Acta Polytechnica CTU Proceedings **26**:144–148, 2020
doi:10.14311/APP.2020.26.0144

available online at https://ojs.cvut.cz/ojs/index.php/app

PRE-PROCESSING OF ADDITIVE MANUFACTURING INPUT FILES FOR NUMERICAL SIMULATION

Jan Voříšek*, Bořek Patzák

Czech Technical University in Prague, Faculty of Civil Engineering, Department of Mechanics, Thákurova 7, 166 29 Prague 6, Czech Republic

* corresponding author: jan.vorisek@fsv.cvut.cz

Abstract. In this contribution, we present the concept of a 3D printer software emulator facilitating the creation of a spatial finite element mesh suitable for the printing process simulation.

The concept is based on gradual processing of a native 3D printer input file (in a G-code format). This file contains a complete description of manufacturing process consisting of series of individual commands interpreted by a printer. The effect of each command needs to be precisely evaluated to obtain the position of the printer head and the volume of the deposited material in any given time. The calculation is performed in the same way as in the Marlin printer firmware using the trapezoidal motion curves and a command buffer. To represent a computational model, a discrete voxel model with variable edge length and time discretization is used. The volume of deposited material is calculated for each voxel as a function of time. The resulting model is suitable for numerical analysis of the printing process.

Keywords: Additive manufacturing, 3D printing, 3D printer emulator.

1. Introduction

Additive manufacturing emerged as an innovative manufacturing process in the field of rapid prototyping. It is based on incremental deposition or sintering of the material in thin layers. Wide range of materials, including plastic, metal or even concrete, can be used to produce spatial objects of complex geometries and shapes (see Figure 1). The objects can be designed using currently available CAD software.

Printing a prototype is often more affordable than making one by hand, and certainly less costly than contracting a manufacturer to do it by conventional manufacturing, especially in the fields where low quantities of parts with a high degree of customization and complexity play the critical role. An extensive overview of the additive manufacturing methods is available in [1].

Additive manufacturing consists of several independent sub-processes. Everything begins with a 3D model which, in order to be printed, must be processed using specialised software called slicer. The slicing process produces a sequence of commands to control the movement and operation of the machine stored in a G-code file line by line. This file describes the complete processing (it is directly interpreted by the printer) and is used as an input for the presented pre-processing tool. Visualization of the movement commands can be seen in Figure 2 and Figure 4.

In recent years, there has been an increasing interest in the simulation of the additive manufacturing process. Parts produced primarily with extrusion of molten thermoplastics often do not reach designed dimensions due to the cracking and warping induced by large temperature gradients [2, 3]. Such issues are also

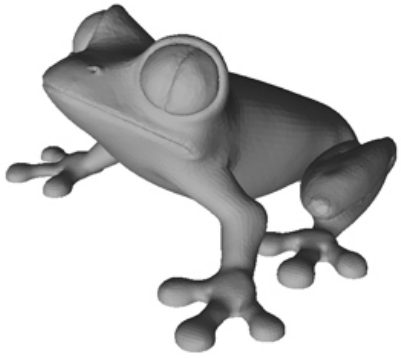

Figure 1. Visualization of the geometry of a printed object in CAD.

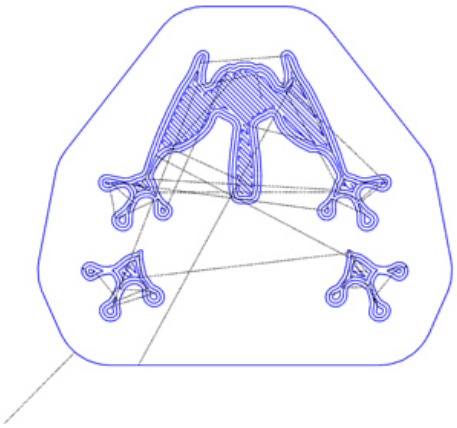

Figure 2. Visualization of the movement commands contained in the input file (single layer). Moves with material extrusion are drawn with blue color, moves without extrusion with black color.

present in conventional manufacturing processes like injection moulding where accurate numerical simulation allows to mitigate such issues by optimising the object geometry or the production process itself [4].

Numerical simulation of the manufacturing process is indispensable to produce tailored, defect-free and optimised products. Due to the complexity of the process, the coupled transient thermo-mechanical analysis is needed for reliable predictions. Some research of the additive manufacturing has been already conducted using the Finite Difference Method [3].

