2018

G-code Modeling for 3D Printer Quality Assessment

Tyler Welander

Ronald Marsh
University of North Dakota, ronald.marsh@UND.edu

Md Nurul Amin
m.amin@ndus.edu

G-code Modeling for 3D Printer Quality Assessment

Tyler Welander
Department of Computer Science
University of North Dakota Grand Forks, ND, 8203
tyler.welander@und.edu

Ronald Marsh
Department of Computer Science
University of North Dakota Grand Forks, ND, 58203
rmarsh@cs.und.edu

Md Nurul Amin
Department of Computer Science
University of North Dakota Grand Forks, ND, 58203
m.amin@ndus.edu

Abstract

The Department of Computer Science at the University of North Dakota (UND) has been evaluating optical/imaging methods for measuring the quality of 3D printed parts. In particular, we are interested in optical/imaging methods that can detect and measure such quality issues as layer shifting, layer separation and splitting, overheating, dimensional accuracy, and infill errors.

This paper describes our work towards the analysis of infill errors as the quality of the infill does impact the structural integrity of the part being made. Externally, a part may look acceptable, but if the infill is faulty the part may be structurally unsound. Furthermore, once a part is finished printing it is usually not possible to see the infill. Therefore, monitoring of the infill must be done while the part is being printed.

Introduction

The Department of Computer Science at the University of North Dakota (UND) has been evaluating optical/imaging methods for measuring the quality of 3D printed parts. In particular, we are interested in optical/imaging methods that can detect and measure such quality issues as layer shifting, layer separation and splitting, overheating, dimensional accuracy, and infill errors.

To evaluate infill error, we used both a camera to acquire the images and a computer model of part being made. The intent is to be able to generate a model of the internal structure of the part being made by parsing the G-code (RS-274) layer by layer and then comparing the computer model with an image of the part being generated. For those unfamiliar with numerical control (NC) programming languages, G-code is a language which instructs computerized machine tools how to make something by telling the device's motors where to move, how fast to move, and what path to follow. For this project the G-code is parsed and a 3D computer model, using OpenGL, is created. This allows us to rotate or tilt the computer model and adjust the lighting model to best match the physical environment. In our case, we then used a Logitech Brio 4K webcam to acquire the images and a XYZ printing da Vinci 1.0 3D printer to fabricate the parts. In this paper we will provide details as to the G-code parsing and OpenGL model, examples, and the results of the study.

Background

3D printing is an additive manufacturing process where products are produced layer by layer in cross sectional slices [1]. 3D printing is becoming commonly used in producing customized and low production run (low volume) products. Examples of where 3D printing has been used as a meaningful and reliable way of manufacturing include: replacement parts of machine, prototypes, and medical components. Prototyping, in particular, is an area where 3D printing has multiple advantages: 1) ease of duplicating products, 2) security and privacy consideration of product design, and 3) low cost in customized manufacturing [1].

3D printing was first introduced in 1970 [2] and 3D printers work in a way similar to traditional laser or inkjet printers; however, instead of using multi-colored inks, they use filament which slowly builds the piece layer by layer by melting down and depositing the extruder material [1]. Software is used to create thousands of cross-sectional slices of each design to determine which layer is to be constructed and how to construct it. A 3D printer can produce a simple object like a gear in less than an hour [1].

The 3D printing type varies on the materials used. Experiments have been done to produce a versatile range of products like biodegradable materials [3], pharmaceuticals [5,6], microfluids [8], imaging apertures [4] and nanocomposites [7] all of these have been 3D printed. It can be used for preserving historical objects [9] and creating educational excitement [10].

3D printing is an immerging field, as such, we need to be aware of the drawbacks of this technology. Some researchers have found carcinogenic (radioactive) emissions from 3D printed materials when they are melted for printing/extruding [11]; and, this may have a huge impact on society, but analysis of those impacts is beyond the scope of this paper. Our goal is to determine if we can find defects in 3D printed objects at the time of printing through analysis of images. As a typical 3D printer can't detect defects in the products, our concern is to provide a mechanism that can operate during the print phase, a mechanism that can detect faults while the part is being printed and eventually be able to then instruct the printer to stop printing (i.e. abandon the part).

To find defects in already printed parts, we intend to scan the part using a 3D high resolution camera. Such approaches are already commonly in use. Three-dimensional scanning is used for measuring object shape and size [12], evaluation of cosmetic products used in reshaping human bodies [13], and in creating custom wear clothing based on body shape [14]. 3D scanning also has many applications in medical science such as in finding defects of a person's skeletal structure [15], and in validating the quality of automotive products [16] such as turbine blades.

