

Inverse Discrete Design

by

Amira Abdel-Rahman

B.S., The American University in Cairo (2014)

M.Des, Harvard University (2017)

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN MEDIA ARTS AND SCIENCES

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2025

© 2025 Amira Abdel-Rahman. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Amira Abdel-Rahman
Program in Media Arts and Sciences
August 22, 2025

Certified by: Prof. Neil Gershenfeld
Director, MIT Center for Bits and Atoms, Thesis Supervisor

Accepted by: Prof. Joe Paradiso
Academic Head
Program in Media Arts and Sciences

Inverse Discrete Design

by

Amira Abdel-Rahman

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning,
on August 22, 2025 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN MEDIA ARTS AND SCIENCES

ABSTRACT

Historically, engineering and manufacturing have been dominated by analog processes—manual, continuous, and heavily reliant on human intuition and trial-and-error. Digital technologies have revolutionized communication and computing, enabling unprecedented scalability, precision, and complexity, yet our methods for designing and fabricating physical structures remain largely analog, fragmented across disciplines, and resistant to integration, significantly limiting innovation across complex, multidisciplinary domains. We need a parallel digital revolution in the physical realm that implements principles of discretization, modularity, hierarchy, and error correction to create new possibilities for physical design and fabrication. By applying these principles to physical structures, we can enable precise placement of functional materials with embedded electrical and mechanical properties. These ‘Digital Material’ structures bring digital programmability to the physical realm, mirroring the transformative impact of digital technologies in other fields.

This transformation requires new design methodologies to faithfully model our complex environment and enable us to design a new world where the physical and digital become indistinguishable. My thesis introduces a fully declarative and inverse workflow for designing and building scalable Digital Material systems. These workflows allow users to model and design systems that span scales (micro, meso, macro) and disciplines (electrical, mechanical, aerospace, architectural engineering) without being experts in all—or any—of these fields. The introduced workflow uses domain knowledge as priors and universal design representations across all stages of design, simulation, optimization, fabrication, and control.

Leveraging the discrete and hierarchical nature of the Digital Material system, I first introduce a multi-scale multi-physics simulation tool for mechanical metamaterial. I then present target-based multi-scale optimization workflows for the design of the geometry as well as the growth of cellular structures and soft robots. Next, I introduce advanced path planning algorithms—swarm, recursive, and hierarchical—for scalable robotic swarm assembly; as well shape-control optimization for the reconfiguration of these robots. The work extends to introduce a comprehensive design tool to design and build intelligence and physical computing, and culminates in an end-to-end inverse design workflow of electromechanical structures, that transforms text specifications into physical designs. The workflow is used to design a plethora of static and dynamic structures, ranging from bridges and shelters to aerospace structures, robots, and electronics.

Thesis supervisor: Prof. Neil Gershenfeld

Title: Director, MIT Center for Bits and Atoms

INVERSE DISCRETE DESIGN

by
Amira Abdel-Rahman

This dissertation has been reviewed and approved by the following committee members:

THESIS SUPERVISOR

Prof. Neil Gershenfeld
*Director, The Center for Bits and Atoms
The Massachusetts Institute of Technology*

THESIS READERS

Prof. Katia Bertoldi
*William and Ami Kuan Danoff Professor of Applied Mechanics
Harvard University*

Prof. Kirstin Petersen
*Associate Professor, Department of Electrical and Computer Engineering
Cornell University*

*To my parents,
with deepest gratitude for your unwavering love, support, and sacrifices,
which made this work possible.*

Acknowledgments

First and foremost, I am deeply grateful to my committee members for their guidance and support throughout my doctoral studies.

To Prof. Neil Gershenfeld, thank you for your never-ending support, for always pushing me to be a better researcher and scientist, and for reminding me to have fun with my projects. I will carry the lessons I learned from you for life. Thank you so much.

To Prof. Katia Bertoldi, your class was one of the most inspiring and well-prepared experiences of my PhD. I had followed your work for years, even before starting my graduate studies, and your research has been a constant source of inspiration. I feel very fortunate to have had you on my committee.

To Prof. Kirstin Petersen, thank you for your encouragement throughout the years—at conferences, during my qualifying exam, and in countless conversations. You have always been generous with your time and feedback, and I deeply admire your example as a scientist and mentor. I am especially grateful for your recommendation during my faculty applications, which played an important role in my next step at Cornell. I very much look forward to being your colleague, while continuing to value you as a mentor.

I am grateful for the funding and collaborations that made this work possible, including DARPA, ARL, NVIDIA, and Dassault Systèmes, as well as the colleagues I had the pleasure to work with: Prof. Aaron Becker, Andy Spielberg, Christopher Camron, Faraz Farouqi, Christine Gregg, Kenny Cheung, Gavriel State, and many others.

To my MIT CBA family: Kara Pendlebury, Joe Murphy, Candace Rosary, Sherry Lassiter and CBA labmates: Miana, Ben, Alfonso, David, Jake, Erik, Camron, Vlasta, Patricia, Kat, Zach, Alan, Quentin, Jack, Jiri, and so many more. Thank you for creating such a unique, inspiring environment. Special thanks to Miana, Ben, and Alfonso your collaboration, support, and friendship. I learned so much from each of you, and our time working together has deeply shaped my research and made my experience at CBA especially meaningful.

I was also fortunate to cross paths with my Harvard crew: Elnaz and Nada, and I feel lucky to have had you in my life since those days.

To Sally, thank you for being one of my constants over the last six years; your support has meant more than I can say.

To Sarah, my sister in every way that matters, I could not imagine these last fifteen years without you. From recklessly climbing mountains to diving deep in the sea, you've shared every adventure and challenge with me. You are also the best aunt Layla could ask for, and I can't wait for all that lies ahead.

I am grateful for my extended family, especially to my aunt Manal, a source of inspiration and a second mother, and to Malik's family, who truly became a second family to me.

To Tarek, thank you for always making me laugh, for paving the way for my success, and for giving me the best gifts, Hoda and Karim.

To my parents, none of this would have been possible without you.

