Transmitting G-Code with Geometry Commands for Extrusion Additive Manufacturing

Alex Roschli[1], Michael Borish[1], Liam White[1], Cameron Adkins[1], Celeste Atkins[1], Abigail Barnes[1], Brian Post[1], Zac DiVencenzo[2], Charlie Dwyer[2], Gaven Rudiak[2], Brian Zellers[2]

[1]Oak Ridge National Laboratory, Manufacturing Demonstration Facility

[2]Juggerbot3D

Abstract

G-code refers to text-based commands used to instruct a 3D printer how to construct an object. G-code is generated to represent each toolpath during the slicing process. Each toolpath is represented as a list of points that define the trajectory of the path to be printed. Additional commands are included to define the motion velocity and extrusion rate, called the feeds and speeds. These toolpaths and commands must be generated specific to the machine, material, and calibration settings that will be used during the print. This paper outlines a new approach for the slicing and g-code creation process that eliminates the need for outputting feeds and speeds in the slicing process. Instead, the slicer outputs g-code that defines the desired bead geometry as printed. The 3D printer can then read this geometry data and calculate the necessary feeds and speeds based on internal calibration data to successfully print the object.

Keywords: G-code, 3D printer calibration, flowrate, feedrate, bead width

Introduction

The design of most 3D printers originates from the design of CNC (computer numerically controlled) machines, which have been around since the 1940s. These CNC machines are programmed with computers that define the toolpaths the machine must traverse to complete the manufacturing operation. These toolpaths are relayed to the CNC machine using a g-code file that is exported from the computer and uploaded to the CNC machine. A g-code file is a text-based file that contains commands for motion of the system as well as controls for operating various inputs and outputs of the machine.

G-code is a programming language developed for CNC machines. The 'g' in g-code stands for geometry because g-code is used to represent the geometry information needed by the manufacturing process. G-code was first introduced at MIT in the 1950s to replace the tape and punch card instruction sets used to program machines [1]. The punched tape stored instructions, like g-code does today, but with limitations for what data can be conveyed to the system. The original implementation was developed for vertical mills, but today g-code can be used for a variety of manufacturing systems including 3D printers, mills, lathes, and more [2].

While g-code is now the industry standard for controlling CNC machines, g-code itself is not standardized across all equipment and processes. This is due in part to the fact that such a variety of systems, such as 3D printers and lathes, use the same g-code instruction sets to convey commands to the machine. There are standard commands, such as G1 for linear motion and G4 for pauses, but there are many commands that are not standardized or commands that have varying implementations based on the machine interpreting the command. Most of these non-standardized commands are m-codes rather than g-codes (still contained within a g-code file), which are machine commands. The lack of standardization can be confusing, but also allows for unique and machine-specific commands and interpretations to be added. This is particularly useful for developing new functionality that could not have been thought of by the original designers.

The G1 command, a linear motion, implementation allows the user to define any of the following parameters: feedrate (F), X position (X), Y position (Y), Z position (Z), W position (W), A position (A), B position (B), C position (C), spindle feedrate (S), and extruder position (E). For each of the position parameters, the system moves from the current position to the commanded position to complete the move using the commanded feedrate value, which is a velocity value. The S and F parameters are used to define speeds. The F value is a commanded feedrate and the S value is a commanded RPM; both stay enabled and constant until updated.

This data from the G1 parameters is communicated to the machine via the g-code, and most systems simply read and execute the instructions without any error checking or calculations. This means that the operator must properly calibrate the machine, configure the g-code, and know exactly what data needs to be sent to the system for successful execution of the program. This setup relies heavily on operator knowledge and can be easily broken when a slight change to the system is required. Because of this, a new implementation of g-code is needed to transmit additional data to the system and offload some of the processing to the machine, rather than the operator. This paper seeks to define those means of transmitting additional geometry information from the slicing process via the g-code file to be used by 3D printers to allow more data and control of the process. While the focus will be the slicing and g-code for this new approach, this paper will also talk about necessary changes to the machine itself to properly interpret and execute the instructions.

