A GRAPH GRAMMAR BASED APPROACH TO 3D PRINT AND ASSEMBLE FURNITURE

Sulabh Gupta and Rahul Rai

Design Analytics Research and Technology (DART) Lab,
Department of Mechanical and Aerospace Engineering,
University at Buffalo-SUNY, Buffalo-NY, 14260

Abstract

A Plethora of user generated 3D models are available online. With rapid proliferation and diffusion of additive manufacturing machines in households, it has now become possible to download these virtual objects and print them out as physical parts. Although printing small size parts (within print volume of low cost 3D printers) is relatively an easy task, additive fabrication of large size parts (part volumes greater than print volume of low cost 3D printer) remains a challenging task for novice 3D printer users. In this paper the authors present a computational pipeline to 3D print large size 3D models that can be easily downloaded from online websites. The pipeline essentially enables decomposition of large objects into smaller parts that can be 3D printed and then assembled. To assemble the printed parts a three-pronged approach is outlined. First, an interface based on graph grammar rules has been developed to generate assembly instructions. Second, an interactive segmentation of the desired 3D model is carried out using a Segmentation Guide Interface (SGI). SGI has been developed to assist a user to carry out component to sub-component segmentation. Third, we have also developed an interface that aids a user in printing small size pieces that can be printed in print volume of a commercial 3D printer (such as Makerbot®) and then assembled to create components that are too large to be printed in print volumes of low cost 3D printers. We demonstrate the efficacy of developed pipeline by creating assembly instructions for multiple large sized 3D table models available online.

Introduction

The rapid development in 3D printing field has resulted in the beginning of a new era in personal fabrication. 3D printers are easy to use and are capable of printing almost any 3D CAD model. The widespread availability of 3D CAD models online has further stimulated personal fabrication. For example, online repositories have far more varieties of virtual furniture models than real ones in your nearby furniture store.

Our goal is to enable individual users to 3D print their favorite 3D models by leveraging online shape repositories. In personal fabrication domain 3D printing until now has mostly been used for printing objects for trivial use or showpieces. In order to achieve large scale acceptance 3D printers should break free from their stereotypical personal fabrication use and print objects that have more practical utility. It has been observed that the maximum size of an object that a domestically affordable 3D printer can fabricate in one pass (the printing volume) is limited by practical considerations. This problem has curbed the large scale adoption of low cost 3D printers. Larger objects must therefore be printed as multiple separate parts and assembled.

In this paper, we outline a computational pipeline to 3D print large size 3D models available online. Using the computational pipeline a user can semantically segment the 3D model into its component parts. Once the segmentation is completed the user is provided guidance in slicing the individual components into primitive shapes (sub components) which can be printed in the limited build volume of commercially available 3D printers. The pipeline provides basic information to a user such as the number of sub-components required for each component, the shape of each sub-component and how to assemble the sub-components once they are printed. Additionally, we have developed a graph grammar based technique to generate assembly instructions. The generated assembly instruction joins components using off-the-shelf connectors found in furniture assembly domain.

Our key contributions are:

(1) We outline a computational pipeline that guides a user to 3D print large parts in commercially available 3D printers by segmenting them into smaller and printable sizes.

(2) We have developed a set of new graph grammar rules that can be used for generating assembly instructions to assemble furniture like tables.

We demonstrate the validity of our computational pipeline by generating assembly instruction of 3D models of IKEA-style tables downloaded from Google 3DWarehouse, Princeton Shape Benchmark, and Polantis. All our test models have arbitrary topology and structure. None of the 3D table models have the geometry or connector information needed for fabricating them.

Related Work

In the present section, three key areas related to the presented research namely (1) 3D printing (2) Graph Grammar, and (3) Mesh segmentation are briefly reviewed.

3D Printing

With the advancement of 3D printing we are witnessing the first stages of manufacturing democratization. It is believed that individual users will play a role in designing and creating their own products in the future [Gross 2007, Landay 2009]. Users can readily build their own customized products like plush toys [Mori, et.al 2007], chairs [Luo, et.al 2012], furniture [Lau, et al. 2011], garments [Umetani et al. 2011], Burr puzzles [Xin et al. 2011], and planar sections [McCrae et al. 2011, Hildebrand et al. 2012]. The current trend portends to revolutionize design and fabrication and could possibly give rise to a new class of creators and products [Mota, 2011]. We aim to propel this trend further. While the professional and higher end 3D machines are capable of producing large complex objects, the personal and less costly 3D printers are constrained by their print size and cost. The limited print volume of affordable 3D printers is the main cause of their limited usage. Efforts are been made to overcome these barriers.

Luo et.al, [2012] has developed a new method called "chopper". Chopper can be used to chop a given large objects into smaller pieces. These smaller pieces can be printed in 3D printers and assembled to create a large object. This method automatically generates male and female connectors on the cut surfaces. Once the object is printed, the user can assemble the parts without any assembly instructions. This method focuses on printing showpieces and the connectors generated do not provide much structural rigidity to the printed parts. In the computational pipeline outlined in this paper, a set of assembly instructions are generated to guide the user in assembling the 3D printed parts. The connectors used to assemble the parts are off-the-shelf connectors and the placement of these connectors provides structural rigidity to the final assembled part.

Lau et.al, [2011] developed a graph grammar technique to create assembly instructions for virtual furniture models. In their work, the end user has the ability to use the assembly instructions to build furniture models with standard tools and wooden materials. In the presented work, we guide the user to segment the 3D model into small parts. These segmented parts can be 3D printed in the limited build volume of low cost 3D printers. We then use graph grammar rules to generate assembly instructions for the printed models. The user can have little to no experience of working with tools.

Graph Grammar

Graph grammars have found applications in various fields such as concurrent systems, engineering, programming languages, and biology. It has been used to develop methods for understanding and modelling new architecture [Stiny 1980], cities [Parish, et.al 2001], buildings [Muller, et.al 2006] and details of facades [Wonka, et.al 2003]. Coffee maker grammar was one of first examples of using grammars for product design [Agarwal, et.al 1998]. Their grammar described a language that generates a large class of coffee makers. Shea et.al, [1997] presented a parametric shape grammar for the design of truss structures that uses recursive annealing techniques for topology optimization. Other engineering applications include lathe grammar [Brown, et.al 1997], grammar for machine design [Schmidt, et.al 1995], grammars for mechanical clocks [Starling, et.al 2003] and gear trains in [Starling, et.al 2005]. One of the interesting implementations of graph grammars is in the area of function-based design [Sridharan, et.al 2004]. In more recent applications graph grammar has been used to develop new design concepts to represent different topologies, configurations or shapes within a single search space [Rai, et.al 2011]. Lau et.al, [2011] have used graph grammar techniques to generate assembly instructions for shelves, cabinets, and tables to build with standard wooden materials. We have developed graph grammar rules for generating assembly instructions for printing large size 3D printed objects. The novelty of our work is in the fact that a novice user can use our pipeline to easily assemble the 3D printed parts.

