Modern Environmental Science and Engineering (ISSN 2333-2581)
December 2019, Volume 5, No. 12, pp. 1143-1151
Doi: 10.15341/mese(2333-2581)/12.05.2019/008
Academic Star Publishing Company, 2019
www.academicstar.us

Caterpillar — A Gcode Translator in Grasshopper*

Hao Zheng[1], Barrak Darweesh[2], Heewon Lee[3], and Li Yang[4]

1. Weitzman School of Design, University of Pennsylvania, Philadelphia, USA

2. Massachusetts Institute of Technology, Boston, USA

3. University of California, Berkeley, Berkeley, USA

4. University of Sydney, Sydney, Australia

Abstract: Additive manufacturing has widely been spread in the digital fabrication and design fields, allowing designers to rapidly manufacture complex geometry. In the additive process of Fused Deposition Modelling (FDM), machine movements are provided in the form of Gcode - A language of spatial coordinates controlling the position of the 3D printing extruder. Slicing software use closed mesh models to create Gcode from planar contours of the imported mesh, which raises limitations in the geometry types accepted by slicing software as well as machine control freedom. This paper presents a framework that makes full use of three degrees of freedom of Computer Numerically Controlled (CNC) machines through the generation of Gcode in the Rhino and Grasshopper environment. Eliminating the need for slicing software, Gcode files are generated through user-defined toolpaths that allow for higher levels of control over the CNC machine and a wider range of possibilities for non-conventional 3D printing applications. Here, we present Caterpillar, a Grasshopper plug-in providing architects and designers with high degrees of customizability for additive manufacturing. Core codes are revealed, application examples of printing with user-defined toolpaths are shown.

Key words: 3D printing, gcode, grasshopper, modelling, simulation

1. Introduction

1.1 Project Goal

3D printing, as the most common usage of additive manufacturing technology, also known as Fused Deposition Modelling (FDM), has been widely applied to design and architectural manufacturing field, which leads a revolution of producing geometries. Often as an end application of techniques, architectural design regards 3D printing as a method to produce entities from digital models.

However, as the geometries become more and more complex, designers are no longer satisfied with the conventional printing technique, which prints the object from bottom to top, layer by layer, actually a 2.5D printing. A more flexible control method of 3D printers is required for making non-conventional 3D printings, which makes full use of 3 degrees of freedom (X, Y, and Z) of the printer.

Especially in recent researches, this trend of non-conventional 3D printing becomes clearer. Wang Sheng et al. (2018) [1-5] applied a robotic arm with a printing extruder to print concrete with customized toolpaths. Seibold Hinz et al. (2018) [6-8] also refined a robotic arm to print ceramic. Mostafavi Kemper et al. (2018) [9] used similar robotic technique to print soft silicone on curved surfaces. Zheng and Schleicher (2018) [10] drew inspiration from the spinning behaviours of insects to program a 3D printer to print along given curves (Fig. 1). Yi-Chia and June-Hao (2018) [11] focused on printing wire structures directly from the extruder, without breaking the continuity. Molloy and Miller (2018) [12] constructed

* This paper was originally published in CAADRIA 2019 conference.

Corresponding author: Hao Zheng, Ph.D. Candidate; research areas/interests: machine learning, robotic technology, mixed reality, generative de-sign. E-mail: zhhao@design.upenn.edu.

a robotic system to print continuous filaments and lattice structures.

So in order to spread the idea of the non-conventional 3D printing technique and achieve it in cheap desktop level 3D printers rather than expensive industrial robotic arms, we present Caterpillar, a Grasshopper plug-in helping designers transform geometries into Gcode, which runs the 3D printer according to customized toolpaths, with all 3 degrees of freedom.