There are a lot of 3D scanning technologies and some are good for short distant scanning and some are better suited for mid or long-range scanning. Some 3D scanning technologies are better for small objects and some are better for large objects like aircraft [17]. Short range scanners typically use laser triangulation or structured light technology. For longer range scanning, the scanner projects a laser line or multiple lines onto an object and then captures the reflection with single or multiple sensor(s). It is critical that the sensors be located in a known position/distance in order to be able to accurately measure the structure of experimental object. Is such cases, the reflection angel is considered when finding actual points of construction in certain areas.

In this work, instead of using laser, we used a 4k camera to monitor the shape of the part under construction inside the 3D printer. We also used the 4k camera to analyze the parts after completion where we evaluated the parts with respect to warp, tilt, deformation, and delamination; However, those analyses are not of interest to this paper.

G-code Parser

As noted above, G-code [18, 19] is a numerical control (NC) programming language which instructs computerized machine tools how to make something by telling the device's motors where to move, how fast to move, and what path to follow. In this case, the G-code is comprised of numerous commands that are mostly used for initializing the printer and are not used in the generation of the part. These commands are not of interest to us. The OpenGL parser only used two commands from the entire set of possible G-code commands. The two commands are G0 and G1. All the other commands are used to either set up the printer, such as setting the bed temperature or adjusting the bed height, or switching the coordinate system between absolute positing or relative positing. Absolute positing is the coordinates from the nozzle origin which was in the lower left-hand corner

of our particular printer. Relative positing is used to move the nozzle a certain distance from its current location, such as moving the nozzle up five units. The units are generally in millimeters. The reason why we didn't have to worry about which coordinate system we are in is because the printer switches between these two systems at very specific times during the printing. At these times the nozzle wasn't extruding any filament, so it wouldn't affect the OpenGL model.

The G0 and G1 commands have three components that we have to worry about. The first component is the actual G0 or G1 part. The parser uses this component to determine what should be done next (in regards to generating the OpenGL model). The next component are the position coordinates. There is at least an X, Y, or Z coordinate. This first component is where the parser extracts the coordinate points for the vertices of the OpenGL model. The last component is the for the extruder and tells the printer to extrude filament. For the parser, we only need to know if we, or are not, extruding filament. If we are then the parser would automatically calculate the correct amount of filament to extrude (in the OpenGL model). If filament is not to be extruded, we would use this parameter to keep track of the nozzle position (in the OpenGL model). Figure 1 depicts a sample of the G-Code.

```
G1 X149.699 Y143.200 E10.11763
```

Figure 1: Example G-code Command.

The parser has two main stages. The first stage is the extraction stage of the necessary data from the G-Code. This stage would go through the G-code file line by line and get the coordinate points for each vertex and determine if there is filament between two vertices. All the data for the coordinate points and the filament extraction is held in a structure (shown in figure 2). The X, Y, and Z coordinates are held in GLDouble variables. The filament extraction value is held as an boolean value. The second stage of the parser is where the program goes through the coordinate points and renders the OpenGL model.

```
struct coordinatepoint
{
      GLdouble xCoordinate;
      GLdouble xCoordinate;
      GLdouble xCoordinate;
      bool draw;
};
```

Figure 2: Structure for G-code Data.

The extraction stage of the parser has a few steps to it. The first step is to remove all the comments. G-code has two ways of denoting comments. The first way is with a semi-colon. The parser will scan a line for a semi-colon. If it finds one, it will remove everything after the semi-colon. The second way of denoting a comment is with the use of a set of parentheses. If the parser finds an opening parenthesis, it will scan the remainder of the line for the closing parenthesis. After it finds the closing parenthesis, it will remove the substring where the comment is.

The extraction stage second step is to determine if it is a G0 or a G1 command. It does this by testing the first character of the line. If it is a 'G', then it tests the next character of the

line. If the first character is not a ‘G’, then it moves on to the next line. If the second character is a ‘0’, it sends the remainder of the line to be processed a function that corresponds to the G0 command. If the second character is a ‘1’, the remainder of the line is sent to a function for the G1 command.

