From GCode to STL: Reconstruct Models from 3D Printing as a Service

Felix W Baumann[1,2,3], **Martin Schuermann**[1], **Ulrich Odefey**[1] **and Markus Pfeil**[1]

[1] TWT GmbH Science & Innovation, D-70565 Stuttgart, Germany
[2] University of Stuttgart, Institute of Computer-aided Product Development Systems, D-70569 Stuttgart, Germany

[3] Author to whom any correspondence should be addressed: felix.baumann@twt-gmbh.de

Abstract. The authors present a method to reverse engineer 3D printer specific machine instructions (GCode) to a point cloud representation and then a STL (Stereolithography) file format. GCode is a machine code that is used for 3D printing among other applications, such as CNC routers. Such code files contain instructions for the 3D printer to move and control its actuator, in case of Fused Deposition Modeling (FDM), the printhead that extrudes semi-molten plastics. The reverse engineering method presented here is based on the digital simulation of the extrusion process of FDM type 3D printing. The reconstructed models and pointclouds do not accommodate for hollow structures, such as holes or cavities. The implementation is performed in Python and relies on open source software and libraries, such as Matplotlib and OpenCV. The reconstruction is performed on the model's extrusion boundary and considers mechanical imprecision. The complete reconstruction mechanism is available as a RESTful (Representational State Transfer) Web service.

1. Introduction

With 3D printing it is possible to create physical objects from a variety of materials using only a digital model and a specialised tool, the 3D printer. The manufacturing of the models works by extrusion of material, curing of material, fusion of material or other generative material manipulation techniques. It is carried out by a special machine that executes a machine-dependent instruction set. This instruction set, also known as machine code, is commonly used in a standardized format called GCode [1]. The machine code is generated by a software called slicer. Numerous slicers exist, as open-source software, such as *Slic3r*, freeware, such as *Cura*, or commercially available software, such as *Simplify3D*. For an overview of the availability and influence of this software see [2].

The resulting GCode is partially machine or vendor specific, as vendors implement parts of the instruction set or extensions thereof. It is furthermore dependent upon the slicing software. The proposed implementation is developed for machine code generated with *Slic3r*, version 1.2.9.

For Fused Deposition Modeling (FDM) type 3D printing, the machine code contains information as to where the printhead moves, what the movement speed is, and how much material is to be extruded along the movement path. This code is generated from an intermediary file format, commonly STL (Stereolithography, file format) or AMF (Additive Manufacturing File Format). For this paper, STL and AMF are considered as equal for the purpose presented and only STL is discussed where necessary. The STL file format contains the surface representation of the original model as oriented triangles. STL is the resulting file format for the reconstruction method of this paper. The details of the file transformations are described in Sec. 2. In 3D printing the resulting physical object is an approximation to the original digital 3D model. The latter model is often created using CAD software. The object is only an approximation as information is lost during the file conversion steps. Information is also lost due to 3D printing strategies. Furthermore, information is lost through strategies to compensate for technology specific alterations such as shrinkage. These strategies are employed by the slicing software to various degrees. The physical creation is, furthermore, limited by mechanical constraints. In cases where the creation of an object must be certified to adhere to the original template, i.e., the digital model, it is required to make statements on the degree of conformity or identity. Alterations to the model are not possible when only the GCode is available. Reverse engineering does not completely recover the original model, as information is lost due to approximations.

Currently, a limited number of implementations for model reconstruction from GCode exists. One such solution is commercially available at *Makeprintable*, another one is presented by [3]. To the knowledge of the authors, no open source service is available.

In this paper, a method to reconstruct the 3D model based on the available GCode is presented. This method is implemented and the implementation simulates the extrusion process of FDM type 3D printing graphically. From the slice-wise graphical extrusion, the layer geometry is reconstructed and then triangulated to create the respective STL model. The resulting model is suitable to be compared to the original model, and to calculate difference or identity metrics. The implementation of the method for simulation and the model reconstruction is performed in *Python* and relies on open source software as described in Sec. 4. The reconstruction mechanism is available as a RESTful Web service. The focus is on GCode files created for FDM 3D printing with a Cartesian geometry using *Slic3r*. Model reconstruction is important for two reasons. Firstly, the question whether the fabricated model is still corresponding to the original model or if it was altered during the transfer from the user to the printer can be answered. Secondly, model reconstruction is important for reverse-engineering of existing GCode files for which no corresponding digital model is available.