1.2 Work Flow

Fig. 2 shows the work flow of Caterpillar in Grasshopper. First, users should adjust the default settings to match the parameters of the printer they are going to use. Second, users can either input the printing and slicing geometries then use the slicer to produce the toolpaths, or directly input the user-defined toolpaths as curves. Last, the generator will output the Gcode text in a panel. By saving the text as a Gcode file and load it to the printer, the customized printing will start working. Optionally, when a Gcode text is inputted into the decoder component, the toolpaths will be read and generated as geometries. Users can apply this optional function to rebuild the toolpaths from any Gcode files in modelling software.

Fig. 1 Bio-inspired 3D printings by Zheng and Schleicher (2018) [10].

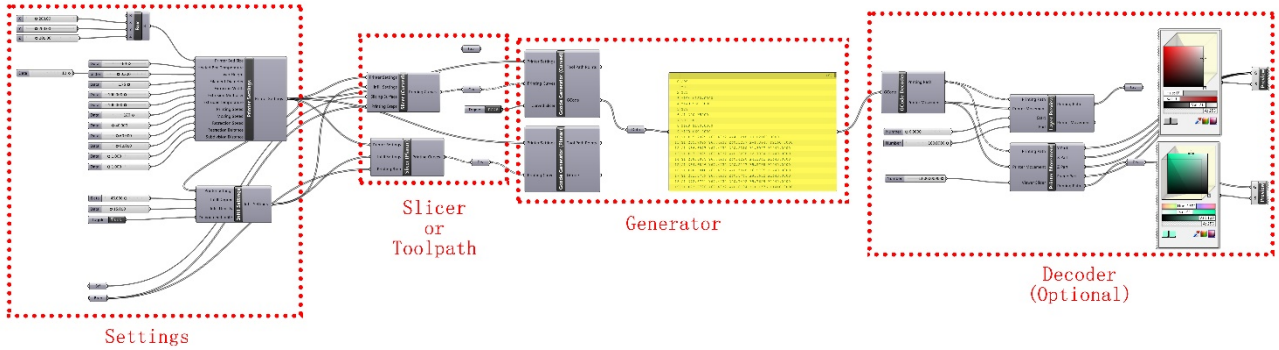

Fig. 2 Work flow of Caterpillar in Grasshopper.

2. Settings

To make use of Caterpillar correctly, the first step is to tell it the settings of the printer and the printing object.

2.1 Printer Settings

For the printer settings, there are totally 14

parameters. Printer bed size (MM) contains three numbers (x, y, z), indicating the maximum printing size of the printer. Heated bed temperature (°C), extruder temperature (°C), and filament diameter (MM) are based on the printing material, which normally will not be changed once settled. Layer height (MM) and subdivision distance (MM) control the precision of the printing, while printing speed (%), moving speed (%), retraction speed (%), and retraction distance (MM) control how fast the printer will act when printing, moving without printing, and retracting materials. Extruder width (%) and extruder multiplier (%) together decide the width of the printed toolpaths.

For conventional printing, the default settings are recommended. But changing some parameters may cause special effects, such as enlarging the extruder multiplier and lessening the printing speed to print very thick paths, or enlarging the extruder temperature to melt a base material along toolpaths. There is no limitation for the settings, so users can simply regard the printer as a moving machine with an extruder of any degrees of temperature.

2.2 Infill Settings

Another setting is for the infill, only needed when the users want to slice the model and apply infill inside. Infill degree defines the rotating degree of the web-like infill, while infill density represents the distance between each grid. More types of infill patterns are being developed, but users can also model the customized infill with the printing object, to achieve more flexibility.

3. Slicer and Toolpath

Next, geometric manipulations will be executed to the printing objects to generate the toolpaths. As Fig. 3 shows, there are three options for the users, planar slicer like most of the slicing software, curved slicer for printing on existing objects or curved toolpaths for special usages, and user-defined toolpath inputted from Rhino or Grasshopper curves.