Mama, thank you for always putting us first, for your prayers, your support, and for being my friend before being my mother. Words will never be enough to express how grateful I am.

Daddy, thank you for being the best role model I could have asked for. Your discipline, creativity, and dedication to your work have been my greatest inspiration.

To Malik, thank you for being the best thing that happened to me at MIT. Every corner of campus holds a memory with you, and you made this experience uniquely special. Thank you for your constant support throughout this journey. I am so excited for the chapters ahead.

Finally, and most importantly, to my daughter Layla. God knows how much we waited for you. You are the greatest blessing of my life, shifted my priorities to better ones, and made everything more meaningful. Now that this chapter is done, I cannot wait to devote more of myself to being the mother you deserve.

Contents

<i>List of Figures</i>	15
<i>List of Tables</i>	19
1 Introduction	21
1.1 Problem Statement	21
1.2 Motivation	23
1.3 Background and Related Work	24
1.4 Digital Material	24
1.5 Digital Design Tools	24
1.5.1 Design Representation	25
1.5.2 Inverse Design	25
1.5.3 End-to-End Design Integration	26
1.6 Thesis Structure	27
2 Simulation: MetaVoxels Simulation Tool	29
2.1 Introduction	29
2.2 Structural Mechanics Model	30
2.2.1 Euler-Bernoulli Beam Model	30
2.2.2 Beam Model	31
2.2.3 Dynamic Parallel Model	32
2.2.4 Integration	33
2.2.5 Damping	33
2.2.6 Timestep calculation	34
2.2.7 Collision and Friction	34
2.3 Benchmark	34
2.4 Validation and Convergence	35
2.5 Hierarchical Modeling	36
2.6 Multi-Scale Hierarchical Modeling	37
2.6.1 Case Study: Morphing Robotic Lattices Modeling	38
2.7 Fluid-Solid Interaction	39
2.7.1 Case Study: MADCAT	39
3 Inverse Design of Macro-scale Discrete Mechanical Metamaterial	43
3.1 Introduction	43
3.2 Results	44

3.2.1	Inverse Design Workflow	44
3.2.2	Bottom-Up Optimization	46
3.2.3	Top-Down Optimization	47
3.2.4	Fabrication and Testing	52
3.3	Discussion	55
4	Learning Morphogenesis: Differentiable Physical Growth	57
4.1	Introduction	57
4.2	Results	58
4.2.1	NCA Architecture	58
4.2.2	Static Structures	59
4.2.3	Dynamic Structures	64
4.3	Recap	68
5	Self-Replicating Hierarchical Modular Robotic Swarms	71
5.1	Introduction	71
5.2	Results	73
5.2.1	Scaling Robotic Construction	73
5.2.2	Material-Robot System	76
5.2.3	Robot Self-Replication and Hierarchy	77
5.2.4	Inverse Problem: From Structure to Swarm	80
5.3	Methods	88
5.3.1	Hardware Fabrication and Components	88
5.3.2	Voxel Manufacturing	88
5.3.3	Simulation Environment	88
5.4	Discussion	90
5.5	Recap	91
6	Shape-Control Optimization of Reconfigurable Digital Material Robots	93
6.1	Introduction	93
6.2	Results	94
6.2.1	Modular Robot Design Tool	94
6.2.2	Shape-Control Optimization	96
6.3	Recap	100
7	Design Integration: Physical Computing Design Tool	101
7.1	Introduction	101
7.2	Results	102
7.2.1	Voxel Invention Kit	103
7.3	Co-Design of Mechanical and Electrical Components	104
8	Case Study: How can Anyone Design and Build a Pavilion in One Day?	105
8.1	Overview	105
8.2	Design: Text to Viable Design	105
8.3	Digital Material as a Sustainable Alternative in AEC Industry	107

8.4	Microstructure Optimization	109
8.5	Swarm Robotic Construction	109
8.6	Design of Reconfiguration	110
8.7	Recap	113
9	Conclusion	115
9.1	Summary of Contributions	115
9.2	Open-Source Tools	116
9.3	Impact	116
9.4	Future Directions	116
	<i>References</i>	119

List of Figures

1.1	Traditional Design Workflows.	22
1.2	Inspirations for the Physical Embodiment and Digital Design Workflows. . .	23
1.3	Thesis Structure.	27
2.1	MetaVoxels Simulation Tool Overview.	29
2.2	FEA vs MetaVoxels (GPU and CPU) Performance Comparative Study . . .	35
2.3	FEA vs MetaVoxels Validation for Linear Static Analysis	35
2.4	Final Node Displacement for the 7 by 7 by 7 Lattice - FEA vs Parallel Dynamic Solver	36
2.5	Simulation Examples.	37
2.6	Multiscale Hierarchical Modeling	37
2.7	Comparative Study between Oriented, Hierarchical, and Multiscale Particle Simulation (Cantilever)	38
2.8	Morphing Robotic Lattices - MetaVoxels Hierarchical Modeling vs Robots .	39
2.9	MADCAT Representations	39
2.10	MADCAT Comparison	40
2.11	Multiphysics (Fluid Structure Interaction) Workflow	40
2.12	MADCAT Simulation - Exaggerated Loads - MetaVoxels and LBM Simulation	41
3.1	Inverse Design Workflow Overview.	45
3.2	Bottom-Up Optimization.	46
3.3	Top-Down 2D Optimization.	48
3.4	Top-Down 3D Optimization.	50
3.5	3D Top-Down Optimization from 2D Pieces.	51
3.6	2D Material Library.	52
3.7	Effect of Discretization.	53
3.8	Fabrication and Testing.	54
4.1	Neural Cellular Automata (NCA) Overview.	59
4.2	Synchronous vs Asynchronous NCA Growth.	60
4.3	Examples of Different Boundary Conditions.	61
4.4	NCA Training Progress Example.	62
4.5	Extension to Different Physics and Dimensions.	62
4.6	NCA Stability and Resilience.	63
4.7	Hierarchical Growth with Multiple Microstructures.	64

