Procedural Grammars for Truss Microstructure Design

Ane Zuñiga
anezu@mit.edu
MIT EECS

Jorge Cañada
jcanadap@mit.edu
MIT EECS

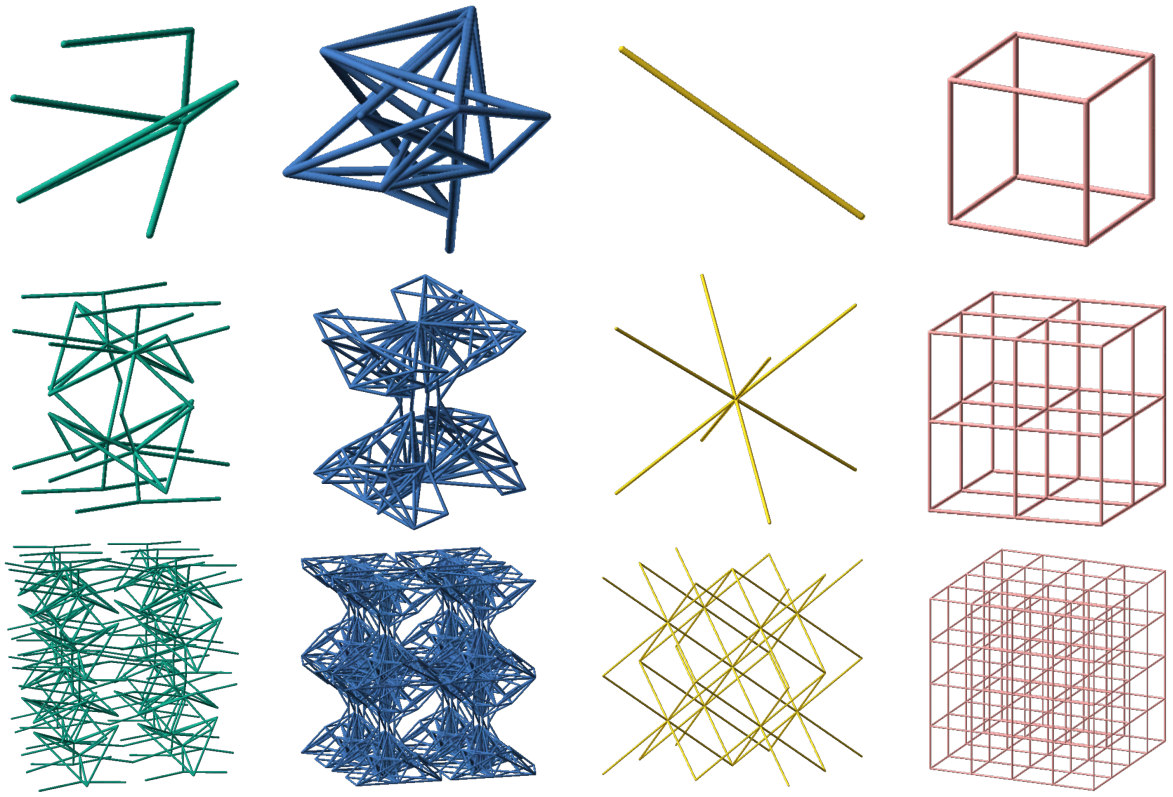

Figure 1: Diverse truss structures generated using the proposed procedural grammar.

ABSTRACT

Truss structures play a pivotal role across diverse domains like civil engineering, space exploration, and healthcare, owing to their adaptability and structural efficacy. Despite the longstanding success of conventional design approaches in crafting large-scale truss configurations, the realm of microscale truss design remains largely untapped. In this project, we leverage procedural grammars, computational tools for design space exploration, to systematically generate cube-shaped truss structures. By defining a procedural grammar tailored for truss design, we aim to expand the design space beyond typical structures and explore novel configurations. We provide open-source code implementing the proposed grammar and generate a dataset of cube-shaped truss structures. Through this work, we contribute to the advancement of truss design methodologies, enabling designers to explore non-traditional truss configurations and expand the possibilities for structural optimization.