Mesh Segmentation

In our work we segment the 3d model manually into its component parts. This is enabled through mesh segmentation techniques. Mesh segmentation techniques can be based on curvature and geodesic distances [Mangan, et.al 1999], dihedral angles [Shlafman, et.al 2002], Planarity and normal direction [Coheh-Stiner, et.al 2004, Attene, et al 2006], and Slippage [Gelfand, et.al 2004]. These attributes are sensitive to local surface features and to pose changes. Therefore, these techniques are not suitable for segmenting the same object in different poses. Topology-based approaches like spectral analysis [Liu, et.al 2004], average geodesic distance (AGD) [Hilaga, et.al 2001] and Reeb-graphs [Attene, et.al 2003] can be used for mesh segmentation of object in different poses. Nevertheless, they are vulnerable to topological and connectivity changes and do not distinguish well between shape differences.

In our work we use Shape Diameter Function (SDF). The Shape-Diameter Function (SDF) is a scalar function defined on the mesh surface. It expresses a measure of the diameter of the object's volume in the neighborhood of each point on the surface. Using the SDF one can easily segment 3D models and perform part-retrieval [Shapira, et.al 2008].

Basic Components of Computational Pipeline

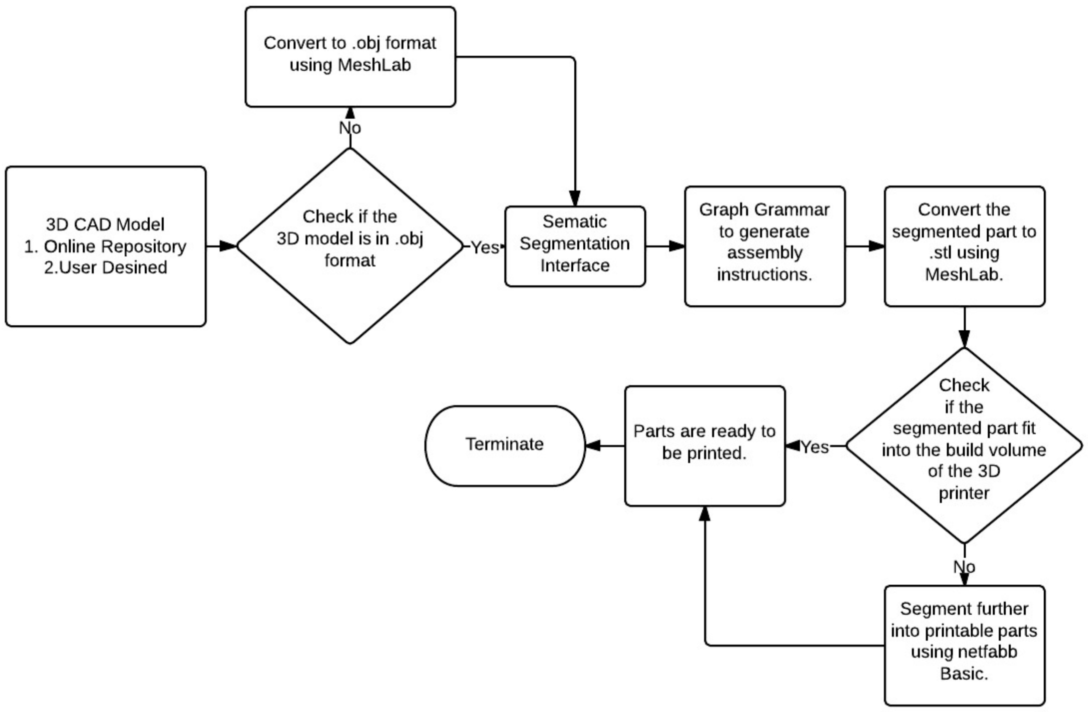

Convert to .obj format using MeshLab
No
3D CAD Model
1. Online Repository
2.User Desined
Check if the 3D model is in .obj format
Yes
Sematic Segmentation Interface
Graph Grammar to generate assembly instructions.
Convert the segmented part to .stl using MeshLab.
Check if the segmented part fit into the build volume of the 3D printer
Yes
Parts are ready to be printed.
Terminate
No
Segment further into printable parts using netfabb Basic.

Figure 1: Flowchart showing different steps of computational pipeline for printing and assembling parts bigger than the print volume of 3D printers.

(Figure 1) illustrates the overall framework of the developed pipeline. The pipeline can accept 3D CAD models as input in any one of the following two ways: (1) A 3D CAD model created using CAD software like SolidWorks, CATIA, or PRO-E, or (2) A 3D CAD model obtained from online repositories such as Google Warehouse, Princeton Shape Benchmark, and Polantis.

The Semantic Segmentation Interface (SSI) is used to segment the inputted 3D CAD models into component parts. Each segmented part is saved individually. The SSI is based on SDF [Shapira, et.al 2008]. In order to carry out Semantic Segmentation using SDF vertex and face information is required. As a result it has to be ensured that the CAD model is in .obj format. If the input 3D model is not in .obj format then MeshLab software can be used to convert the 3D CAD file into the required (.obj) format.

GraphSynth (a graph grammar software) is used to create assembly instructions to assemble the segmented parts. For assembly instruction generation we have developed twenty-four rule graph grammar that defines different ways to assemble tables (Table 1). By simply executing different combinations of grammar rules, assembly instructions for a variety of tables can be generated. A user skilled in wood working can use the instructions generated by GraphSynth to make the object using wood and simple fasteners.

In our proposed approach it has been assumed that a user has little or no experience with wood working tools. As a result our computational pipeline enables a not-so-skilled user to 3D print the parts. 3D printers require the input files to be in .stl format as a result the segmented components that are in .obj format have to be converted to .stl format. .obj to .stl format conversion can also be carried out using MeshLab. In case segmented 3D components do not fit in the build volume of the 3D printer they are further segmented. We further segment the components into sub-components using slicing software called netfabb Basic.

The printed sub-components are assembled to create components. The components are then assembled to create the overall object by using the assembly instructions generated by graph grammar interface. The tool components and the associated methods are described in detail in the following sub-sections.