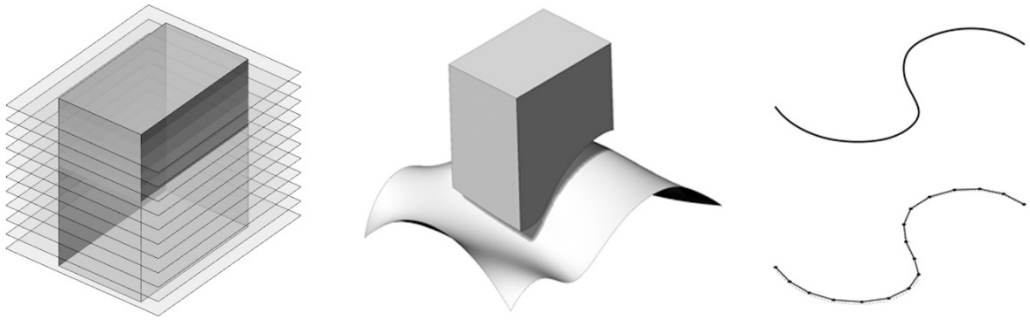

Fig. 3 Planar Slicer (left), Curved Slicer (middle), User-defined Toolpath (right).

3.1 Planar Slicer

For planar slicer, since the printing objects are always closed meshes, Grasshopper command ‘Contour’ here acts as the main role to generate the printing boundaries. Then infill settings will be applied, mapping infill pattern into the boundaries. Then all curves will be sorted to make sure the end point of the current printing path is close to the start point of the next printing path, avoiding time waste when moving the extruder.

3.2 Curved Slicer

Different from other slicing software, Caterpillar provides users with an option to slice the printing objects with a curved plan. This function was originally designed to help generate toolpaths when users want to print on an existing object but don’t know how to draw or generate toolpaths directly. Other than the slicing process, the infill producing and curves sorting processes are the same as the planar

slicer. Users can transform a real-world object into a 3D model by 3D scanning or modelling it directly, generate the toolpaths, and then put the object in the right position on the printing bed. The printer will firstly move beyond the top of the object, and then start printing on it.

3.3 User-Defined Toolpath

For advanced usages, it's recommended to directly input the toolpaths as user-defined curves, telling the printer to move and extrude exactly along the defined paths. And the program will not sort the curves, but only move the extruder from curves to curves. The toolpaths can be any 3D curves, no need to be planar. This enables the full control of the printer.

4. Gcode Generator

Then, the toolpaths will be sent to the Gcode generator, the program will deconstruct the curves and generate Gcode text file based on the printer settings.

4.1 Generator Work Flow

Fig. 4 shows the work flow of the Gcode generator. First, since the Gcode only recognizes spatial points with the coordinate of (x, y, z), the inputted curves will be divided into points based on the setting of subdivision distance, which approximately represent the curves.

Then, the toolpaths will be created based on the points. But different from other Gcode, here since it's acceptable if the users input 3D spatial curves, the printer should move up its extruder after finishing printing the current curve, then move to the position above the start point of the next curve, to avoid the collision with printed objects (Fig. 5). The height that the extruder should be lifted can be figured out before printing by referring the current top point in the printed objects.

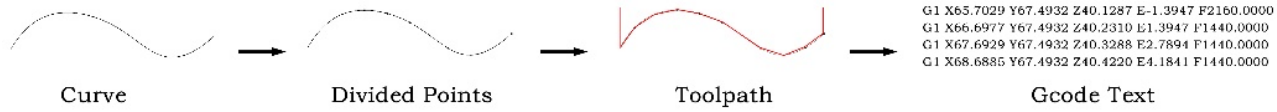

Fig. 4 Work flow of gcode generator.