4.8	Generalization to Changing Boundary Conditions.	65
4.9	Design of Targeted Growth.	66
4.10	Soft Robot NCA Reconfiguration Workflow.	67
4.11	Soft Robot Mass Change.	67
4.12	Soft Robot Reconfiguration.	68
4.13	Generalization: Learning Walking Gait.	68
5.1	Relative Robot: Material-Robot System.	72
5.2	Swarm Assembly.	72
5.3	Swarm Assembly Scaling and Limitations.	73
5.4	Comparative Analysis of Different Robotic Assembly Strategies vs Nature.	74
5.5	Detailed Assumptions for Comparative Cube Construction Study.	75
5.6	Modular Building Blocks and Assembler.	77
5.7	Voxel manufacturing.	78
5.8	Actuator Components.	78
5.9	Robot Self-replication and Hierarchy.	80
5.10	Hierarchical Robots Design Space.	81
5.11	Adaptive Shape Compiler.	83
5.12	Robot Decision Tree.	84
5.13	Specific Speedup and Optimum Configuration.	85
5.14	Effects of Structure Resolution on Total Construction Time.	86
5.15	Hierarchical Construction.	87
6.1	Discrete Robotic Toolset.	94
6.2	Modular Robot Design Tool.	95
6.3	Design Tool sim2Real	95
6.4	Robot Zoo.	96
6.5	Robot Control Reinforcement Learning.	96
6.6	Robot Zoo and Graphs.	97
6.7	Multi-Task Robot Reconfiguration Strategy.	97
6.8	Shape-Control Hierarchical Optimization Workflow.	98
6.9	Tree Search.	99
6.10	Integrated RL and Simulation Workflow.	99
6.11	Sim2real: Control.	100
7.1	Tiny and Meso DICE Parts.	101
7.2	Physical Computing Design Tool.	102
7.3	The Voxel Invention Kit (VIK).	103
7.4	Using Physical Computing Design Tool to Design a Chair using the Voxel Invention Kit	104
7.5	Using Physical Computing Design Tool to Design the Body and Brain of Digital Material Robots	104
8.1	Discrete Inverse Design Overview.	106
8.2	Large Scale Digital Fabrication Methods Comparison.	107
8.3	Digital Material.	108

8.4	Optimized Microstructures.	109
8.5	Swarm Robotic Construction.	110
8.6	Pavilion Swarm Robotic Construction.	111
8.7	NCA Reconfiguration.	112
8.8	Ant Reconfiguration.	112

List of Tables

2.1	Comparison between SOA Mechanical Metamaterial (Beam Based) Simulation tools and MetaVoxels	30
2.2	MADCAT Comparison (Coarse vs Adaptive/Multiscale vs Fine)	39
5.1	Robot Material System Bill of Materials.	89

Chapter 1

Introduction

1.1 Problem Statement

Throughout human history, no innovation has progressed as rapidly as digital technologies [1]. This has profoundly transformed our physical world; with the rise of the Internet of Things, our built environment is becoming smarter, more dynamic, and increasingly complex. Every year, the line between physical and digital becomes more blurred. However, if we look at the history of computer-aided design (CAD), engineering (CAE), and manufacturing (CAM) tools, there have been very few radical advancements since the invention of Sketchpad in 1963, the very first interactive CAD system [2]. These radical advances are needed to not only faithfully model and respond to the complex environment around us; but enable us to imagine and design a new world where the physical and digital are indistinguishable. An indispensable requirement for this new generation of design tools is the ability to model and design systems that span (1) scales (nano, micro, meso, macro) and (2) disciplines (electrical, mechanical, aerospace, architectural engineering) (3) without the user being an expert in all (or any) of these fields.

If we look at current design workflows, even though there exist integrated design, simulation, and fabrication workflows that span across different disciplines (see Figure 1.1); these tools have a lot of limitations. First, these linear workflows rely heavily on domain knowledge and expertise and require that the user knows exactly what they want to design before they use the tools. Moreover, CAD, CAE, and CAM tools are developed independently, each with independent geometrical representations and solvers, which means there is no way to propagate errors forward or backward, which makes it harder to visualize the impact of a design decision on the final product being built. That's why most multi-disciplinary design optimization workflows are slow and inefficient. Finally, even in more advanced, integrated CAD/CAE/CAM tools, feedback cannot be returned to the system to improve the design after the object has been built.

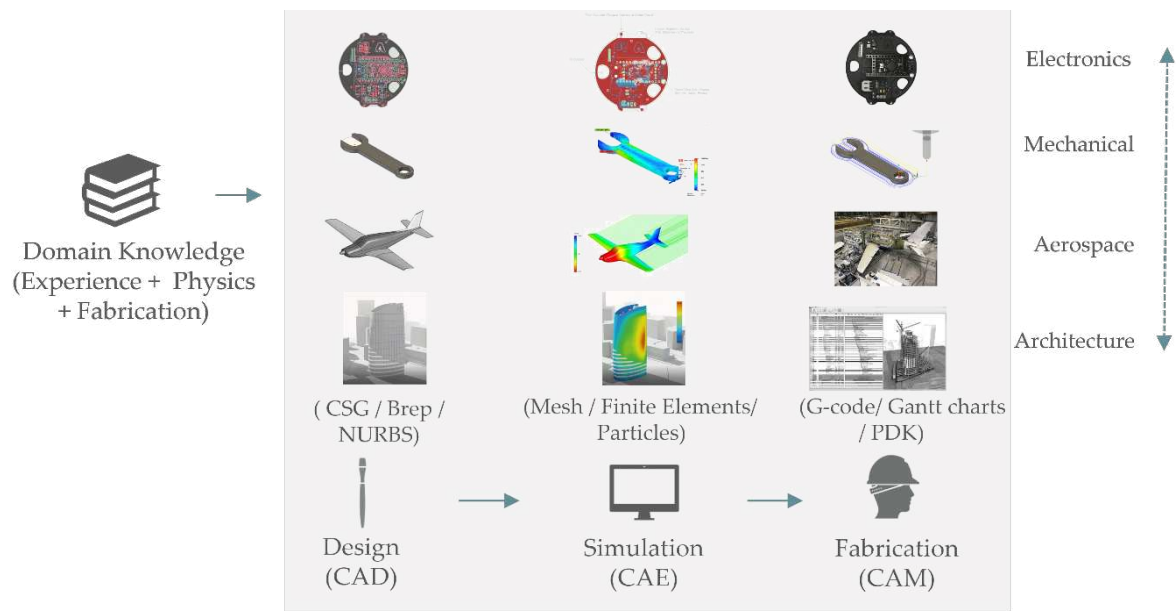


Figure 1.1: Traditional Design Workflows.

