

Feature Based Tool Path Generation System for FDM

V Durga Rao¹, S Rajasekhar², N.V.S.Shankar³

^{1,2}Department of Mechanical Engineering, Kakinada Institute of Technology and Science, Divili, Kakinada, AP, India

Department of Mechanical Engineering, Kakinada Institute of Technology and Science, Divili, Kakinada, AP, India

³Department of Mechanical Engineering, Swarnandhra College of Engineering and Technology, Seethampuram, AP, INDIA

Abstract— Integration of Computer Aided Design and Manufacturing process is important for any firm during its business operations. An important aspect in this step is modelling of the part using manufacturing features. These features explicitly capture manufacturing attributes. This process is implemented in many of the CAM software for conventional manufacturing processes. But when it comes to special processes like 3D printing, this philosophy has not been implemented. The way the code is generated for 3D printing is quite different when compared to conventional manufacturing. Slicing is used for generating the code during 3D Printing. The generated part model is exported to stl file and this stl file is then sliced along z-direction to facilitate code generation. The current work investigates into use of feature based modelling to simplify slicing techniques. Based on the modelling, a z-buffer of loops is generated. This z-buffer is modified each time a feature is added to a part. This z-buffer is then used to generate the code. This process helps in quicker modification of code by eliminating the need of exporting and slicing each time. This helps integrating the 3D printing module into any CAD software thus eliminating the slicing procedure partially. This also helps in controlling the errors and tolerances in the part generated by 3D Printing. As a part of implementing this principle, AutoCAD customized using VB.NET.

Keywords— 3D Printing, Feature Based Modeling, Automatic code generation, Z-Buffer, CAD.

I. INTRODUCTION

3D Printing is finding a lot of applications in daily life especially when it comes to prototyping, irrespective of the field in which it is being employed. Ze-Xian Low, et al [1] compared 3D printing techniques and conventional manufacturing techniques that can be used for membrane manufacturing. Spahiu, et al [2] described how 3D printing can be used in textile industry. Wong [3] reviewed the application of 3D printing in orthopedics. Abdul Azeez, et al [4] used 3D printing techniques for fabricating support for hand drop disorder. P Vivek, et al [5] used 3D printing to fabricate a low cost prosthetic hand with EPS foamed fingers.

Basic 3D printing machines are G-Code based. The G-Codes are listed in [6]. There are multiple ways in which these codes are generated. Starly, et al [7] extracted the geometry from STEP files and then sliced the geometry to generate toolpath. Udriou, et al [8] reviewed Polyjet printing process for different applications. Donghong Ding, et al [9] reviewed good amount of literature relating to slicing of objects. The problems during generation of stl file and using them to generate the tool path for 3D printing, tolerancing methods are discussed. Various criteria based on which tool path is computed is also presented in the same paper. Marco Livesu, et al [10] reviewed various aspects of process planning relating to 3D printing. Various aspects involve solving problems during import of geometry, planning for 3D printing like orientation of the object, determination of use of supports and the way to be generated, selection of process parameters etc. Kao & Prinz [11] discussed a method to reduce the gaps and errors that are produced in 3D printing when printing objects with sharp corners. Pandey, et al [12] reviewed classification of slicing algorithms, the algorithms involved and tolerancing methods currently in practice. Siraskar [13] used octree technique to determine layer height for improving accuracy and reduce stair case effect. Lensgraf and Mettu [14] described an algorithm to reduce waste moves using the existing methods but also utilizing local dependencies between parts of the model to ensure that printing and movement in three dimensions produces the target model and is collision-free. G Q Jin, et al [15] developed an algorithm for generating zig-zag tool path using hybrid and adaptive techniques. Mateusz Wojcik, et al [16] generated a tool path for fused deposition process using Modified Zig Zag pattern optimized using GA. Yu-An Jin [17] developed a slicing methodology for generating tool paths for 3D printers based on FDM techniques taking surface accuracy and fabrication efficiency into consideration. Various examples are presented in this regard. Kai-Yin Fok, et al [18] devised a modified travelling sales man algorithm to optimize the toolpath generated for

FDM printers. Yasusi Kanada [19] patented a 3D printing method which solves the asperity by controlling the flow of the molten material. Zhiwen Zhao and Luc Laperrière [20] proposed a slicing algorithm which adjusts the layer thickness to be printed depending on the curvature of the outer surface to be achieved.