In this contribution we assume that the Finite Element Method (FEM) will be used to solve the problem. Our goal is to develop a software tool able to automatically prepare inputs for the FEM model (domain discretization, boundary conditions, etc.) straight from the additive manufacturing input files. Not only should our tool generate a suitable 3D mesh, but it should also provide the history of the infill percentage of the single mesh elements. By knowing the infill percentage, we will be able to consider the two-phase characteristics of the voxels (thermoplastics and air). The process is, in a sense, an inverse process of slicing as it produces the simplified (voxelized) version of the original object.

Several G-code processing utilities are available online [5, 6]. These tools provide print time estimation and filament consumption. However, they only consider the printer moves as vectors, and thus constructing the voxel model with the material extrusion history is not possible. Our solution extends the idea of the movement commands into the 3D space considering each of them as a block of material with a finite volume.

2. The model

Finite element model can generally consist of elements of various shapes. We have chosen a discrete voxel model with variable edge length along each axis due to its simplicity.

Assuming all the edge lengths equal leaves us with a simple cube. This simplification will allow the precomputation of shape function values and their derivatives as they are same for all the elements. On top of that, such a hexahedral element is commonly used in the FEM software packages.

3. G-code pre-processing

The first step into the simulation of additive manufacturing is to pre-process the raw input files. These files contain a line-separated set of instructions for the 3D printer to follow. A complete overview of all the available G-codes is available online [7]. Only a small subset of the commands, which controls the deposition of the material, is considered.

Processing of the printer input file begins with determining the duration of individual commands and particularly of printer head moves. This step is performed only once and results can be reused to generate simulation inputs with different resolution.

3.1. Duration of moves

It is essential to capture the real printing speed in the sequence of point-to-point moves. The target printer speed is defined in the G-code file for each move, yet the actual motion profile is to be decided by the individual printer.

Constant acceleration is the most used planning strategy resulting in a trapezoidal velocity profile due to its mathematical simplicity and limited computing power of the 3D printers microprocessors. The printer stores a limited number of moves, usually 16, in a buffer and recalculates the acceleration profile allowing the printer to be able to stop after the last move, if needed. The typical printing speed of one print layer can be observed in Figure 3.

Maximal accelerations and velocities must be respected in the direction of each axis. These limits can be taken from the 3D printer specification or the source code of the printer controller firmware (Marlin, RepRap).

Duration of the movements can be determined using the calculated velocity and known distance. By gradually summing up the duration of individual commands, we can obtain the current time for any given print head position. This information is essential for numerical simulation as we can reverse this procedure to calculate the position of the print head for any given time (time step).

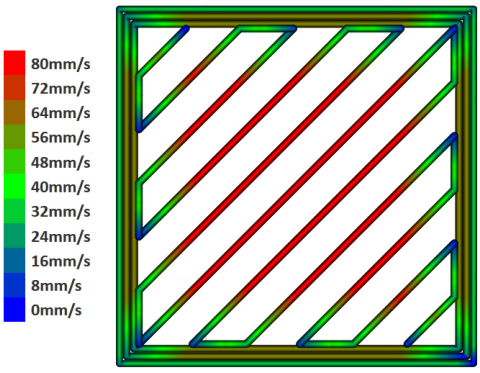

Figure 3. Visualization of the printer head speed during processing of a single layer assuming the trapezoidal velocity profile.

3.2. Reconstructing the extrusion width

Extrusion width is essential for calculating the intersection of the extrusion volume and the individual voxels to determine the amount of material deposition inside each voxel. It is difficult to define the exact shape of the extrusion cross-section. Therefore slicers use various simplified shapes for their internal calculations, most notably a rectangle with curved sides (Fig. 6) or a simple rectangle [8].

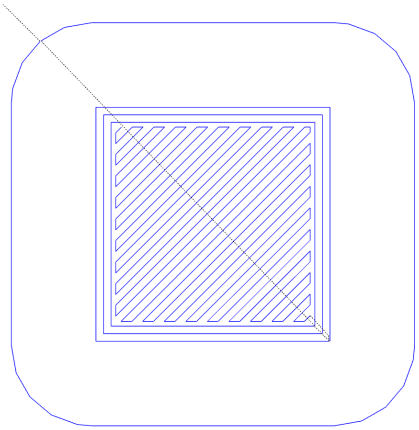

Figure 4. G-code visualization of the first layer of a $20 \times 20 \times 20$ mm cube.