KEYWORDS

Truss microstructures, procedural grammars, computational design

1 INTRODUCTION

Trusses (i.e., assemblies of elements such as beams that form a rigid structure) are one of the most fundamental construction structures. Their applications span a broad variety of fields, including civil engineering [Tian et al. 2019], space exploration [Onoda 1988], and healthcare [Bagheri et al. 2018]. Additionally, truss structures are suitable for construction in a wide range of sizes, spanning from megastructures (e.g., Eiffel Tower, Forth Bridge), to micro- and nanostructures (e.g., DNA origami trusses [Matthies et al. 2016]).

While the use of trusses for the construction of large structures is well established and has been applied for centuries, the fabrication of microscopic truss structures has only become possible in recent years, with the development of precise nanofabrication technologies.

Furthermore, in addition to the intrinsic material efficiency of trusses at any scale, the fine-tuning of geometrical parameters at the microscale can influence macroscopic mechanical properties [Yan et al. 2006], [He et al. 2017], making the study of truss microstructures increasingly relevant. However, the infinite possibilities of

truss arrangement make the design space unfathomable through traditional design techniques.

Recent works on truss structures propose the use of computational methods (e.g., topology optimization [Asadpoure et al. 2011], simulated annealing [Lamberti 2008], procedural graphs [Makatura et al. 2023]) for truss design optimization. Although these works claim to provide effective solutions to the inverse design problem (i.e., taking a desired behavior as input, outputting an optimized design), they can only address specific problems and do not allow to generalize.

Procedural grammars are computational tools for the exploration of design spaces that allow to generate samples following a fixed set of rules. The rules are defined taking into account the nature of the problem and are applied sequentially in order to generate the samples. Examples of the use of grammars for design space exploration include urban design [Beirão 2012], chemical engineering [Guo et al. 2022] and mechanical design [Agarwal et al. 2000], among others.

This work aims at widening the view on the truss design space by systematically generating truss designs that are structurally feasible and do not necessarily resemble typical structures. This computation of new truss designs is meant to explore the largely unknown design space of trusses and inform designers about non-typical configurations. Furthermore, the generated designs can readily be introduced into a mechanical simulator in order to also populate the performance space of trusses.

The contributions of these paper can be summarized as follows:

- We define a procedural grammar for the systematic generation of cube-shaped truss structures.
- We provide open-source code that implements the proposed grammar.
- We generate a dataset of cube-shaped truss structures generated using the proposed grammar.

The rest of this paper is structured as follows: the Preliminaries section contains an introduction to the foundational concepts that this paper builds on (i.e., grammars, shape grammars, context-sensitive grammars); the Related Work section expands upon previous work on truss design; the Methodology section exposes the proposed grammar for truss design; the Results section delves into the implementation of the proposed grammar and shows examples of its use; the Discussion section examines the benefits and limitations of using the proposed grammar; and the Conclusion section presents final remarks.

2 PRELIMINARIES

Grammars are rule-based systems for the description and generation of designs. They typically comprise a set of symbols and rules. The rules code transitions between those symbols and they are defined following the laws that naturally govern the design space. For example, if a simple grammar has two symbols: A and B, and two rules: *p1: A turns into AA*, and *p2: A turns into B*, then starting with one symbol A, the application of the rules a number of times and in some order will yield a chain of As and Bs (e.g., $ABBAAA$). For example, in a grammar where symbols A and B represent molecules and the rules are defined according to specific chemical reactions, the use of such grammar can yield the products synthesized in a reactor.

Shape grammars are a specific type of grammar that generate designs by computing directly with shapes in two or three dimensions, rather than with symbols or other indirect representations of shapes [Knight and Stiny 2015]. Like their name suggests, shape grammars are particularly suitable for working with shapes, as they are able to represent them in a more intuitive way than a chain of symbols could. Shape grammars can be, however, difficult to automate as a result of their non-abstract nature [Stiny 2006].

The grammar developed in this work is a shape grammar that directly represents the physical location of edges and vertices in trusses.

The other grammar category relevant to this paper is context-sensitive grammars, also referred to as Type-1 grammars according to Chomsky's hierarchy [Chomsky 1956]. The rules of context-sensitive grammars consider not only the symbol to which they are applied, but also their context (i.e., other symbols around them). Therefore, using context-sensitive grammars, it is possible to implement conditional rules that accurately mimic real processes (e.g., chemical synthesis [Guo et al. 2022]).