Most of the above methods are based on tessellated solid models or stl definitions of the geometry. There is no attempt to directly use the geometry information that is generated when modeling is being done. This is the area that is concentrated in this paper introduction of the paper should explain the nature of the problem, previous work, purpose, and the contribution of the paper. The contents of each section may be provided to understand easily about the paper.

II. PROBLEM STATEMENT

As mentioned earlier, most of the methods are based on tessellated solid models. Though there is some work in extracting geometry from step files, this also still requires the solid model to be exported into a separate data exchange format and then imported into another software. The aim of the current work is to avoid this complicated procedure. Most of the software are feature based design software. This principle was the motivation for the current work. The work involved implementing a concept “Design for 3D printing”, during which the model is designed using feature based modeling techniques and 3D printing code generation is then done making use of the features that are used to create the model.

III. PROCEDURE

Fig 1 shows the basic steps followed during design of the tool. The Tool has a feature based 3D Modeler which models the solid in AutoCAD environment. Fig 2 shows the UI that has been written. The model tree shows the features of the model. The number of features that can be handled are now limited to Rectangular Prism, Cylinder. All kinds of Boolean operations can be performed using these. These features define the cross-sections at various levels. Modeling using this tool is the first step in this concept.

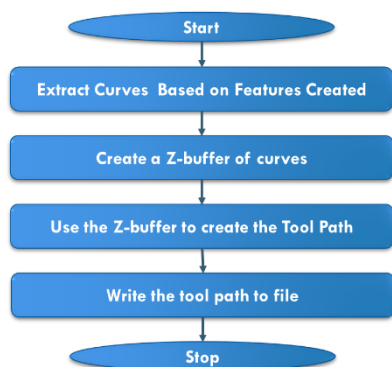


FIGURE 1: FLOW CHART SHOWING BASIC STEPS FOLLOWED IN THE SOFTWARE

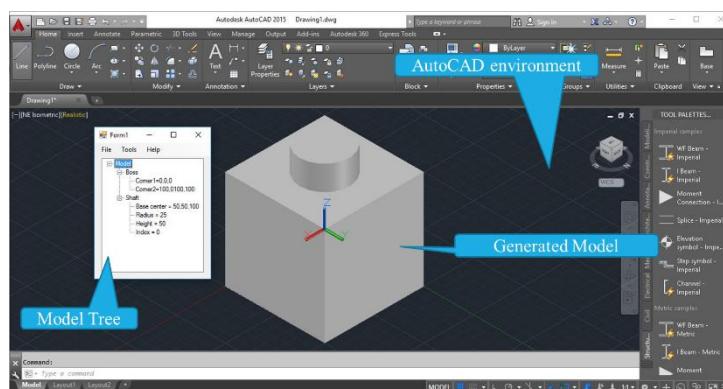


FIGURE 2: UI WITH AUTOCAD

The second step is extracting the curves and arranging them in the increasing order of Z-Value. In other words, Z-buffer of curves is being created. Figure 3 shows the flow chart giving the outline of the procedure followed during this step. Once Z-Buffer is created, this Z-buffer is used to create G-Code. Flow chart in Figure 4 shows the procedure for generating G-Code using Z-buffer. Here, the criterion “*print boundary first*” is used for generating tool path. The infill pattern consists of 45° inclined grid. When generating infill patterns, a line inclined at 45° to X-Axis is first swept from left to right and then intersections are calculated at each step. Once the intersections of each scan line are obtained, the number of intersections are then checked for. If the number of intersections is odd, then a corner point of the profile exists in the tool path. This point is identified and the points are joined in such a way that gaps and this point are avoided. The same procedure is repeated for line inclined at -45°. Based on the obtained intersections, the tool path generated. Figure 5 shows this as a flow chart. The generated code is then written to a file.