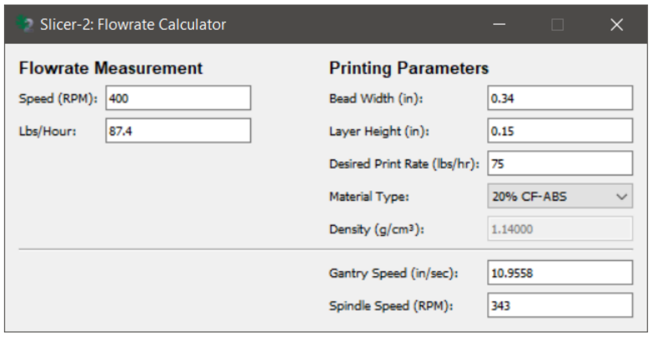
The Reason for Geometry Commands

Most thermoplastic extrusion 3D printers in use today fall into two categories: filament and pellet. Filament systems are positive displacement, meaning that the volume of filament pushed into the hot end and nozzle is directly correlated to the volume of molten plastic that is extruded. If the filament diameter remains perfectly consistent, control of the systems is predictable because the system always knows exactly what is coming out based on knowing exactly what is going in. Filament systems control their extrusion by commanding the length of filament to extrude during each motion, typically by commanding the number of steps for a stepper motor. This is implemented by adding the ‘E’ parameter as part of the G1 linear motion command. For example, the line `G1 X10 E2` would command the system to move to the location X10 and feed 2 units of filament into the extruder.

Pellet systems, unlike filament, are not positive displacement with respect to the extrusion output. This creates a control challenge for correctly calibrating and commanding the system during part construction [3]. These systems can be commanded like their filament counterparts with an E command as part of every G1 motion to define screw rotation distance. However, a more common and intuitive approach is to define an RPM value for the motor to turn for the duration of the entire path, not just the individual move. More advanced systems with PLCs (programmable logic controllers) can modulate this commanded value in real-time based on the current state of the machine. For example, if the machine is slowing down as it approaches a corner, the same slow-down command is applied to the extruder to prevent over-extrusion in the corner that would come from a constant extruder RPM with a not constant motion speed. A similar challenge exists for thermoset materials which are extruded in liquid form using a pumping system with a commanded RPM value [4].

Due to the nature of how pellet thermoplastic extrusion works with a screw-based extruder shearing, melting, compressing, and extruding the plastic material, there are many control variables that impact the volume of material that comes out the extruder given a commanded RPM and defined period of time. These systems require a manual calibration process that often involves timed extrusion tests to measure the mass flow rate and test prints to measure bead width for a variety of motion and extrusion speeds. These calibration tests must be repeated each time the material, temperature, extruder backpressure, or system hardware changes. This calibration process can be quite tedious and requires many hours of labor from a skilled operator.

Once the calibration process is complete, the calibration data can be used to configure settings for the slicing software to prepare an object to be constructed. Tools such as the ORNL Slicer 2 Flowrate Calculator [5], Figure 1, can be helpful for determining the parameters needed from the slicing process given a set of calibration data. However, each time the calibration changes, the slicer must be reconfigured for the new calibration data and all objects must be re-sliced. This is because slicing programs are configured to output precise machine feeds and speeds to tell the machine exactly how to operate. G-code doesn’t offload any processing to the machine because most machines don’t have knowledge of the calibration and output from testing. This is partly because machines don’t typically have a means of self-calibration, but also because machines don’t have the means of storing the calibration data and applying it during the printing process. A machine properly configured to calibrate itself and store the calibration data, such as the Juggerbot3D Tradesman Series, can make use of a g-code file with geometry information.