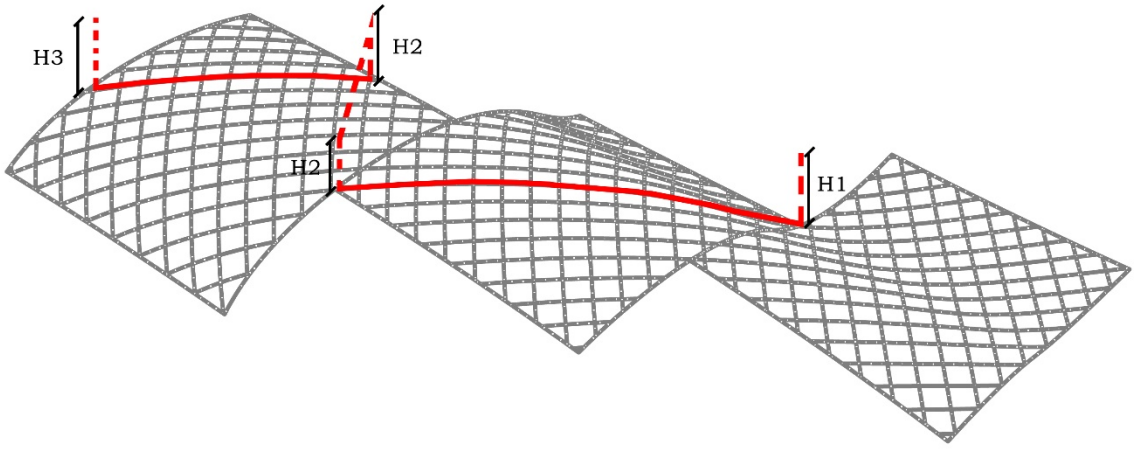

Fig. 5 Toolpath to avoid collision.

Next, the extrusion amount, which is represented as a variable "E", is calculated based on the printer settings of layer height, filament diameter, extrusion width, and extrusion multiplier. To be specific, for a given printing toolpath, which is a single line from the current position point to the next position point, its "E" value is proportional to the settings of layer height, extrusion width, and extrusion multiplier, while that is

inversely proportional to the settings of filament diameter. So the formula can be expressed as following:

E = Height(layer)*Width(extrusion)*Length(curve) *Multiplier(extrusion)/sqr(Diameter(filament)/2)/Pi

What's more, to avoid the over extruding of extra filament, in the moment of finishing printing current curve, the extruding motor should spin back a little to suck back the filament, then after reaching the start point of the next curve, the motor should release the filament again to compensate this distance. This command should also be added into the printing toolpaths.

Finally, the data of (x, y, z) and "E" with the variable "F" in the printer settings, which represents the moving speed of the toolpaths, are formatted into text Gcode file (Fig. 6). Together with the starting and ending codes, a completed Gcode is generated and ready to be inputted into the printers.

4.2 Optimization

Mentioned before, the curves will be divided into points, and each line of the Gcode text represents a point that the extruder should reach. However, different from curved toolpath, there is no need to divide a linear toolpath into points (Fig. 7). Experiments showed that, the file size will be greatly enlarged and exceed the file size limit for most of the printers, if dividing linear toolpaths, which widely appear in traditional planar printing. So before inputting the given curves to the dividing component, the program will detect and separate curved toolpaths and linear toolpaths, then divide the curved toolpaths as usual and extract the start and end points to represent the linear toolpaths.

```
G1 X65.7029 Y67.4932 Z44.1218 E0 F2160.0000
G1 X65.7029 Y68.4932 Z40.1287 E1.3947 F2160.0000
G1 X66.6977 Y67.4932 Z40.2310 E1.5861 F1440.0000
```

Fig. 6 Gcode example.