The G0 command is used to tell the printer to move from one point to another as fast as possible in all axes. For example, if the nozzle needs to move 3 millimeters in the x axis and 6 millimeters in the z axis and the max speed in any axis in 3 millimeters per second, it would move the nozzle 3 millimeters per second in the X direction and 3 millimeters per second in the Z direction. After one second has passed, the speed in the X direction will be reduced to 0 millimeters per second and the speed in the Z direction will remain the same. Once the nozzle reaches its destination, the velocity in all axes is reduced to 0. This was implemented in the parser by breaking the command into multiple commands. The parser would calculate the line that would be produced by the first part of the command until the nozzle switched directions. It would then store the two endpoints of the line in the global vector holding all the vertices. The parser would then calculate the remaining part of the line and put those endpoints in the same vector. The last step for the parser would be to check if there was an ‘E’ component for the G-Code command. If there was, the filament extraction value in the coordinate point struct would be set to true, instructing the parser to draw a section in the OpenGL model. If the ‘E’ parameter doesn’t exist in the G-Code command, the filament extraction value would be set to false, instructing the parser to simply update the position of the virtual print head in the OpenGL model. The parser would still have to worry about that particular G-Code command even if there was no ‘E’ parameter because the parser needs to be able to track where the nozzle is.

The G1 command is used to tell the printer to move from one point to another in one straight line. This is accomplished by the nozzle using different speeds for each axis, so every axis reaches their destination at the same time. The parser simulates this by just adding the destination coordinate to the coordinates vector. The parser would also set the filament extraction value to true if the ‘E’ parameter is present in the particular G-Code command. If the ‘E’ parameter is not present, the filament extraction value is set to false.

The next step of the parser is to take the coordinates and map them to the OpenGL model. This is done is by placing spheres along where the filament would be extruded. This was done so we could include a lighting model and render the OpenGL model as photorealistic as possible. The way we implemented this was by calculating the total amount each axes would change. Then we calculated the total length of the line the spheres would be covering by calculating the Euclidean distance. After that, we divided the distance of the axes by the total length of the line. Then we used these values to increment along the line to place the spheres. Figure 3 provides this code.

```
double xDist = next.xCoordinate – prev.xCoordinate;
double yDist = next.yCoordinate – prev.yCoordinate;
double zDist = next.zCoordinate – prev.zCoordinate;
double totalSpheres = floor(10 * totDist);
double xGap = xDist/totalSpheres;
double yGap = yDist/totalSpheres;
double zGap = zDist/totalSpheres;
```

```
for (int i = 1; i <= totalSpheres; i++)
{
        popMatrix();
        glTranslated(xGap * i, yGap * i, zGap * i);
        glutSolidSphere(1, 10, 10);
        pushMatrix();
}
```

Figure 3: Code for placing spheres.

Once everything is drawn to the screen, we rotate, scale, and translate the model and adjust the lighting model to best match the model being printed and its environment. The rotation, scaling, and translation of the model was done by just mapping different keyboard keys to global variables and applying those values to the OpenGL commands for rotation, scaling, and translation. The lighting model was implemented by just adjusting the different values for each type of light. The types of lights we used were ambient, diffuse, and specular. In this case, since the two physical parts were made of different types of materials, each OpenGL model had to have a unique lighting model even though they were in identical environments.

Results

For the this work we used two different models. While both were of a 10cm cube, each one had different infill patterns and densities. Our first model utilized a honeycomb infill with a density of about 12.5%. The filament was a black color. Figure 4 provides a photograph of the physical part with figure 5 showing the OpenGL model.

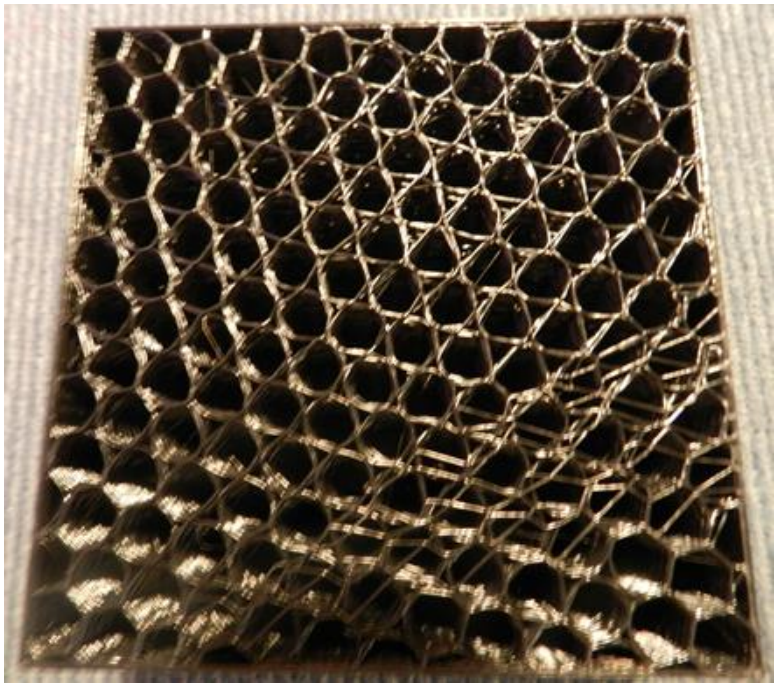

Figure 4: 3D Printed part.

