Rule-based Lightweight Structure Design

Yuan Yao, Wanwan Li, Junsheng Shen
School of MEA
Shanghai University
Shanghai, CN
yaoyuan@shu.edu.cn

Xiaoming Yang
School of Computer Science
Huzhou University
Huzhou, CN
yxm2008@hutc.zj.cn

Abstract

This paper presents a rule-based computer aided design framework for assembling complex internal structure. It defines each voxel with its neighborhood as an intelligent agent, which make it possible to convert the structural assembly problem into an optimization problem of a multi-agent system. We use this method in the lightweight model design. Simple defeasible rules are used to build up the agents behaviors, which bind the external requirements with the internal structures. The target model is generated by the emergence of a large number of agents, which provided an implementation mechanism for the function-oriented design.

1 INTRODUCTION

3D printing technologies provided a solution to the fabrication of complex structures, while it also brought complexity to the design. At present, some 3D printing oriented CAD systems, such as Meshmixer and 3-Matic, have been integrated with mesh model optimization modules to provide mesh hollowing, support structure generation and other operations to support lightweight design. However, there is a lack of effective analytical testing and evaluation tools for inexperienced designers. For example, the structural strength cannot be tested in the process of design. Most CAD methods rely on the third-party simulation and analysis system to validate the design. This process is repeated until the model meets the requirements of functions and the manufacturing process.

Many researchers attempt to improve the design ability by integrating the process of design and analysis into an iterative process. In the lightweight design, changing the internal structures [LSZ+14]and the material distribution[PEVWLSH13], are commonly used methods to reduce the weight while maintaining the strength of model. Wang et al. [WWY+13] proposed a skin-frame structure to save the cost of printing and shorten the printing time. This method was further extended for generating support structures in 3D printing. Telea et al.[TJ11] provided an evaluation method to detect and assess strength problems caused by the slender structures and holes in the model.

In order to establish a reused design framework, Vidimce et al.[VVCEWRKM13] presented a programmable pipeline for specifying multi material distribution in the 3D structures, which provided a programming mechanism for designing complex volumetric structure. Chen et al.[CLD+13] used a reducer tree and a tuner network to translate functional requirements to structures. However, these work cannot avoid complex configuration and programming.

The integration of finite element analysis (FEA) in the iterative makes the design process become a time consuming work, To speed up the design, Chen et al.[CLSM15] provided a data driven finite element method. A composed structure library was built to simulate the real properties of the complex multi-material structure units. These kind of techniques

In: T. Ågotnes, B. Liao, Y.N. Wang (eds.): Proceedings of the first Chinese Conference on Logic and Argumentation (CLAR 2016), Hangzhou, China, 2-3 April 2016, published at http://ceur-ws.org

are not enough to solve the problem. With the increase of design complexity, the mesh model is often divided into high precision voxels. While the computation demand is also becoming huge [KTL+15].

In order to reduce the amount of computation, we provide a structure evolution framework for volumetric structure design. In this framework, a voxel with its neighborhood is defined as an intelligent agent. As a consequence, the volumetric modeling can be converted into a self-organized problem driven by a multi-agent system(MAS). Simple rules are designed to build up the geometric structures of the voxels, and the type of interconnection between them. We describe the MAS based volumetric structure design framework in detail and utilize it in a lightweight model design. By using different group of rules, different forms of structures are generated. While the mechanical simulation result can be fed back into the design process to realize a demand-driven design.

2 Design Framework

The goal of lightweight model design is to reduce the quality of the model according to the requirements of mechanics and functions. It can be defined as the following optimization problem:

$$\arg\min_{v_i} \Sigma v_i \quad \forall v_i \in \mathcal{M} : \mathrm{F}(v_i) < \eta[\sigma]$$

where v_i is a voxel, $\mathcal{M}$ presents the entity part of the model, $\mathrm{F}(v_i)$ represents the stress at v_i, $[\sigma]$ is the allowable stress, and η is a safety coefficient.

In order to meet the design goals, we first integrate the structural design and the structure analysis into a unified volumetric computational domain. As shown in fig. 1, the proposed design framework is divided into four stages: (1)voxelization, (2)initialization, (3)structure evolution, and (4)evaluation (FEA) & feedback. The evolution and evaluation are repeated until achieve the terminal conditions. The output volumetric model can be reconstructed to a closed mesh for 3D printing.