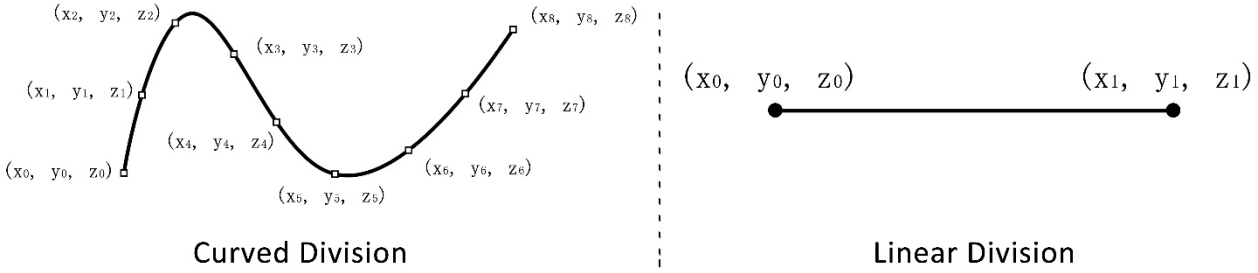

Fig. 7 Different division rules for curved toolpaths and linear toolpaths.

5. Gcode Decoder

In addition to the Gcode generator, our team also developed a Gcode decoder to translate Gcode text file back to the toolpaths as Rhino or Grasshopper geometries, helping designers preview, rebuild, and modify the printing model in modelling software.

5.1 Keywords Extraction

Although there are about 100 commands in Gcode format, most of them are designed to control other hardware, such as the fan speed or the extruder temperature, or the irrelevant parameters of the motors, such as the running speed or the working plane (Fig. 8). The only two commonly used commands to control the position of the extruder (movement of the motors) are "G1" (or "G01") and "G92". "G1" controls the position of the extruder by a following text, such as "G1 X1 Y2 Z3 E4 F5", which tells the extruder to move to the spatial point of (1, 2, 3) with a speed of 5 unit, while extruding 4 mm filament. "G92" can set the current position as a new original state, mostly used by a following text of "G92 E0" to set the position of the extrusion motor to 0, preventing the over increasing of the variable "E".