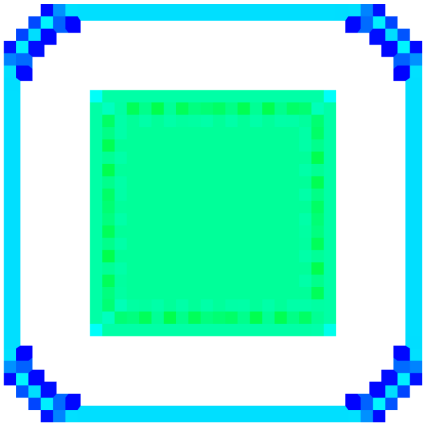

Figure 5. Activated voxel elements of a first layer of a $20 \times 20 \times 20$ mm cube including volume assignment.

Extrusion volume shape is automatically set in the slicing software based on the nozzle diameter or manually input by the user. However, the information is not present in the exported G-code file, and so it must be recalculated using known variables. To make the calculation slicer-independent, we assume a rectangular cross-section of the deposition.

The extruded volume can be calculated as

$$V_{in} = w_i h_i L_i \tag{1}$$

where w_i is extrusion width, h_i is extrusion volume height and L_i is extrusion volume length (see Figure 6). It must be equal to the volume of material coming into the extruder given as

$$V_{in} = \frac{\pi D^2}{4} \Delta E \tag{2}$$

where D is nozzle diameter and ΔE is length of extruded material.

The unknown extrusion width is then expressed as

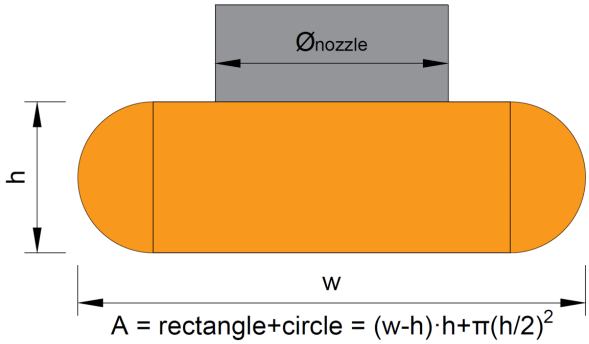

Figure 6. Extrusion width geometry used to produce G-code in Slic3r software [8]

$$w_i = \frac{\pi D^2}{4 h_i L_i} \Delta E \tag{3}$$

The simplified calculation produces a slightly narrower extrusion width than the original Slic3r value assuming oval-shaped sides of the cross-section. The difference is negligible and will be not be taken into account in any further calculations. For slicers assuming rectangular extrusion cross-sections, the calculated width will match the original value.

Extrusion width calculation can be omitted for additive manufacturing processes based on the laser beam sintering of material such as Stereolitography. In such case the width is defined by the beam diameter.

4. Voxel activation and filling

The voxel-activation and the extruded volume assignment is performed for each combination of time-step length and voxel-size. See Figure 5 for an example of a single processed print layer.

4.1. The principle

Single depositions, represented by block geometry, are to be decomposed (projected) into the individual voxel contributions. Calculating the intersection of two generally oriented blocks would be very computation-heavy. Fortunately, the deposition of the material occurs in horizontal planes, and so the intersection can be decomposed into two much simpler processes.

The first process is the so called horizontal decomposition, where extrusion volume (represented as arbitrarily oriented rectangle is decomposed to individual voxel column contributions (rasterization).

In the second step (vertical decomposition), the extruded volume is distributed to individual voxels in a column according to the height and position of extruded material volume.

These two subprocesses together provide history of extruded material volume in each voxel.

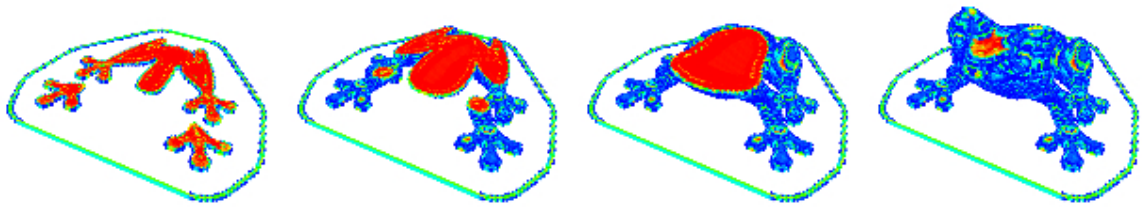

FIGURE 7. Activation sequence of the individual voxel elements. The colour of each voxel corresponds to its infill percentage (VOF). The centre of the print consists mostly of filled elements shown in red colour while blue elements near the surface of the printed part are almost empty.

4.2. Optimization

The bounding box of the depositions is calculated to save computer resources, so only the voxels contained within the bounding box are considered for the intersection calculation. Furthermore, it is possible to use Bresenham's line algorithm to only mark the affected voxels by doing only integer math.