1.2 Motivation

A main driving question for my research is ‘**How to democratize design workflows to span across scales and disciplines?**’ To achieve that, we need (1) novel physical and (2) digital tools that can handle complexity and grow reliably.

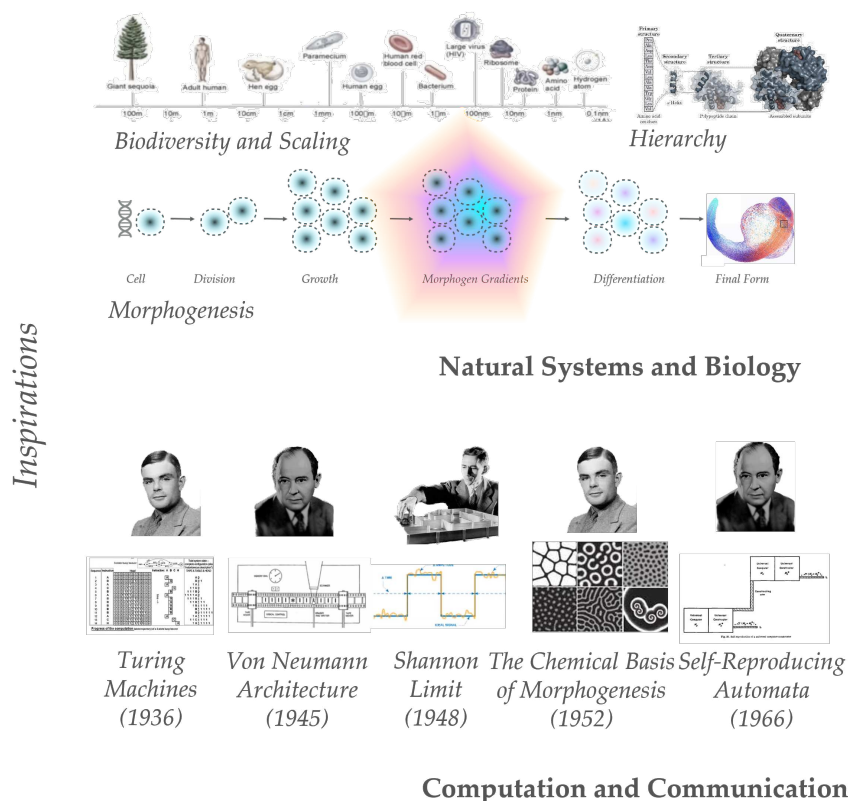


Figure 1.2: Inspirations for the Physical Embodiment and Digital Design Workflows.

The first question is **what inspiration could we draw for the physical embodiment?** The first thing that comes to mind as the most scalable and diverse ecosystem we know is biology. In biological growth and evolution, a dynamic range of structures can be found: primary (protein sequence), secondary (helix), tertiary (chains), to quaternary (assemblies). In the process of morphogenesis, a cell doesn't contain the final blueprint of what it is going to build, it just contains and follows simple instructions on how to build it. It does that reliably because of principles like *discretization*, *distributed computation*, *hierarchy*, and *error correction*.

The second question is: **how do we then create scalable digital tools to model this process?** If we look through history, Turing, Von Neumann, and Shannon, used these same principles to create scalable and reliable communication and computation workflows. At the end of their careers, Turing and Von Neumann studied morphogenesis and self-reproducing automata as a basis for a new direction in computing.

1.3 Background and Related Work

1.4 Digital Material

The digitization of analog signals has revolutionized the communication and computation industry, but there is still lots of potential for the same principles to be applied to the physical world. In my thesis, I focus on the inverse design of Digital Material systems. These are metamaterial structures made of discrete reconfigurable heterogeneous cells, where one controls the global behavior of the structures by changing the properties (geometry, material, capabilities) of the cells. Most importantly, they follow digital scaling principles (modularity, hierarchy, parallelism, and error correction). In these systems errors can be incrementally detected and corrected; as the code of construction is embedded in the individual building block rather than relying on the complexity of the builder. That means that an unreliable simple builder can build a scalable deterministic structure quite precisely, in contrast to state-of-the-art additive and subtractive processes where the precision of the fabricated parts is constricted by the accuracy and complexity systems. Therefore, the cost of developing machines that can build large structures while maintaining a high resolution grows exponentially [3].

Using digital material principles, researchers have developed macro-scale, ultralight lattice materials with record-setting mechanical properties [4] and mass-producibility [5], as well as heterogeneous mechanical [6] and robotic properties [7]. Applications include structural systems for morphing wings [8] and ultralight aerostructures [9], statically reconfigurable infrastructure [10], and designs for large-scale space structures [11].

Even though digital design tools were used to design and model the individual components, none of the presented larger digital material systems were designed using digital design tools. These were all designed by domain experts using slow forward iterative design workflows (mainly through fabrication trial and error) because of limitations in available digital design workflows that could not handle, visualize, or simulate systems that are made of hundreds (or thousands) of components.

Given a fine enough resolution, recreating conventionally manufactured objects using digital materials is a feasible task. This then unlocks the ability to place any functional material at any location making the new design space exponentially larger [12]. Therefore, there is a need for the introduction of discrete design workflows that are capable of modeling, optimizing, and designing new interdisciplinary digital material systems.

1.5 Digital Design Tools

To develop universal workflows for the design of digital material systems that span across scales and disciplines, there is a need for novel (1) **design representations** that help with (2) **end-to-end design integration** across different disciplines and stages of the design, as well as scalable approaches to (3) **automate the design discovery** to find innovated - yet optimum solutions - based on functional specifications.