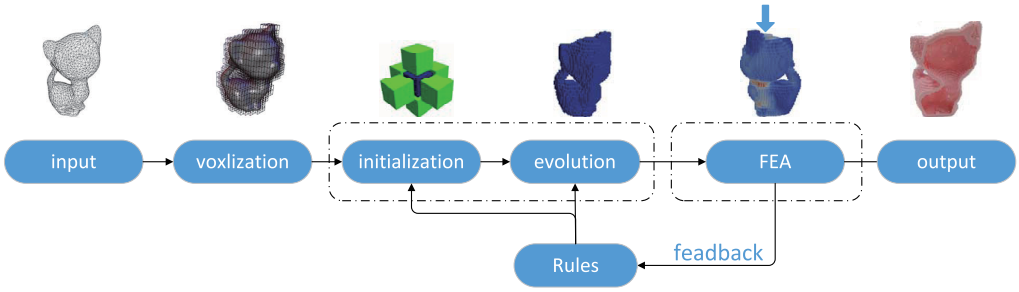

Figure 1: Design framework

2.1 Voxelization

The voxelization stage discretizes the input mesh model into the voxels domian. By using the modern methods, such as the out-of-core sparse voxel octrees [BLD14], highly detailed multi-material structures can be expressed in a single data structure. As a consequence, all design and analysis are able to carried out on this discrete data structure.

2.2 Initialization

In the initialization stage, we define a voxel with its neighborhood as a intelligent agent. Due to the hierarchical structure of the volumetric model, this definition can be done on the different levels of the model. Therefore, different population of agents can be produced. The initial configuration of the system is composed of a population of agents that fixed at the corresponding voxels. Each agent can change its own structure based on the environment. We thus convert the volumetric model into a multi-agents system (MAS). The function oriented design can also be described by a problem of MAS based optimization.

2.3 Evolution

The structure evolution stage is an iterative process. During each time step, an agent decides its action according to the environment. The environment contains many attributes, each attribute has a different impact on the agent. However, the agent decides to attempt only one action. The agent's behavior will change the local structure and induce reactions of the environment. Therefore, the volumetric structure will continue to change until it meets the design requirements.

2.4 Evalutation

In the context of lightweight model design, the process of evalution & feedback is used to convert the user requirements to the data format can be perceived by the agent. That is, it builds up a sensible environment for the agents, together with the geometric constraints. More evaluation processes can be added in the framework to support other design requirements.

3 MAS Expression of the Framework

MAS is preferred for expressing a group of heterogeneous entities. It provides a methodology for designing systems, from the view of the agents. In order to maintain consistency with the design framework provided in section 2, we assume that MAS is a discrete and finite system. The environment and the agents embedded in the environment are defined as follows.

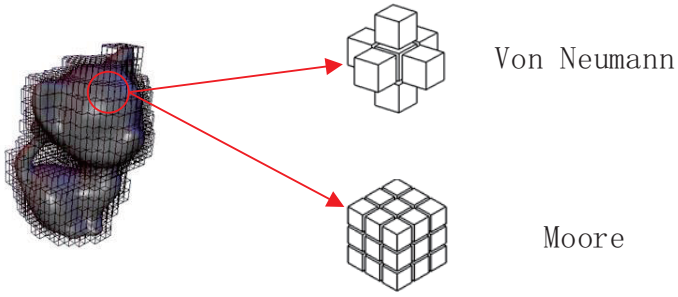

Figure 2: Environment and agent

The environment is a 3D grid generated from a closed triangle mesh. The element in the grid is called voxel, denoted by v . Each v has different attributes.

The agent is fixed in a voxel and its neighbourhood. The type of neighborhood can be Von Newmann or Moore. This structure is called a design unit. The voxels in a design unit can be changed and removed. A voxel can belong to different design units (Fig.2). However, there cannot be more than one agent occupy the whole design unit. The entity part of the model, denoted by $\mathcal{M}$, is composed of the design units. Each design unit in $\mathcal{M}$ is characterized by a state vector $\mathbf{s}_a$, which contains a set of variables that characterize the performance of the current structure. The behaviors of the population of agent determine the final structure of $\mathcal{M}$.