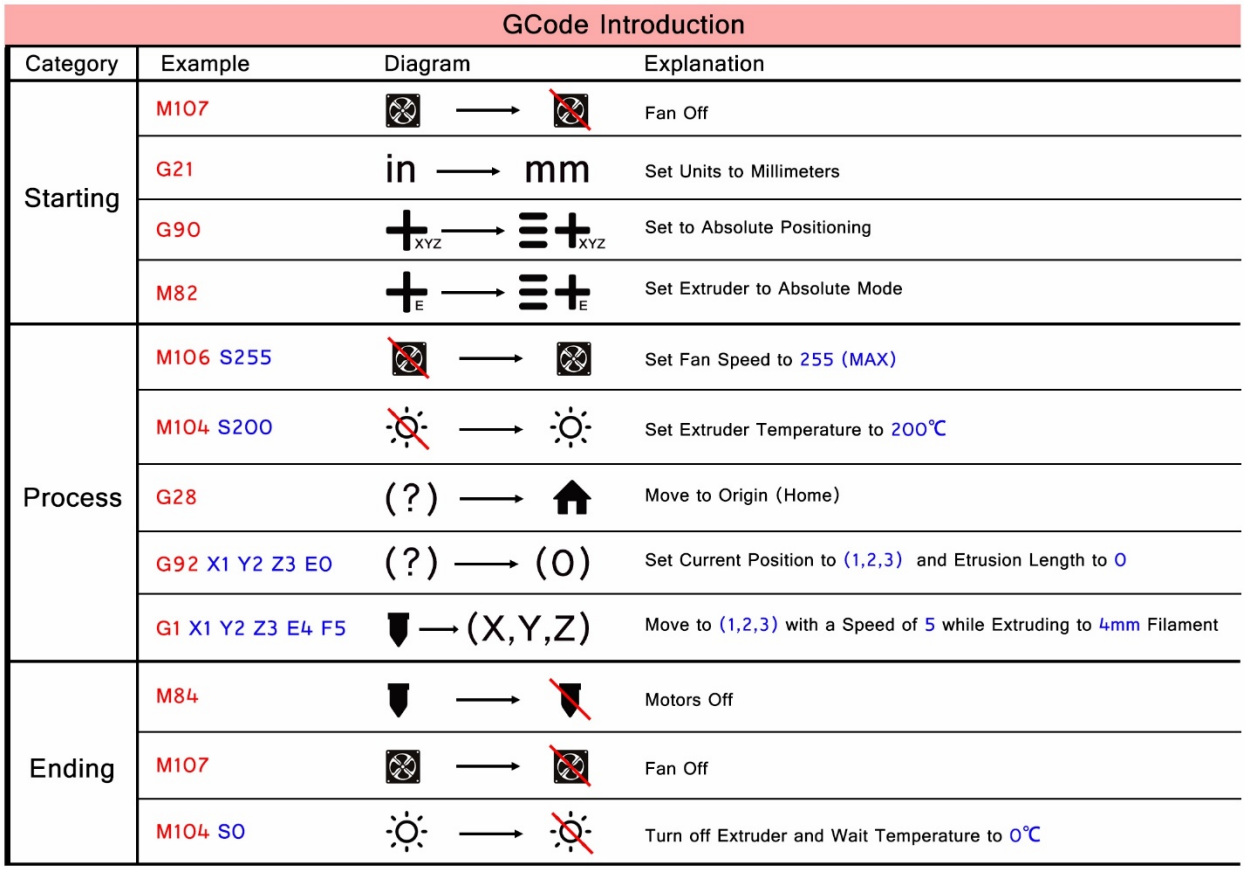

Fig. 8 Commonly-Used Gcode.

So the rule to read and translate Gcode text is quite simple. Every time, when a line with a beginning code of “G01” or “G1” appears, the program will read the numbers after the characters of “X”, “Y”, “Z”, and “E”, then the numbers are the coordinate of one of the points in the toolpaths. Also, when “G92” appears, the following numbers should be recorded as the reference point, which acts to transform the current coordinate to the world coordinate.

5.2 Model Rebuilding

After extracting all points from the Gcode text file, the program will check the recorded “E” value to separate the toolpaths for printing and the toolpaths for moving. To be specific, if the “E” value of the current point is the same or less than that of the previous point, the curve from the previous point to the current point is a moving path, which should not be rebuilt as the printing model. On the contrary, if the “E” value keeps increasing, the points should be included in the printing paths.

Then, a printing simulator was developed, which takes in the printing paths, and outputs a dynamic model showing how the 3D printer will work to print the Gcode file (Fig. 9). By adjusting a slider, the users can check the printed geometries and the position of the extruder in different printing percentage.

Also, the users can bake the printing paths to Rhino as geometries for analysis or model rebuilding. This component not only enables the preview of printing objects in Rhino, but also provides a convenient method to adjust the Gcode by remodel the objects directly.

6. Customized Printing Examples

Last, several examples are shown to discuss the advanced usages.

The best way to make full use of the plug-in is to input user-defined toolpaths directly from users' design. Fig. 10 shows the design "Pimples", in which small circular disturbances will be applied to the printing paths in different positions, so that there will be buckles in the surface of the printing.

With the same technique, but larger layer height and extrusion multiplier, the printer will keep extruding thick filament and produce a cup-like art work as Fig. 11 shows.

Also, users can reform the printer with extra toolkits or replace parts of the components. Fig. 12 shows a printer with a replaced sleeve motor, which supports a printing while rotating the printing platform. The Gcode can be generated based on the scaled toolpaths.

By replacing the extruder with other tools such as a brush, the printer can become a writer to produce calligraphy, with customized toolpaths and Gcode generated by Caterpillar (Fig. 13).