1.5.1 Design Representation

The choice of the correct representation and parameterization of design space is one of the most crucial steps in approaching design. “This view can be extended to all problem-solving — solving a problem simply means representing it so as to make the solution transparent” [13]. In their review paper, Bermano et al. [14] present a comparative analysis of fabrication-based design representations in terms of conciseness, expressiveness, differentiability, and easiness of use with minimal post-processing for subsequent processes such as physics simulation and manufacturing processes (mainly additive manufacturing). The conclusion was that there wasn’t a “dominant representation” that performed best in all criteria, however, there usually is one for a specific application and fabrication process in mind. In my thesis, I will use the same criteria to choose and evaluate the design representations that work best for digital material design and manufacturing.

1.5.2 Inverse Design

The automatic discovery of innovative design alternatives given high-level functional requirements has been a long-standing goal in engineering design. In contrast to the traditional imperative design process, where designers heavily rely on their expertise to suggest new design solutions in a linear tedious process, in declarative design workflows, the designer only focuses on architecting the functional requirements. Achieving fully declarative design workflows is an indispensable step in democratizing interdisciplinary engineering design for the masses.

It could be argued that one of the first examples of the use of AI for automatic design synthesis is the Westinghouse Electric Corporation programs that were used to design electric motors, generators, and transformers in 1956 [15]. Then in Herbert Simon’s 1969 paper “The Science of Design: Creating the Artificial” he introduced foundational principles of how to approach inverse design and underscored the importance of choosing the correct design representation (in particular using hierarchy in the design process to reduce the complexity), use of resource efficiency and parallelism when evaluating design alternatives, and use of state-of-the-art search methods and statistical decision theory to evaluate and choose design alternatives [13].

Recently, there has been an emergence of different inverse design or generative design approaches. Topology optimization is one of the main generative design approaches used across a variety of engineering disciplines. Cagan et al. [16], [17] introduced “a framework for computational design synthesis” that was used for the design of soft robots morphology and controls using evolutionary search [18], [19]. Data-driven machine learning has been widely incorporated in different ways in the inverse design process, either in a supervised way to create surrogate models for expensive design evaluation processes which are cheaper to evaluate and differentiable; or in an unsupervised way to create generative models. These are however data-hungry methods that are usually not generalizable and do not provide a guarantee either for the correctness of the surrogate model or for the optimality of the design alternatives created by the generative design process.

Physics-informed machine learning on the other hand offers a better alternative than data-driven machine learning approaches as it solves both problems. The emergence of

differentiable programming tools helped with the ability to create differentiable simulation engines that could be incorporated into the design search processes. The ability to use the generated gradient of the cost function against the design parameters makes the search process less random hence more efficient.

1.5.3 End-to-End Design Integration

Most of the inverse design or automated design workflows discussed earlier were used for specialized engineering domains (i.e., material science, aerospace, architecture, mechanical, electrical engineering) and usually focus on one evaluation criteria/objective (usually functional performance at one moment in time) while ignoring other important parameters like fabricability, and other life-cycle factors like control, reconfiguration, and maintenance. There are only a very few examples in the literature of end-to-end tools that can automatically generate interdisciplinary designs from high-level functional requirements, and these are mainly (if not exclusively) for robotics design and manufacturing [20], [21], [22], [23], [24] and are not extendable for other applications.

Two examples of end-to-end workflows for digital material design were presented in Hiller’s Ph.D. thesis [12] and Ghasseini’s Master’s thesis [25]. Hiller presented a novel workflow for digital material multi-material “printing” as well as simulation and optimization workflows using his tool VoxCad [12]. He only focused on digital material on the microscale with base units that have limited capabilities. Ghasseini introduced DMDesign a CAD, simulation, and CAM workflow for digital material design [25]. The tool used hierarchical abstractions to represent different types of electro-mechanical digital material components. In my thesis, I plan to build on and extend these two bodies of work to develop end-to-end (CAD, CAE, CAM, and control) **inverse** design workflows (starting with target functional specifications) with a focus on **scalability** (abilities to handle thousands/millions of components). The goal is for these tools to be used for applications on different scales and disciplines.

1.6 Thesis Structure

In my thesis, I investigate two critical questions: (1) How can physical design be made as programmable, modular, and scalable as digital code? and (2) How can discrete, modular design principles unlock scalable, cross-disciplinary design workflows for engineering complex systems?

Analogous to how the transition from analog to digital logic enabled the exponential scaling of advancements in digital technologies, in my thesis I will introduce the concept of **discrete design**, a design paradigm that uses principles of (1) *discretization*, (2) *distributed computing*, (3) *hierarchy*, and (4) *error correction* to develop *scalable, integrated* workflows for *digital material* design. I created **inverse design** a workflow for these systems by using *domain knowledge as priors* and a *universal design representation* across all design stages (simulation, optimization, fabrication, and control). These tools have been used to design a plethora of structures, ranging from bridges to shelters, airplanes, robots, and electronics.