Context-sensitive grammars, as defined above, only capture local conditions, i.e., when applying a rule, they take into consideration the immediate surroundings of a specific symbol, but fail to assess global information. However, it is possible to extend them by introducing global parameters that capture system-level information [Guo et al. 2022]. This allows to keep track of additional information that can also be used to decide which rules are applicable. The following is an example of a grammar rule that takes into account both the local context and global parameters:

$$\text{p3: } B \text{ <A> } B : \text{N} < 10 \rightarrow AAA$$

This rule allows to turn A into AAA only if there is one B on each side of A, and if the parameter N has a value lower than 10.

This work proposes a context-sensitive shape grammar with global parameters for the computational generation of truss microstructures.

3 RELATED WORK

Symmetry plays a critical role in the design of truss structures, particularly for simplifying manufacturing and optimizing mechanical properties [Tang and Whitcomb 2003]. Traditionally, symmetric truss designs have been employed where a single octant of a unit cell is designed and then mirrored across planes to form the full lattice structure. This approach ensures that the microstructure exhibits uniform properties across different directions when translated throughout a lattice. Such techniques have been foundational in the development of materials that require isotropic properties or specific directional strengths.

In previous work [Panetta et al. 2015], a novel approach is detailed for the design of parametric, tileable 3D patterns suitable for additive manufacturing. This method employs a combinatorial search over potential topologies combined with a subsequent shape optimization phase, enabling the creation of a diverse array of symmetric 3D patterns. The primary innovation lies in the parametric nature of the design process, which facilitates extensive customization of the patterns' geometric and mechanical properties. These

parametric patterns are particularly advantageous for achieving varied elastic material properties, as they can be precisely tailored to meet specific functional requirements. The design process is inherently modular, leveraging the parametric system to adjust dimensions and shapes dynamically.

[Zok et al. 2016] explore the classification and design space of periodic trusses using principles from crystallography and geometry. They introduce a taxonomy for describing the structure types of trusses, which significantly simplifies the search for optimal structures in various applications such as structural engineering, materials science, and more. These applications often exploit trusses for their advantageous properties in load bearing, thermal management, and fluid flow facilitation. While [Zok et al. 2016] provide a comprehensive overview of typical truss structures and propose a generalized taxonomic classification to aid in navigating the design space, their exploration remains within the bounds of conventionally used truss configurations.

4 METHODOLOGY

In this section, we outline our approach to generating procedural grammars for microstructure truss design. We start by defining the context in which our project operates, including the assumptions and constraints we consider. Then, we detail how we address these constraints and develop the procedural grammar rules. The primary aim of this section is to provide a clear understanding of our systematic approach to generating truss structures.

4.1 Context: Assumptions and Constraints

4.1.1 Assumptions. In this section, we delineate the fundamental assumptions guiding our approach to truss structure generation. We assume that our target structures are cubic-shaped and possess symmetry. Consequently, our design methodology focuses on defining the truss structure within a single octant of a unit cell. Subsequently, this octant is mirrored across the three axes to construct the complete unit cell. The lattice structure is then formed by translating this unit cell in space. For a visual representation of this process, please refer to Figure 3.

4.1.2 Constraints. In addressing feasibility constraints, we prioritize continuity throughout the truss structure generation process. Specifically, we ensure the absence of floating vertices or disconnected substructures within the cubic-shaped symmetrical trusses we design.

To achieve this, all vertices within the octant must be interconnected, either directly or indirectly. This interconnection guarantees structural integrity and eliminates the presence of isolated components within the truss.

Furthermore, we enforce continuity preservation during the mirroring and translation operations. To achieve this, we require at least one vertex to be present on each face of the octant. In cases where a vertex is shared between two or three faces (e.g., edges), it is counted as contributing to the continuity of multiple faces.

Additionally, each vertex within the truss structure must satisfy specific connectivity conditions to ensure structural integrity and eliminate floating edges. Here, we introduce the variable *FulfilledVertex*, which indicates whether a vertex is satisfying connectivity conditions or not. These conditions vary depending on the vertex's location within the structure.