Slicer-2: Flowrate Calculator

Flowrate Measurement

Speed (RPM):	400
Lbs/Hour:	87.4

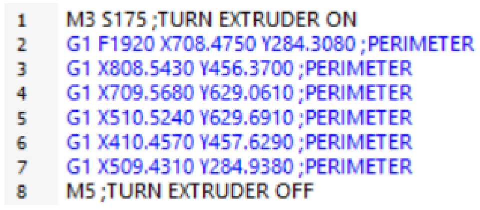
Printing Parameters

Bead Width (in):	0.34
Layer Height (in):	0.15
Desired Print Rate (lbs/hr):	75
Material Type:	20% CF-ABS
Density (g/cm^3):	1.14000
Gantry Speed (in/sec):	10.9558
Spindle Speed (RPM):	343

Figure 1: The ORNL Slicer 2 Flowrate Calculator used to take mass flowrate data and desired bead geometry to calculate the optimal print speed and extruder RPM for a given material.

G-Code Implementation

As previously mentioned, standard g-code implementations for pellet-based thermoplastic 3D printers involve RPM control for the extruder. At the start of the path, an RPM value is commanded, and this value stays constant until the path is completed and the extruder is turned off. This g-code structure can be seen in Figure 2 which details the lines of code needed to print a hexagon shape. This g-code snippet was generated using the ORNL Slicer 2 program, an open-source slicing software package designed for large format additive manufacturing [5].



```
1   M3 S175 ;TURN EXTRUDER ON
2   G1 F1920 X708.4750 Y284.3080 ;PERIMETER
3   G1 X808.5430 Y456.3700 ;PERIMETER
4   G1 X709.5680 Y629.0610 ;PERIMETER
5   G1 X510.5240 Y629.6910 ;PERIMETER
6   G1 X410.4570 Y457.6290 ;PERIMETER
7   G1 X509.4310 Y284.9380 ;PERIMETER
8   M5 ;TURN EXTRUDER OFF
```

Figure 2: G-code generated by ORNL Slicer 2 for a hexagon-shaped closed-contour with RPM commands for the extruder.

In Figure 2, line 1 turns the extruder on using the M3 command and sets the RPM to 175 via the S parameter. Lines 2-7, colored in blue, are G1 motion commands with X and Y values that represent the six vertices of a hexagon. Line 2 also contains an F parameter, which is used to set the feedrate, or the velocity, of the end effector. Units vary from system to system, but the most common units for this F parameter are millimeters per minute. These lines, 2-7, do not include an S or E parameter, meaning they are not commanding the extruder. The extruder was enabled at 175RPM on line 1 and continues at that constant speed until a new value is commanded or the extruder is disabled. The exception to this is systems that can modulate the extruder speed based on the acceleration and deceleration of the motion platform; however, the command value remains the same and the controller will do it's best to maintain 175RPM whenever the system is at steady state. The last line, line 8, contains the M5 command to turn the extruder off which effectively sets the extruder to 0RPM.

To get away from this constant commanded RPM value, the units and implementation of the g-code can be modified. Rather than interpret the value following the S parameter for motion commands as an RPM, the 3D printer can be configured to read the S parameter value as cross-sectional area or bead width. In either instance, an area or a width command, the system can use the new value with bead/layer height and existing calibration info to determine what RPM is needed to create the desired output bead geometry. For polymer systems, the assumed bead geometry can be represented as a rectangle with a half circle on each end (Figure 3).

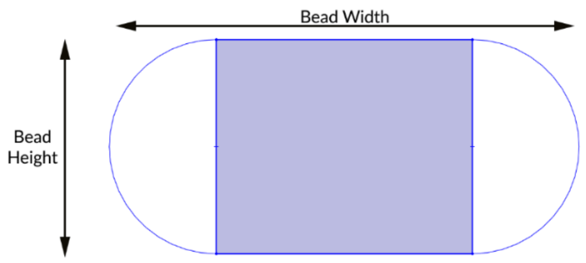


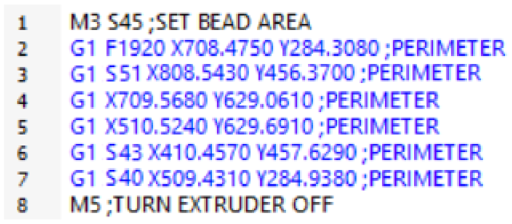
Figure 3: Cross-section of a 3D printed bead of plastic, approximated with a rectangle (shaded) and two half circles (white).

