube

An algorithm is presented that categorizes the grid points as being either outside or inside of a given STL object. Rays shot from the grid points of a cube may intersect with the object present inside the cube at various points. Valid intersection points are to be determined for every triangular facet of the object inside the cube. Valid intersection points lie within the bounds of both the cube height and the triangle facet. Intersection points lying outside the ray or the triangle facet are discarded. Sometimes a ray may lie in the same plane as a triangle, intersecting with the triangle at infinite points. All such intersections will be neglected.

For each ray, the normal of the first intersecting triangle is the key factor in determining the grid point's position with respect to the STL file facets. In an STL representation of the object, the normals of all the triangles face outwards and therefore, if the face normal of the first intersecting triangle is in general opposite to the ray direction, then the grid-point is outside the object, or else it is considered to be inside the object.

From Figure 20, it is clear that the ray from the grid point P is directed opposite to the normal of the first intersecting triangle and therefore the grid point 'P' lies outside the green colored object.

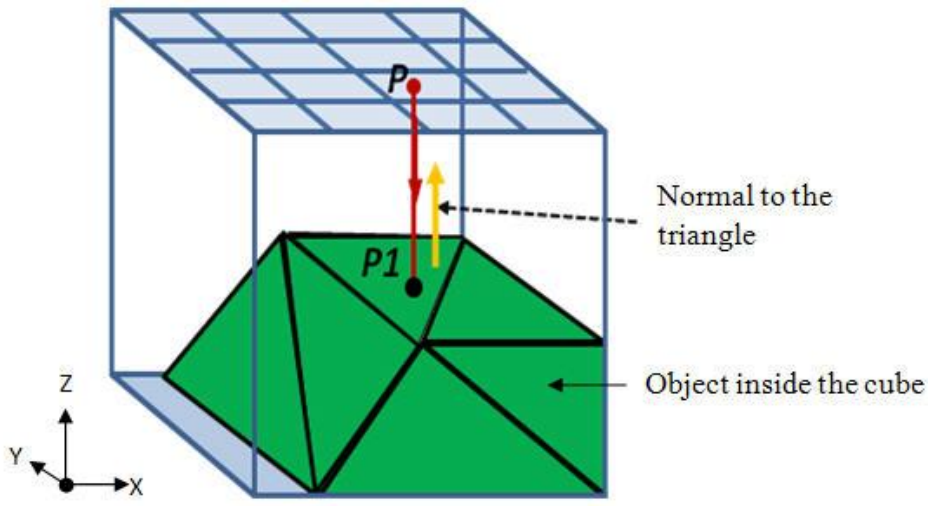


Figure 20: Intersection of a casted ray and triangles inside the cube

Some grid-points may cast rays that do not intersect with any of the triangles inside the cube. Positions of such grid-points can be decided using grid-points with known locations. As shown in the Figure 21, the ray from grid point Q does not intersect with any of the triangles inside the cube. To determine location of such a grid point, another ray is extended from Q to P, whose location is known. If there is an even number of intersections or no intersections with the triangles then the point Q is also outside the object. If there is an odd number of intersections then the point Q is inside the object.

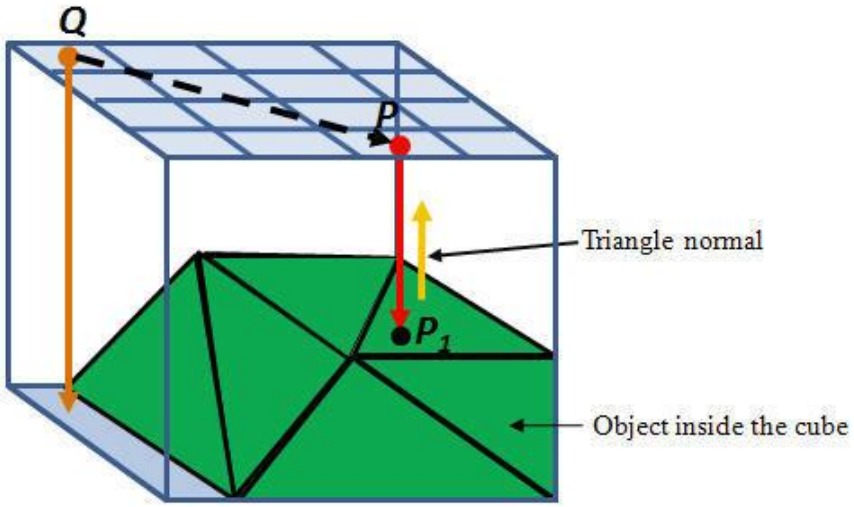


Figure 21: Location of grid point A1 with respect to object inside the cube

At the end of the test, all the grid points are separated into those that lie within the object and those that lie outside the object.

3.1.4.1 Special Case: Intersection at a Common Edge or a Common Vertex

If a ray shot from a grid point intersects with two or more triangles at a common vertex or a common edge then the grid point is shifted in the same plane by a very small value 'δ' (relative to the grid size). This resolves the issue of more than one intersecting triangle with normals in

different directions. In Figure 22, the ray shot from the grid point P intersects with two different triangles at point P_1 at a common edge. Grid point P is shifted by a very small distance 'δd' in the same plane to avoid the multiple intersecting triangles. This process is repeated until the rays intersect triangles at distinct points.

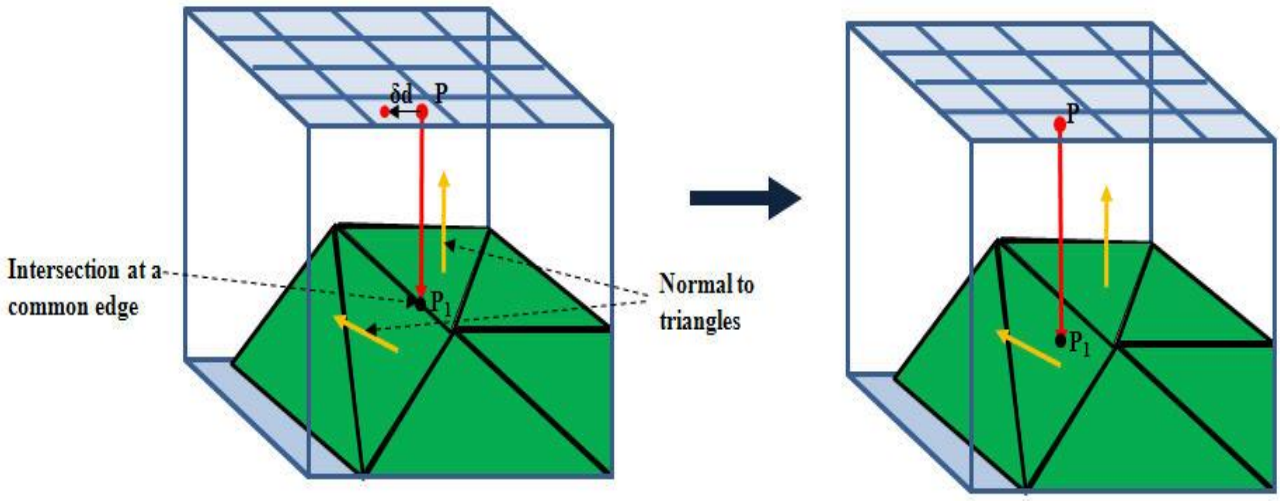


Figure 22: Ray intersecting at a common edge or a common vertex of triangles

3.1.4.2 Segment of a Ray Outside the Object

For a casted ray, once valid intersections with the triangles inside a cube are calculated, in the next step the segments of the intersecting rays lying outside the object are computed. The segment preceding the first intersection will always be outside for a ray casted from a grid point lying outside the object. Following the first intersection every alternate segment will lie outside the object. In Figure 23, the green polygon is a two-dimensional representation of the object while the square represents the intersecting cube. The grid point P lies outside the object and hence the segment from the grid point to the first intersection, P-P1, is outside the object. Again the segment from the second intersection to the third intersection, P2-P3, lies outside the object

and so on. A ray from a grid point outside the object that does not intersect with any triangle lies totally outside the object.

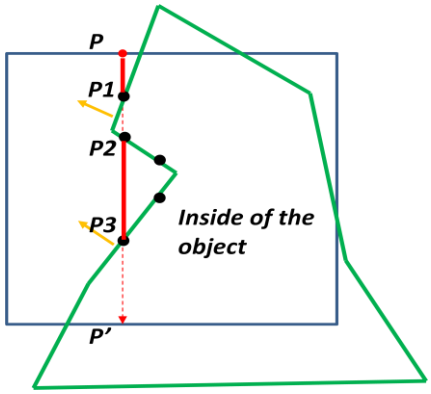


Figure 23: Section view of a cube, STL file and a casted ray from outside grid point

Similarly, for a grid point lying inside an object, the segment between the first and second intersections and every alternate segment thereon lie outside the object. The grid point Q in Figure 24 is inside the object and the ray segment between the first and second intersections, Q1-Q2, is outside the object and so on.

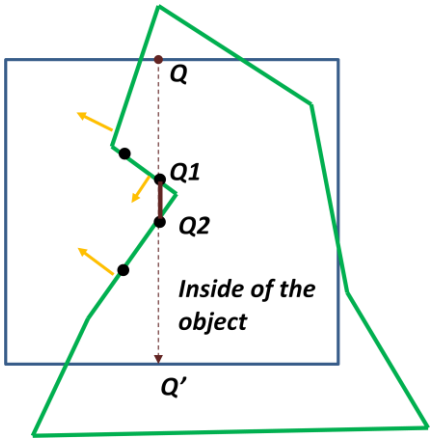


Figure 24: Section view of a cube, STL file and a casted ray from inside grid point

The algorithm thus computes all the segments of every ray that lie outside the object which are then used to calculate the volume of the cube which remains empty.

3.1.4.3 Volume of a Cube outside an Object

With the knowledge of all the ray segments lying outside the object, the following formula is used to calculate the volume of the cube outside the object.

$$\% \text{ volume of a cube outside an object} = \frac{\sum(\text{length of segments of all rays outside the object x } 100)}{(\text{Total number of rays X Cube length})} \quad (6)$$

The percentage volume of the cube outside an object is multiplied by total volume of the cube to calculate the volume of the cube outside the object, as given below.

Vol. of a cube outside an object = Cube vol. x *Percentage vol. of the cube outside the object* (7)

The volume of the cube inside an object is calculated by subtracting the total cube volume from the above derived empty cube volume, as given below.

Vol. of a cube inside an object = Cube vol. – Vol. of the cube outside the object (8)

Volume of cube outside the object is then compared to the user defined volumetric tolerance. The volumetric tolerance is the maximum volume of a gray node that may remain outside the given object. It is a user desired value for the entire octree representation. If a given gray node satisfies the volumetric tolerance provided by the user, then it is termed as a terminal black node, else it is further divided into 8 child cubes. These child cubes then undergo the entire series of processes that its parent cube was subjected to.

In the case that the volume of the object lying within a given cube is less than the volumetric tolerance, the cube is discarded, since the removal of that cube from the octree representation does not significantly affect the surface quality of the object.

At the end of this MBODS algorithm, the STL file is converted to terminal black nodes of different sizes. These terminal black nodes are then utilized to compute the slice thicknesses at different levels.

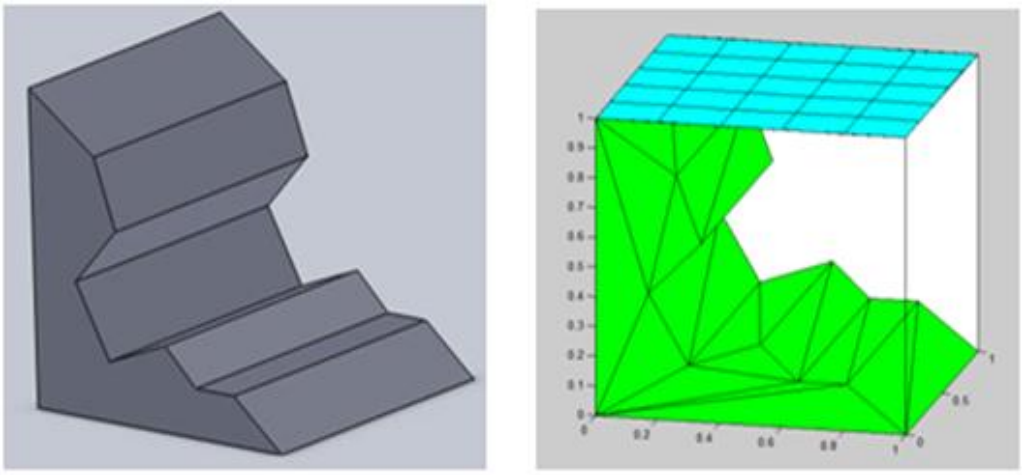
3.1.4.4 Volume of the Cube outside an STL file: examples

The ray casting algorithm is used to determine the volume of any gray node outside an STL file. Two examples are presented below to explain and validate the algorithm.

Example 1: An object, as shown in Figure 25, is modeled using a CAD package (SolidWorks) and exported as an STL file. A cube is created encompassing the STL file. The top face of the cube is divided into a square matrix with the centers being grid points and rays are extended from these grid points, parallel to the negative Z axis, reaching the bottom face of the cube.

Part specifications:

- Side of the cube = 1 mm
- Volume of the cube = 1 mm^3
- Actual volume of the CAD object = 0.37 mm^3
- Actual volume percentage of the cube outside the object = 63%



(a) (b)

Figure 25: (a) CAD model (b) STL file with grid points for example 1

The ray casting algorithm is then applied several times to the object and the cube, by varying the number of grid points each time. The corresponding results are presented in Table 3.