- A vertex situated within the body of the structure must have at least two connections to any other vertices.
- A vertex located on a face of the structure must fulfill one of the following conditions: it must have at least two connections within the same face; alternatively, it must have at least one connection to a vertex that is not within the same face.

4.2 Grammar Rules

In our approach to generating procedural grammars for microstructure truss design, we define specific symbols to denote various elements within the grammar:

- **E**: Essential vertices, referencing vertices on faces to maintain continuity feasibility.
- **B**: Additional vertices inside the body, representing any extra vertices within the body of the structure.
- **F**: Additional vertices on faces, indicating any extra vertices located on the faces of the structure.
- **h**: Non-terminal edge, representing a connection between two vertices that is yet to be determined.
- **I**: Terminal connection, denoting a connection between two vertices.
- **J**: Terminal disconnection, representing a non-connection between two vertices.

The additional parameters used in this approach include:

- N: Total number of vertices.
- M: Maximum number of interconnected vertices.
- *FullfilledVertex*: Indicates if a vertex fulfills connectivity conditions.

We utilize context grammars to define the rules governing the generation of truss structures. These rules are described below:

(1) **P1: Initial Vertex and Non-terminal Edge Generation** This rule serves as the initial vertex and non-terminal edge generator. It generates n_e essential vertices, along with any optional n_b additional vertices within the body and n_f on faces. Additionally, it generates all non-terminal edge connections among each pair of vertices. The sum of $n_e + n_f + n_b$ equals the total number of vertices, denoted by N.

$$\text{p1: None } <\mathcal{X}> \text{ None : None} \rightarrow n_e E, n_b B, n_f F, \frac{N(N-1)}{2} h$$

(2) **P2: Vertex Connection** Irrespective of the current state, this rule allows us to convert a non-terminal edge into a terminal connection. Upon application, it updates the count of interconnected vertices, M, and updates the connectivity status of the two connected vertices.

$$\text{p2: None } <\text{h}> \text{ None : None} \rightarrow I, \text{ Update } \textit{FulfilledVertex} \text{ and } M.$$

(3) **P3: Vertex Disconnection** This rule governs the disconnection of vertices, converting a non-terminal edge into a terminal disconnection. It is context-sensitive and can only be applied when all vertices are interconnected, and both

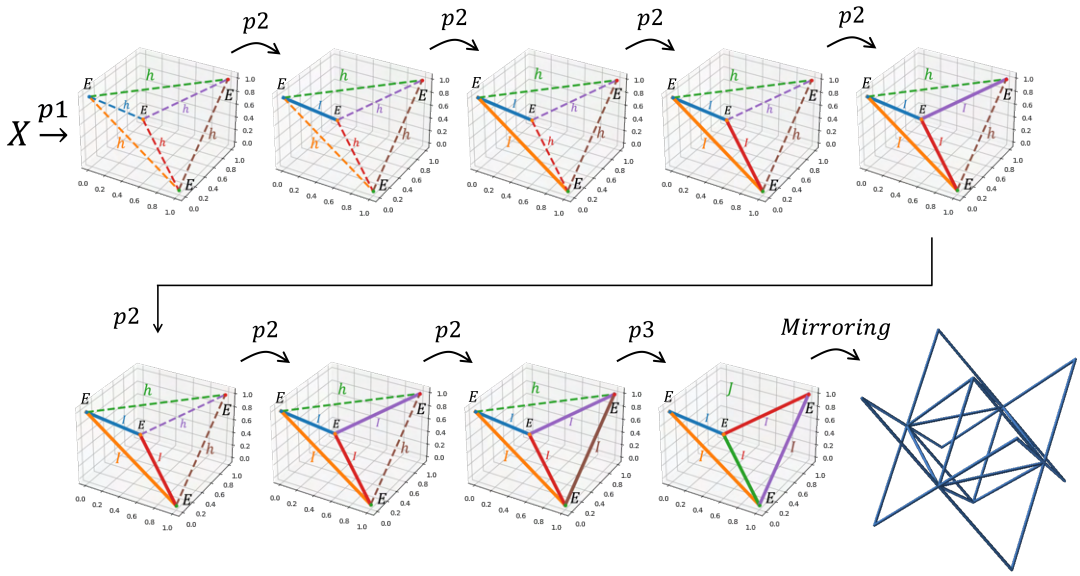

Figure 2: Step-by-step evolution of a truss microstructure with four vertices, demonstrating the application of grammar rules. Each stage of the design process showcases the sequential application of rules, resulting in the gradual formation of the final structure.