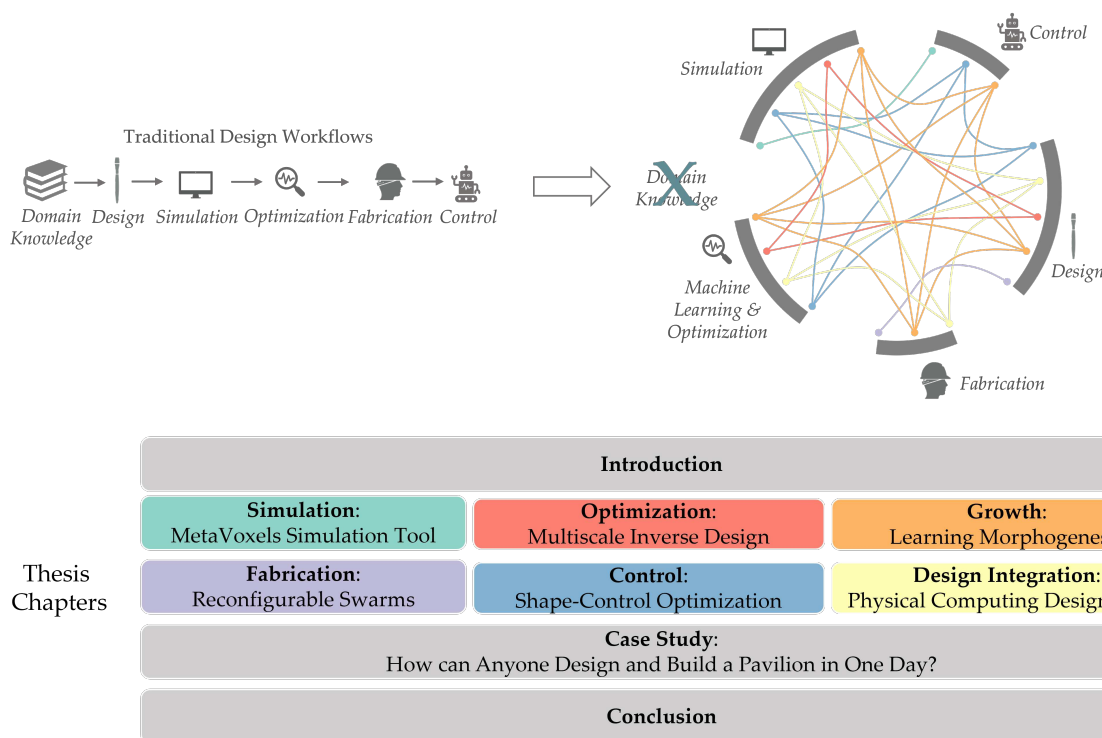


Figure 1.3: Thesis Structure.

Chapter 2

Simulation: MetaVoxels Simulation Tool

2.1 Introduction

To enable inverse design across scales and disciplines, I first address a foundational limitation: the lack of simulation tools capable of handling the scale, hierarchy, and complexity of systems composed of thousands of components. Therefore, I developed MetaVoxels [26] as a scalable tool to simulate large metamaterial structures in a fast and efficient manner. It takes advantage of the inherent modularity and hierarchy of such structures by using efficient parallelization and multiscale simulation.

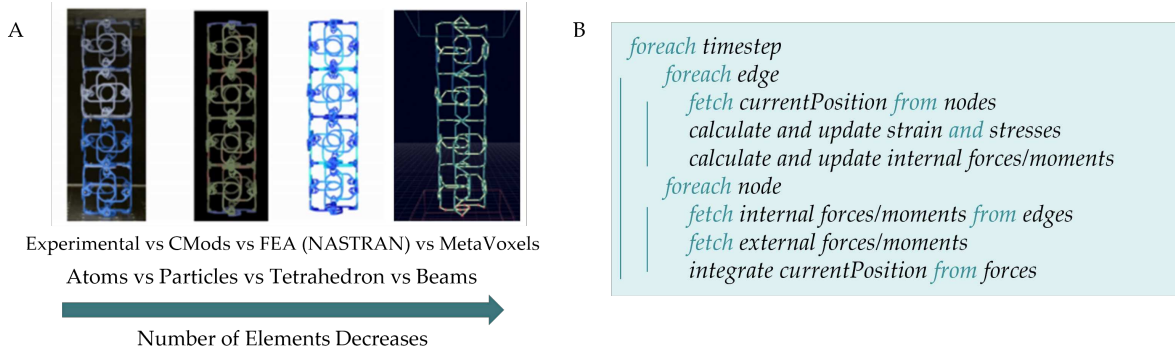


Figure 2.1: **MetaVoxels [26] Simulation Tool Overview.** (a) Discretization Comparison between different Tools [27] (b) Overall Workflow

When looking at state-of-the-art (SOA) simulation tools to simulate metamaterial and lattice structures, one will find an abundance of FEA tools to simulate individual units in detail. However, when simulating 3 by 3 by 3 units or more, it becomes infeasible to use the same detailed discretization and the same solvers that have polynomial or even exponential scaling with the problem size. Therefore, there is a need to create flexible tools with different types and levels of discretization based on the simulation scale and fidelity. Particle methods such as mass spring systems are widely used in computer graphics for dynamic simulations and simulating structures with large deformations. The structure is discretized to point masses and one can tailor the force laws that are between these masses to simulate different physical behaviors. Then, all that is needed is to solve a system of ODEs, which describes how

these masses move through time under the forces using a chosen integration scheme. These equations are much simpler than FEA and could be all solved in parallel which makes it much more computationally efficient (see Figure 2.1). However, the accuracy of the simulation is highly dependent on the force laws used and the integration scheme. MetaVoxels addressed this point by introducing different levels of abstractions for the particle system (oriented particles and hierarchical particles) with more complex interparticle forces inspired by FEA and homogenization theory. The next table shows a comparison between SOA tools that are currently used to simulate mechanical metamaterial. MetaVoxels was first developed based on VoxCAD [24] but went further by adding GPU computation, differentiability, multiscale particles, arbitrary angle beam solving, and multiphysics modeling (fluid-solid interaction).

	Frame3DD	GSA (Oasys)	Abaqus	VoxCAD	MetaVoxels
GPU	x	x	x	✓	✓
Statics	✓	✓	✓	✓	✓
Dynamics	x	x	✓	✓	✓
Non-linearity	✓	x	✓	✓	✓
Large Angles	✓	x	✓	✓	✓
Arbitrary Angles	x	✓	✓	✓	✓
Multi-scale	x	x	x	✓	✓
Multi-physics	x	x	x	x	✓
Differentiability	x	x	x	x	✓

Table 2.1: Comparison between SOA Mechanical Metamaterial (Beam Based) Simulation tools and MetaVoxels

2.2 Structural Mechanics Model

2.2.1 Euler-Bernoulli Beam Model

MetaVoxels’ simulation uses oriented particles with 6 degrees of freedom: three rotational degrees of freedom in addition to the traditional three translational degrees. I then used Bernoulli-Euler beam theory to prescribe the biaxial bending, transverse shear, and axial stretching that is exerted on each beam [28]. At each timestep, the code goes through all nodes and calculates the forces exerted on them from the external forces applied to the structures, as well as the internal forces that are exerted from the stresses in the attached beams. It then integrates through time to get the new nodes’ position/rotations using a double Euler integration scheme. Finally, it goes through all the beams and calculates the new internal forces caused by the nodes’ displacements/rotations. After enough time steps, all the forces propagate through the entire structure, and the solution converges to the analytical solution.