Table 3: Volume of cube outside STL file for example 1

No of grid points	% by which the cube is empty (using ray casting method)	Actual % by which the cube is empty
9	57.974	63
25	60.542	
36	61.6287	
64	61.8148	

It is clear from Table 3 that the degree of accuracy in calculating the volume of a given cube outside an object increases as the number of grid points increases. With 9 grid points, there is a volumetric error of 7.97% whereas the error reduces to 1.88% for 64 grid points.

As can be gathered from the graph in Figure 26 the accuracy in computing the cube's unoccupied volume percentage increases steeply as the grid size increases. As the number of grid points increases beyond a certain level, the predicted value asymptotically approaches the actual fraction of the cube's empty volume.

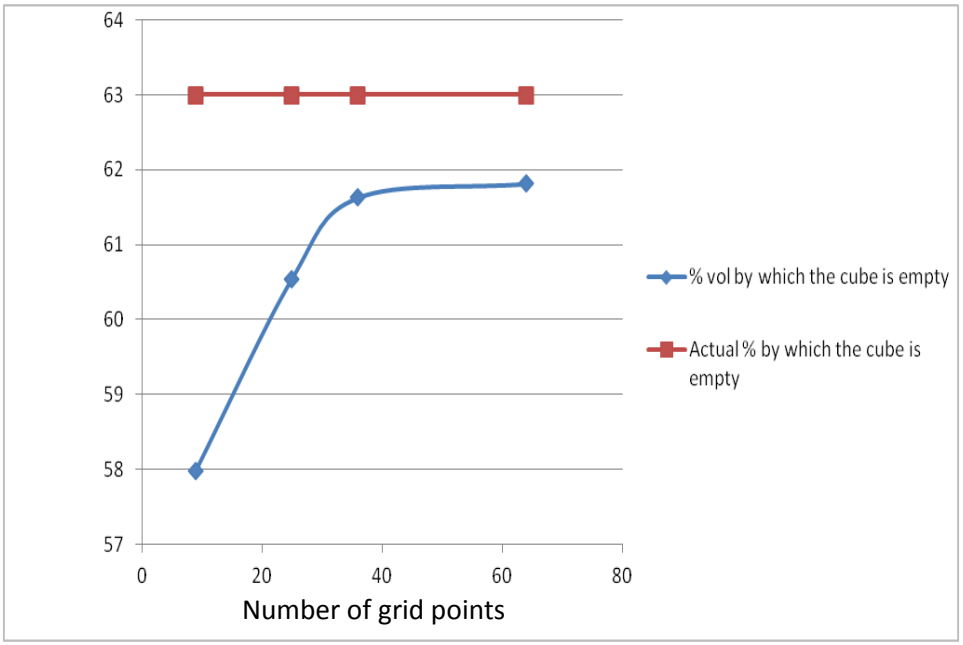


Figure 26: Percentage volume by which the cube is empty against the number of grid points for example 1

Example 2: A section of a sphere, in the first quadrant of the Cartesian coordinate system is used as the second test case. The test object is modeled in SolidWorks and exported as an STL file. A cube is then created enclosing the entire object.

Part specifications:

- Sides of the cube = 1 mm
- Volume of the cube = 1 mm^3
- Actual volume of the object = 0.52 mm^3
- Actual volume % of the cube outside the object = 48%

Figure 27 shows the STL file of the test object in the first quadrant enclosed within a unit cube. The ray casting algorithm is used to calculate the unoccupied volume of the cube. This calculated value is then compared to the actual volume the cube that is empty.

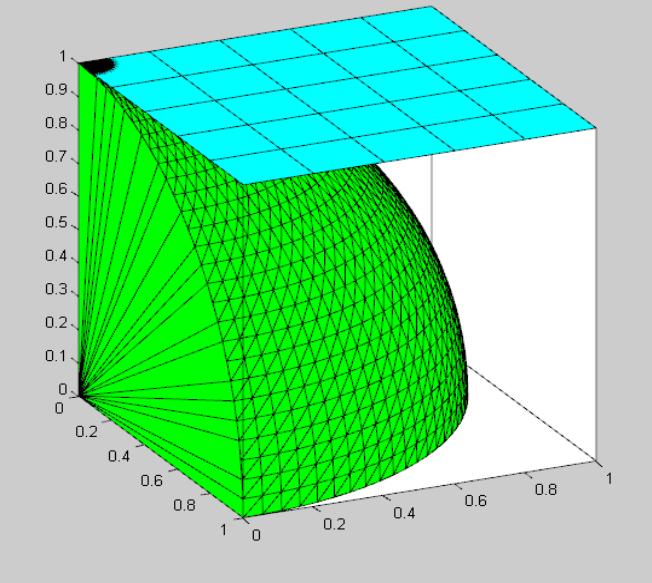


Figure 27: STL file with grid points for example 2

The top face of the cube is divided into different number of grid points and the ray casting algorithm is applied to the different grid sizes. The results are presented in Table 4.

Table 4: Volume of cube outside STL file for example 2

No of grid points	% by which the cube is empty (using ray casting method)	Actual % by which the cube is empty
9	58.248	48
25	52.365	
36	50.459	
64	49.636	

Data from Table 4 confirms that the accuracy of the ray casting algorithm increases as the number of grid points is increased. For 9 grid points, there is a 21 % error in accuracy between the calculated and the actual volumes of the empty space in the cube, whereas there is only a 3.4% error in the accuracy when 64 grid points are used. The variation of the cube's unoccupied volume percentage with the grid size exhibits trends similar to those observed in the previous example, as illustrated in Figure 28.

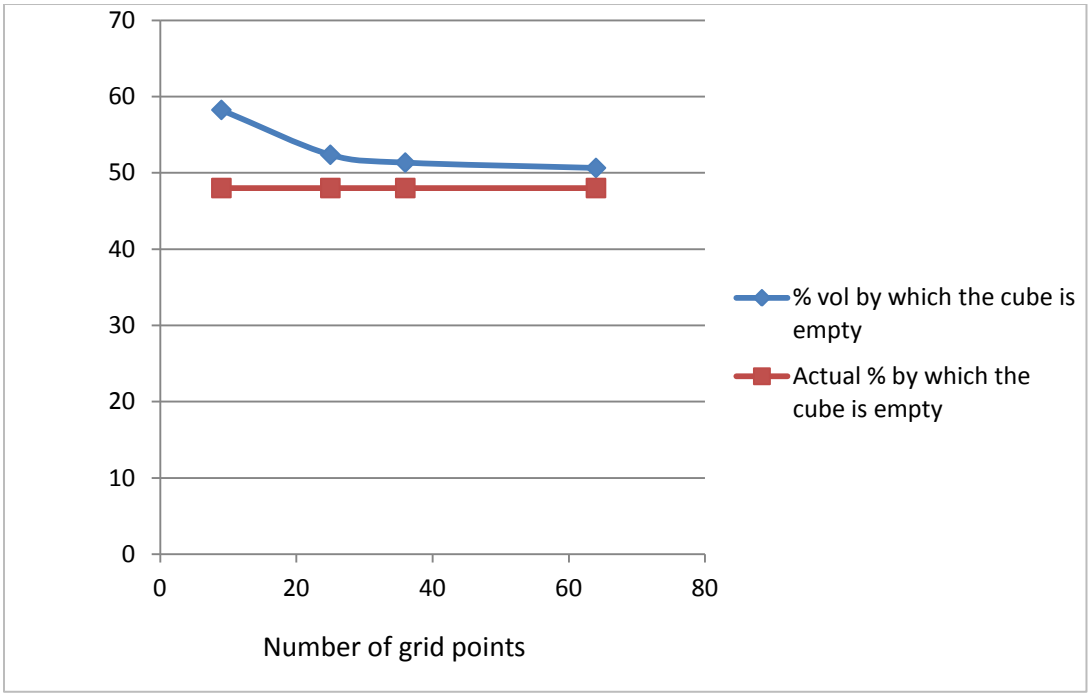


Figure 28: Percentage volume by which the cube is empty against the number of grid points for example 2

3.2 Computation of Slice Thicknesses from the MBODS

The terminal black nodes obtained from the MBODS algorithm present an approximation of the STL file based on volumetric error. Their information is utilized to compute the local slice thicknesses which are employed in building the part. The overall approach is that the local slice thickness at a particular level is decided based on the size of the smallest cube at that level. The steps involved in the calculation of the slice thickness are presented below:

Step 1: A cube, having the smallest size amongst the terminal black nodes derived from the MBODS algorithm is selected. There may be many such cubes which satisfy this criterion and any of these cubes may be initially selected. The slice thicknesses at each cube's level are then fixed within the 'Z' coordinates of the selected cube. In this algorithm, all the terminal black nodes are arranged in the ascending order of their size. An array with the terminal black nodes structures in ascending order is shown below.

Terminal cube ascending order = [Smallest black nodes …… Largest black nodes]

The first smallest cube is selected and the slice thickness at that level is fixed between the two Z coordinates of the specific cube. For example, if a cube's origin is located at (x_1, y_1, z_1) and its height is T, then the local slice thickness is fixed between z_1 and z_1+ T.

$$z_1 \leq \text{Local slice thickness} \leq z_1+T$$

Step 2: As mentioned previously, the AM machine has specific layer thickness parameters, namely, minimum allowable slice thickness (T_{min}) and maximum allowable slice thickness (T_{max}). Slice thickness at any level has to be within this range. If size of the selected cube lies within T_{max} and T_{min} then the cube height is selected to be the slice thickness for that level. In Figure 29, the size of the selected cube 'T' lies within the minimum and maximum range and therefore T is chosen to be the slice thickness at that level. There are three cases to consider.

- Case 1: $T_{min} \leq T \leq T_{max}$; Final slice thickness = T

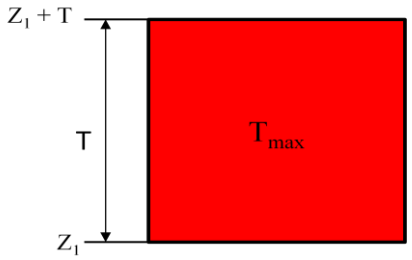


Figure 29: Slice thickness at level z1 for case 1

If the selected cube has a height greater than T_{max} then it is divided into two parts, a multiple of T_{max}, and the residual height. As shown in Figure 30, the cube height is greater than twice the value of $T_{max.}$ The corresponding slice thicknesses would be T_{max}, T_{max} and the residual height of the cube. In this case the residual slice would be greater than T_{min} and less than $T_{max.}$

- Case 2: $T_{max} \leq T$; Residual slice $\geq T_{min;}$ Final slice thicknesses = T_{max}, T_{max} and the residual slice

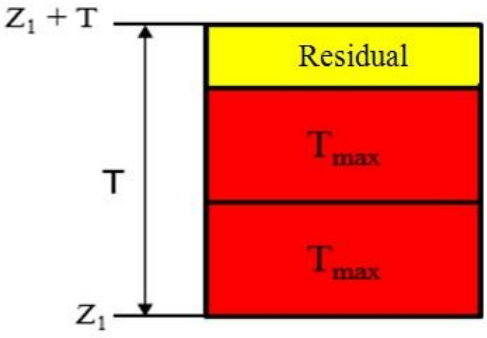
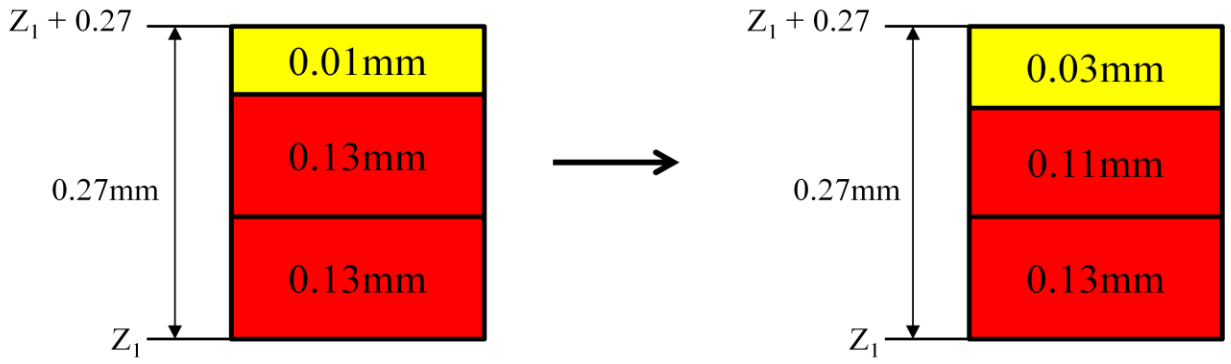


Figure 30: Slice thickness at level z1 for case 2

If the residual slice is less than T_{min}, its dimension is changed to match with T_{min} and corresponding changes are applied to the thickness of the preceding slice. For the example shown in Figure 31(a), T_{min} is 0.03 mm and T_{max} is 0.13 mm and the residual slice thickness is

0.01 mm which is less than T_{min}. Thus, as shown in Figure 31 (b), the residual slice thickness is changed to 0.03 mm (T_{min}) and the thickness of the prior slice before that is modified with a corresponding value.

- Case 3: $T_{max} \leq T$; T_{max} = 0.13 mm and T_{min} = 0.03 mm; Initial slice thicknesses = 0.13 mm, 0.13 mm and 0.01 mm; Residual slice = 0.01 mm $\leq T_{min;}$ Final slice thicknesses = 0.13 mm, 0.11 mm and 0.03 mm



(a) (b)

Figure 31: Slice thickness at z1 for case 3; (a) In-correct slice thicknesses computation (b) Correct slice thicknesses computation

In a similar manner, slice thicknesses are determined over a range (z_1 to z_1+T in the given example) of the selected cube.