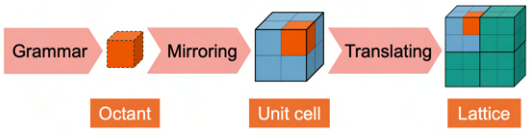

Figure 3: Illustration depicting the generation of cubic-shaped truss structures with symmetry. The truss structure is defined within a single octant of a unit cell and then mirrored across the three axes to construct the complete unit cell. Finally, the lattice structure is formed by translating this unit cell in space.

vertices connected by the edge satisfy the vertex connection criteria.

p3: *FulfilledVertex* <h> *FulfilledVertex* : M == N $\rightarrow$ *J*,

Update *FulfilledVertex* and M

These rules collectively define the procedural grammar used to generate truss structures, ensuring continuity feasibility and structural integrity throughout the design process.

5 RESULTS

Figure 2 illustrates the step-by-step design of a structure composed of 4 vertices. Initially, Rule 1 is applied, resulting in the creation of 4 vertices and 6 non-terminal edges, distinctly marked with dashed lines. Subsequently, connections are established through the application of Rule 2. Finally, Rule 3 is utilized to execute a terminal disconnection. Here note that this rule can be applied because all vertices are already interconnected, and the disconnected edge's vertices fulfill the vertex connectivity criteria. To complete the process, the octant structure is mirrored across all three axes, resulting in the formation of the unit cell structure.

Additionally, we also generate some other random structures. For the code implementation, we set the maximum number of extra vertices in faces to be 5 and the maximum number of extra vertices in the body to be 5, which are decided randomly. Once the initial set of vertices and non-terminal edges are generated, the non-terminal edge where the next rule will be applied is selected randomly. If the vertex connectivity and interconnectivity criteria are not met, the connection rule is applied. Otherwise, connection or disconnection is selected with a 10% probability for connection. Some examples of structures generated in this manner are shown in Figure 4

We have released the source code for our implementation as open-source software, available at the following GitHub repository: https://github.com/anezuniga/grammar_truss_microstructure.

6 DISCUSSION

The proposed grammar allows to easily generate a vast dataset of truss microstructure examples. The design space encompassed by this grammar, however, is still limited compared to the infinite design of trusses: our grammar is limited to cube-shaped structures defined through octant mirroring.

Moreover, the conditions we chose to impose to guarantee continuity and structural soundness also restrict the design space that can be accessed through this grammar. For example, as shown in Figure 5 the fundamental octant required to produce a cube unit cell through mirroring does not comply with our restrictions defined in Constraints subsection, for a vertex on a face (although an octant that is already a cube does satisfy the constraints).