Semantic Segmentation Interface (SSI)

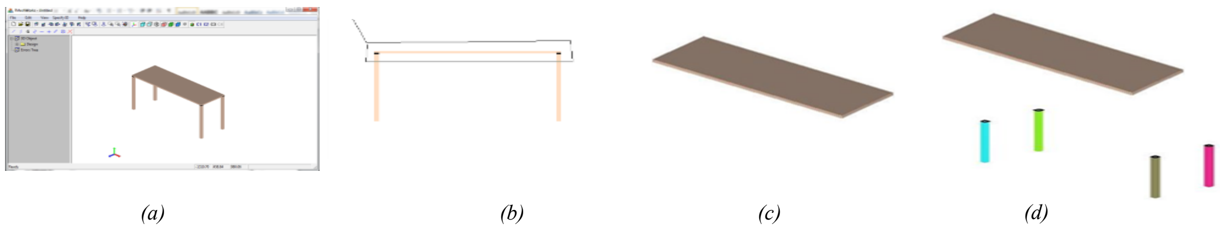


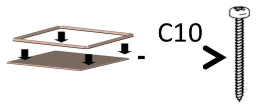
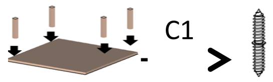
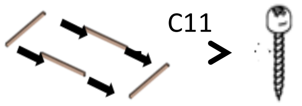
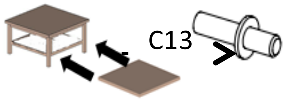
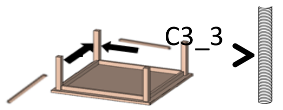
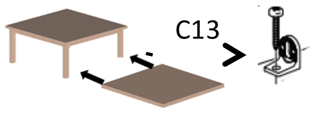
Figure 2: (a) Semantic Segmentation Interface, (b) Semantic segmentation carried out by drawing a boundary around the part to be segmented, (c) Segmented table top, (d) The segmented table into its constituent parts

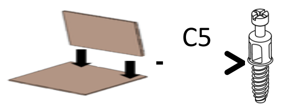
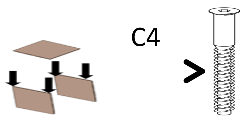
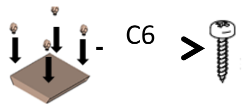
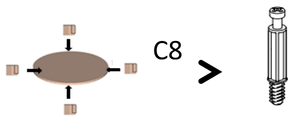
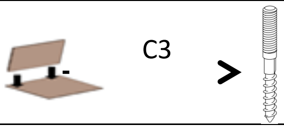
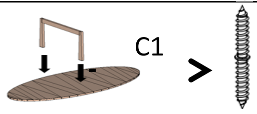
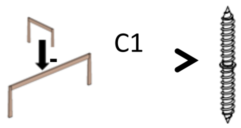
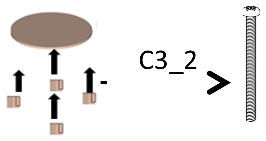
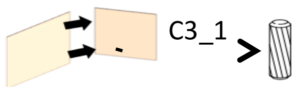
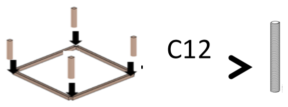
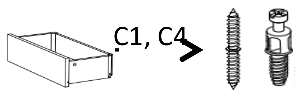
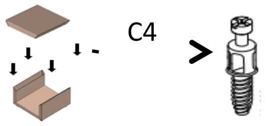
There is lack of assembly information in the 3D models available online and most of the 3D models are stored as a single part. The Semantic Segmentation Interface (SSI) is used to semantically segment 3D CAD models into component parts (Figure 2a). SSI is based on a volume-based shape-function called the shape-diameter function (SDF). The SDF is a scalar function defined on the mesh surface. In essence it represents a measure of the diameter of the object's volume in the neighborhood of each point on the surface [Shapira et al 2008]. In order to carry out Semantic Segmentation using SDF, vertex and face information is required. As a result it has to be ensured that the input 3D CAD model is in .obj format. If the inputted 3D CAD model is not in .obj format then MeshLab can be used to convert it into .obj format. User through interaction with SSI segments the 3D CAD model by drawing a box around the part (Figure 2b). The user can also select different colors to color segmented parts in different colors. This helps in differentiating different segmented parts. Each part can then be exported separately.

Graph Grammar for generating assembly instructions

Graph grammars are comprised of rules for manipulating nodes and arcs within a graph. In our computational pipeline the graph grammar rules specify a formal language for generating assembly instructions from an initial user defined graph (seed graph). The development of these rules encapsulates a set of valid operations that can be used to assemble different parts of a table. Through the application of each grammar rule the current state of graph is transformed into a new state, incrementally evolving towards a desired solution.

A typical graph grammar rule is comprised of a left-hand side (LHS) and a right-hand side (RHS). The LHS contains the conditions, upon which the applicability of a rule is determined. Accordingly, the LHS describes the state of the graph for a particular rule to be applicable. The RHS, on the other hand, contains the resulting graph transformation. It describes the new state of the graph after the application of the rule. We have developed a twenty four-rule graph grammar that defines ways to assemble different parts of a table (Table 1). The set of twenty four-graph grammar rules were created by studying actual assembly instructions used for assembling different types of tables.

Rule	LHS	RHS	Description
Rule 1	TableTop→TopSupport	TableTop→ C10 → TopSupport	
Rule 2	TableTop→Legs	TableTop→ C1 → Legs	
Rule 3	TopSupport→TopSupport	TopSupport → C11 → TopSupport	
Rule 4	ShelfSupport → Shelf	ShelfSupport →C13 →Shelf	
Rule 5	ShelfSupport → Legs	ShelfSupport → C3_3 → Legs	
Rule 6	Legs → Shelf	Legs → C2 →Shelf	

Rule 7	TableTop →VerticalFront	TableTop →C5 →VerticalFront	
Rule 8	VerticalFront →Bottom	VerticalFront →C4 →Bottom	
Rule 9	Bottom →Wheel	Bottom →C6 →Wheel	
Rule10	SideSupport_round →Shelf_Round	SideSupport_round →C8 →Shelf_Round	
Rule 11	TableTop →VerticalSide	TableTop → C3 →VerticalSide	
Rule 12	Legs →M	Legs → C1 →M	
Rule 13	M → M	M →C1 → M	
Rule 14	TableTop_round → SideSupport_round	TableTop_round →C3_2 →SideSupport_round	
Rule 15	TableTop →VerticalSide_Horizontal	TableTop →C3_1 → VerticalSide_Horizontal	
Rule 16	TopSupport→Legs	TopSupport → C12→ Legs	
Rule 17	Drawer	Side→C4→Back→C4→ Side→C1→Front→C1→Side	
Rule 18	VerticalSide → Bottom	VerticalSide →C4 →Bottom	