Step 3: Once the slice thicknesses are fixed for the level ranging from z1 to z1+T, the second smallest cube from the array is selected. The Z coordinates of this cube are checked and the cube is neglected if it lies within the same level (z1 to z1+T). For example, in Figure 32, the levels from 20 to 25 are fixed using the red colored cube and yellow cube is neglected from the algorithm since that level is already fixed using red cube. If the Z coordinates of the cube are at a

different level than that determined above, then step 2 is repeated for this cube too. Then the next smallest cube is selected and the process is repeated until the slice thicknesses for the entire object height are determined.

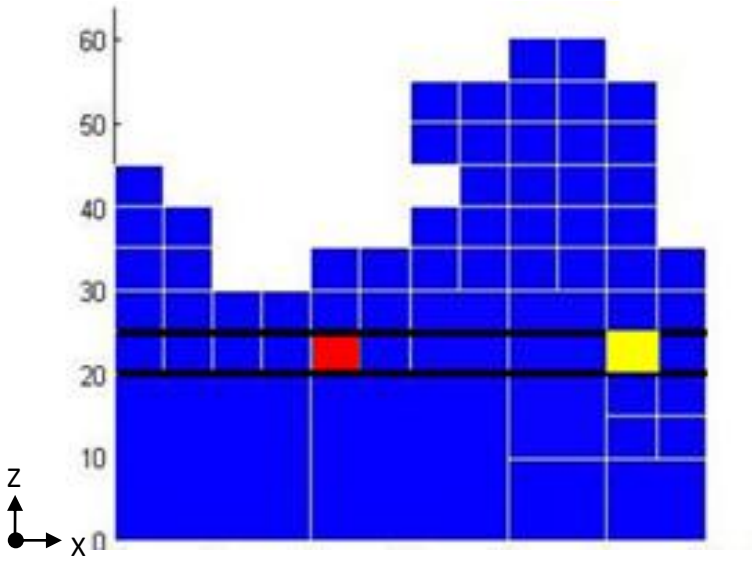


Figure 32: MBODS; X-Z view of an object for case 1

Special cases

There may be some cases where the selected cube has some portion of its height overlapping with an already fixed slice level. In this case, the overlapping portion is neglected and the remaining portion is considered to compute the slice thicknesses. There are two such cases in which the Z coordinates of the selected cube may overlap an already fixed slice levels.

Case 1: In this case, as illustrated in Figure 33, slice thicknesses from Z_1 to Z_1+T_1 are already fixed using a smallest cube at that level and the next smallest cube has a portion of its height overlapping with previous smallest cube. This overlapped part is neglected and only the remaining part of the current smallest cube is considered. That is, only the slice thicknesses from Z_1+T_1 to $Z_1+ (T_2- T_1)$ are decided at that stage.

$$Z_1+T_1 \leq \text{Local slice thickness} \leq Z_1+ (T_2 - T_1)$$

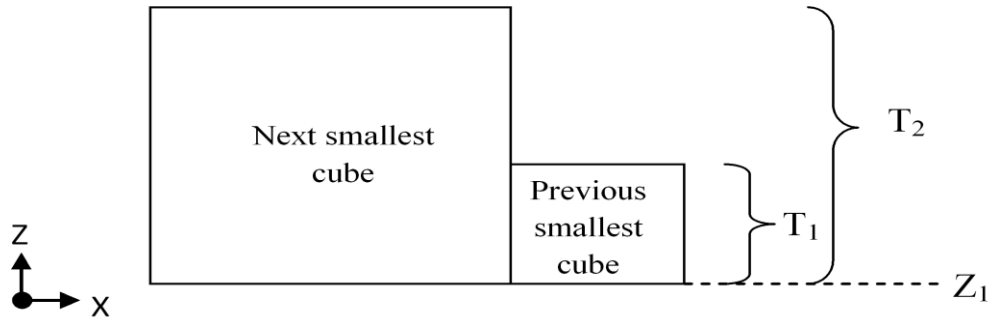


Figure 33: X-Z view of two terminal black nodes for case 1

Case 2: As in the previous case, the slice thicknesses from Z_1 to Z_1+T_1 are already fixed and the next smallest cube has a portion of its height overlapping with previous smallest cube, as illustrated in Figure 34. This overlapped part is neglected and only the slice thicknesses from Z_1 to Z_2 are decided at this stage.

$$\mathbf{Z_2 \leq \text{Local slice thickness} \leq Z_1}$$

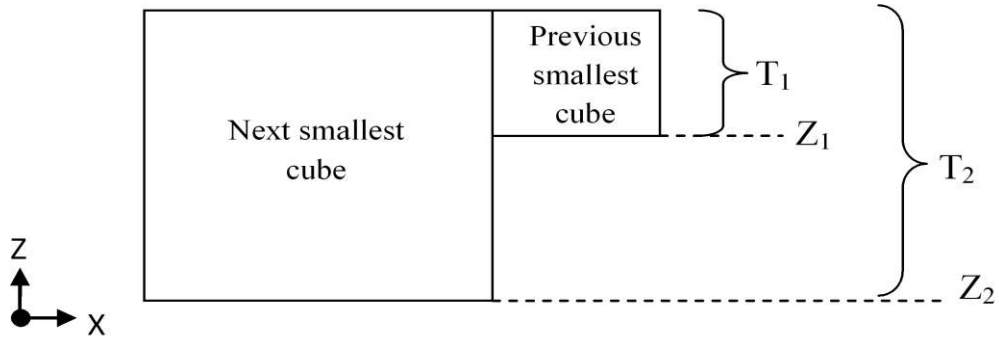


Figure 34: X-Z view of two terminal black nodes for case 2

There is yet another condition where a cube may contain the topmost part of the object as shown in Figure 35. If the object's height is lesser than the upper Z coordinate of that cube, then the slice thickness is decided based on the object's height within. In this case, only the portion of the cube from Z_1 to Z_2 is considered for slice thickness.

$Z_1 \leq$ Local slice thickness $\leq Z_2$

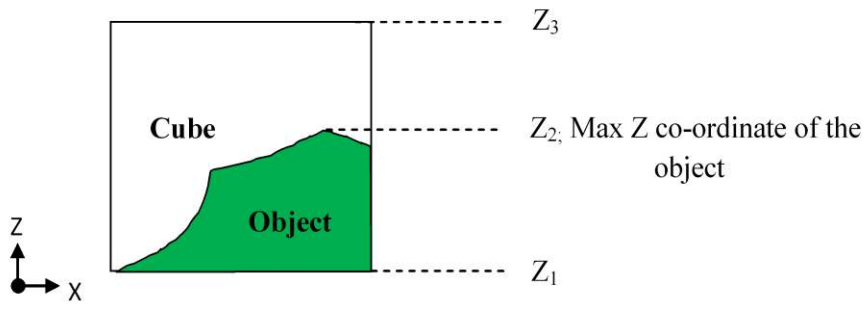


Figure 35: X-Z view of an object tip in a terminal black node

The object is checked against the summation of all the adaptive slice thicknesses computed over the entire algorithm and the algorithm is terminated when all the slice thicknesses sum up to the height of the object. The slice thicknesses thus obtained are used in the subsequent AM process.

3.3 Virtual Manufacturing (VM) of STL file

As mentioned, parts manufactured by AM processes exhibit a staircase effect, thus compromising on the surface quality. Typically, manufactured parts that do not satisfy the tolerance requirements are rejected and rebuilt. Virtual manufacturing plays an important role in identifying and correcting these wastes. A part is fabricated only when it successfully passes through all the inspection tests in the VM phase. Virtual adaptive manufacturing of the part is used to verify part tolerances for the part built with MBODS algorithm.

For the virtual manufacturing of a part its CAD model is extracted as an STL file. Adaptive slice thicknesses, calculated from the MBODS algorithm presented in the previous section, are used in the virtual manufacturing of the object. In VM, the slicing planes are always perpendicular to the Z-axis of the AM machine. At a given slice level, the slicing plane intersects

the triangles of the STL file. These intersection points are designated to be the contour points of that particular level. Figure 36 shows the contour points obtained by slicing the STL file using a slicing plane at a particular height.

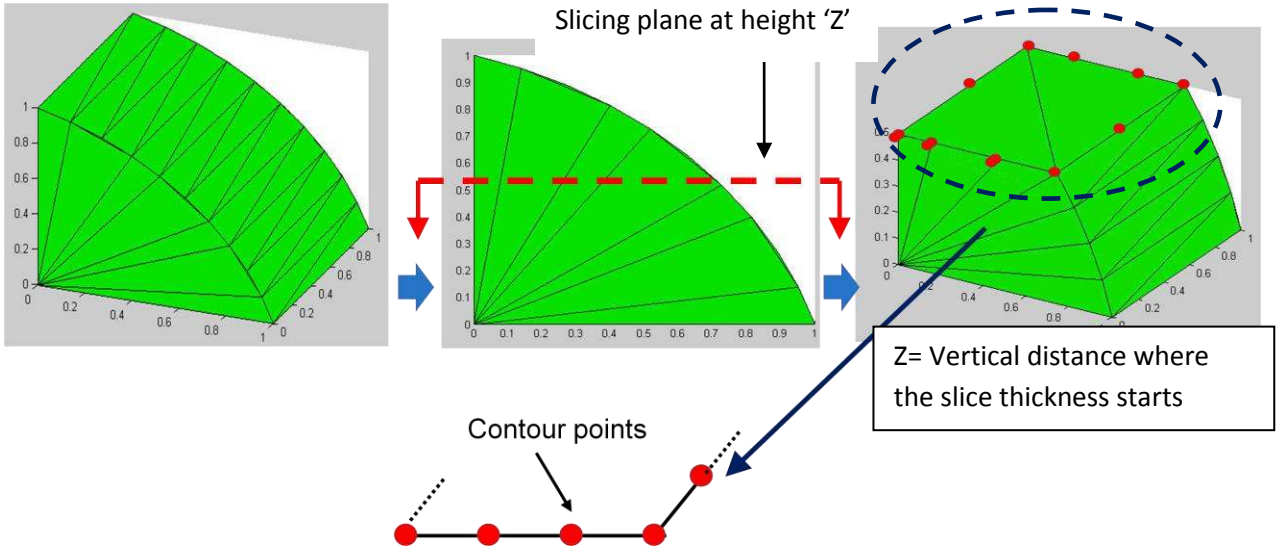


Figure 36: Contour points for a particular slicing plane at height Z

The contour points of a slicing plane are projected on a plane parallel to the slicing plane by an offset value equal to the slice thickness at that level, as illustrated in the Figure 37. Therefore, two sets of points are obtained for any particular slice level. The contour points and their projected counterparts from all the slicing levels are collected in a simulated manufacturing point dataset and are utilized for virtual manufacturing of the object.

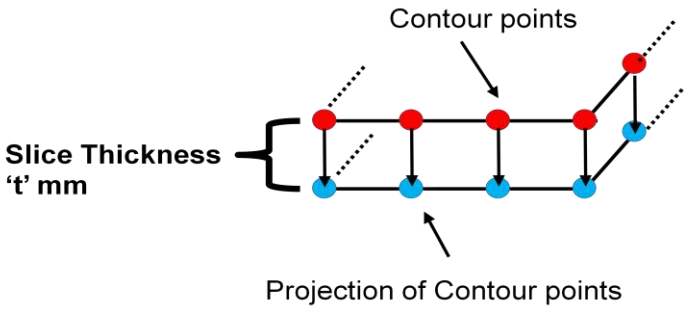
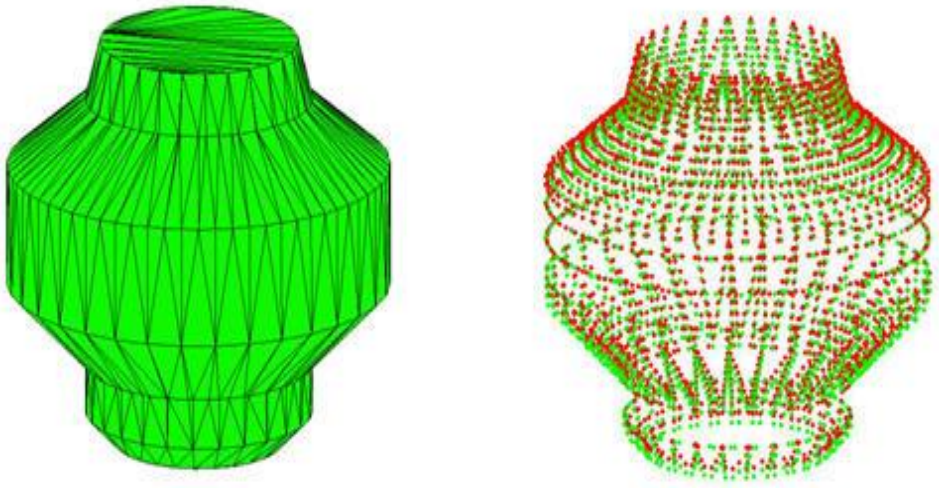


Figure 37: Projected contour points

Figure 38 shows an STL file and the contour points for all the corresponding slices (designated as red dots). Green points in the figure are projections of these contour points on parallel planes offset by the slice thicknesses at the respective levels.