We formulate the agents dynamics by using the defeasible rules [Lou87] . The reason we introduce the defeasible logic is that the mechanism of microscopic structures interaction is very complex. Design requirements and design rules are often changed in different applications and different domains. Those changing factors lead to the occurrence of many abnormal conditions that we cannot predict and handle. For example, when a voxel suffered stress is far less than the safety value, it will be deleted. However, the voxel is just located on the surface of the structure. The deletion will destroy the structure. It is impossible to predict these type of exceptions, especially when more constraints are added in the design process in the form of a group of rules. The defeasible logic is a suitable option, which can solve the conflicts of rules without adding additional explicit knowledge. It provides a simple interface that allows the complex design system to be controlled and guided by the user at a high level.

Based on this, we describe the ability of agent from two aspects: the perception and actions. The perception consists of two functions:

1. $\Gamma_1(state)$ return the agent's own information. This state can be its own attributes $\mathbf{s}_a$ or some indicators computed from these attributes.

2. $\Gamma_2(state)$ return state from outside. In the context of lightweight model design, the only ouput is the feedback from FEA.

The actions of a agent includes the change of its own state, the changing of the geometric structure and materials, etc. In the context of lightweight design, the set of actions includes:

1. Delete(v): a agent delete its voxel.

2. Update(v): a agent update the state of it's voxel.

In order to avoid cross operation, the v is limited to the voxel in the center of the design unit where an agent occupies.

For each agent, the decision process returns an action based on the agent perceptions. Different perceptions need to be combined to generate an action. At the level of the whole model, design requirements are converted to the domain beliefs of the agents, expressed by the defeasible logic. For lightweight model design, these beliefs are listed as follows:

1. Facts:
   (a) $isSafe(\Gamma_2(stress))$, to determine if the equivalent effective stress in a voxel is less than $\eta[\sigma]$;
   (b) $isTenPercent(\Gamma_2(stress))$, to determine if the equivalent effective stress in a voxel is in the minimum of 10% of the total voxels in $\mathcal{M}$;
   (c) $isBoundary(\Gamma_1(location))$, to determine the center voxel of an agent is located on the boundary of $\mathcal{M}$ or not;
   (d) $isNull(\Gamma_1(state))$, to check if the property that indicates the distance to boundary is empty.
   (e) $isMaxInNeibour(\Gamma_1(state))$, to determine if the property that indicates the distance to boundary is the largest in the neighborhood.
2. Strict rules:
   (a) $isNull(\Gamma_1(state)) \rightarrow Update(v)$

   Defeasible rules:
   (a) $isTenPercent(\Gamma_2(stress)) \Rightarrow Delete(v)$
   (b) $isMaxInNeibour(\Gamma_1(state)) \Rightarrow Delete(v)$
3. Defeaters:
   (a) $isBoundary(\Gamma_1(location)) \rightsquigarrow \neg Delete(v)$
   (b) $\neg isSafe(\Gamma_2(stress)) \Rightarrow \neg Delete(v)$

In this simple model, no superiority relations are needed. In the voxels based lightweight model design framework, the agent is covered with each other, which can be used to simulate the interface of materials and structures, where their boundaries are not clear in the real model. Heterogeneous agents can also be added to express the characteristics of the multi-material, flexible structure, and gradient density distribution. In order to simplify the discussion, we only consider the use of isomorphic agent in the context of lightweight model design.

4 Experiments

We use several mesh models to evaluate the features of our method. After voxelization by octree, each voxel with its Von Neumann neighborhood is chosen as a agent. Corresponding geometric editing methods and attribute editing methods are also developed according to the the actions of agent defined in section 3. For instance, the action update(v) is realized to check whether the current property that indicates the distance to boundary is 0, if it is, and its neighborhood location value is not 0, the property is updated with $maximum(\mathcal{N}(v)) + 1$, where $\mathcal{N}(v)$ presents the voxels in the neighborhood of v.

In the voxelization, the octree depth is set to 5. Thus the configuration of agents and the volumetric grid used in FEA can be defined in the same level. In the initialization stage, we first mark all voxels, i.e, the boundary voxels are set to 1, and the interior voxels are set to 0. Then, strict rules (a) is used to generate a signed distance map in the 3D volumetric space, which provides a signed distance map for rule-based structure evolution.

In the evolution stage, The defeasible rule (b) and defeater (a), (b) are used to control the changing of structure together. If we convert the defeasible rule (b) to a strict rule, or delete the defeater (a), all structure units are removed. When the defeasible rule (b) is removed, the evolution process will not be started. Fig. 3 shows the structure evolution process on different mesh models without external load. That means the formation of the structure is only affected by the gravity.