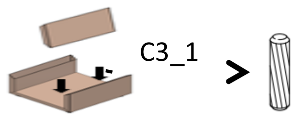
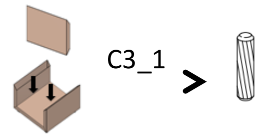
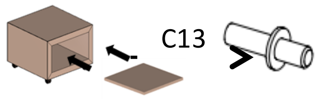
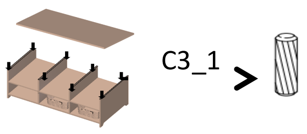
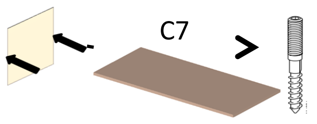
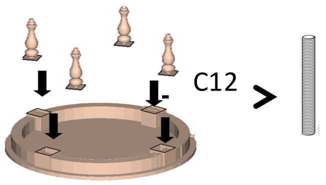
Rule 19	TableTop →VerticalPartition	TableTop →C3_1 → VerticalPartition	
Rule 20	TableTop →HorizontalPartition	TableTop→ C3_1→HorizontalPartition	
Rule 21	VerticalSide_Horizontal→Shelf	VerticalSide_Horizontal → C13 →Shelf	
Rule 22	Bottom →VerticalSide_Horizontal,VerticalPartition, HorizontalPartition,VerticalSide_Horizontal	Bottom →C3_1→VerticalSide_Horizontal, VerticalPartition, HorizontalPartition, VerticalSide_Horizontal	
Rule 23	VerticalSide → Shelf	VerticalSide →C7 →Shelf	
Rule 24	SideSupport_round →Legs	SideSupport_round →C12 →Legs	

Table 1: Rules to generate assembly instructions for tables

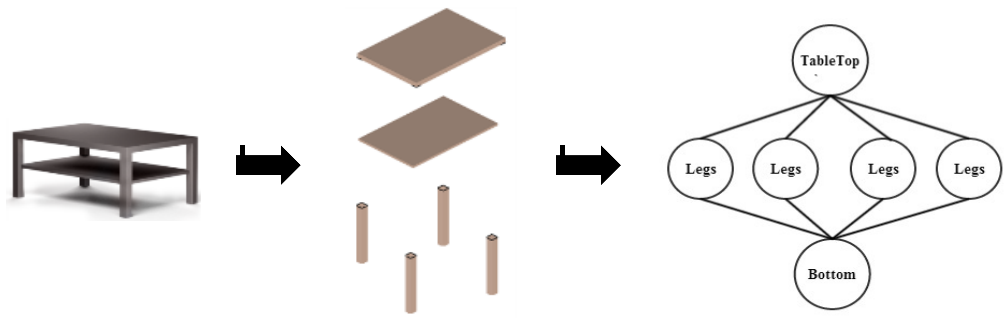


Figure 3: User defined initial contact point ***of*** *the segmented table*

By simply executing different combinations of graph grammar rules, assembly instructions for variety of tables can be generated. The generation of assembly instruction starts with the user defining the initial contact points of the table. This is a simple step and the user inputs contact point information in the form of a seed graph (Figure 3). Once the seed graph is defined, in the following steps different graph grammar rules are used to automatically recognize different connections and appropriate connectors. This continues until all the connectors for the whole assembly are identified.

A complete sequence of application of different assembly instruction graph grammar rules to create assembly instructions for a table shown in (Figure 3) has been illustrated in (Figure 4). In order to generate assembly instructions graph grammar approach starts with a seed graph which is defined by the user. This results in the seed graph to become LHS in step 1. After this stage the process of recognize, choose, and apply is invoked in an iterative manner resulting in a new LHS and RHS at each step. The assembly instruction for the table example shown here can be derived by application of rule sequence {1, 1, 1, 1, 6, 6, 6, 6}. Since tables have repetitive parts like legs we notice that the same rules have been applied more than once. The graph grammar interface can generate assembly instructions for a wide range of tables.