(a) (b)

Figure 38: (a) STL file (b) Simulated virtual manufacturing point dataset

3.4 Evaluation of Errors

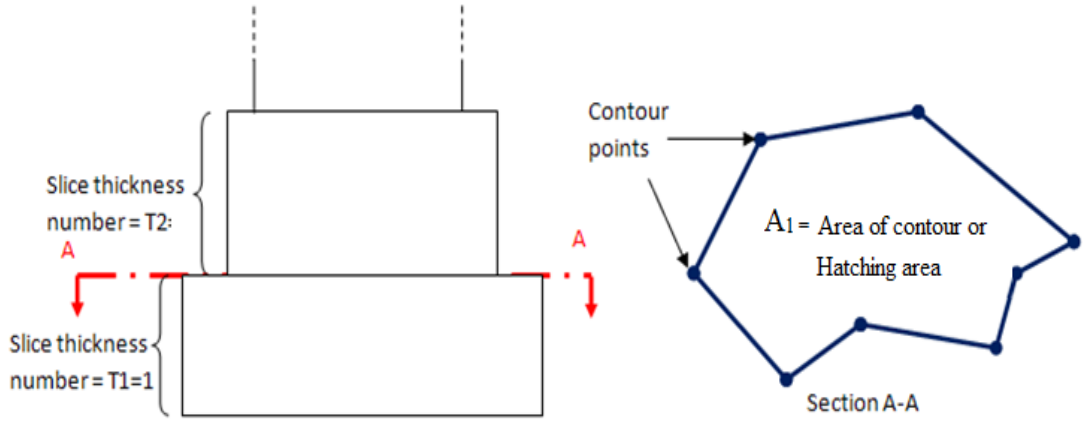
In this section, the procedure for evaluating the volumetric error and the profile error of a given part are discussed. The simulated manufacturing point dataset, evaluated previously is used to calculate the volumetric error of a part. Further, the shortest distances between every point in the simulated point dataset and the CAD surface of the object are calculated which are then used to evaluate the profile error.

3.4.1 Evaluation of Volumetric Error

The volumetric error is defined as difference between the actual volume of an object and the volume of the manufactured object [43]. In the present research, since the object is virtually

manufactured, the volume of the virtually manufactured part is subtracted from the volume of the actual part to calculate the volumetric error.

As shown in Figure 39, the contour points from each slicing level are used to calculate the hatching area for that level. At every level, the entire hatching area is then extended by a value equal to the slice thickness at the respective level.The process is then repeated for all the available slice thicknesses and the object is virtually manufactured layer by layer. The formula to calculate the volumetric error for a given STL file is explained below [43].



(a) (b)

Figure 39: (a) Variable slice thicknesses (b) Contour created by first slice

If **n** is the number of slice thicknesses or number of slicing planes; **V**, the actual volume of the object; $\mathbf{T_1}$, $\mathbf{T_2}$, …, $\mathbf{T_n}$, the adaptive slice thicknesses obtained from slice thickness algorithm and $\mathbf{A_1}$, $\mathbf{A_2}$,…,$\mathbf{A_n}$, the hatching areas created by the contour points at the respective slicing levels**,** then,

$$\text{Volume of the VM object, } V_m = (A_1T_1) + (A_2T_2) + \ldots + (A_nT_n) = \sum_{i=1}^{n}(A_iT_i) \qquad (9)$$

$$\text{Volumetric error in virtual manufacturing} = V - V_m \qquad (10)$$

3.4.2 Evaluation of Profile Error

An STL file, as explained in section 2.2, is an orderly arrangement of triangular facets. In other words, the surfaces of a CAD model are approximated by a series of triangles. As a first step in calculating profile error, the triangles of an STL file related to a particular CAD surface are to be identified. The facet isolation algorithm, developed by Navangul [44] to identify the STL triangles related to a particular surface of an object is used for this purpose. In the next step, the contour points and the offset points obtained during VM of the part corresponding to the selected part surfaces are determined. These points are termed as selected point dataset. Then, the shortest distance between every point in the selected point dataset and the CAD surface is determined using Newton's method [31, 45-47].

In Figure 40, the profile error d_i is the minimum distance between the i^{th} point in selected point dataset and the corresponding CAD surface 'C'. Let $P_i(X_i, Y_i, Z_i)$ represent the points in the selected database and Q_i, the corresponding closest points on the design surface.

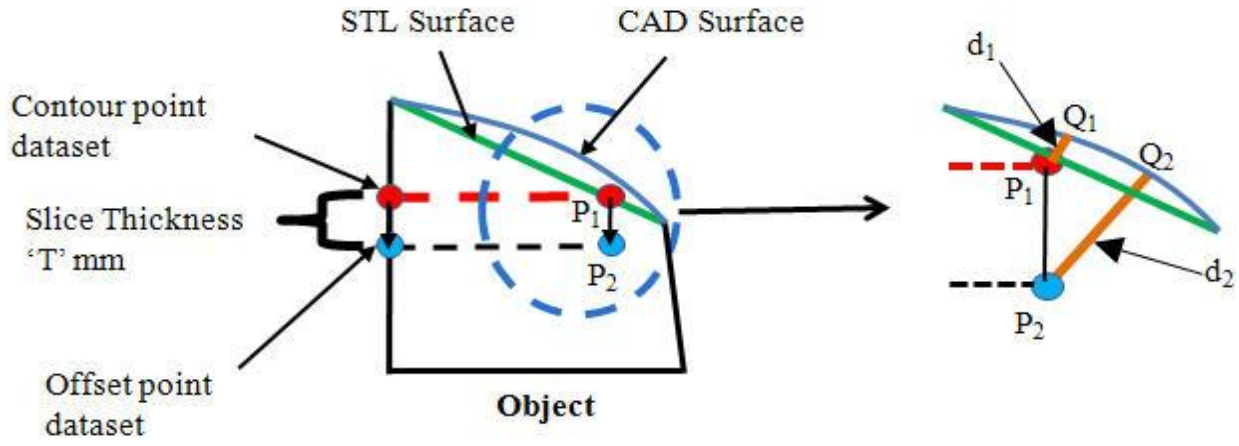


Figure 40: Minimum distance of an offset contour point from object surface

The profile error problem is a constrained non-linear optimization problem in u and v. The initial starting point $Q_g(u_i, v_i)$ that is closest to point dataset P_i is determined using a coarse

optimization algorithm [44]. A grid constructed from the points $Q(X,Y,Z)$ is obtained for different u and v values. 'Q_g', the closest point to P_i is then determined from the grid. The corresponding u and v values of the point Q_g are used as the initial starting values in Newton's method to find d_i, the shortest distance to the CAD surface. The formulation of Newton's method is as follows [44].

$$d_i = P_i - Q_g(u,v)$$

$$\text{Minimize: } \sum(d_i)^2$$

$$\text{Constraints: } 0 \le u \le 1 ,$$

$$0 \le v \le 1$$

where, 'P_i' is the point from the selected point database, and 'Q_i' is the final closest point on the design surface that can be calculated using the (u,v) values obtained from the optimization. The profile error in the part is then evaluated using the following formula.

$$\text{Profile error} = \text{Maximum } (d_i) \qquad (11)$$

where 'd_i' is the shortest distance to the CAD surface for all selected dataset points associated with that surface.

3.4.3 Evaluation of Cylindricity

As per ASME Y14.5, 1994 [48], cylindricity tolerance is defined as: "Cylindricity is a condition of a surface revolution in which all points of the surface are equidistant from a common axis." "Cylindricity tolerance specifies a tolerance zone bounded by two coaxial cylinders within which the surface must lie." [48].

As show in Figure 41, two coaxial cylinders, an inscribed cylinder and a circumscribed cylinder, are the boundary limits for all the point dataset of the cylindrical feature. Any perfect

cylinder description needs the location of the axis and its orientation in space. This perfect cylinder is used as a reference feature to evaluate cylindricity.

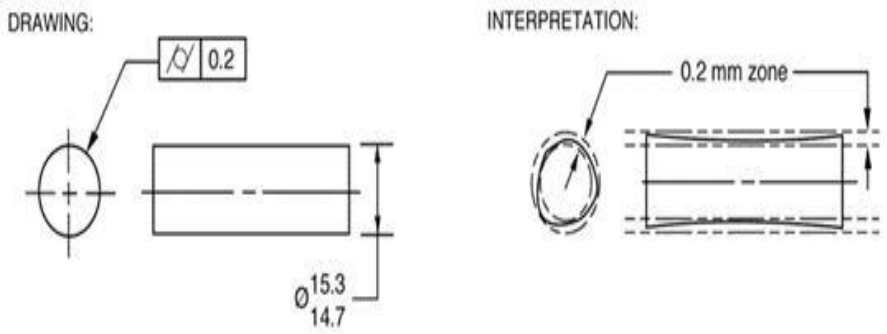


Figure 41: Cylindricity tolerance zone [49]

From Figure 42, let Q_k be a point on the least square axis of the coaxial cylinders and P, the unit direction vector of the axis. Q_i is the manufacturing point dataset from the virtually manufactured part.

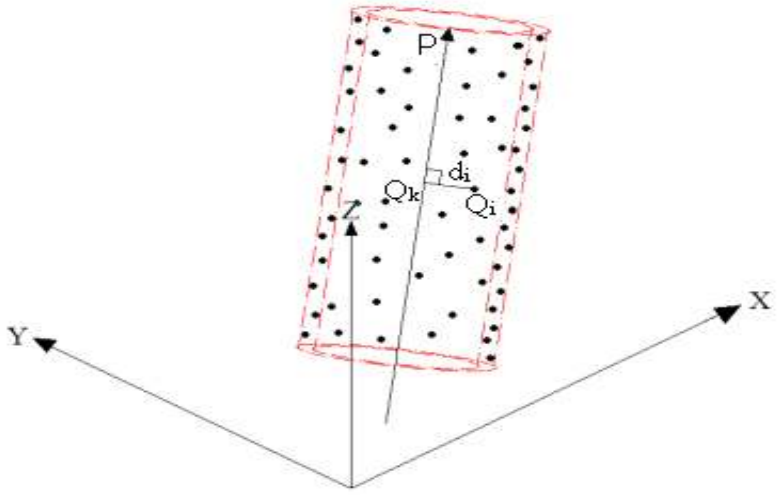


Figure 42: Evaluation of MX cylindricity (adapted from [40])

As explained in section 2.6, the least square and the minimum zone algorithms are usually used for the evaluation of cylindricity. In their cylindricity problem formulation, Carr

and Ferreira [36] used the least square algorithm as an input for the minimum zone algorithm. The contour points on a cylindrical surface and the related offset points from the simulated dataset are selected to evaluate the cylindricity tolerance zone. Contour points and the related offset points of the flat surfaces of the cylinder are insignificant and thus neglected. In this research, the cylindricity formulation developed by Ramaswami [50] is used to evaluate the cylindricity error of a part.

Let, R_1 and R_2 be the radii of the inner (inscribed) cylinder and the outer (circumscribed) cylinder respectively, P (P_x, P_y, P_z) is a unit vector along the axis of cylinder, Q_i (X_i,Y_i,Z_i) are points in the dataset for i=1, 2, 3,.., n and Q_k ($X_k$$Y_k$,$Z_k$) is a point on the axis of the inner and outer cylinders. The starting point Q_k for the algorithm is obtained from the least square axis. The objective is to find the axis of two concentric cylinders such that all the points in the measured dataset (Q_i) lie between the outer and inner cylinders with a minimal radial difference between the two cylinders.

The formulation of the problem can be written as [44, 50].

Minimize (R_2-R_1)

Constraints: $d_i (Q_k, P) \leq R_2$, for i =1, 2, 3…n

$d_i (Q_k, P) \geq R_1$, for i =1, 2, 3…n

Where, $d_i (Q_k, P) = | P \times (Q_i - Q_k) |$, for i=1, 2, 3…n

The first constraint ensures that all the points are on or inside the outer cylinder. The second constraint ensures that all points are on or outside the inner cylinder. The output of this formulation is cylindricity tolerance of the feature.

$$\text{Cylindricity tolerance zone} = (R_2 - R_1) \quad (12)$$

4. RESULTS

In this chapter the MBODS algorithm is applied for different examples and the results for volumetric, profile and cylindricity error are tabulated.

4.1 Modified Boundary Octree Representation and Evaluation of errors of Object 1

The first test object chosen for this research, termed as 'object 1', has a complex structure with a curve surface. The model is created in SolidWorks and exported in an STL file format. The part, AM machine specifications and process parameters are as follows:

Part specifications:

1) Height = 1.5mm
2) Width = 1mm
3) Breadth = 1 mm
4) Actual volume of the object = 1.16mm^3

AM machine specifications:

1) Minimum slice thickness = T_{min} = 0.01 mm
2) Maximum slice thickness = T_{max} = 0.1 mm

Process parameters:

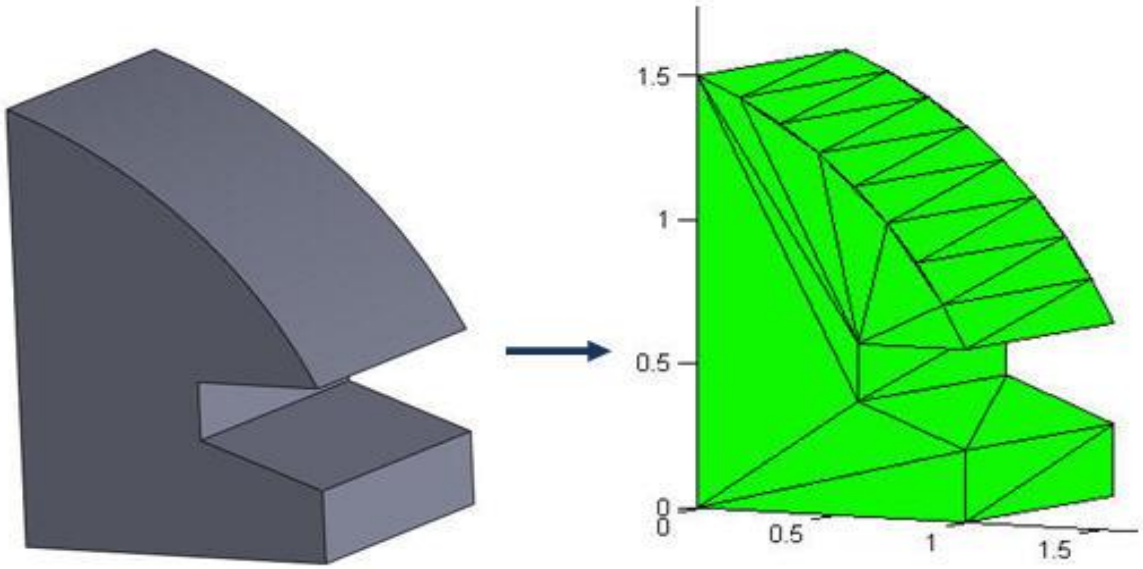
1) Predefined Volumetric tolerance for MBODS = 0.0003 mm^3

Figure 43 shows the CAD model of object 1 and the corresponding STL file that was extracted. The STL file of object 1 is converted using the MBODS algorithm. Figure 44 shows the MBODS model of the STL file of object 1.