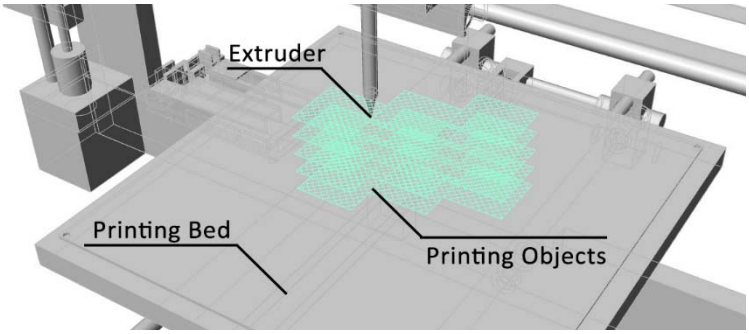

Fig. 9 Printing simulation.

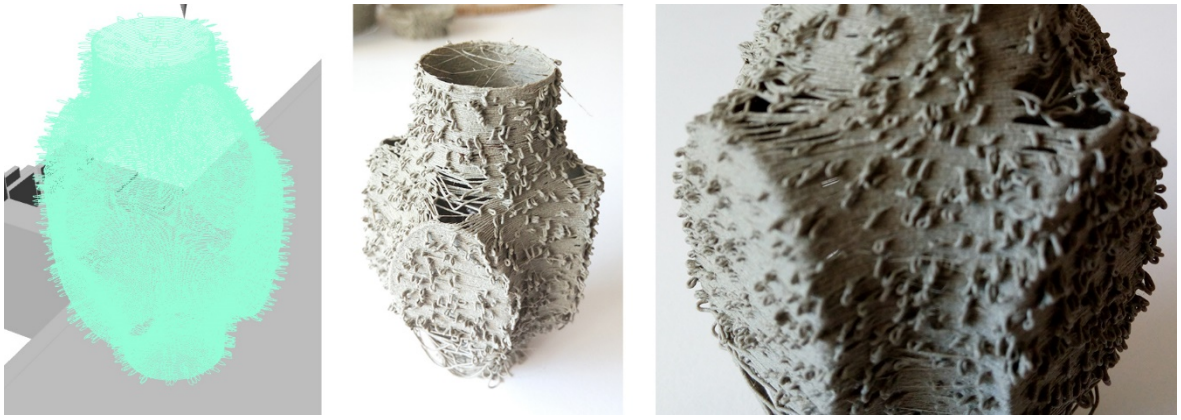

Fig. 10 Pimples.

Fig. 11 Hair.

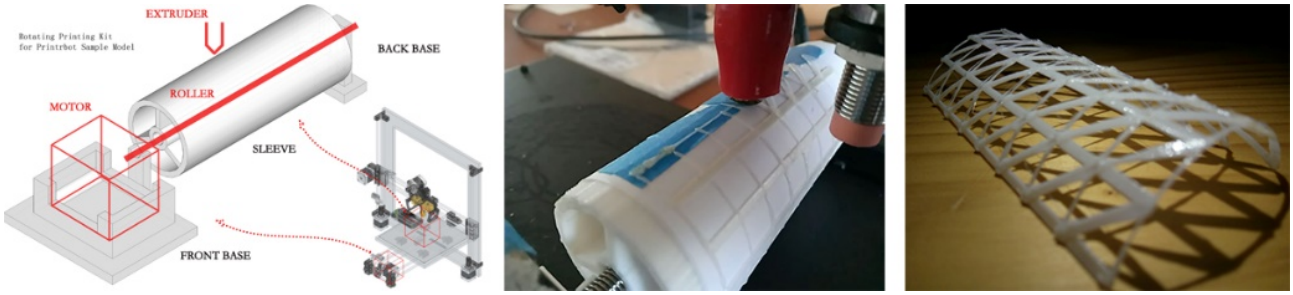

Fig. 12 Roof.