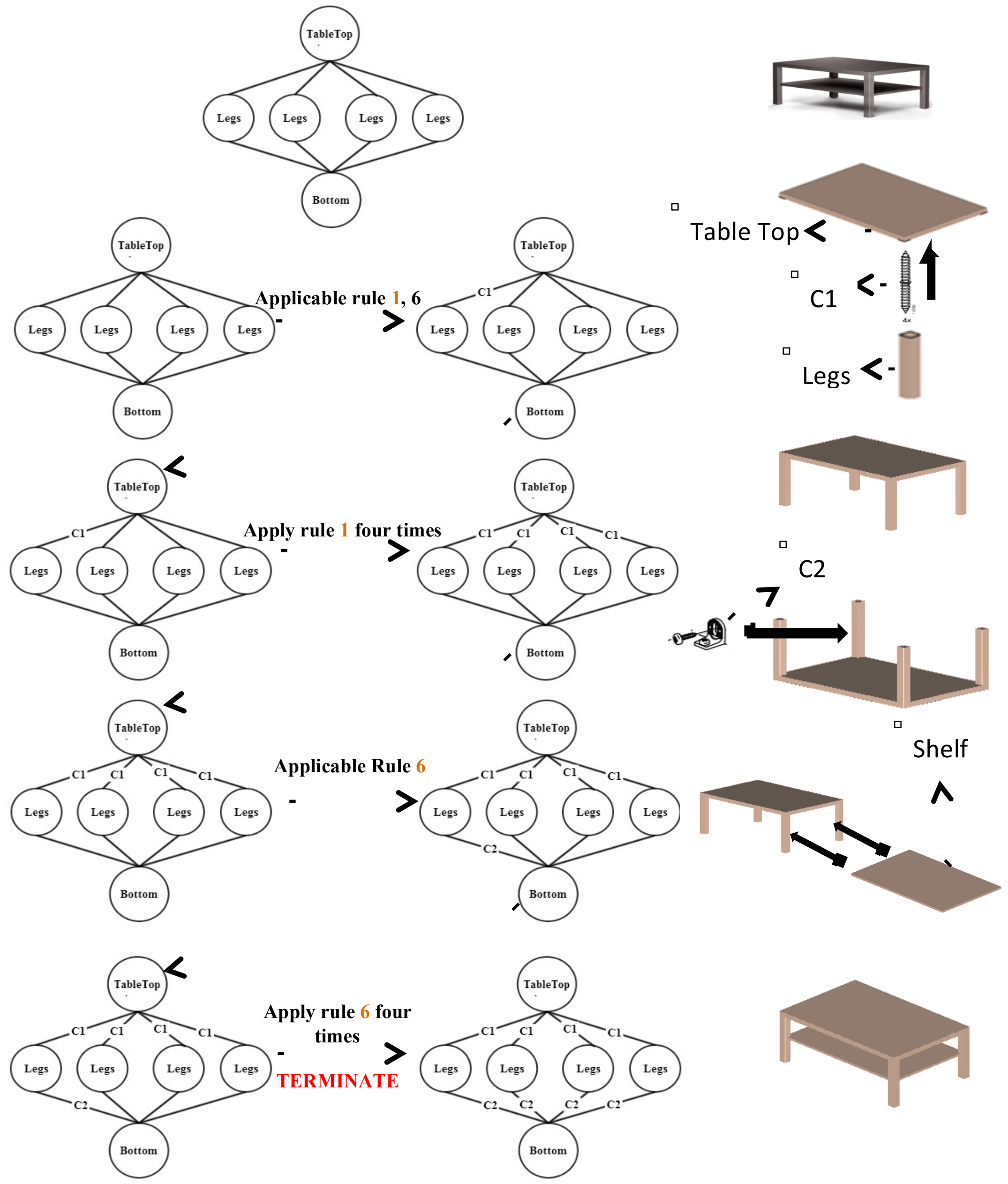


Figure 4: Application of rule sequence {1, 1, 1, 1, 6, 6, 6, 6} creates the assembly instructions for the example table from initial seed graph

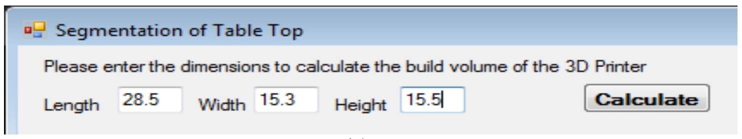
Segmentation Guide Interface (SGI)

A single component can be too big to 3D print at one go. The idea behind the development of SGI is to guide the user to carry out further segmentation of the segmented components. SGI enables the user to compare the build volume of the 3D printer to the volume of the component. If the print size of the component is smaller than the print volume of 3D printer then the part is ready to be printed. Otherwise SGI provides information regarding the shape and number of sub-components that the component should be segmented into. SSI also provides the user with visual information regarding type of connectors and how to use those connectors to join the sub-components.

The Segmentation Guide Interface has been developed in Visual C#. Visual Studio supports Visual C# with a full-featured code editor, compiler, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The .NET Framework class library provides access to many operating system services and other useful, well-designed classes that speed up the development cycle significantly.

A sequence illustrating the interaction with SGI is presented below:

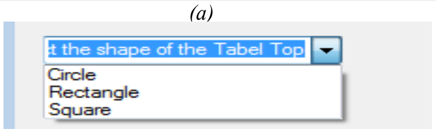
1) Prompt the user to enter the build dimensions of the printer. Maximum printable area and volume is calculated (Figure 5a).
2) The user is then prompted to enter the shape of the table top. A choice of three shapes is displayed (Figure 5b).
3) Once the user selects the shape of the table top. The user is prompted to enter the dimensions of the table top. Using this information area and the volume of the table top are calculated (Figure 5c).
4) A list of possible connector types is displayed (Figure 5d) to join the sub-components. A (.gif file) appears to help the user visualize the selected connector.
5) The user then clicks on the segmentation button. The interface responds by displaying the number and shape of each sub-component that the user needs to generate for creating the overall component (Figure 5e).
6) Similar steps are followed to segment the Legs of the table.


Segmentation of Table Top

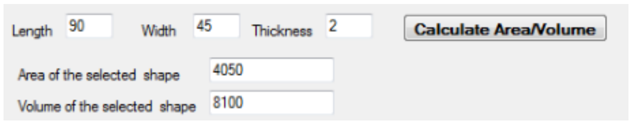
Please enter the dimensions to calculate the build volume of the 3D Printer

Length 28.5 Width 15.3 Height 15.5

(a)



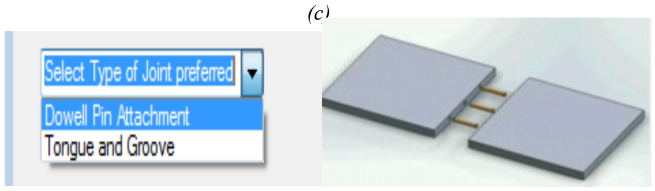
(b)


Length 90 Width 45 Thickness 2

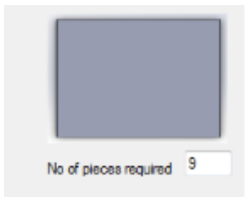
Area of the selected shape 4050

Volume of the selected shape 8100

(c)



(d)


No of pieces required 9

(e)

Figure 5: Segmentation Guide Interface working for segmenting the Table top

The suggestions provided by the segmentation guide interface are passed on to netfabb Basic software. Netfabb Basic is very simple to use software and is often used to segment parts that are bigger than the build volume of 3D printers. netfabb Basic is a freeware for handling .stl files. The freeware includes advanced model browsing, STL fixing, measurement and quality management. The freeware also includes a basic slicing module and assists the user in data preparation and 3D Printing.

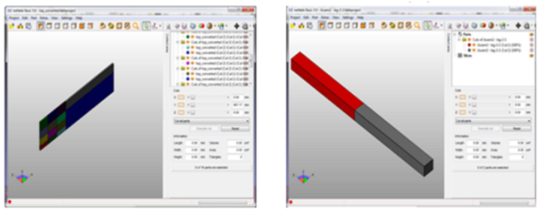


Figure 6: Semantically segmented components being segmented into sub-components using netfabb Basic.

The .stl files of the components are imported into netfabb Basic and cut option is used to slice portions of the model (Figure 6). The number of pieces and the shape to be segmented is based on feedback from SGI. Each single piece is exported separately. User can often save time if the parts repeat themselves; in such cases only one sub-component is exported. These parts are then separately exported to the 3D printer's software to begin printing.