4.2.1 Effect of MBODS on Volumetric Error of Object 1

Once the adaptive slices are computed from MBODS of the STL file, it is virtually manufactured. A simulated manufacturing point dataset is obtained using the algorithm

mentioned in the earlier chapter. This simulated dataset is used as an input in the computation of the volumetric error, as described in section 3.4.1. For a comparison, the virtual manufacturing process is repeated for different uniform slice thicknesses. The comparison between the different slicing approaches are presented in Table 5.



(a) (b)

Figure 43: (a) CAD Model (b) STL File for object 1

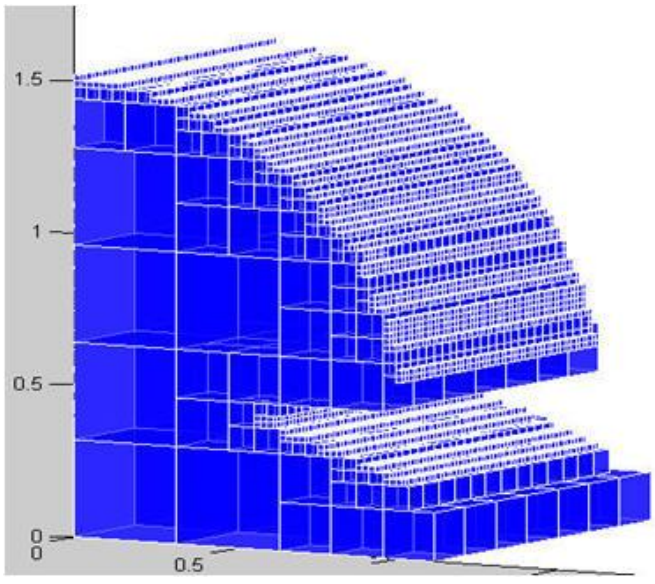


Figure 44: MBODS of STL for object 1

Three different uniform slice thicknesses are used in this test: 0.025 mm, 0.05 mm and 0.1 mm. A total of 60 slices are required with 0.025 mm uniform slicing approach as compared to adaptive slicing, which requires only 48. The percentage volumetric error for adaptive slices is less than that of 0.025 mm uniform slicing. This result validates that a combination of improved build time (reduced number of slices) and better part quality (reduced volumetric error) can be achieved using the adaptive slicing approach described in this research.

Table 5: Volumetric error for adaptive and uniform slicing methods for object 1

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness**		
		0.025 mm	**0.05 mm**	**0.1 mm**
Number of slices	48	60	30	15
Volume of CAD file	1.16	1.16	1.16	1.16
Volume of VM part	1.1426	1.1392	1.1206	1.078
% Volumetric error	1.501	1.7966	3.3959	7.0663

Figure 45 compares the percentage volumetric errors of the different slicing approaches used above. It is clear from the graph that the adaptive approach has a clear advantage over the uniform slicing method. Table 6 shows a comparison between the two slicing approaches utilizing the same number of slices, 48. The corresponding slice thickness in the uniform slicing method is calculated to be 0.03125 mm. As can be seen, the volume of the virtually manufactured part differs from the volume of the CAD files by 1.501 % and 1.933 % for adaptive and uniform slicing methods respectively, resulting in a smaller volumetric error for the former approach as compared to the latter.

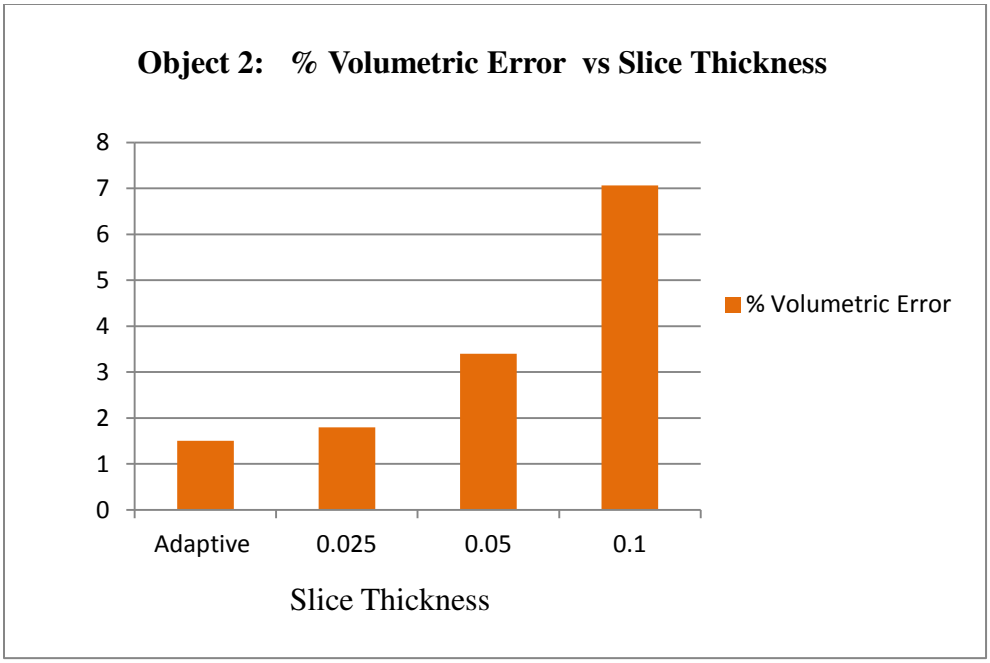


Figure 45: Comparison of percentage volumetric error vs slice thicknesses for object 1

Table 6: Volumetric error for adaptive and uniform slicing methods using same number of slices

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness of 0.03125 mm**
Number of slices	48	48
Volume of CAD file	1.16	1.16
Volume of VM part	1.1426	1.1376
% Volumetric error	1.501	1.933

4.2.2 Effect of MBODS on Profile Error of Object 1

For the purpose of evaluating the profile error encountered in the VM of object 1, only the curved portion of the object is being considered as shown in Figure 46. The facet isolation algorithm [44] mentioned in chapter 3 is used to identify the triangles related to this surface.

The contour points and the corresponding offset points related to this surface are separated from the simulated manufacturing database obtained during the virtual manufacturing of the object.

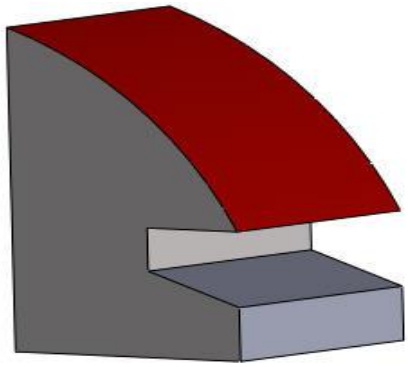


Figure 46: Selected surface of a CAD model of object 1

Figure 47 shows the STL file and the point database corresponding to the curved surface.

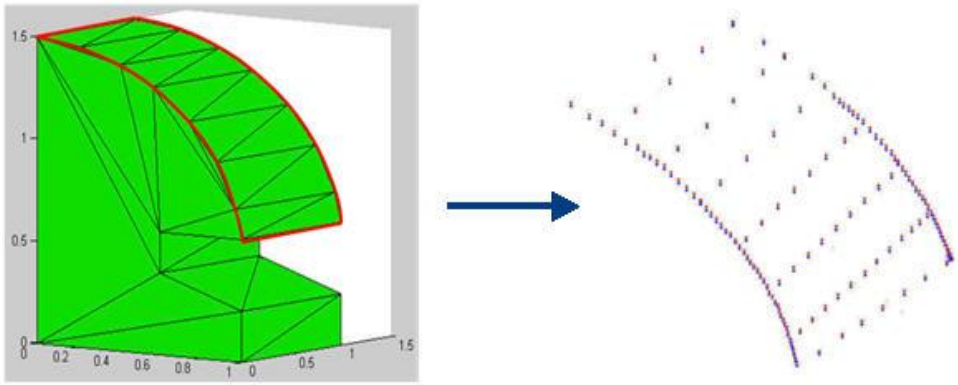


Figure 47: Selected surface from STL file for object 1

Then, the minimum distance between each point of the selected database and the CAD surface is determined using algorithm described in section 3.4.2 [44]. The maximum of all the above evaluated minimum distances gives the profile error for the curved surface of the object. Once again the same profile error is evaluated for different values of uniform slice thicknesses. The

results are compared to MBODS adaptive slicing, presented in Table 7. The profile error for adaptive slicing is seen to be less than that for the uniform slicing method with slice thicknesses of 0.025 mm, 0.05 mm and 0.1 mm.

Table 7: Profile error for adaptive and uniform slicing methods for object 1

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness**		
		0.025 mm	**0.05 mm**	**0.1 mm**
Number of slices	48	60	30	15
Profile Error	0.032	0.036	0.06	0.11

It can be seen from Figure 48 that the profile error is much more for 0.1 mm uniform slicing than for that of adaptive slicing. Even the errors encountered as the uniform slice thicknesses decrease to 0.05 mm and 0.025 mm are seen to be higher than the corrspconding error in adaptive slicing.

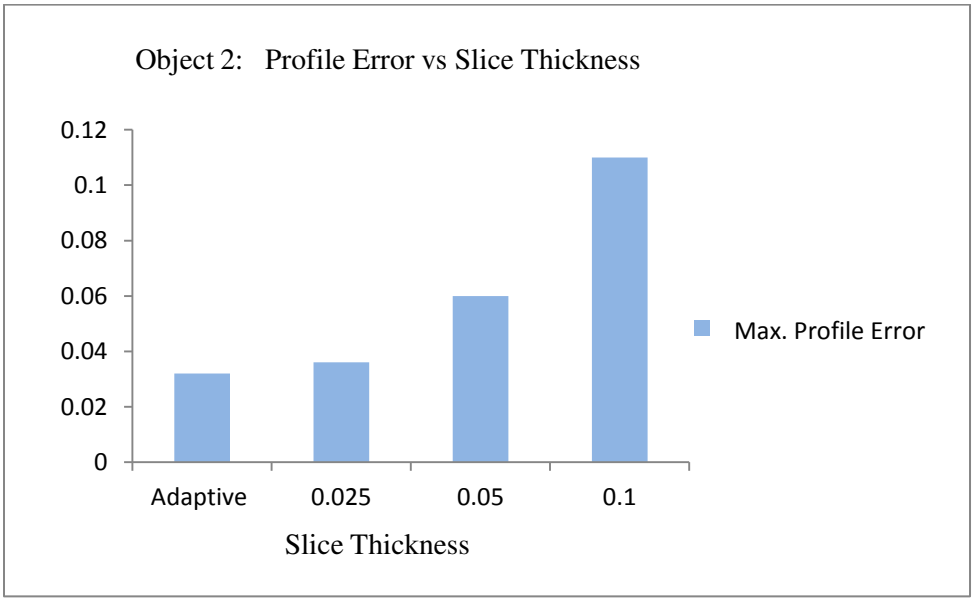


Figure 48: Comparison of profile error vs slice thicknesses for object 1

As seen from Table 8, the profile error resulting from the application of the two slicing methods with the same number of slices is seen to be smaller for the MBODS based adaptive slicing approach than the uniform slicing approach by 30.43%.

Table 8: Profile error for adaptive and uniform slicing methods using same number of slices

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness of 0.03125 mm**
Number of slices	48	48
Profile Error	0.032	0.046

4.3 Modified Boundary Octree Representation and Evaluation of Errors of Object 2

For the second test, an object with flat surfaces, termed as object 2, is modeled using SolidWorks. The object is exported as an STL file and the CAD to STL conversion in this case does not have any geometry distortion as the object is devoid of any curved features. The part specifications, AM machine specifications and process parameters are as follows:

Part specifications:

1) Height = 1.2 mm
2) Width = 1 mm
3) Breadth = 1.5 mm
4) Actual volume of the object = 1.12 mm^3

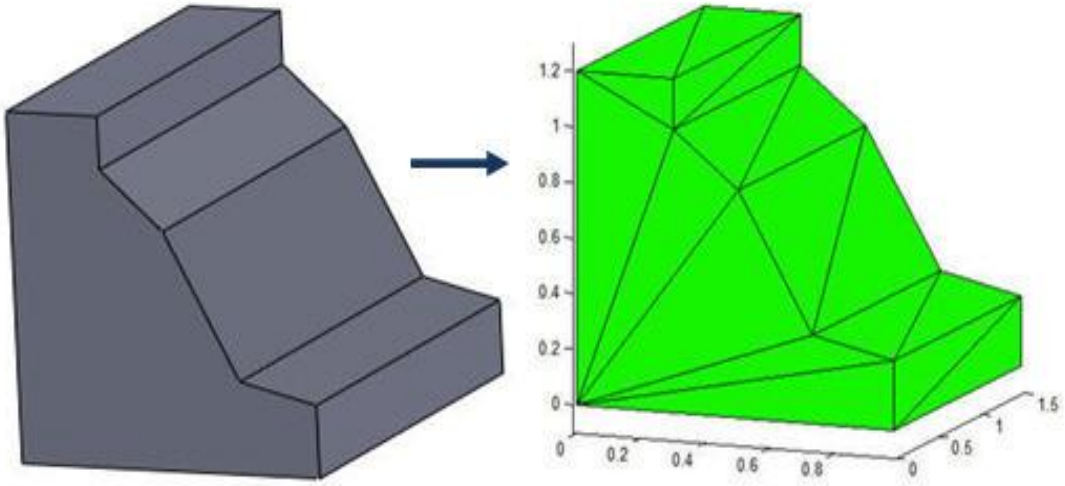
AM machine specifications:

1) Minimum slice thickness = T_{min} = 0.025 mm
2) Maximum slice thickness = T_{max} = 0.1 mm

Process parameters:

1) Predefined Volumetric tolerance for MBODS = 0.0003 mm^3

Figure 49 shows the extracted STL file from the CAD object. The STL file is converted using the MBODS algorithm and the resulting representation of the STL file is depicted in Figure 50.