The remainder of this paper is structured as follows: In Sec. 2, an explanation and introduction to 3D printing is provided. In Sec. 3 relevant and related work to this approach is examined and discussed. The main work, i.e., the presentation of the reconstruction approach, is outlined in Sec. 4. The implementation overview for the Web service is provided in Sec. 5. The proposed approach is finally discussed in Sec. 6, whereas a conclusion and an outlook are provided in Sec. 7.

This paper is an extension to [4] focusing on the reconstruction aspect implemented as a Web service.

2. Additive Manufacturing

Additive Manufacturing (AM) and 3D printing are used synonymously in the literature. Further synonyms include Rapid Manufacturing, Generative Manufacturing, Additive Layer Manufacturing, Layered Manufacturing or Rapid Prototyping, with an additional meaning of the last term for a broader method of prototyping. 3D printing has the tendency to be used for consumer-focused additive fabrication, whereas AM commonly denotes the industrial application of these technologies. Both terms describe a number of different technologies that are used to create physical objects directly made from different materials using only a 3D printer as a manufacturing tool. Furthermore, robots or CNC machines [5] equipped with special actuators can be used for 3D printing. The materials available for processing [6] include plastics, metals, ceramics, bio-inspired or bio-compatible materials, and glass [7]. The various technologies and techniques differ fundamentally from each other so that files that must be prepared in the 3D printing process differ as well. The 3D printing process, see also [8-10], can be described in seven steps with the first step being the model generation in a CAD (Computer Aided Design) environment or another software. The second step is the conversion stage from the accurate digital 3D CAD model to an AM intermediary file format such as STL or AMF. These files are digital representations with reduced accuracy and information, suitable for 3D model interchange and interoperability. The third step is the placement and orientation of the 3D model within the virtual build environment of the 3D printer. The fourth step is the creation of the specific machine code from the digital model. This machine code adheres mostly to GCode [1] standard. The 5th step is the actual fabrication of the object in the 3D printer which can be error prone, especially in consumer-grade 3D printers, due to unaligned axes or build plate, imprecise parts or lack of adhesion of the object. Following step five, the object must be post-processed, with varying intensity and goals. The seventh step is the testing of the object according to a test protocol. In literature, an eighth step, which is the actual usage, is sometimes regarded as being part of the 3D printing process.

3. Related Work

Tsoutsos et al. [3] presented a very detailed description of the distributed process chain for AM to help secure 3D printing. The authors proposed a reverse engineering mechanism to reconstruct digital 3D models from acquired GCode toolpaths. In their work, the authors also analyse the resulting models using finite element analysis (FEA). Their simulation creates basic geometric shapes according to the extrusion which are then fused and processed to result in an approximation of the original model. The difference between their approach and the one proposed here is that they use solid geometry generation directly on the GCode simulation, whereas the approach in this paper creates an intermediary point cloud first. Gao et al. [11] proposed a reverse engineering method for blade reconstruction in the aerospace industry. In their work, they reconstructed a worn-out turbine blade of an engine. The reconstruction utilised a digitiser and a CAD model reconstructed with *CATIA V5*. The authors implemented their approach with an integrated additive and machining based manufacturing. The work by Thrimurthulu et al. [12] discussed the simulation of FDM type 3D printing to optimise surface roughness and part orientation. Relevant to the work presented herein is the discussion on process parameters and influences on the material deposition. Okarma and Fastowicz [13] presented an approach to assess the quality of FDM printed objects based on feature similarities from image analysis. The authors did not reconstruct the shape or model of the original but compared the resulting surfaces graphically. Zhang and Chout [14] developed an FEA-based simulation method for FDM 3D printing. The simulation was used for a specific specimen to assess the influence of process parameters on the resulting object. Shrivastava and Modi [15] presented an approach to reconstruct the digital model from a point cloud acquired by a coordinate measuring machine. The authors used the *CATIA V5* software to construct the CAD representation of the physical model. This method, like the approach presented here, is only suitable for the reconstruction of the outer shape, with limitations on holes and other concave structures. The difference is that Shrivastava and Modi relied heavily on the reconstruction via the CAD software and did not discuss reconstruction principles.

4. Model Reconstruction

Figure 1 shows the proposed concept of the model reconstruction. GCode files are used as input and no additional comments in the file are considered. The input file is preprocessed to remove comments and unexpected instructions, and to rewrite certain instructions. A dialect that is compatible with the *Sailfish* firmware is implemented in the simulation core.