In Fig. 3, the change of 3D structure is given by the odd lines. And the even number of lines give the structure change on the longitudinal section. Where N is the number of iterations of evolution. In the experiment, most of the process are convergence after 8 times of iterations.

By using of the new generated volumetric structure, the evaluation process further divide it into tetrahedral data. We deploy OOFEM [Pat12] to carry out the FEA process. Considering the FEA process is the most time consuming operation

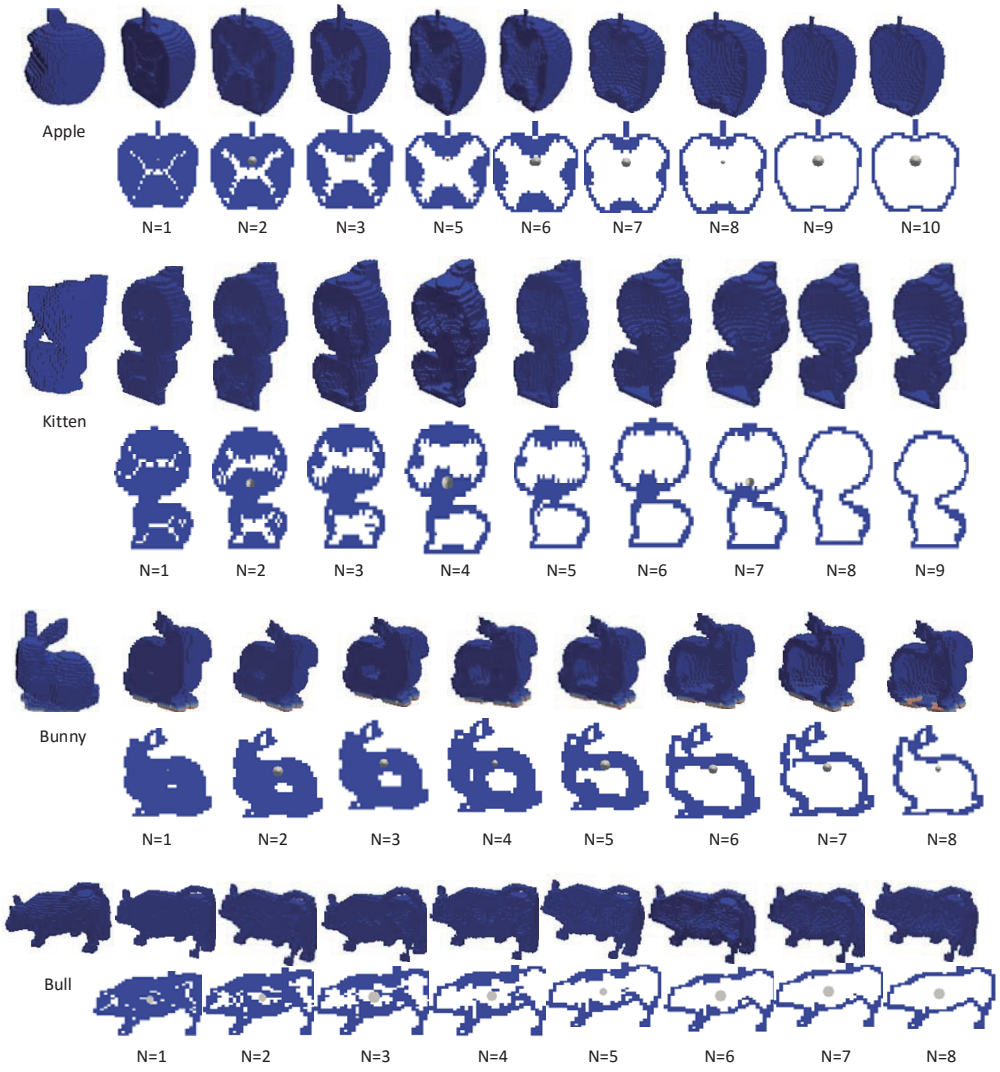

Figure 3: Structure evolution process without load

in the process of evolution. To further speed up the design process, we can utilize the low-resolution voxels for FEA analysis, while use the high-resolution voxels to assemble the structure.

We also test the structure design under the external loading condition. In this situation, The defeasible rule (a) and defeater (a), (b) are used to control the structure evolution process. The same terminal conditions is used, however, we get the different structure. Fig. 4 shows the change of structure during the evolution process. We can see a support structure is generated inside the model, which is just located in the direction of the external force.