The cross-sectional area (CSA) of this bead can be found using equation 1, which provides the bead area in terms of the bead width (BW) and bead height (BH). The bead height can be passed as a parameter at the start of the program or user defined within the machine interface. Alternatively, the height can be commanded during every extrusion move by adding an additional parameter and passing the height value. For example, the line `G1 F1800 S40 H4 X10 Y0` can be used to set a feedrate of 1800, a bead area of 40, a height of 4, an X position of 10, and a Y position of 0. This is especially useful for systems with variable layer height. For this research, the height is passed in the header at the start of the print using a machine specific command, such as `M740 H4`, to set a height of 4 for the entire build.

Equation 1

$$CSA = \pi * \left(\frac{BH}{2}\right)^2 + BH * (BW - BH)$$

For a system with the ability to parse the geometry information, the bead geometry can be commanded on a per path or per segment basis, just like the traditional RPM control. This means that the bead geometry can be commanded at the start of the closed loop, like in Figure 2, so that the same bead geometry is applied to every motion segment until a new value is commanded. Alternatively, the bead geometry can be commanded for each segment over multiple lines as shown in Figure 4.



```
M3 S45 ;SET BEAD AREA
G1 F1920 X708.4750 Y284.3080 ;PERIMETER
G1 S51 X808.5430 Y456.3700 ;PERIMETER
G1 X709.5680 Y629.0610 ;PERIMETER
G1 X510.5240 Y629.6910 ;PERIMETER
G1 S43 X410.4570 Y457.6290 ;PERIMETER
G1 S40 X509.4310 Y284.9380 ;PERIMETER
M5 ;TURN EXTRUDER OFF
```

Figure 4: G-code generated by ORNL Slicer 2 for a hexagon-shaped closed-contour with RPM commands for the extruder.

To maintain backwards compatibility with other slicing programs and standard operating modes, specifically the commanding of RPM values, an m-code command should be added to the header of the g-code file to tell the machine that the G1 and M3 commands are passing geometry information, rather than speed information. This is called defining the active mode. The m-code for this command can be machine specific, based on available m-code commands. When this command is not present at the start of a file, the machine will operate with the traditional interpretation of commands where the S parameter passes an RPM value.

System Design

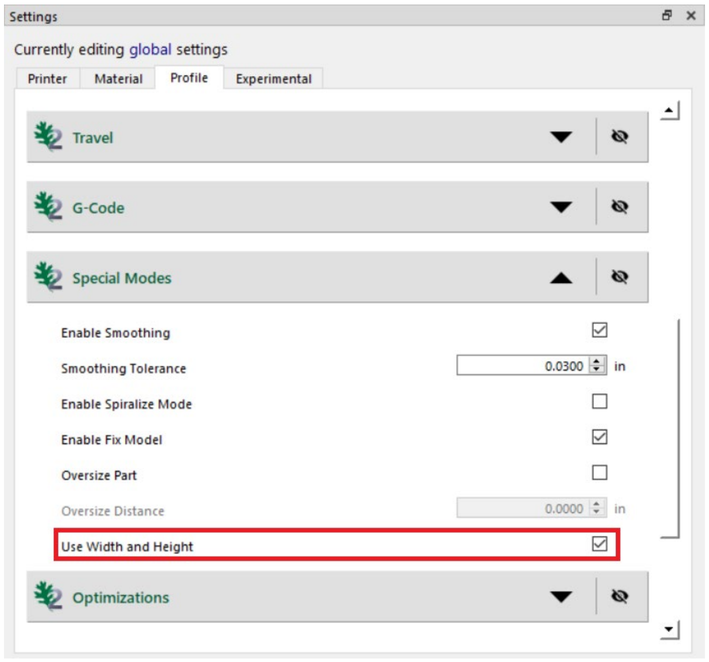
For the previously mentioned g-code changes to be properly implemented, a machine must have the proper systems in place for calibration and data storage. Calibration can be done by hand by printing test objects at various bead widths and extruder speeds, and then the operator can measure the output with calipers. This data can be logged and ultimately converted into a mathematical function to define the machine flowrate based on commanded extruder RPM. However, this is a tedious method that requires an operator to meticulously measure various points along the print and input the data. A more robust implementation is discussed by Chesser is his paper about calibration for accurate extrusion width [3]. He mentions using a laser profilometer to measure the height and width of the bead in real-time as a gantry system moves along the extrusion path. This system works by projecting a laser line bead across the bead and monitoring the reflection of the light to calculate width and height of the object being measured. The computer can use the measurement data and calculate the cross-sectional area, which can be correlated to the path length to calculate volume.