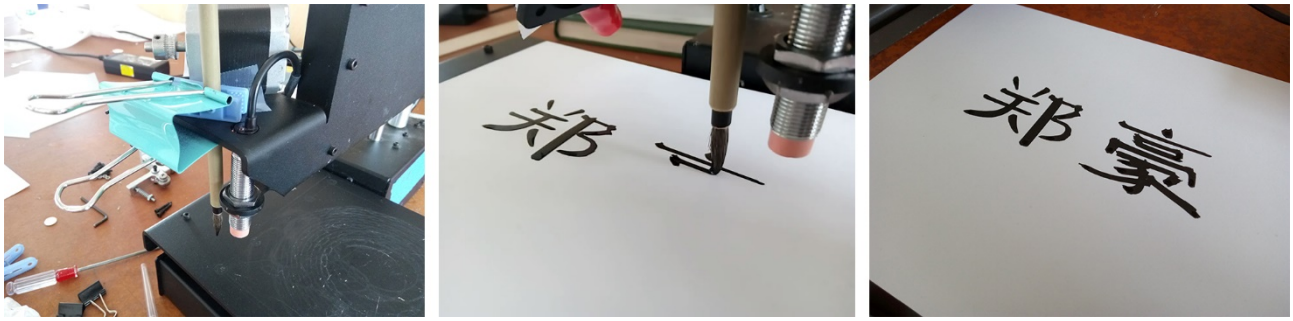

Fig. 13 Calligraphy.

7. Conclusion

Caterpillar is a powerful plug-in for Grasshopper, which helps generate 3D printing Gcode from geometries, visualize toolpaths, and remodel printing objects from Gcode. It supports printing with the movement of all 3 axes, releasing all 3 degrees of freedom of the 3D printer. Customized printings can be achieved by modifying the hardware and generating Gcode by Caterpillar.

In the future, non-conventional customized 3D printing will be highly developed for both educational and industrial purposes. Low-cost 3-axis 3D printers with extra toolkits can handle a variety of tasks, providing an alternative for expensive robotic fabrication.

References

[1] S. Y. Wang et al., Transient materialization, in: *Learning, Prototyping and Adapting*: *Short Paper Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018*. Beijing, China, 2018.

[2] S. Sun et al., 3D printing concrete system design and fabrication process research based on robotic Arm, in: *Learning, Prototyping and Adapting, Short Paper Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018*, Beijing, China, 2018.

[3] H. C. Im et al., Responsive spatial print — Clay 3D printing of spatial lattices using real-time model recalibration, in: *Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture*, Mexico City, Mexico, 2018.

[4] W. R. L. D. Silva et al., 3D concrete printing of post-tensioned elements, in: *Proceedings of the IASS Symposium 2018*, Boston, USA, 2018.

[5] C. A. Battaglia et al., Sub-additive 3D printing of optimized double curved concrete lattice structures, in: *Robotic Fabrication in Architecture, Art and Design 2018*, Zurich, Switzerland, 2018.

[6] Z. Seibold et al., Ceramic morphologies — Precision and control in paste-based additive manufacturing, in: *Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture*, Mexico City, Mexico, 2018.

[7] J. Carvalho et al., 3D printed ceramic vault shading systems, in: *Proceedings of the IASS Symposium 2018*, Boston, USA, 2018.

[8] S. Bhooshan et al., Function representation for robotic 3d printed concrete, *Robotic Fabrication in Architecture, Art and Design 2018*, Zurich, Switzerland, 2018.

[9] S. Mostafavi et al., Multimode robotic materialization, *Robotic Fabrication in Architecture, Art and Design 2018*, Zurich, Switzerland, 2018.

[10] H. Zheng and S. Schleicher, Bio-inspired 3D printing experiments, in: *Learning, Prototyping and Adapting, Short Paper Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018*, Beijing, China, 2018.

[11] H. Yi-Chia and H. June-Hao, The composite material wire printing of robotic fabrication and construction process, in: *Learning, Prototyping and Adapting, Short Paper Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018*, Beijing, China, 2018.

[12] I. Molloy and T. Miller, Digital dexterity — Freeform 3D printing through direct toolpath manipulation for crafted artifacts, in: *The International Conference on Association for Computer Aided Design in Architecture (ACADIA)*, Mexico City, Mexico, 2018.