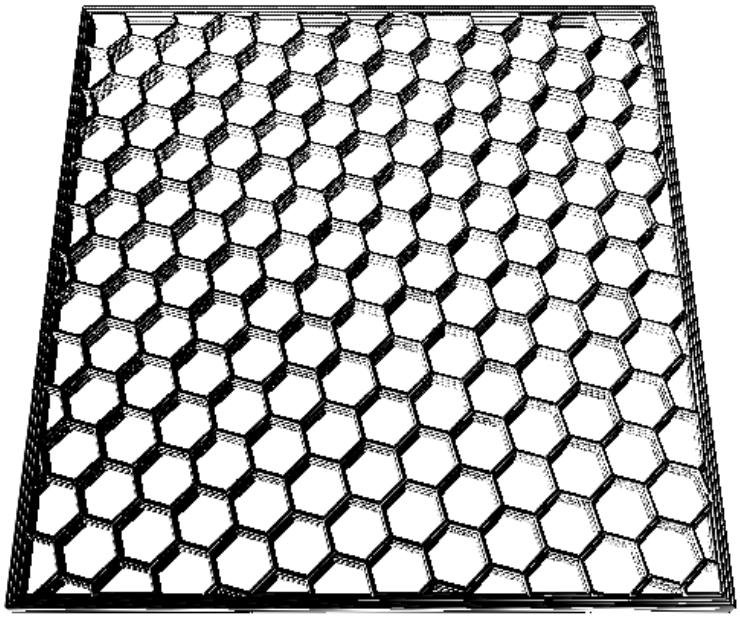

Figure 5: OpenGL Model

Our second model utilized a rectilinear/diagonal infill with a density of about 50%. The filament was a red color. Figure 6 provides a photograph of the physical part with figure 7 showing the OpenGL model.

Figure 6: 3D Printed Model

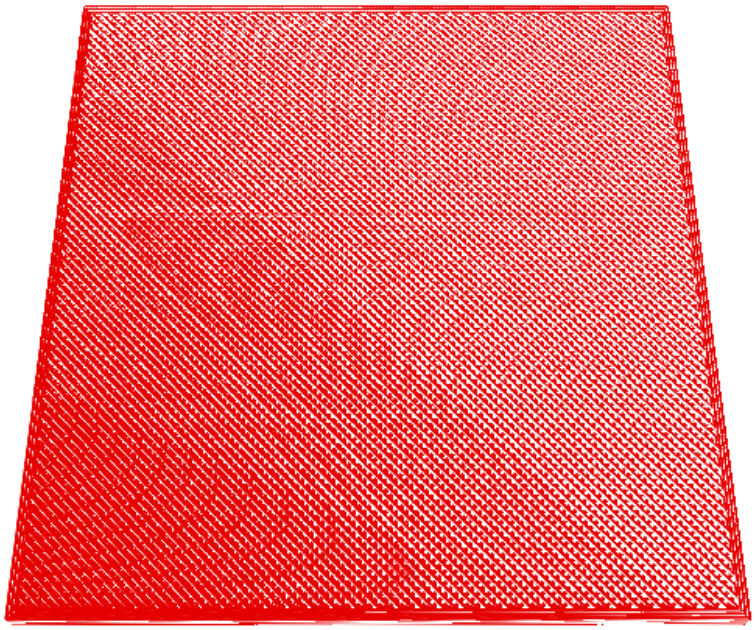

Figure 7: OpenGL Model

Conclusion

This paper describes our work towards the analysis of infill errors as the quality of the infill does impact the structural integrity of the part being made. Externally, a part may look acceptable, but if the infill is faulty the part may be structurally unsound. Furthermore, once a part is finished printing it is usually not possible to see the infill. Therefore, monitoring of the infill must be done while the part is being printed.

The intent was to use a camera to take pictures of a part as it was being printed and to compare those images with an image generated by an OpenGL model. A model that would be generated from the printer's G-code at the same time and same rate as the part being printed. Theoretically, if done properly, the OpenGL model and the actual physical part would coincide in every visible way. The OpenGL model would be scaled and rotated to match the camera angle and the OpenGL lighting model would be tailored to match that of the printing environment.