Whether an automated calibration system is implemented, or manual measurements are taken, the system must be able to equate a commanded RPM value to the resultant output bead cross-sectional area. With this information, the machine can then parse the desired bead area or bead width from the g-code command and calculate what RPM value is needed to achieve the desired geometry. For this to work, the system needs to know what material and calibration profile to apply to the print. A material for the printing operation can be selected through a variety of methods such as adding a material parameter to the header of the g-code file with an m-code, having the operator select the material from a dropdown menu on the machine interface, or scanning a barcode from the material being loaded so that the machine knows exactly what material and version of the material (such as the manufacturing lot number) is in use.

For all the options of parsing and implementing the g-code with geometry data, the machine itself needs to understand what is meant by the selected material for the printing operation. This is best done with a database of the calibration data for all the various materials that could be used on the machine. A more robust implementation would also contain data points for the other variables that can impact extrusion output such as temperature, extruder pressure, extruder torque, nozzle diameter, and more. As more variables are accounted for, the mathematical function that the machine must calculate in real-time gets more complex. Future work for system manufacturers will involve pushing the limit to build a database with as much information as possible to calculate as accurate of an output while still maintaining real-time control.

The Results

This new g-code format and slicing approach has been developed into the ORNL Slicer 2 software package. The code for these changes is available open source [5]. Currently, the functionality has been added to machines running a modified Marlin firmware to accept the bead area commands. A setting has been added for the user to enable or disable transmitting the geometry information in the g-code, seen in Figure 5. When enabled, the current layer height and commanded bead width are used with Equation 1 to calculate the bead area to be transmitted in the g-code. This mode also enables the output of a machine specific m-code to the header of the file to ensure the machine is in the proper mode.


Settings

Currently editing global settings

Enable Smoothing	☑
Smoothing Tolerance	0.0300 in
Enable Spiralize Mode	☐
Enable Fix Model	☑
Oversize Part	☐
Oversize Distance	0.0000 in
Use Width and Height	☑

Figure 5: New setting added to enable transmitting of bead geometry information.

As shown in Figures 2 and 4, the resultant g-code file is similar to the formatting of the standard g-code file with RPM control. The file length and size are relatively unchanged, excluding a few g-code lines added to the header to define the layer height and establish the printing mode. The time to slice an object and write g-code within the slicer is also nearly identical to the standard approach.

To test the functionality, a Juggerbot3d Tradesman Series P3-44 was configured to parse the geometry information from the g-code. Calibration data was stored on the machine, so that the system could properly calculate and apply the correct extruder control for an accurate print. Figure 6 shows a completed chair, 3D printed with this new functionality.