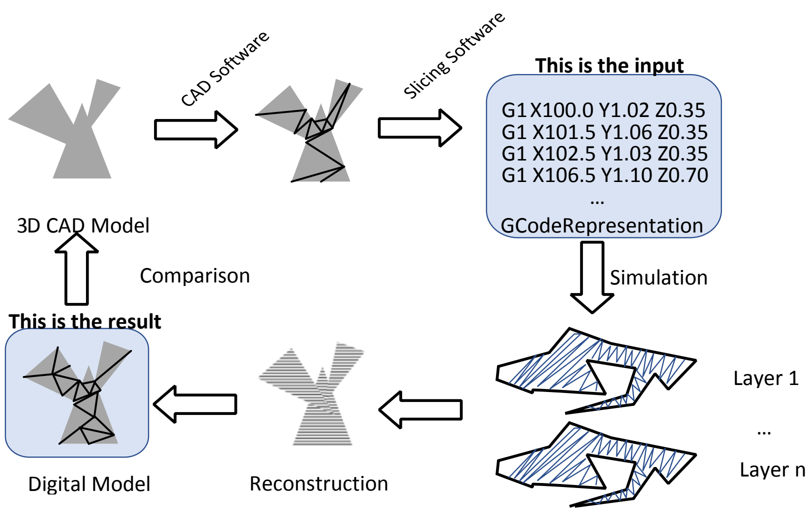

Figure 1. Flow of Reconstruction

4.1. GCode and Simulation Basics

The simulation core mainly processes **G1** (linear movement) instructions that contain information about the **X**, **Y**, and **Z** coordinates to which the printhead and the build plate are moved. Furthermore, these instructions contain information about the movement speed (**F**) towards these coordinates which are given as target movement speeds in mm/min. Information on the extrusion material, material extruded in mm along the path, is given in the **G1** commands per separate extruder, denoted by **A** and **B**, or cumulatively, denoted as **E**. In case of using the E parameter, the GCode allows for the switching of extruders which are also known as tools. The following two lines demonstrate the structure of the **G1** and **G0** commands.

G1 F1800 X89.999 Y89.150 E49.60291 G0 X89.999 Y89.716 F9000

G0 is used for rapid linear movement and **G1** for normal linear movement. Most implementations ignore this difference and treat them the same. Our system rewrites **G0** instructions to **G1** instructions. As shown in the example, the position of the parameters for each instruction is flexible. Instructions with partial parameter sets, such as *G1 X89.111* preserve parameters set previously, such as the **Y** and **Z** coordinates missing in this example. Every instruction parameter is active until set otherwise.

In GCode, all given coordinates are target coordinates, meaning that they must be interpreted within a context, e.g., the last target, to infer information about path lengths and extrusion volumes. The extrusion volume is also dependent on the filament diameter. Common diameters for FDM 3D printers are either 3.0 mm or 1.75 mm. A variation of 10 % of the diameter of the filament can result in approximately 20 % variation of the resulting material extrusion, for an exemplary extrusion of 10 mm material over 10 mm.

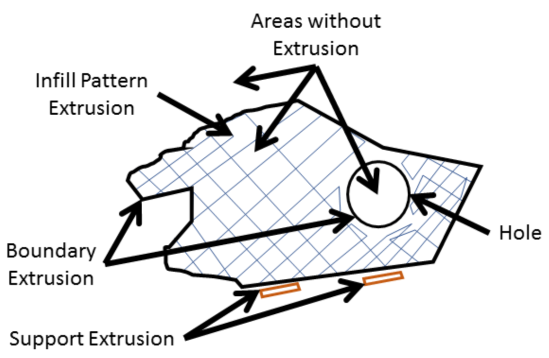

Figure 2. Extrusion Information per Layer

The simulation is based on physical processes to describe the movement of and within the 3D printer. All axes are moved with pre-defined acceleration profiles and can be exercised at a specific maximum velocity. The extrusion simulation is performed on a virtual canvas provided by *Matplotlib*. This extrusion is then visually thresholded with *OpenCV* to fill the resulting holes. These internal holes are either from gaps between extruded material, or from internal structures of the model, or from support structures within the model introduced by the slicing software. Boundary detection is then performed using *OpenCV* functionality. The boundaries are collected for each layer and exported to a file containing X, Y and Z coordinates in a Cartesian coordinate system. To reconstruct surfaces that are flat in one respective layer correctly, random sampling of points is performed. A percentage (5 %) of the area of the total image area is populated with random points. These points are further for structures and areas between individual layers to avoid random points inside the reconstructed model. *MeshLab* is used to process the resulting XYZ files via batch processing. The following operations are used to produce a mesh representation that is then exported as an STL file:

1. Poisson-disk sampling to reduce the number of points to operate on (Param.: 150 000 samples, Base mesh subsampling enabled, other parameters are at their default value),
2. surface reconstruction by ball pivoting (Param.: Ball radius 15 %, other parameters at their default value),
3. hole closing (Param.: Max size to be closed 40, other parameters are at their default value), and
4. removal of zero area, unconnected, null, and duplicate vertices and faces.

In Figure 2, the extrusion structure is depicted. The model is bounded by an outer shell that is indicated by a thick black line. This line also indicates where in reality material is extruded. The blue criss-cross pattern inside the object is for stability, as the object is otherwise printed hollow. The material extrusion of the outer shell and the inner infill-pattern is hard to distinguish as the GCode does not differentiate between them, and the amount of material dispensed is identical per the same length. In this figure, the object contains a hole in this layer which can be going through the complete object or be just part of this specific layer. The differentiation between designed holes, as in this case, and holes or other structures created by the slicing software for stability purposes is indistinguishable, as described earlier. The current implementation does, therefore, not support internal structures, holes or other concave structures. The figure also shows support structure extrusion indicated by red coloured lines, which are also indistinguishable in the GCode due to the reasons discussed above.

4.2. Reconstruction Algorithm

The simulation adds slight random errors in the positioning of the coordinates to simulate machine imprecision due to underlying hardware.

The following algorithmic description details the implementation of the reconstruction method.

1. Simulate printhead movement and material extrusion layer-wise, with random errors introduced to simulate machine imprecision.
2. From each simulated layer, extract geometrical information by:
   (a) Threshold the graphical representation to identify extruded material against the background;
   (b) Perform hole closing to ignore infill strategy;
   (c) Find contours of the resulting graphical representation;
   (d) Identify suitable contours according to their hierarchy and size;
3. Export of X, Y, Z information to a separate file
4. Perform mesh reconstruction in *MeshLab* via batch processing
5. Compare distances from created mesh to the STL representation to create a metric for identity, see [13, 16, 17] for information on similarity metrics.

In Figure 3, the intermediary reconstruction phases are depicted on a model acquired from *thingiverse*.

The leftmost image (a) depicts the simulated extrusion path, based on the GCode. Subfigure (b) is the thresholded representation of the extrusion path with holes filled. Based on Figure 3(b), the boundaries are extracted using the *OpenCV* findContours function, see Figure 3(c). Part (d) of the

Figure shows the extrusion process by placing coloured disks at the individual target points and smaller coloured disks on the respective movement in between the GCode instructions.

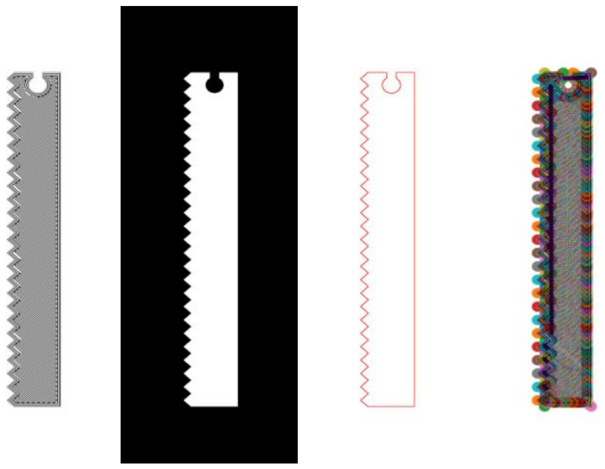
Figure 3. Reconstruction Stages for Model "weigh spoolholder" by Nefi Velarde, https://www.thingiverse.com/thing:1000042; (a) Simulated Extrusion (b) Thresholded Representation (c) Boundary Detection (d) Simulated Movement