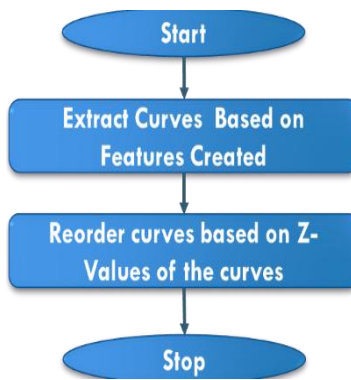


FIGURE 3: FLOW CHART OF THE PROCEDURE FOLLOWED DURING CREATING Z-BUFFER

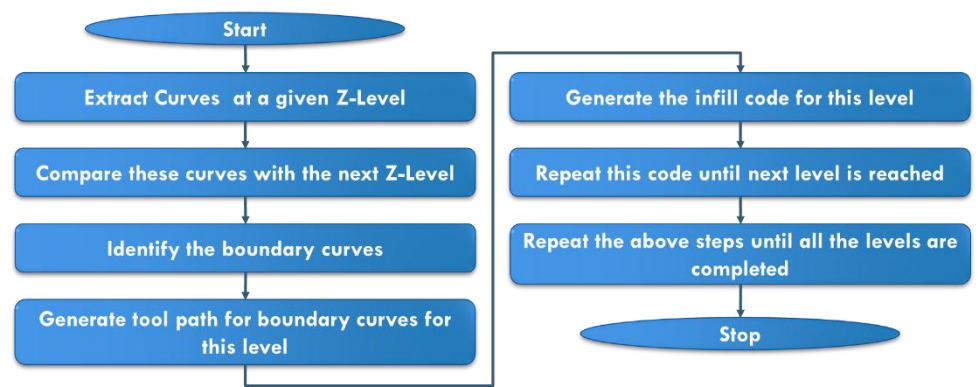


FIGURE 4: FLOW CHART FOR CREATING TOOL PATH

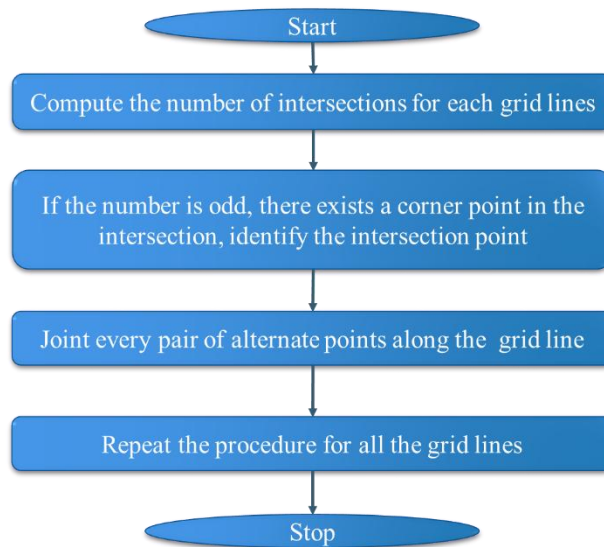


FIGURE 5: FLOW CHART SHOWING THE IN-FILL GENERATION ALGORITHM.

Parametric representation of curves is used to compute the intersections. The curves that we encounter when working with this software are line and circle. The parametric representations of the lines are given by equations (1) & (2) while parametric representation of a circle is given by equation (3).

$$\left. \begin{aligned} x(t) &= x_1 + (x_2 - x_1)t \\ y(t) &= y_1 + (y_2 - y_1)t \\ 0 &\leq t \leq 1 \end{aligned} \right\} \quad (1)$$

$$\left. \begin{aligned} x(u) &= x_3 + (x_4 - x_3)u \\ y(u) &= y_3 + (y_4 - y_3)u \\ 0 &\leq u \leq 1 \end{aligned} \right\} \quad (2)$$

$$\left. \begin{aligned} x(u) &= r \cdot \cos(2\pi u) \\ y(u) &= r \cdot \sin(2\pi u) \\ 0 &\leq u \leq 1 \end{aligned} \right\} \quad (3)$$