Fig. 5 shows examples of the generated solid model, fabricated by a common desktop FDM printer. The material is PLA.

5 Conclusion

Simple defeasible rules can be used to establish relationship between simple properties and local structures. By using our framework and the existing logical reasoning method, we can build up a large number of rules to map the relationship between complex structures and the functions. This provides an effective mechanism for design of functional structures that are difficult to be expressed directly. In addition, the external information, such as the load, can be evaluated through FEA and fed back into the design process as a driving force for structures evolution. Our framework can also have the

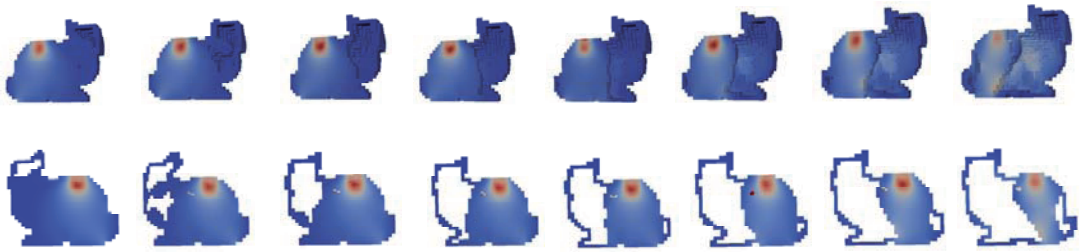

Figure 4: Structure evolution process with load

Figure 5: 3D print model

potential value to simulate the growth of real biological tissues.

It is simple to extend the framework by adding new rules to support different design requirements and structure patterns. Therefore, designing more rules to test the ability of the method is a natural direction in the future. We will work on design of argumentation application to avoid the conflict problem in reasoning with a large number of design rules.

6 Acknowledgements

This research is partly funded by a grant from the Huzhou city public key technology research projects, China(No.2013GZ02) and Public welfare technology research project of Zhejiang province, China(No.2014C31084).

References

[BLD14] Jeroen Baert, Ares Lagae, and Ph Dutré. Out-of-core construction of sparse voxel octrees. In *Computer Graphics Forum*, volume 33, pages 220–227. Wiley Online Library, 2014.

[CLD+13] Desai Chen, David I. W. Levin, Piotr Didyk, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Spec2fab: a reducer-tuner model for translating specifications to 3d prints. *ACM Trans. Graph.*, 32(4):135:1–135:10, 2013. Spec2Fab 2013.

[CLSM15] Desai Chen, David IW Levin, Shinjiro Sueda, and Wojciech Matusik. Data-driven finite elements for geometry and material design. *ACM Transactions on Graphics (TOG)*, 34(4):74, 2015.

[KTL+15] Zhong Xun Khoo, Joanne Ee Mei Teoh, Yong Liu, Chee Kai Chua, Shoufeng Yang, Jia An, Kah Fai Leong, and Wai Yee Yeong. 3d printing of smart materials: A review on recent progresses in 4d printing. *Virtual and Physical Prototyping*, 10(3):103–122, 2015.

[Lou87] Ronald P Loui. Defeat among arguments: a system of defeasible inference. *Computational intelligence*, 3(1):100–106, 1987.

[LSZ+14] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. Build-to-last: Strength to weight 3d printed objects. *ACM Transactions on Graphics (TOG)*, 33(4):97:1–97:10, 2014.

[Pat12] Bořek Patzák. Oofemłan object-oriented simulation tool for advanced modeling of materials and structures. *Acta Polytechnica*, 52(6), 2012.

[PEVWLSH13] Romain Pr E Vost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make it stand: balancing shapes for 3d fabrication. *ACM Transactions on Graphics (TOG)*, 32(4):81:1–81:10, 2013.

[TJ11] Alexandru Telea and Andrei Jalba. Voxel-based assessment of printability of 3d shapes. In *Mathematical Morphology and Its Applications to Image and Signal Processing*, pages 393–404. Springer, 2011.

[VVCEWRKM13] Kiril Vidim V C E, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. Openfab: A programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics*, 32(4):136:1–136:12, 2013. Open Fab 2013.

[WWY+13] Weiming Wang, Tuanfeng Y. Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. Cost-effective printing of 3d objects with skin-frame structures. *ACM Transactions on Graphics (TOG)*, 32(6):177:1–177:10, 2013.