We ran into a number of problems. Firstly, the Logitech Brio 4K webcam has a very wide field of view creating a lot of distortion when trying to image objects up close, as in our case. Therefore, the images presented were actually taken with a Nikon Coolpix camera. Secondly, using a low-cost da Vinci 1.0 3D printer resulted in a variety of infill variations. When using a rectilinear/diagonal infill pattern with a density of about 50%, the physical part and OpenGL model correlated well. However, this infill pattern generated a lot of heat and the 3D printer's build plate eventually shattered. When using the honeycomb infill pattern with a density of about 12.5%, probably the most common infill pattern and density, the physical part and the OpenGL model did not match very well at all. At this point we

are of the opinion that this approach will only be truly feasible when higher density infills are used.

Acknowledgements

This work was funded by the North Dakota Department of Commerce award #16-02-J1-114.

References

1. Berman, B. 3-D printing: The new industrial revolution. *Business Horizons*. 2012, 55, 155–162.
2. Bowyer, A. 3D Printing and Humanity's First Imperfect Replicator. *3D Printing and Additive Manufacturing*. 2014, 1, 4–5.
3. Serra, T.; Planell, J.A.; Navarro, M. High-resolution PLA-based composite scaffolds via 3-D printing technology. *Acta Biomaterialia*. 2013, 9, 5521–5530.
4. Miller, B.W.; Moore, J.W.; Barrett, H.H.; Fryé, T.; Adler, S.; Sery, J.; Furenlid, L.R. 3D printing in X-ray and Gamma-Ray Imaging: A novel method for fabricating high-density imaging apertures. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment.* 2011, 659, 262–268.
5. Khaled, S.A.; Burley, J.C.; Alexander, M.R.; Roberts, C.J. Desktop 3D printing of controlled release pharmaceutical bilayer tablets. *International Journal of Pharmaceutics*. 2014, 461, 105–111.
6. Sanderson, K. Download a drug, then press print. *New Scientist Magazine*. 2012, 214, 8–9.
7. Campbell, T.A.; Ivanova, O.S. 3D printing of multifunctional nanocomposites. *Nano Today* 2013, 8, 119–120.
8. Bonyár, A.; Sántha, H.; Ring, B.; Varga, M.; Gábor Kovács, J.; Harsányi, G. 3D Rapid Prototyping Technology (RPT) as a powerful tool in microfluidic development. *Procedia Engineering*. 2010, 5, 291–294.
9. Abate, D.; Ciavarella, R.; Furini, G.; Guarnieri, G.; Migliori, S.; Pierattini, S. 3D modeling and remote rendering technique of a high definition cultural heritage artefact. *Procedia Computer Science*. 2011, 3, 848–852.
10. Eisenberg, M. 3D printing for children: What to build next? *International Journal of Child-Computer Interaction*. 2013, 1, 7–13.
11. Stephens, B.; Azimi, P.; El Orch, Z.; Ramos, T. Ultrafine particle emissions from desktop 3D printers. *Atmospheric Environment*. 2013, 79, 334–339.
12. DeMatto, A. 5 Ways Body Scanners Could Make Fitting Rooms Obsolete. Available online: http://www.popularmechanics.com/technology/gadgets/a5909/3d-body scanning-technologyapplications/ (accessed on 13 March 2018).
13. Ares, M.; Royo, S.; Vidal, J.; Campderrós, L.; Panyella, D.; Pérez, F.; Vera, S.; Ballester, M.A.G. 3D Scanning System for In-Vivo Imaging of Human Body; *Fringe* 2013; Springer: Berlin/Heidelberg, Germany, 2014; pp. 899–902.

14. King, R. 3D Imaging Spreads to Fashion and Beyond. Available online: http://www.hpcwire.com/2008/10/06/3d_imaging_spreads_to_fashion_and_beyond/ (accessed on 13 March 2018).
15. Stephan, C.N.; Guyomarc'h, P. Quantification of Perspective-Induced Shape Change of Clavicles at Radiography and 3D Scanning to Assist Human Identification. *Journal of Forensic Sciences*. 2014, 59, 447–453.
16. Voicu, A.; Gheorghe, G.I.; Badita, L. 3D Measuring of Complex Automotive Parts Using Video-Laser Scanning. Available online: http://fsim.valahia.ro/sbmm.html/docs/2013/mechanics/18_Voicu_2013.pdf (accessed on 1 April 2015).
17. ems-usa.com, Types of 3D Scanners and 3D Scanning Technologies. Available online: https://www.ems-usa.com/tech-papers/3D%20Scanning%20Technologies%20.pdf (accessed on 13 march 2018).
18. G-code, Available online: https://en.wikipedia.org/wiki/G-code (accessed on 13 march 2018).
19. Smithy CNC, http://server2.smithy.com/media/pdf/G-codes%20Program%20Guide.pdf (accessed on 13 march 2018).