4.1 Calculating Line-Line Intersections

When calculating intersection between the lines (1) & (2), we use the relations in equation (4) i.e. the co-ordinates at the

point of intersection on both the curves are the same.

$$\begin{cases} x(t) = x(u) \\ y(t) = y(u) \end{cases} \quad (4)$$

On solving the above system of equations, we have equation (5) defining the value of t .

$$t = \frac{(y_1 - y_3)(x_3 - x_4) - (x_1 - x_4)(y_3 - y_4)}{(y_3 - y_4)(x_1 - x_2) - (x_3 - x_1)(y_2 - y_1)} \quad (5)$$

If $0 \leq t \leq 1$, intersection exists. Lest intersection does not exist. The intersection, if exists can be computed using (1).

4.2 Calculating Line and Circle Intersections

The intersection point of line and circle defined by equation (1) & (3) is given by equation (4). On equating, the obtained equation is given by (6).

$$\begin{cases} x(t) = x(u) \Rightarrow x_1 + (x_2 - x_1)t = r \cdot \cos(2\pi u) \\ y(t) = y(u) \Rightarrow y_1 + (y_2 - y_1)t = r \cdot \sin(2\pi u) \end{cases} \quad (6)$$

Squaring and adding the expressions in equations set (6), we have,

$$at^2 + bt + c = 0 \quad (7)$$

Where

$$\begin{aligned} a &= (x_2 - x_1)^2 + (y_2 - y_1)^2 \\ b &= 2x_1(x_2 - x_1) + 2y_1(y_2 - y_1) \\ c &= x_1^2 + y_1^2 - r^2 \end{aligned}$$

For equation (7), intersection exists iff $b^2 - 4ac \geq 0$ and the intersection exists at

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

IV. RESULT

Using the above equations, program is written which generates the G-Code for 3D printing. The code is generated as a text file with extension “.gcode”. Fig 6 shows a part that is modeled using the tool created. Z-Buffer generated is shown in Fig 7 and the generated G-Code is shown in Fig 8. Since the generated code is of huge number of lines, total code is not shown here.

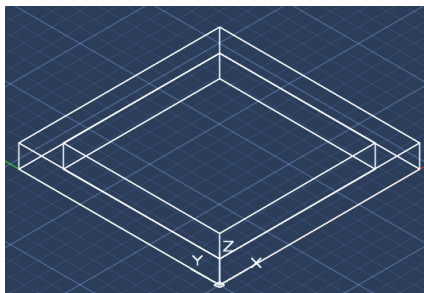


FIGURE 6: MODEL GENERATED USING THE TOOL

```
rectangle|0|0|180|180$$$$0
rectangle|0|0|180|180$$$$20
rectangle|20|20|140|140$$$$0
rectangle|20|20|140|140$$$$20
+++++
exit for
rectangle|0|0|180|180$$$$0
rectangle|20|20|140|140$$$$0
rectangle|0|0|180|180$$$$20
rectangle|20|20|140|140$$$$20
+++++
```

FIGURE 7: OUTPUT SHOWING Z-BUFFER

in *ICONS 2015 : The Tenth International Conference on Systems MZZ-GA*, 2015, no. c, pp. 30–35.

- [17] Y. A. Jin, Y. He, and J. Z. Fu, “An Adaptive Tool Path Generation for Fused Deposition Modeling,” *Adv. Mater. Res.*, vol. 819, no. September, pp. 7–12, 2013.
- [18] K. Fok, C. Cheng, C. K. Tse, and N. Ganganath, “A Relaxation Scheme for TSP-based 3D Printing Path Optimizer.”
- [19] Y. Kanada, “3D printing method that generates fine asperity on solid surface,” US 2016/0214316 A1, 2016.
- [20] Z. Zhao and L. Laperrière, “Adaptive Direct Slicing of the Solid Model for Rapid Prototyping,” *Int. J. Prod. Res.*, vol. 38, no. 1, pp. 69–83, 2000.