4.3. Issues

Some issues emerged after implementing the above-described process. Slicers usually produce overlapping extrusions (see Figure 8) to improve the bonding of the parallel extrusions. The resulting infill history contained over-filled voxels (volume over 100%). The issue is also caused by considering the depositions as non-continuous (see Figure 9). The second issue can be eliminated by considering the intersection of successive depositions. The overlapping volume is rotated by 180 degrees and moved into the affected voxels as can be seen in Figure 9.

FIGURE 8. Overlap of the two concurrent depositions (overlapping volume marked with red).

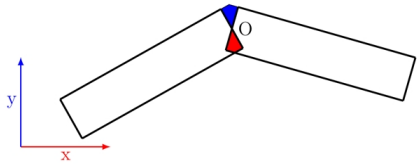

FIGURE 9. Overlap of the two non-straight depositions (overlapping volume marked with red). The red volume must be redistributed into the green area.

The voxel volume is checked after each volume difference assignment, and the surplus volume (over 100%; if applicable) is redistributed into the eight horizontal neighbouring elements to keep the maximum volume of each voxel at 100%.

5. Conclusions

Prototype G-Code pre-processor and voxel-model generator were implemented in JavaScript based on the previously formulated methodology. Several real-world files were processed using the tool. Example of complex geometry can be seen in Figure 7.

So far, it appears that it is possible to create voxel-based geometry. The developed tool enables to observe the time sequence of element activation immediately after the input file analysis. For example the model in the Figure 7 was processed on a generic consumer laptop in one thread using a $1 \times 1 \times 1$ m voxel grid in 2.6 seconds.

It is desirable to perform 3D numerical analysis of the additive manufacturing process using the knowledge of element activation sequence and the VOF distribution over time.

6. Future work

Our main goal is to create a robust, user-friendly tool for the numerical analysis of additive manufacturing and its optimization. The tool should be able to take the G-code input file, process it to obtain mesh with element infill history and finally, run FEM analysis and post-process the results. This paper solves the pre-processing part as the first step in an overall complex process.

For proper numerical simulation, it is desired to identify the thermal, mechanical, as well as process-related properties of the materials involved. We foresee the formulation of suitable constitutive material models and development of suitable adaptive FE solver. Furthermore, we also expect to conduct lab-scale measurements to determine constitutive parameters and develop and calibrate suitable material models.

The idea of gradual voxel activation could be extended from to the activation of generally shaped elements of a finite element mesh. The elements generally do not need to form any regular grid. Therefore current advanced mesh generators could be used to discretize printed object. Such elements can be gradually activated and filled in a same principle as the voxels. However, we expect the intersection of the extrusion block and a general element to be very computationally expensive.

Acknowledgements

The financial support of this research by the Grant Agency of the Czech Technical University in Prague (SGS project No. SGS19/032/OHK1/1T/11) is gratefully acknowledged.

References

[1] H. Bikas, P. Stavropoulos, G. Chryssolouris. Additive manufacturing methods and modelling approaches: a critical review. *The International Journal of Advanced Manufacturing Technology* **83**(1-4):389–405, 2015. DOI:10.1007/s00170-015-7576-2.

[2] Y. Zhang, K. Chou. A parametric study of part distortions in fused deposition modelling using three-dimensional finite element analysis. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **222**(8):959–968, 2008. DOI:10.1243/09544054jem990.

[3] T. Stockman, J. A. Schneider, B. Walker, J. S. Carpenter. A 3d finite difference thermal model tailored for additive manufacturing. *JOM* **71**(3):1117–1126, 2019. DOI:10.1007/s11837-019-03338-6.

[4] R. Johansson, D. Konijnendijk. *INJECTION MOULDING SIMULATION — A finite element approach to analyse thethermodynamics in an IM tool.* Master's thesis, Queensland University of Technology, 2007.

[5] A. Ustyantsev. gcodevisualizer - a web-based visual gcode viewer and analyzer. `https://github.com/hudbrog/gCodeViewer`. Accessed: 2019-07-11.

[6] gcodeanalyser.com. G-code analyser. `http://www.gcodeanalyser.com/`. Accessed: 2019-07-11.

[7] RepRap. G-code — reprap. `https://reprap.org/wiki/G-code`. Accessed: 2019-07-10.

[8] G. Hodgson, A. Ranellucci, J. Moe. Slic3r manual — flow math. `https://manual.slic3r.org/advanced/flow-math`. Accessed: 2019-07-15.