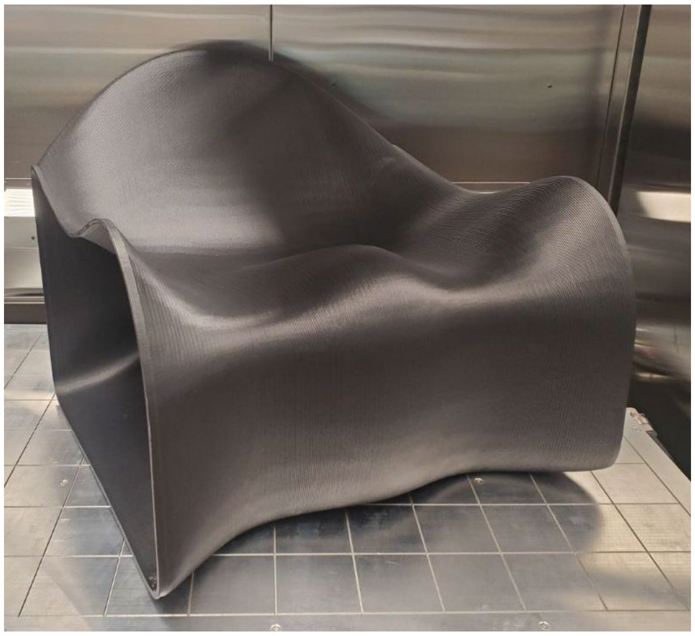


Figure 6: A chair 3D printed using g-code with geometry information.

When implemented correctly, the results of this work are nearly indistinguishable from a well calibrated machine with calibrated slicing software. The benefit of this work is the time savings: the reduction in man hours to calibrate a machine, the reduction in time slicing the object and creating proper material profiles, and the reduction in time spent reslicing parts based on minor adjustments needed for material or flow changes.

Conclusion

Standard g-code implementations utilize hard coded parameters that tell the 3D printer exactly what to do for each step of the construction process. More specifically, these g-code files define the extrusion rate, typically through RPM commands, that must be user-defined during the slicing process. If the material or machine configuration changes, the g-code must be recreated to adapt to the change in condition. The method described in this paper allows the g-code to be created based on part geometry, without any knowledge of material or machine calibration data. This same file can be used for multiple materials or system configurations. This approach allows the designer to do the slicing and be decoupled from the printing process. For this method to succeed, the system must be calibrated by an operator or an automated system. This calibration process gives the machine the necessary data to interpolate the necessary extrusion rate based on commanded bead geometry.

Future Work

The current implementation outlined in this paper focuses on g-code toolpaths with bead geometry cross-sections in place of extrusion commands. This is only the first step towards a more robust g-code file. Future work will involve commanding a layer time along with total layer path distance, rather than a feedrate, so that the machine itself can determine the ideal feedrate based on calibration data for flowrate. Additional work will explore the possibility of transmitting entire layer geometry data in the g-code. This would include the size and shape, perhaps as a polygon, of how the entire layer should appear after printing. If the machine can properly read this data, as well as measure the geometry of the printed layer, a quick comparison can be calculated to understand if the layer was printed correctly. This implementation would require sensors for measuring the printed layer. Similar research is documented by Borish in a recent paper, but the implementation requires a second PC to parse the data as well as additional files output from the slicer rather than one single g-code file containing all the data [6].

References

1. Fox, Chris. “A Little CNC History.” Tormach Blog, blog.tormach.com/cnc-history. Accessed 30 June 2023.
2. Pabla, B. S., and M. Adithan. *CNC machines*. New Age International, 1994.
3. Chesser, Phillip, et al. "Extrusion control for high quality printing on Big Area Additive Manufacturing (BAAM) systems." *Additive Manufacturing* 28 (2019): 445-455.
4. Hershey, Christopher, et al. *Large-scale reactive extrusion deposition of sparse infill structures with solid perimeters*. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2019.
5. Roschli, Alex, Borish, Michael, Barnes, Abigail, Wade, Charles, Crockett, Breanne, White, Liam, and Adkins, Cameron. ORNL Slicer 2 - Open Source Copyright. Computer Software. https://github.com/ORNLSlicer/Slicer-2. Web.
6. Borish, Michael, et al. "Real-time defect correction in large-scale polymer additive manufacturing via thermal imaging and laser profilometer." *Procedia Manufacturing* 48 (2020): 625-633.