5. Webservice Implementation

The implementation of the reconstruction method is performed in *Python* utilising the *web.py* framework. With RESTful Web services, every object is considered a digital resource that is accessible through a URL (Uniform Resource Locator). Based on the implemented and exposed methods a number of operations is facilitated. The resources in this case are the original GCode file, the resulting STL model and digital renderings of the model. User management and security are not considered at this point. User management and security are highly dependent on the infrastructure and environment where such an implementation is deployed and requires adaption. The operations or methods described in the following are invoked by a user, either programmatically or using a Web browser. For example, the following URL is exposed by the service:

https://www.example.com/ModelRecSrv/GCode/10442/convertToSTL

In this example, *https://www.example.com* denotes the transport protocol and base host name for the service. Furthermore, *ModelRecSrv* denotes the name of the service. The following part, *GCode*, denotes the realm of the objects to operate on, here it is GCode. Furthering this example, calling *ModelRecSrv/Rendering/10442* results in the identical response, a digital rendering of the model that is stored under the identifier 10442 in the PNG format, as a call to *ModelRecSrv/GCode/10442/renderModel*. The identifier part is unique per uploaded GCode. In future, more security-aware implementations are required. In the example, *convertToSTL* is the name of the operation that is called. **POST** and **GET** are different ways for a user to call the Web service. The service implements the following **POST** operations:

– **uploadGCode**; A method that expects the GCode file as payload using multipart/form-data encoding and stores the file on the server. A unique identifier is responded to the uploader.

– **convertGCodeToSTL**; A method that expects a GCode file as payload using multipart/form-data encoding and returns the reconstructed STL model to the uploader.

The service also implements the following **GET** operations:

– **convertToSTL**; A method that is expected to be called on a file object identified by the unique identifier to initiate the conversion to a STL file. If the file has previously been converted, the reconstructed model is sent back to the caller from local storage or cache. If the file is not converted yet, the conversion is performed and the result is send back to the caller when available.

– **GCodeStatistics**; A method that is expected to be called on a file object identified by the unique identifier, for detailed information on the original GCode file. This includes information on the number of instructions, file size, and extrusion length per extruder

– **renderModel**; A method that is also expected to be called on a file object identified by the unique identifier that returns the rendering of the resulting STL model as a PNG (Portable Network

Graphics) file. This method takes parameters to adjust the canvas size of the rendering and the rotation of model along its three axes. Furthermore, methods to delete models, files, and renderings from the server are implemented. The uploaded and rendered files will remain on the server indefinitely when not manually removed. The local caching and storage is implemented using a *SQLite* database.

6. Discussion

To discuss this approach, a number of problems encountered in the creation of this method are presented:

– Support material: Support material is indistinguishable in GCode, especially in case of one extruder. With two extruders, it could be derived from the amount of material extruded and then ignored. With a single extruder, heuristics are possible that would work on structural information, e.g., thin, column-like structures that are connected to the base plate and the main object with little surface area.

– Additional material: Some slicing programs instruct to extrude material in a specific pattern to clean the nozzle before the actual 3D print. With multi nozzle printing, it is also common to have “wiping” structures for nozzle cleaning. These structures are indistinguishable in the GCode.

– Simulation of mechanical imprecision: The actual fabrication of objects is limited by the mechanical precision of the AM machine and material behaviour, such as semi-molten flow.

– Adaptive and uneven layer height: Currently, the algorithm expects uniform layer heights for the mesh reconstruction.

– Internal structures: These structures are currently unsupported as the reconstruction is based on boundary reconstruction. Internal structures, such as holes, are hard to distinguish from artificial internal structures constructed by the slicing software, e.g., internal support structures.

– Objects printed larger than supported by the build envelope: The algorithm works by simulating the extrusion on a virtual build plate that is mapped to a digital canvas of predefined dimensions. Reconstructing such large models fails in the current stage. Actual 3D printing of such models would also fail.

– Flat surfaces: The current implementation fails in some cases to correctly close the models with flat surfaces. In future iterations, detection of flat bottom and top structures is improved by the introduction of randomly sampled points on these surfaces.

The rationale to create an intermediary pointcloud instead of layer-wise meshing is as follows:

– With layer-wise meshing, (internal) supporting structures will be meshed too. This will lead to a flawed reconstruction as most objects, in FDM 3D printing, are printed partially hollow, with specific support or infill structure provided by the slicing software, e.g., honey-comb structure or criss-cross pattern, see Figure 1.

– The meshing of the object’s surface structure is made complicate with layer-wise meshing as per-layer meshing is discontinous.

– With layer-wise meshing, most of the generated mesh structure must be removed in the aggregation step as it would describe internal surface structures that are not surfaces in reality, in the model or in reconstruction resulting in illegal STL structures.