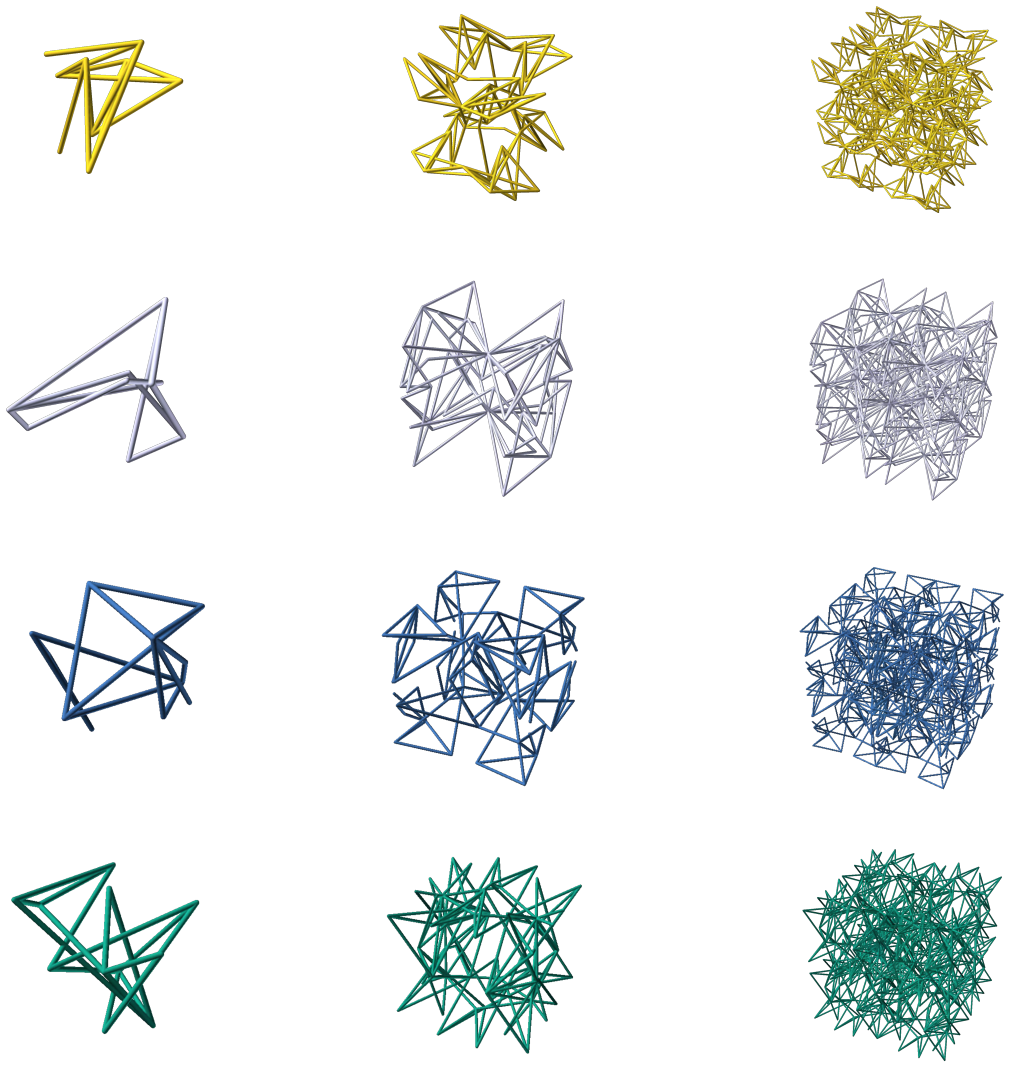

Figure 4: Examples of truss microstructures generated through random application of grammar rules. The structures include octant configurations, unit cell representations, and lattice arrangements.

Additionally, our grammar formulation does not account for volume and does not consider the possibility of edges intersecting each other. This circumstance could be addressed by detecting the occurrence of intersections and assigning additional vertices to them, or by imposing additional restrictions that avoid the appearance of intersections.

Some grammars found in the literature explicitly include all the information used for decision-making in the rules formulation [Guo et al. 2022]. Such approach allows to keep the grammar well defined and self-contained, but it increases the complexity of the rules. Our grammar, on the other hand, relies on computation steps that are not explicitly included in the rules, such as the computation of the values of the parameter *FulfilledVertex*. While this approach makes the grammar depend on algebra that is not fully contained within its rules, it allows to keep the grammar formulation clear and concise. Approaches to turn this grammar into a self-contained grammar include introducing additional symbols that identify the position of the vertices (e.g., vertice inside body, vertice on a face), and the use of parameters that propagate information along the shape as each rule is applied.

7 CONCLUSION

The formulation of a context-sensitive shape grammar for truss microstructure design has been presented. The proposed grammar is capable of generating a diverse set of examples of cube-shaped truss structures produced through the mirroring of an octant across three axes. While the grammar is concise (it consists of only three rules and two parameter update expressions), it is capable of covering a

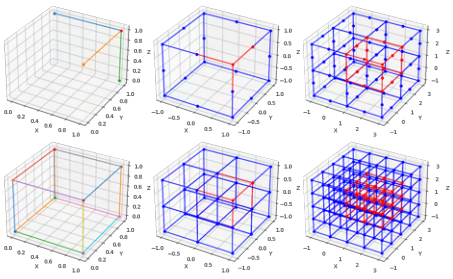

Figure 5: Example of design space limitation resulting from the imposed constraints. Top: building sequence of a cube unit cell from an octant that does not comply with our constraints. Bottom: building sequence of a similar structure from an octant that complies with our constraints.

wide design space that includes some of the most commonly used truss designs (e.g., octet). This grammar enables the systematic exploration of the truss design space, allowing to discover numerous designs in an otherwise unfathomable design space. In combination with simulation tools, the proposed grammar can allow the discovery of optimal designs for specific applications. To facilitate its use, the code implementation and a dataset of generated examples are made available online.