(a) (b)

Figure 49: (a) CAD model (b) STL file for object 2

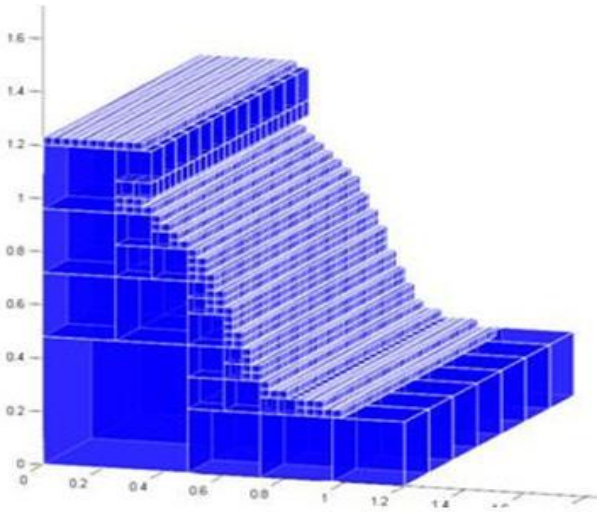


Figure 50: MBODS of STL for object 2

4.3.1 Effect of MBODS on Volumetric Error of Object 2

The adaptive slice thicknesses used in the additive manufacturing of the object 2 are computed in the MBODS algorithm. The simulated manufacturing point dataset used in the VM of object 2 is obtained by slicing the STL file and the volumetric error calculations are completed using the formulae described in section 3.4.1.

The object 2 is then virtually manufactured using three different uniform slice thicknesses of 0.025 mm, 0.05 mm and 0.1 mm, respectively, and the results are compared in Table 9. A total of 32 slices are required with the adaptive slice thicknesses computed using MBODS, with resulting volumetric error of 1.9375%. A tradeoff between the number of slices (build time) and part surface quality (with increased volumetric error) can be seen for the uniform slicing approach.

Table 9: Volumetric error for adaptive and uniform slicing methods for object 2

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness**		
		0.025 mm (T_{min})	**0.05 mm**	**0.1 mm**
Number of slices	32	48	24	12
Volume of CAD file	1.12	1.12	1.12	1.12
Volume of VM part	1.0983	1.1073	1.073	1.065
% Volumetric error	1.9375	1.137	4.188	4.876

As can be gathered from Figure 51, a volumetric error of less than 2% can be achieved using the adaptive slicing approach using the MBODS algorithm. A uniform slice thickness of 0.025 mm, with a 50% increment in the number of slices, and the associated manufacturing time, has a lesser volumetric error than the adaptive approach.

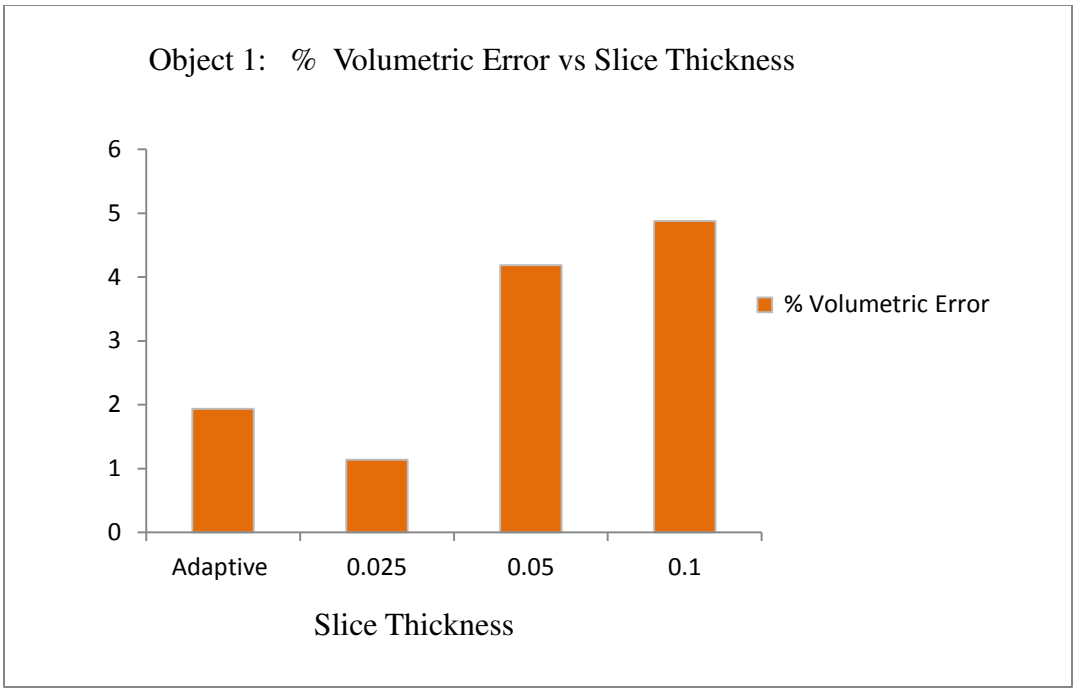


Figure 51: Comparison of percentage volumetric error vs slice thicknesses for object 2

Table 10 shows a comparison between the two slicing approaches utilizing the same number of slices, 32. The corresponding slice thickness in the uniform slicing method is calculated to be 0.03750 mm.

Table 10: Volumetric error for adaptive and uniform slicing methods using same number of slices

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness of 0.03750mm**
Number of slices	32	32
Volume of CAD file	1.12	1.12
Volume of VM part	1.0983	1.0852
% Volumetric error	1.9375	3.107

As can be seen, the volume of the virtually manufactured part differs from the volume of the CAD files by 1.9375 % and 3.107 % for adaptive and uniform slicing methods respectively, resulting in a smaller volumetric error for the former approach as compared to the latter.

4.3.2 Effect of MBODS on the Profile Error of Object 2

In the current case, the evaluation of the profile error is based on the STL surfaces, as they do not differ from the original CAD surface due to the lack of any CAD to STL conversion errors. The manufacturing point dataset, related to the selected STL surfaces as shown in Figure 52, is extracted and the minimum distance between every point in the database and the selected profile is determined. The maximum of all these minimum distances gives the profile error for the selected surfaces of object 2.

Figure 52 shows the STL file of object 2 and the extracted manufacturing point database corresponding to the selected profile. Once again the profile error is evaluated for different values of uniform slice thicknesses and MBODS based adaptive slice thicknesses and compared. The results are presented in Table 11.

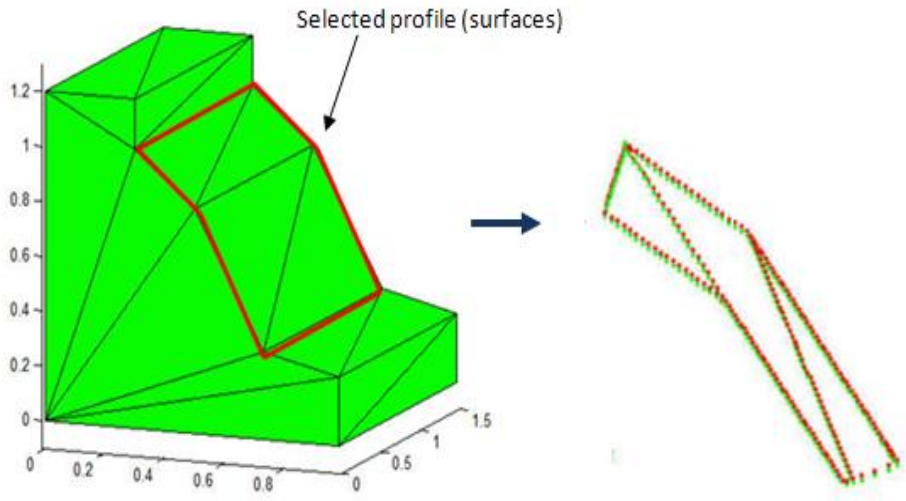


Figure 52: Selected surface from STL file for object 2

Results from the Table 11 confirm the efficiency of adaptive slicing since only 32 slices are required as compared to 48 slices utilized in the 0.025 mm uniform slice thickness approach. A profile error of 0.045 mm is observed when the adaptive approach is applied whereas larger profile errors of 0.07 mm and 0.14 mm are observed with uniform slicing approaches of 0.05 mm and 0.10 mm respectively. A graphical comparison of the profile errors for different slicing approaches is presented in Figure 53.

Table 11: Profile error for adaptive and uniform slicing methods for object 2

Slicing method	**Adaptive slice thicknesses by MBODS (mm)**	**Uniform slice thickness**		
		0.025 mm (T_{min})	**0.05 mm**	**0.1 mm**
Number of slices	32	48	24	12
Profile Error (mm)	0.045	0.033	0.07	0.14

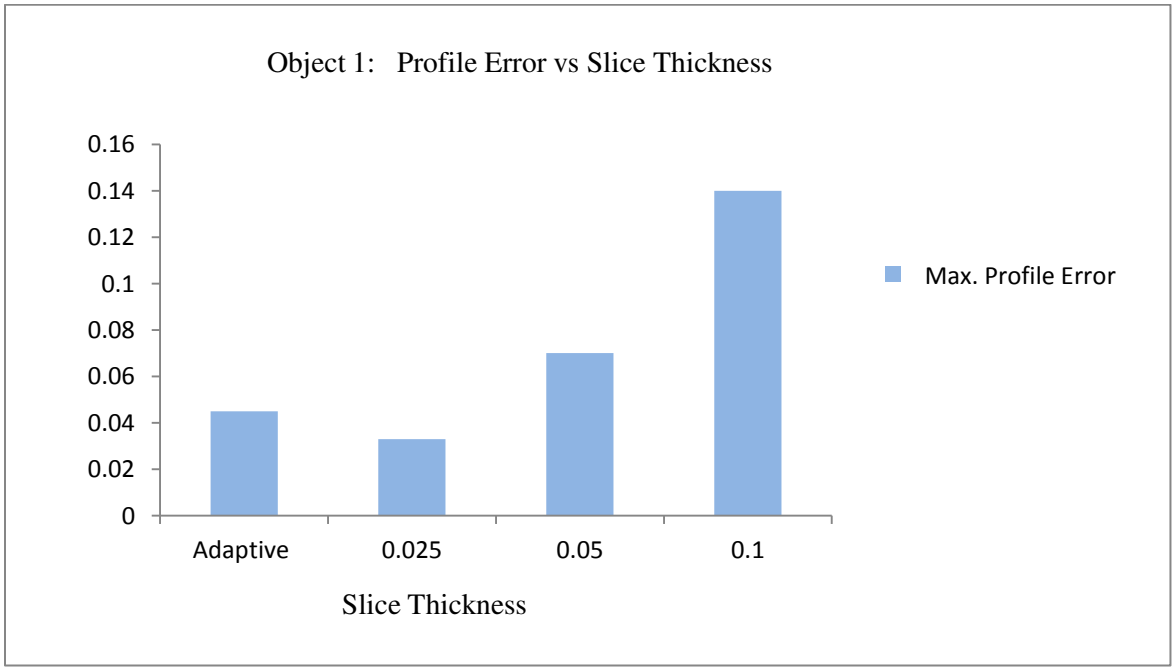


Figure 53: Comparison of profile error vs slice thicknesses for object 2

As seen from Table 12 given below, the profile error resulting from the application of the two slicing methods with the same number of slices is seen to be smaller for the adaptive slicing approach than the uniform slicing approach by 11.76%

Table 12: Profile error for adaptive and uniform slicing methods using same number of slices

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness of 0.03125 mm**
Number of slices	32	32
Profile Error	0.045	0.051

4.4 Modified Boundary Octree Representation and Evaluation of Errors of Object 3

Another complex part, termed as object 3, is selected as the next test case in validating the algorithms developed in this study. The part is modeled in SolidWorks and then exported as an STL file. The CAD to STL conversion tolerance is maintained at low levels to reduce the chord error during the conversion. The number of STL triangles increased with the complexity of the part thus utilizing larger computational time and resources. The part specifications, AM machine specifications and process parameters are presented below.

Part specifications:

1) Height = 0.9147 mm
2) Width = 0.9 mm
3) Breadth = 0.9 mm
4) Actual volume of the object = 0.37 mm^3

AM machine specifications:

1) Minimum slice thickness = T_{min} = 0.01 mm

2) Maximum slice thickness = T_{max} = 0.1 mm

Process parameters:

1) Predefined Volumetric tolerance for MBODS = 0.00003 mm^3

Figure 54 shows the extracted STL file from the CAD model of the object 3. Figure 55 depicts the MBODS of the STL file.