Results

The outlined computational pipeline is tested on a set of five table CAD models. The CAD models for these tables were downloaded from Google 3D Warehouse, Polantis, and Princeton Shape Benchmark websites. An illustrative example shows how the computational pipeline could be used to 3D print and assemble a simple table (Figure 7). The run time of the entire proposed approach depends on the user familiarity with the different software involved. The software involved is very easy to learn and use.

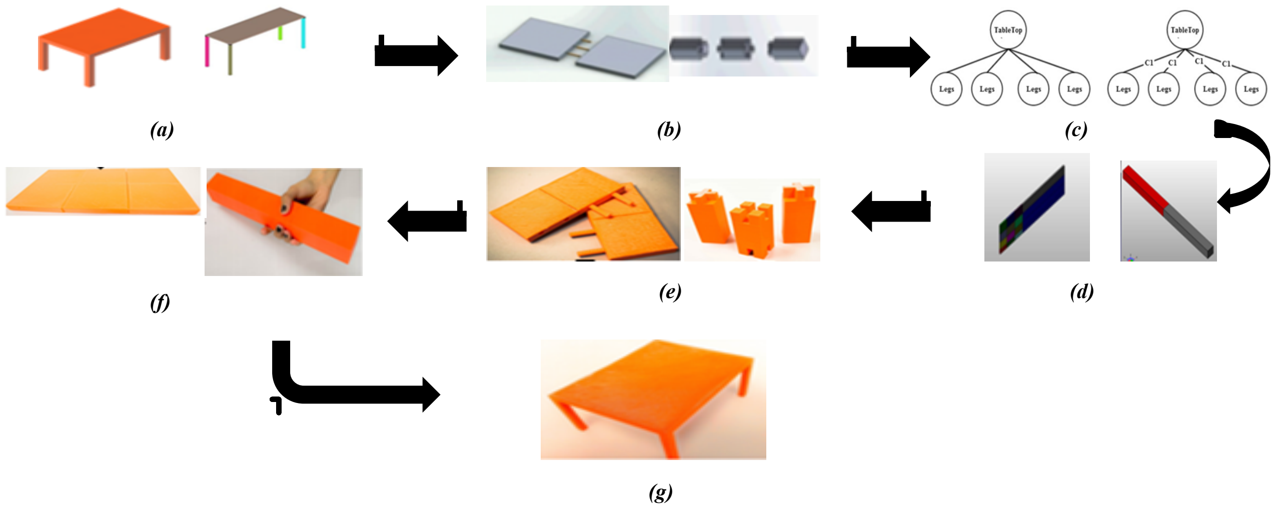
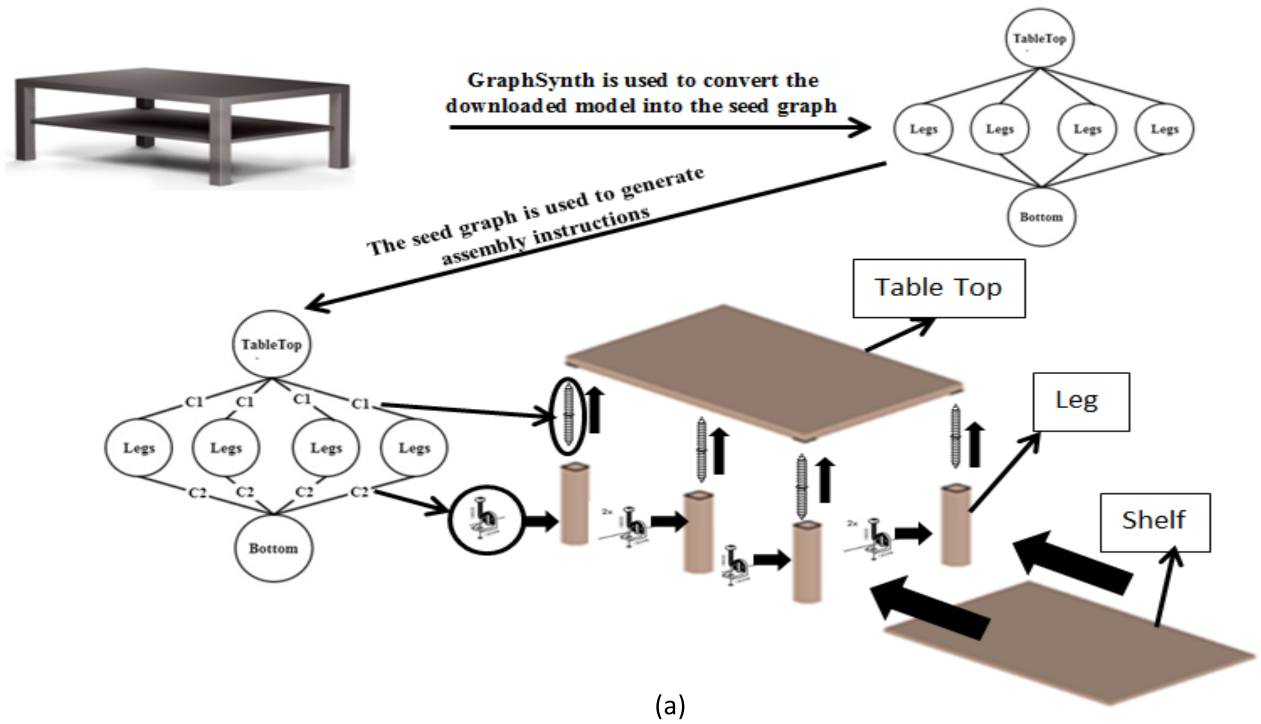
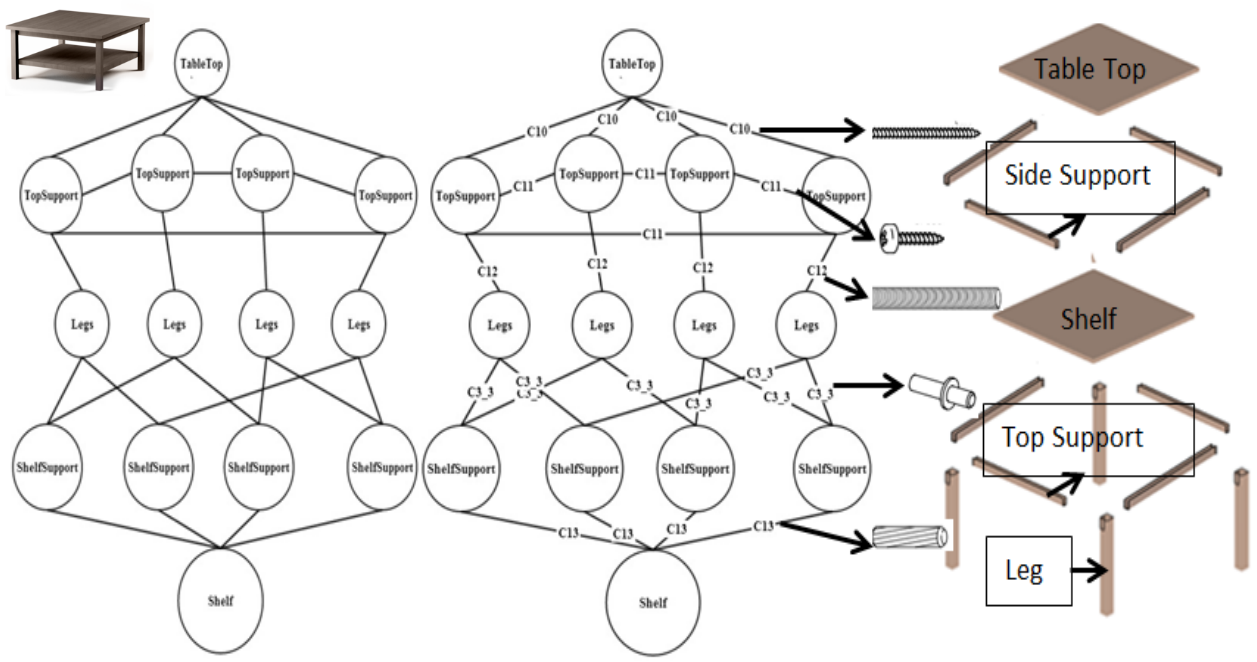