2.2.2 Beam Model

$$\begin{bmatrix} F_n \\ M_n \end{bmatrix} = [K] \begin{bmatrix} D_n \\ \theta_n \end{bmatrix} \quad (2.1)$$

Each element (beam) has 12 degrees of freedom:

$$\begin{bmatrix} F_{x_1} \\ F_{y_1} \\ F_{z_1} \\ M_{x_1} \\ M_{y_1} \\ M_{z_1} \\ F_{x_2} \\ F_{y_2} \\ F_{z_2} \\ M_{x_2} \\ M_{y_2} \\ M_{z_2} \end{bmatrix} = [K] \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ \theta_{x_1} \\ \theta_{y_1} \\ \theta_{z_1} \\ X_2 \\ Y_2 \\ Z_2 \\ \theta_{x_2} \\ \theta_{y_2} \\ \theta_{z_2} \end{bmatrix} \quad (2.2)$$

$$[K] = \begin{vmatrix} a_1 & 0 & 0 & 0 & 0 & 0 & -a_1 & 0 & 0 & 0 & 0 & 0 \\ & b_1 & 0 & 0 & 0 & b_2 & 0 & -b_1 & 0 & 0 & 0 & b_2 \\ & & b_1 & 0 & -b_2 & 0 & 0 & 0 & -b_1 & 0 & -b_2 & 0 \\ & & & a_2 & 0 & 0 & 0 & 0 & 0 & -a_2 & 0 & 0 \\ & & & & 2b_3 & 0 & 0 & 0 & b_2 & 0 & b_3 & 0 \\ & & & & & 2b_3 & 0 & -b_2 & 0 & 0 & 0 & b_3 \\ & & & & & & a_1 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & b_1 & 0 & 0 & 0 & -b_2 \\ & & & & & & & & b_1 & 0 & b_2 & 0 \\ & & & & & & & & & a_2 & 0 & 0 \\ & & & & & & & & & & 2b_3 & 0 \\ & & & & & & & & & & & 2b_3 \end{vmatrix} \quad (2.3)$$

(sym)

$$\begin{aligned} a_1 &= \frac{E_c A}{l} \\ a_2 &= \frac{G_c J}{l} \\ b_1 &= \frac{12E_c I}{l^3} \\ b_2 &= \frac{6E_c I}{l^2} \\ b_3 &= \frac{2E_c I}{l} \end{aligned} \quad (2.4)$$

$$\begin{aligned}
E_c &= \frac{2E_1E_2}{E_1 + E_2} \\
G_c &= \frac{2G_1G_2}{G_1 + G_2} \\
\mu_c &= \frac{E_c}{2G_c} - 1 \\
I &= \frac{bh^3}{12} \\
J &= \frac{bh(bb + hh)}{12} \\
G &= \frac{E}{2(1 + nu)}
\end{aligned} \tag{2.5}$$

where K is the stiffness matrix, I is the bending moment of inertia, J is the polar moment of inertia, E_c is the composite stiffness of a bond between two dissimilar materials, G_c is the composite shear modulus, μ_c is the composite Poisson's ratio and l is the beam length.

2.2.3 Dynamic Parallel Model

Based on [12] and [29], at each time step, in order to simplify the computation, transformations are computed and applied to align each element from its original orientation to face the positive X direction, effectively translating the position and angle of the first voxel to zero. Considering that the first node of each element to be located on the primary X axis, $X_1, Y_1, Z_1, \theta_{x_1}, \theta_{y_1}, \theta_{z_1}$ will all be zeros and the transformation into the X-direction stiffness matrix is a trivial calculation. I then transform $X_2, Y_2, Z_2, \theta_{x_2}, \theta_{y_2}, \theta_{z_2}$ to the original rotation matrix. Therefore, the calculation of the larger matrix is reduced to the following equations:

$$\begin{aligned}
F_{x_1} &= -a_1X_2 \\
F_{y_1} &= -b_1Y_2 + b_2\theta_{z_2} \\
F_{z_1} &= -b_1Z_2 + b_2\theta_{y_2} \\
M_{x_1} &= -a_2\theta_{x_2} \\
M_{y_1} &= b_2Z_2 + b_3\theta_{y_2} \\
M_{z_1} &= -b_2Y_2 + b_3\theta_{z_2} \\
F_{x_2} &= -F_{x_1} \\
F_{y_2} &= -F_{y_1} \\
F_{z_2} &= -F_{z_1} \\
M_{x_2} &= a_2\theta_{x_2} \\
M_{y_2} &= b_2Z_2 + 2b_3\theta_{y_2} \\
M_{z_2} &= -b_2Y_2 + 2b_3\theta_{z_2}
\end{aligned} \tag{2.6}$$

Then the forces F_t and moments M_t on each node are calculated as follows:

$$\begin{aligned} F_t &= \sum_{b=1}^{b=n} \vec{F}_b \\ M_t &= \sum_{b=1}^{b=n} \vec{M}_b \end{aligned} \quad (2.7)$$

2.2.4 Integration

Double Euler integration is used from timestep t_n to $t_{n+1} = t + dt$:

$$\begin{aligned} \vec{P}_{t_{n+1}} &= \vec{P}_{t_n} + F_t dt \\ \vec{D}_{t_{n+1}} &= \vec{D}_{t_n} + \frac{\vec{P}_{t_{n+1}}}{m} dt \\ \vec{\phi}_{t_{n+1}} &= \vec{\phi}_{t_n} + M_t dt \\ \vec{\theta}_{t_{n+1}} &= \vec{\theta}_{t_n} + \frac{\vec{\phi}_{t_{n+1}}}{I} dt \end{aligned} \quad (2.8)$$

where $\vec{D}$ is the three-dimensional position vector, $\vec{\theta}$ is the three-dimensional rotational, $\vec{P}$ is the three dimensional linear momentum, $\vec{\phi}$ is the rotational momentum. Finally, m is the mass and I is the rotational inertia.