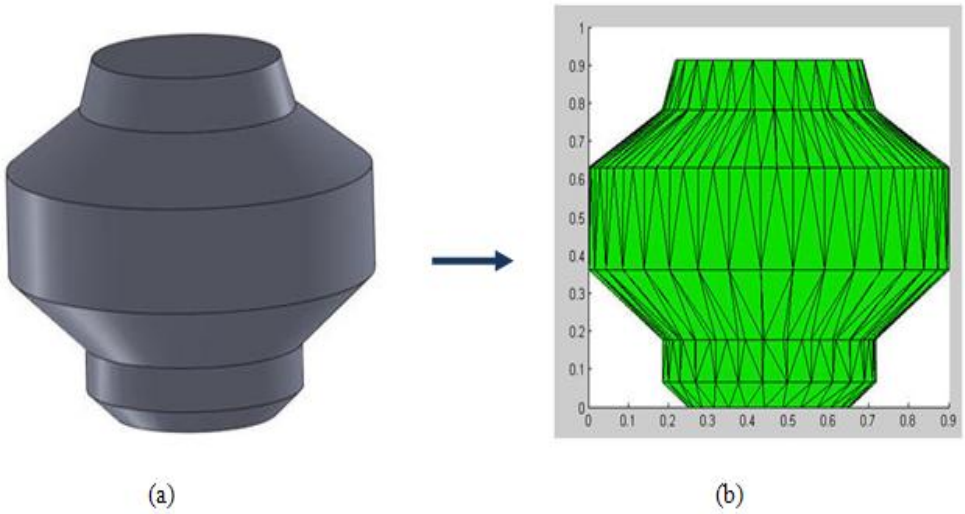


Figure 54: (a) STL file (b) MBODS of STL for object 3

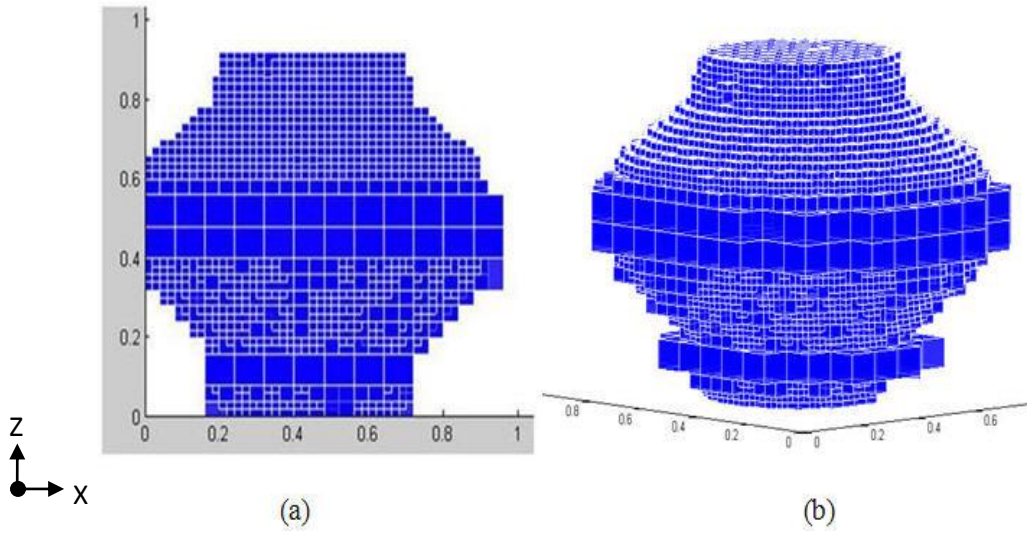


Figure 55: MBODS of STL for object 3, (a) X-Z view (b) 3-D view

4.4.1 Effect of MBODS on Volumetric Error of Object 3

Adaptive slices are computed in a similar way as mentioned in above test cases and the variable thicknesses are used to slice the STL file. The contour area of each layer is calculated and the areas of all the layers and the corresponding slice thicknesses are used to compute the volume of the virtually manufactured part. The object is virtually manufactured using different uniform slice thicknesses as well.

A comparison of the number of slices required for different slicing approaches and the corresponding percentage volumetric error is presented inTable 13. The percentage volumetric error reduces as the uniform slice thickness decreases from 0.1 mm to 0.025mm accompanied by an increase in the number of slices from 10 to 37. Thus there is always a compromise between the build time and the surface quality of a part for the uniform slicing approach. The adaptive slicing approach on the other hand has the least volumetric error with a lesser number of slices as compared to the 0.025 mm uniform slicing.

Table 13: Volumetric error for adaptive and uniform slicing methods for object 3

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness**		
		0.025 mm	**0.05 mm**	**0.1 mm**
Number of slices	31	37	19	10
Volume of CAD file	0.37	0.37	0.37	0.37
Volume of VM part	0.3688	0.3681	0.367	0.365
% Volumetric error	0.3247	0.51351	0.899	1.358

The graph in Figure 56 gives a visual interpretation of the comparison of the volumetric errors.

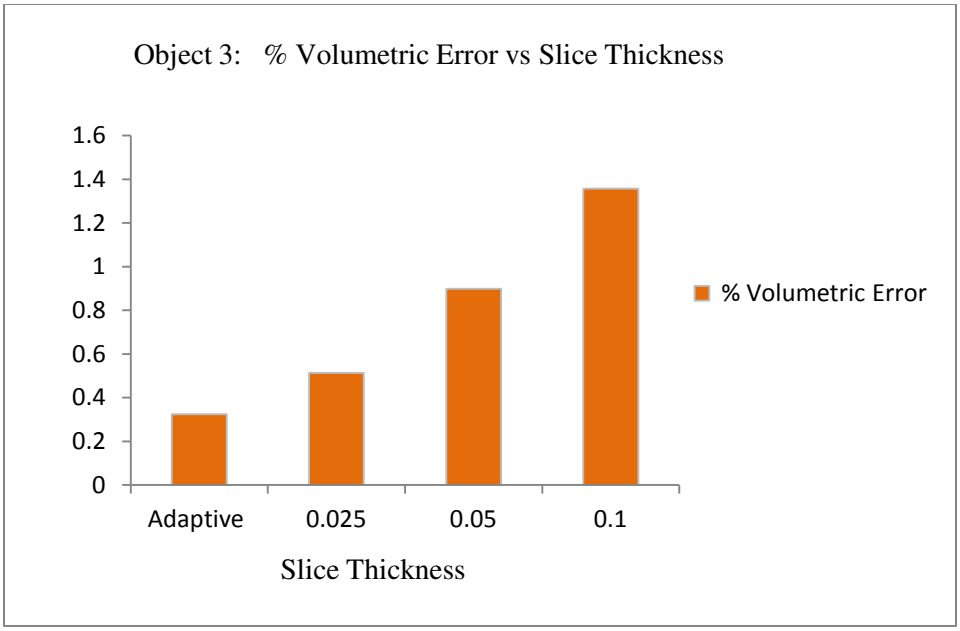


Figure 56: Comparison of percentage volumetric error vs slice thicknesses for object 3

Table 14: Volumetric error for adaptive and uniform slicing methods using same number of slices

Slicing method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness of 0.02950 mm**
Number of slices	31	31
Volume of CAD file	0.37	0.37
Volume of VM part	0.3688	0.3678
% Volumetric error	0.3247	0.5946

4.4.2 Effect of MBODS on the Profile Error of Object 3

Figure 57 shows the highlighted portion of object 3, the profile error of which will be evaluated in this section. The facet isolation algorithm [44] is used to extract triangles related to the selected design surface from the STL file.

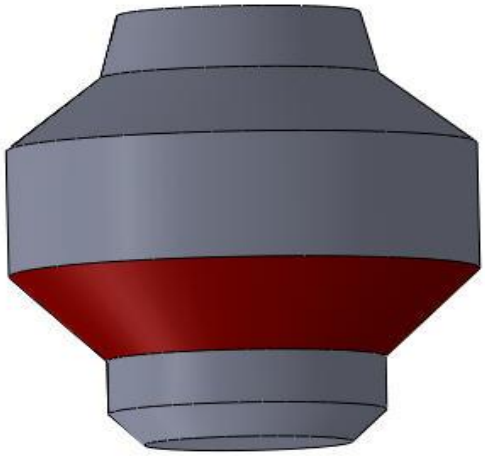


Figure 57: Selected surface of a CAD object for object 3

Figure 58 shows the corresponding points separated from the simulated manufacturing point database. The profile error for the surface is evaluated using the algorithm explained in section 3.4.2. A comparison of the MBODS based adaptive slicing and uniform slicing approaches is presented in Table 15.

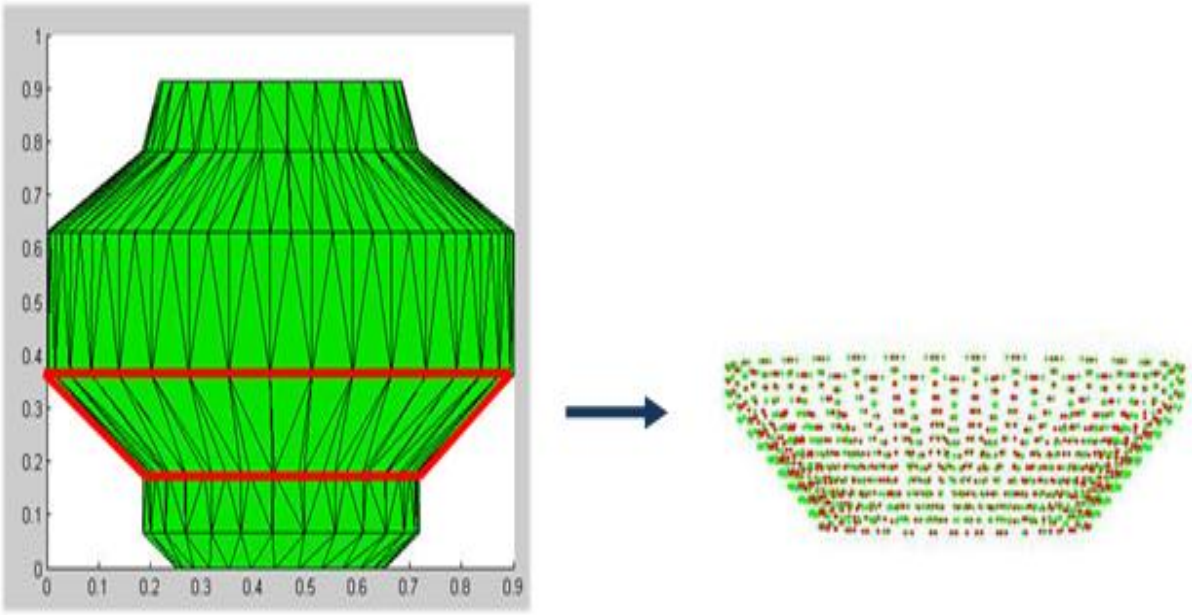


Figure 58: Selected surface from STL file for object3

Table 15: Profile error for adaptive and uniform slicing methods for Object 3

Slice method	**Adaptive slice thicknesses by MBODS**	**Uniform slice thickness**		
		0.025 mm	**0.05 mm**	**0.1 mm**
Number of slices	31	37	19	10
Profile Error	0.0113	0.0127	0.024	0.049

The results in Table 15 confirm the trend seen in the other tests. A balance between the optimum number of slices and enhanced part quality is achieved using adaptive slicing. The profile error for uniform slices increases as the uniform slice thickness is increased from 0.025 mm to 0.1 mm. Once again the error is least in the adaptive approach compared to any of the uniform slicing methods. A graphical comparison of the error results are shown in Figure 59.

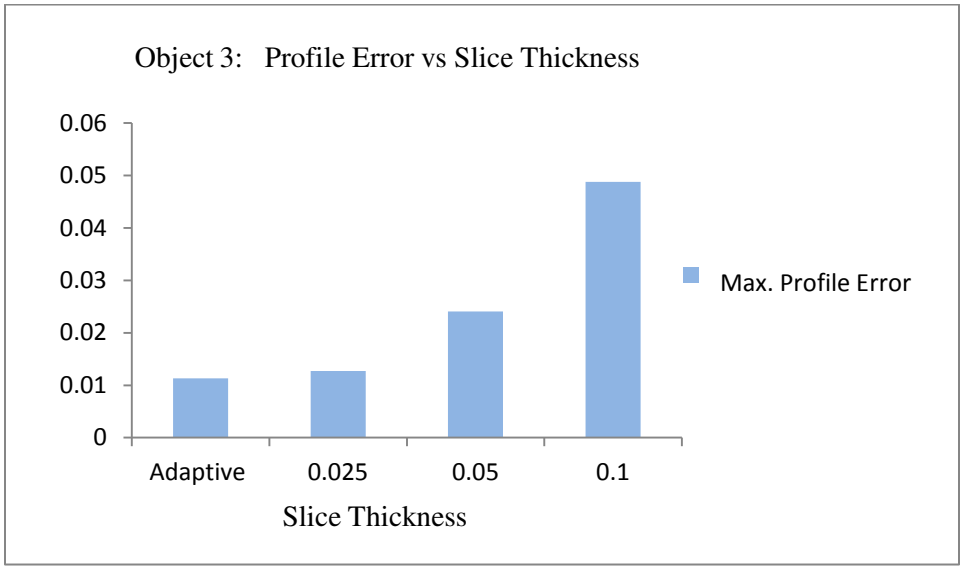


Figure 59: Comparison of profile error vs slice thicknesses for object 3

As can be seen from Table 16, the profile error resulting from the application of the two slicing methods with the same number of slices is seen to be smaller for the case of adaptive slicing approach than the uniform slicing method by 23.64%.

Table 16: Profile error for adaptive and uniform slicing methods using same number of slice

Slicing method	Adaptive slice thicknesses by MBODS	Uniform slice thickness of 0.02950mm
Number of slices	31	31
Profile Error	0.0113	0.0148

Thus, the robustness of the STL-MBODS conversion, volumetric error evaluation, and profile error evaluation algorithms developed in the present study have been validated using three different test objects of varying complexity.