Figure 7: (a) Downloaded model is segmented into its constituent components using SSI, (b) SGI is used to determine the shape and number of sub-components to be created, (c) GraphSynth is used to generate assembly instructions, (d) Segmented sub-components are 3D printed, (e) 3D printed sub-components are assembled, (f) The final table is assembled using assembly instructions generated by GraphSynth. and (g) the final assembled table

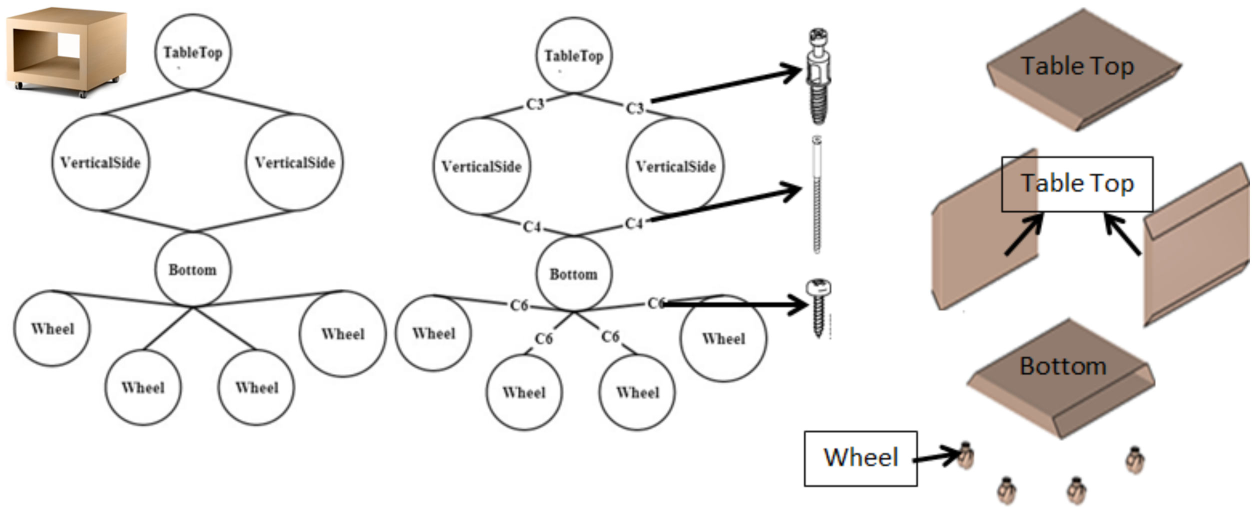
Assembly instructions generated by the piepline for four other tables is shown in Figure 8.


GraphSynth is used to convert the downloaded model into the seed graph
The seed graph is used to generate assembly instructions
TableTop
Legs
Bottom
C1
C2
Table Top
Leg
Shelf
2x

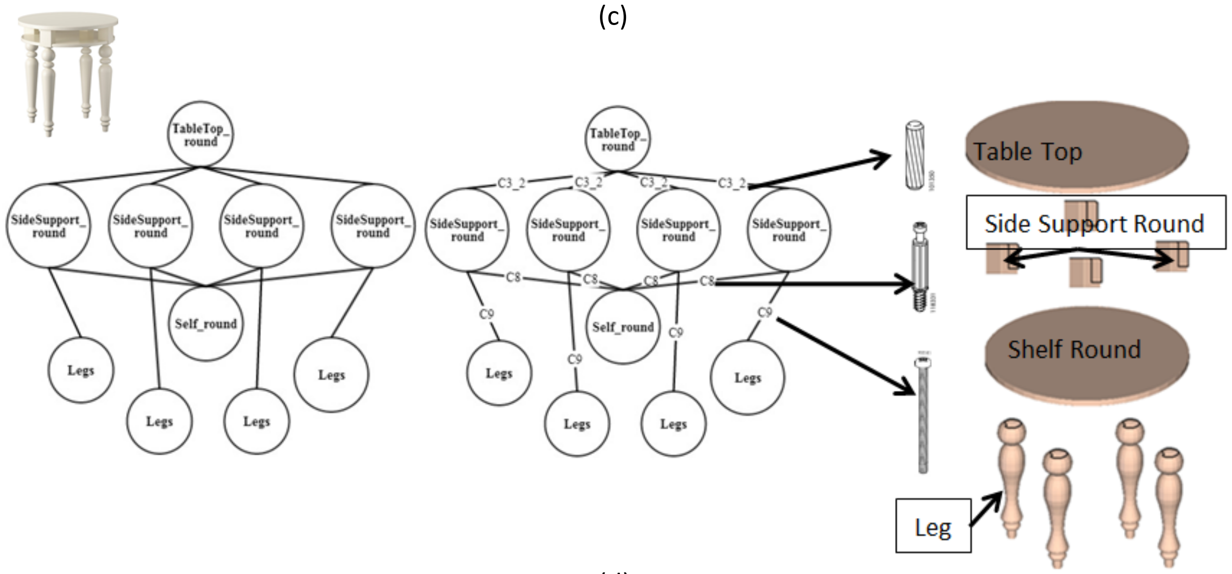
(a)