ACKNOWLEDGMENTS

We extend our sincere gratitude to Liane Makatura, PhD student at the MIT Computational Design and Fabrication Group, for her input and guidance throughout this project.

REFERENCES

Manish Agarwal, Jonathan Cagan, and George Stiny. 2000. A micro language: generating MEMS resonators by using a coupled form—function shape grammar. *Environment and Planning B: Planning and Design* 27, 4 (2000), 615–626.

Alireza Asadpoure, Mazdak Tootkaboni, and James K Guest. 2011. Robust topology optimization of structures with uncertainties in stiffness–Application to truss structures. *Computers & Structures* 89, 11-12 (2011), 1131–1141.

Ali Bagheri, Irene Buj-Corral, Miquel Ferrer Ballester, Maria Magdalena Pastor, and Francesc Roure Fernandez. 2018. Determination of the elasticity modulus of 3D-printed octet-truss structures for use in porous prosthesis implants. *Materials* 11, 12 (2018), 2420.

José Beirão. 2012. *CItyMaker: designing grammars for urban design.* TU Delft.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2, 3 (1956), 113–124. https://doi.org/10.1109/TIT.1956.1056813

Minghao Guo, Wan Shou, Liane Makatura, Timothy Erps, Michael Foshey, and Wojciech Matusik. 2022. Polygrammar: grammar for digital polymer representation and generation. *Advanced Science* 9, 23 (2022), 2101864.

ZeZhou He, FengChao Wang, YinBo Zhu, HengAn Wu, and Harold S Park. 2017. Mechanical properties of copper octet-truss nanolattices. *Journal of the Mechanics and Physics of Solids* 101 (2017), 133–149.

Terry Knight and George Stiny. 2015. Making grammars: From computing with shapes to computing with things. *Design Studies* 41 (2015), 8–28.

L Lamberti. 2008. An efficient simulated annealing algorithm for design optimization of truss structures. *Computers & Structures* 86, 19-20 (2008), 1936–1953.

Liane Makatura, Bohan Wang, Yi-Lu Chen, Bolei Deng, Chris Wojtan, Bernd Bickel, and Wojciech Matusik. 2023. Procedural metamaterials: A unified procedural graph for metamaterial design. *ACM Transactions on Graphics* 42, 5 (2023), 1–19.

Michael Matthies, Nayan P Agarwal, and Thorsten L Schmidt. 2016. Design and synthesis of triangulated DNA origami trusses. *Nano letters* 16, 3 (2016), 2108–2113.

Junjiro Onoda. 1988. Two-dimensional deployable truss structures for space applications. *Journal of Spacecraft and Rockets* 25, 2 (1988), 109–116.

Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. 2015. Elastic textures for additive fabrication. *ACM Trans. Graph.* 34, 4 (2015), 135–1.

George Stiny. 2006. *Shape: talking about seeing and doing.* MIt Press.

Xiaodong Tang and John D Whitcomb. 2003. General techniques for exploiting periodicity and symmetries in micromechanics analysis of textile composites. *Journal of composite materials* 37, 13 (2003), 1167–1189.

Zhijuan Tian, Yongjian Liu, Lei Jiang, Weiqing Zhu, and Yinping Ma. 2019. A review on application of composite truss bridges composed of hollow structural section members. *Journal of Traffic and Transportation Engineering (English Edition)* 6, 1 (2019), 94–108.

Jun Yan, Gengdong Cheng, Shutian Liu, and Ling Liu. 2006. Comparison of prediction on effective elastic property and shape optimization of truss material with periodic microstructure. *International Journal of Mechanical Sciences* 48, 4 (2006), 400–413.

Frank W Zok, Ryan M Latture, and Matthew R Begley. 2016. Periodic truss structures. *Journal of the Mechanics and Physics of Solids* 96 (2016), 184–203.