7. Conclusion & Outlook

In this paper, the authors demonstrated a method to reconstruct the digital model from a GCode file which is implemented as a RESTful Web service. This reconstruction is based on the virtual simulation of the extrusion process. The reconstruction is intended for FDM type 3D printing and GCode generated for a Cartesian geometry. The underlying simulator is capable of interpreting *Sailfish* targeted machine code. Mechanical accuracy of the fabrication is simulated in the reconstruction via random inaccuracies. The reconstruction uses existing open source software and libraries, such as *OpenCV* and *MeshLab* to reconstruct the model based on the simulated boundaries. The method is unable to reconstruct internal structures. Improvements of the method and its implementation are discussed in the previous Section. The future implementation will support the detection of internal structures, detection of support material and supplemental material extrusions. This work is intended to be part of a complete model reconstruction framework that will help to create a chain of trust in AM. This will allow to answer the question whether a specific machine code represents an original 3D model or not. This reconstruction is not only reasonable for reverse engineering purposes but also within the context of the proposed chain of trust. Model reconstruction is important to certify that a machine code instruction file corresponds to its respective digital model.

References

[1] ISO 6983-1:2009 2009 Automation systems and integration - Numerical control of machines - Program format and definitions of address words

[2] Baumann F W, Bugdayci H, Grunert J, Keller F and Roller D 2016 Influence of slicing tools on quality of 3D printed parts *Computer-Aided Design and Applications* **13** 1 pp 14–31

[3] Tsoutsos N G, Gamil H and Maniatakos M 2017 Secure 3D printing: reconstructing and validating solid geometries using toolpath reverse engineering *Proc. of the 3rd ACM Workshop on Cyber-Physical System Security (CPSS '17)* pp 15–20

[4] Baumann F W, Schuermann M, Odefey U and Pfeil M 2017 Model reconstruction from machine code instructions *Proc. of the 6th International Conference on Mechanical Engineering & Mechanics: Industrial Engineering, Mechatronics and Industry Future (ICMEM)* In Preparation.

[5] Chakraborty D, Reddy B A and Choudhury A R 2008 Extruder path generation for curved layer fused deposition modeling *Computer-Aided Design* **40** 2 pp 235–43

[6] Wong K V and Hernandez A 2012 A review of additive manufacturing *ISRN Mechanical Engineering* **2012**

[7] Klein J, Stern M, Franchin G, Kayser M, Inamura C, Dave S, Weaver J C, Houk P, Colombo P, YangM Oxman N 2015 Additive manufacturing of optically transparent glass *3D Printing and Additive Manufacturing* **2** 3 pp 92–105

[8] Baumann F W and Roller D 2016 3D printing process pipeline on the internet *Proc. of the 8th Central European Workshop on Services and their Composition (ZEUS 2016)*, pp 29–36

[9] Gibson I, Rosen D and Stucker B 2015 Additive Manufacturing Technologies - 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing. Springer New York, 2 edition

[10] Zeltmann S E, Gupta N, Tsoutsos N G, Maniatakos M, Rajendran J and Karri R 2016 Manufacturing and security challenges in 3D printing *JOM* **68** 7 pp 1872–81

[11] Gao J, Chen X, Yilmaz O and Gindy N 2008 An integrated adaptive repair solution for complex aerospace components through geometry reconstruction *The International Journal of Advanced Manufacturing Technology* **36** 11 pp 1170–79

[12] Thrimurthulu K, Pandey P M and Reddy N V 2004 Optimum part deposition orientation in fused deposition modeling *International Journal of Machine Tools and Manufacture* **44** 6 pp 585–94

[13] Okarma K and Fastowicz J 2017 Quality assessment of 3D prints based on feature similarity metrics *Proc. of the International Conference on Image Processing and Communications* pp 104–11

[14] Zhang Y and Chou K 2008 A parametric study of part distortions in fused deposition modelling using three-dimensional finite element analysis *Proc. of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **222** 8 pp 959–68

[15] Shrivastava R and Modi Y K 2015 Reverse engineering approach for rapid manufacturing of freeform components *Proc. of the 2nd International Conference on Science, Technology and Management* pp 2086–95

[16] Adan A and Adan M 2004 A flexible similarity measure for 3d shapes recognition *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** 11 pp 1507–20

[17] Shum H-Y, Hebert M and Ikeuchi K 1996 On 3d shape similarity *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '96)* pp 526–31