(b)



(c)



(d)

Figure 8: Assembly Instruction for IKEA style furniture

Conclusion

We have developed a computational pipeline to enable a user to 3D print parts that are larger than the print volume of 3D printers. To demonstrate the utility of the developed pipeline, we 3D print segmented parts of five different tables using MakerBot® Replicator™ 2X and assemble these parts using the assembly instructions generated by the computational pipeline. Some printed parts had warping problems resulting in significant distortion of the printed parts. Sometimes post processing of parts like enlarging hole size and hammering certain joints is required. The finish quality of the parts is dependent on the size and alignment of the print. For example, the segmented legs when printed in upright position gave better fit compared to horizontal prints. We believe that with further advancement in 3D printing processes these issues can be easily addressed.

To make our framework more general, one can extend the presented pipeline to work with any arbitrary 3D model. This could be achieved by extending the initial set of 24 rules presented in this paper. The graph grammar rules defined in the pipeline have been derived manually. Automatic graph grammar rule generation for an arbitrary 3D model is an avenue for future research. The interfaces used in our approach require user interaction to carry out manual segmentation. It would be beneficial to enable automatic segmentation of the components and the sub-components.

References

[1] Gross, M. "Now more than ever: computational thinking and a science of design." *Japan Society for the Science of Design* 16, no. 2 (2007): 50-54.

[2] Landay, James A. "Technical perspective Design tools for the rest of us."*Communications of the ACM* 52, no. 12 (2009): 80-80.

[3] Mori, Yuki, and Takeo Igarashi. "Plushie: an interactive design system for plush toys." In *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 45. ACM, 2007.

[4] Luo, Linjie, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. "Chopper: partitioning models into 3D-printable parts." *ACM Trans. Graph.* 31, no. 6 (2012): 129.

[5] Lau, Manfred, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. "Converting 3D furniture models to fabricatable parts and connectors." In *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 85. ACM, 2011.

[6] Umetani, Nobuyuki, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. "Sensitive couture for interactive garment modeling and editing." *ACM Trans. Graph.* 30, no. 4 (2011): 90.

[7] Xin, Shiqing, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. "Making burr puzzles from 3D models." In *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 97. ACM, 2011.

[8] McCrae, James, Karan Singh, and Niloy J. Mitra. "Slices: a shape-proxy based on planar sections." *ACM Trans. Graph.* 30, no. 6 (2011): 168.

[9] Hildebrand, Kristian, Bernd Bickel, and Marc Alexa. "crdbrd: Shape fabrication by sliding planar slices." In *Computer Graphics Forum*, vol. 31, no. 2pt3, pp. 583-592. Blackwell Publishing Ltd, 2012.

[10] Mota, Catarina. "The rise of personal fabrication." In *Proceedings of the 8th ACM conference on Creativity and cognition*, pp. 279-288. ACM, 2011.

[11] Stiny, George. "Introduction to shape and shape grammars." *Environment and planning B* 7, no. 3 (1980): 343-351.

[12] Parish, Yoav IH, and Pascal Müller. "Procedural modeling of cities." In*Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 301-308. ACM, 2001.

[13] Müller, Pascal, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. *Procedural modeling of buildings*. Vol. 25, no. 3. ACM, 2006.

[14] Wonka, Peter, Michael Wimmer, François Sillion, and William Ribarsky.*Instant architecture*. Vol. 22, no. 3. ACM, 2003.

[15] Agarwal, Manish, and Jonathan Cagan. "A blend of different tastes: The language of coffee makers." (1996).

[16] Shea, Kristina, Jonathan Cagan, and Steven J. Fenves. "A shape annealing approach to optimal truss design with dynamic grouping of members." *Journal of Mechanical Design* 119, no. 3 (1997): 388-394.

[17] Brown, K. N., and Jonathan Cagan. "Optimized process planning by generative simulated annealing." *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* 11, no. 03 (1997): 219-235.

[18] Schmidt, Linda C., and Jonathan Cagan. "Recursive annealing: a computational model for machine design." *Research in Engineering Design* 7, no. 2 (1995): 102-125.

[19] Starling, Alex C., and Kristina Shea. "A grammatical approach to computational generation of mechanical clock designs." In *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*. 2003.

[20] Sridharan, Prasanna, and Matthew I. Campbell. "A grammar for function structures." In *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 41-55. American Society of Mechanical Engineers, 2004.

[21] Rai, Rahul, Pranay Kilaru, Ravi Vallepalli, and Matthew I. Campbell. "A Novel Search Algorithm for Interactive Automated Conceptual Design Generator (ACDG)." In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 987-996. American Society of Mechanical Engineers, 2011.

[22] Mangan, Alan P., and Ross T. Whitaker. "Partitioning 3D surface meshes using watershed segmentation." *Visualization and Computer Graphics, IEEE Transactions on* 5, no. 4 (1999): 308-321.

[23] Shlafman, Shymon, Ayellet Tal, and Sagi Katz. "Metamorphosis of polyhedral surfaces using decomposition." In *Computer Graphics Forum*, vol. 21, no. 3, pp. 219-228. Blackwell Publishing, Inc, 2002.

[24] Cohen-Steiner, David, Pierre Alliez, and Mathieu Desbrun. "Variational shape approximation." In *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 905-914. ACM, 2004.

[25] Attene, Marco, Bianca Falcidieno, and Michela Spagnuolo. "Hierarchical mesh segmentation based on fitting primitives." *The Visual Computer* 22, no. 3 (2006): 181-193.

[26] Gelfand, Natasha, and Leonidas J. Guibas. "Shape segmentation using local slippage analysis." In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 214-223. ACM, 2004.

[27] Hilaga, Masaki, et al. "Topology matching for fully automatic similarity estimation of 3D shapes." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001.

[28] Attene, Marco, Bianca Falcidieno, and Michela Spagnuolo. "Hierarchical mesh segmentation based on fitting primitives." *The Visual Computer* 22, no. 3 (2006): 181-193.

[29] Shapira, Lior, Ariel Shamir, and Daniel Cohen-Or. "Consistent mesh partitioning and skeletonisation using the shape diameter function." *The Visual Computer*24, no. 4 (2008): 249-259.