4.5 Effect of MBODS on Cylindricity Error of a Cylindrical Feature at Different Orientations

Two different orientations of a cylinder inclined at 45^{o} and 15^{o} respectively, with the cylinder axis along positive Z-axis are modeled using SolidWorks. The two CAD models are exported as STL files. The STL files are then converted into MBODS and the corresponding adaptive slice thicknesses are computed using the algorithms developed in the previous chapters. The manufacturing point dataset is then obtained for each cylinder by virtual manufacturing. Only the contour points on the curved surface of the cylinders and the related offset points are selected to evaluate the cylindricity tolerance zone. The following model specifications, AM machine specifications and model orientations are used.

Model specifications:

For all models

1) Cylinder length = 2mm
2) Cylinder radius = 0.5 mm

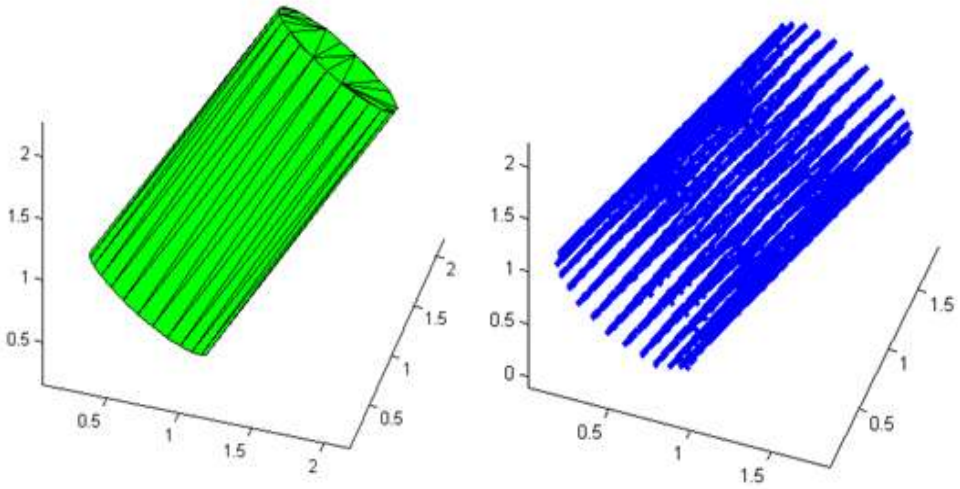
AM machine specifications:

1) Minimum slice thickness = T_{min} = 0.025 mm
2) Maximum slice thickness = T_{max} = 0.1 mm

Model orientations:

1) Model 1; Angle of the cylinder axis with the Z-axis = 45^o
3) Model 2; Angle of the cylinder axis with the Z-axis = 15^o

Figure 60(a) shows the STL format of the cylindrical feature at a 45^o angle with the Z-axis. This STL file is converted into MBODS and then sliced using adaptive slices in accordance with the explained algorithm. Figure 60(b) illustrates the manufacturing point dataset used for the evaluation of cylindricity. The dataset contains the contour points on the cylindrical surface and the corresponding offset points.



(a) (b)

Figure 60: (a) STL format of cylinder with 45^o the Z-axis (b) Points dataset for cylindricity evaluation

This point dataset is then used in the algorithm explained in section 3.4.3 and the cylindricity tolerance for the object is obtained. Subsequently, the entire procedure is repeated

with uniform slice thicknesses (0.025 mm, 0.05mm and 0.1 mm) and the corresponding cylindricity tolerance zones are computed using the algorithm developed by Ramaswami [50]. The process is repeated again for the cylinder that is oriented 15^{o} with the Z-axis and the corresponding results are documented in Table 17.

Table 17: Comparison of cylindricity of a cylindrical feature at different orientations

Model	**Orientation of cylinder axis**	**Unit direction vector along cylinder axis**	**Cylindricity error**							
			Adaptive slice thicknesses by MBODS (mm)	**No. of slices for MBODS**	**Uniform slice thickness**					
					0.025 mm (T_{min})	**# slices**	**0.05 mm**	**# slices**	**0.1 mm**	**# slices**
Model_1	45^{o}	(0.5, 0.5, 0.707)	0.0631	55	0.0647	57	0.0841	29	0.0994	15
Model_2	15^{o}	(0.183, 0.183, 0.966)	0.0289	71	0.0317	78	0.0463	39	0.0632	20

Thus the robustness of the algorithm developed in this research is supported by the cylindricity results for a cylindrical feature at different orientations. The adaptive slicing based on the MBODS algorithm results in consistently lesser value of cylindricity error with reduced number of slices.

5. CONCLUSION AND FUTURE SCOPE

A novel approach to accomplish adaptive slicing using the Modified Boundary Octree Data Structure (MBODS) of a part in AM processes has been developed and presented. Several beneficial features of octree data structures such as space decomposition capability, the ability to store object information, simplicity, and ease of computation prove to be advantageous while evaluating the variable slice thicknesses using the MBODS algorithm. This algorithm converts STL file of a part to boundary octree data structure. A subsequent algorithm computes the variable slice thicknesses using the MBODS representation of the part and virtually manufactures the part utilizing the calculated slice thicknesses. AM machine parameters and user defined volumetric tolerance values are applied in the algorithm which ensures that the fabricated part satisfies the required geometric tolerances. This research presents several test cases which were successfully converted into MBODS using the conversion algorithm. These parts were virtually manufactured using the computed slice thicknesses and then inspected for the volumetric, profile and cylindricity form errors. It is observed that the required geometric tolerances have been met by using the variable slice thicknesses obtained from the MBDOS algorithm. Results reported in this study shows that the MBODS method appears to be more efficient and accurate over other approaches used for slicing a part in additive manufacturing.

As opposed to using the STL file, future research may involve a direct conversion of the CAD object surface to MBODS or a different boundary representation to compute adaptive slice thicknesses. While the ray casting algorithm has been used to calculate the volume of a given cube outside the STL file in the present study, other approaches with voxels or maximum distance between a cube ‘s facets and its intersecting triangular facets as a criterion can be used to limit the depth of division of an octree representation.

REFERENCES

[1] Source: http://en.wikipedia.org/wiki/3D_printing

[2] Source: http://www.xpress3d.com/Default.aspx

[3] Samet H, Webber RE. Hierarchical Data Structures and Algorithms for Computer Graphics. IEEE Computer Graphics and Applications Journal 1988; 8(4): 59-75

[4] Kalpakjian S, Schmid SR. Manufacturing, Engineering & Technology, Fifth Edition 2006; ISBN 0-13-148965-8. Pearson Education, Inc, Upper Saddle River, NJ.

[5] Kulkarni P, Marsan A, Dutta D. A review of process planning techniques in layered manufacturing. Rapid Prototyping Journal 2000; 6(1):18 – 35.

[6] Byun HS, Lee KH. Determination of the optimal build direction for different rapid prototyping processes using multi-criterion decision making. Robotics and Computer-Integrated Manufacturing 2006; 22: 69–80.

[7] Sreeram, P, Dutta D. Determination of Optimal Orientation Based on Variable Slicing Thickness in Layered Manufacturing. Technical Report UM-MEAM-TR-9414,1994; Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI.

[8] Fadel G, Kirschman C. Accuracy issues in CAD to RP translations. Rapid Prototyping Journal 1996; 2(2): 4-17.

[9] Source: http://www.rapidtoday.com/stl-file-format.html

[10] ASTM International. Standard Specifications for Additive Manufacturing File Format (AMF). Version 1.1, 2011; ASTM F2915 – 12.

[11] Pandey P, Reddy N, Dhande S. Slicing procedures in layered manufacturing, a review. Rapid Prototyping Journal 2003; 9(5): 274–288.

[12] Dolenc A, Makela I. Slicing Procedures for Layered Manufacturing Techniques. Computer-Aided Design 1994; 26(2):119-126.

[13] Sabourin E, Houser SA, Bohn JH. Adaptive slicing using stepwise uniform refinement. Rapid Prototyping Journal 1996; 2(4): 20-26.

[14] Sabourin E, Houser SA, Bohn JH. Accurate exterior, faster interior layered manufacturing. Rapid Prototyping Journal 1997; 3(2): 44-52.

[15] Carmier D, Unnanon K, Sanni E. Specifying non-uniform cusp heights as a potential for adaptive slicing. Rapid Prototyping Journal 2000; 6(3): 204-211.

[16] Jamieson R, Hacker H. Direct slicing of CAD models for rapid prototyping. Rapid Prototyping Journal 1995; 3(1):12-19.

[17] Jackins CL, Tanimoto S. Oct-trees and their use in representing three-dimensional objects. Computer Graphics and Image Processing 1980, 14(3): 249-27.

[18] Meagher D. Geometric modeling using octree encoding. Computer Graphics and Image Processing 1982; 19(2):129-147.

[19] Yamaguchi K, Kunii TL, Fujimura K, Toriya H. Octree related data structure and algorithms. IEEE Computer Graphics and Applications 1984; 4(1): 89-95.

[20] Source: http://en.wikipedia.org/wiki/Octree

[21] Medellin H, Corney J, Davies JBC, Lim T, Ritchie JM. An automated system for the assembly of octree models. Assembly Automation 2004; 24(3):297–312.

[22] Samet H. An overview of quadtrees, octrees and related hierarchical data structures. Theoretical foundations of computer graphics and CAD 1988; NATO ASI series vol F40.

[23] Allada V, Anand S. Manufacturing applications of octrees. Journal Article, Computers & Industrial Engineering 1992; 23(1-4):37-40.

[24] Ayala D, Brunet P, Navazo I, Juan R. Object Representation by means of Non-minimal Division of Quadtrees and Octrees. ACM Transactions on Graphics 1985; 4(l): 41-59.

[25] Ayala D. Boolean operations between solids and surfaces by octrees: models and algorithm. Computer Aided Design 1988; 20(8): 452–465.

[26] Anand S and Knott K. An algorithm for converting the boundary representation of a CAD model to its Octree Representation. Computers in Industrial engineering 1991; 21(1-4): 343-347.

[27] Lawrence Associates Inc. Virtual Manufacturing Technical Workshop. Technical Report 1994; Dayton, OH.

[28] Marinov V. On some Aspects of the Virtual Manufacturing Concept. In: Proceedings of the Third International Conference on Metal Cutting and High Speed Machining 2001; Metz, France.

[29] Masood S, Rattanawong W. A Generic Part Orientation System Based on Volumetric Error in Rapid Prototyping. International Journal of Advanced Manufacturing Technology 2002; 19(3):209–216.

[30] SU N, Hui G. Research on the Profile Error Evaluation and Visualization of Free-Form Surface. Applied Mechanics and Materials 2011; 43: 560-564.

[31] Besl P, Mckay N. A Method for Registration of 3 D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 1992; 14(2): 239–256.

[32] Gunnarson KT, Prinz FB. CAD Model Based Localization of Parts in Manufacturing. Computer 1987; 20(8): 66–74

[33] Murthy T, Abdin S. Minimum Zone Evaluation of Surfaces. International Journal of Machine Tools Manufacturing 1980; 20(2):123-136.

[34] Shunmugam MS. On assessment of geometric errors. International Journal of Production Research 1986; 24(2):413-425.

[35] Dhanish PB, Shunmugam MS. An algorithm for form error evaluation – using the theory of discrete and linear Chebyshev approximation. Computer Methods in Applied Mechanics and Engineering 1991; 92 (3): 309-324.

[36] Carr K, Ferreira P. Verification of form tolerances Part II: Cylindricity and straightness of a median line. Precision Engineering 1995; 17(2):144-156.

[37] Lai J, Chen I. Minimum Zone Evaluation of Circles and Cylinders. International Journal of Machine Tools & Manufacture 1996; 36(4):435-451

[38] Huang S, Fan K, Wu J. A New Minimum Zone Method for Evaluating Flatness Errors. Precision Engineering Journal 1993; 15(1):25-32.

[39] Kovvur Y, Ramaswami H, Anand RB, Anand S. Minimum-Zone Form Tolerance Evaluation using Particle Swarm Optimisation. International Journal of Intelligent Systems Technologies and Applications 2008; 4(1):79-96.

[40] Ramaswami H, Kovvur Y, Anand S. Accurate Size Evaluation of Cylindrical Components using Particle Swarm Optimization. Transactions of the North American Manufacturing Research Institution of SME 2006; 34:229-236.

[41] Ramaswami H, Anand S. Accurate Size Evaluation of Cylindrical Components. International Journal of Advanced Manufacturing Technology 2010; 49(9-12):1079-1092.

[42] Roth SD. Ray Casting for Modeling Solids. Computer Graphics and Image Processing 1982; 18(2):109–144.

[43] Fadel G, Kirschman C. Accuracy issues in CAD to RP translations. Rapid Prototyping Journal 1996; 2(2): 4-17.

[44] Navangul GD. Stereolithography (STL) File Modification by Vertex Translation Algorithm (VTA) for Precision Layered Manufacturing. MS Thesis 2011; University of Cincinnati.

[45] Huang X, Gu P. CAD-model based inspection of sculptured surfaces with datums. International Journal of Production Research 1996; 36(5):1351–1367.

[46] Gudla R. Sculptured Surface Localization using Generalized Hopfield Networks. MS Thesis 2000; University of Cincinnati.

[47] Patrikalakis N, Bardis L. Localization of Rational B-Spline Surfaces. Engineering with Computers 1991; 7(4):237-252.

[48] American Society of Mechanical Engineers. Mathematical definition of dimensioning and tolerancing principles. ASME Standard Y14.5.1M-1994; ASME Press, NewYork.

[49] American Society of Mechanical Engineers. Geometric dimensioning and tolerancing (GD&T). ASME Y14.5-2009.

[50] Ramaswami H. An integrated framework for virtual machining and inspection of turned parts. PhD Thesis 2010; University of Cincinnati.