2.2.5 Damping

Damping should be applied at the local interactions between neighboring nodes, rather than simply adding a force proportional to each node's overall velocity, which would also suppress rigid body motion. By introducing damping at the node-to-node level, we prevent modal vibrations at the scale of individual voxels from building up [12]. For each edge, first the average position, velocity, and angular velocity are calculated. The velocity of the second voxel relative to the first:

$$\vec{V}_{2 \rightarrow 1} = (\vec{V}_2 - \vec{V}_a) + (\vec{D}_2 - \vec{D}_a) * \vec{\omega}_a \quad (2.9)$$

$\vec{V}_2$ is the velocity of the second voxel, $\vec{V}_a$ is the average velocity of two voxels, $\vec{D}_2$ is the position of the second voxel, $\vec{D}_a$ is the average position, $\vec{\omega}_a$ is the average angular velocity. For each voxel, I calculate the damping force as

$$F_d = 2\xi\sqrt{mk}V_r \quad (2.10)$$

F_d is the damping force, m is the mass attached to a edge with stiffness k and a relative velocity V_r . The damping ratio ξ is normally selected as 1, corresponding to critical damping. Angle velocities are calculated as follows:

$$M_d = 2\xi\sqrt{Ik_\phi}\omega_r \quad (2.11)$$

M_d is the damping force with rotational inertia I , with edge of stiffness k and relative angular velocity ω_r , rotational angular ratio ξ is also 1.

2.2.6 Timestep calculation

Before each simulation, the timestep dt is calculated as follows:

$$dt < \frac{1}{2\pi\omega_{0_m}} \quad (2.12)$$

ω_{0_m} is the maximum natural frequency of any bond, which is calculated as follows:

$$\omega_{0_{max}} = \sqrt{\frac{k_b}{m_m}} \quad (2.13)$$

k_b is stiffness of the bond, m_m is the minimum of either mass connected to the bond.

2.2.7 Collision and Friction

The effective normal stiffness on the floor is constrained by the maximum stiffness among connected masses. As a result, each node in contact with the floor adopts the local stiffness of the floor at that point. Although this can lead to noticeable floor penetration in some instances, overall behavior remains qualitatively accurate [12]. A Coulomb friction model is used where the node will resist motion until:

$$|F_l| > \mu_s f_n \quad (2.14)$$

F_l is the horizontal force parallel to the ground, μ_s is the coefficient of static friction between the voxel and ground, and F_n is the normal force pressing the voxel into the plane of the ground.

Similarly to [12]’s implementation, a boolean flag is used to indicate that the node is temporarily restricted from lateral movement, while still allowing motion in the direction normal to the ground - enabling it to become unweighted before shifting sideways. When the static friction threshold is exceeded during any time step, the flag is cleared, permitting the voxel to initiate movement in the appropriate lateral direction. The voxel is allowed to move in any direction, but a friction force is applied in the opposing lateral direction.

$$|F_l| = \mu_d f_n \quad (2.15)$$

μ_d is the dynamic coefficient of friction. To accurately detect when a node has stopped lateral movement, a minimum motion threshold must be defined. In order to detect a stopping node, the node is halted when:

$$V_l \leq \frac{f_n \mu_d dt}{m} \quad (2.16)$$

Collisions are also damped in the direction normal to the contact surface, using a user-defined damping ratio that ranges from zero (no damping) to one (critical damping)[12].

2.3 Benchmark

To compare the scaling performance of MetaVoxels (CPU and GPU) in comparison to linear FEA (static linear analysis), I created a benchmark test comparing the time needed to

calculate the vertical uniform load on n by n by n cuboctahedron lattice structures where $n=1,2,3,4,5,6$ (see Figure 2.2). The benchmarking was done using the GeForce GTX 1070 Ti GPU. The graph shows that even though the CPU serial code is faster for small problems, the FEA code scales exponentially while the GPU solver stays nearly constant (the only big cost is the GPU initialization).

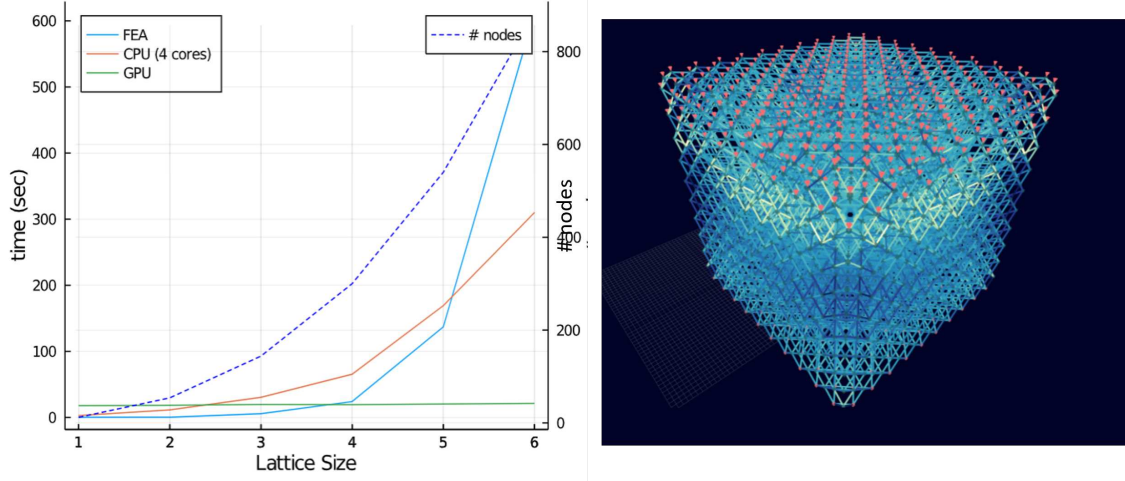


Figure 2.2: FEA vs MetaVoxels (GPU and CPU) Performance Comparative Study

2.4 Validation and Convergence

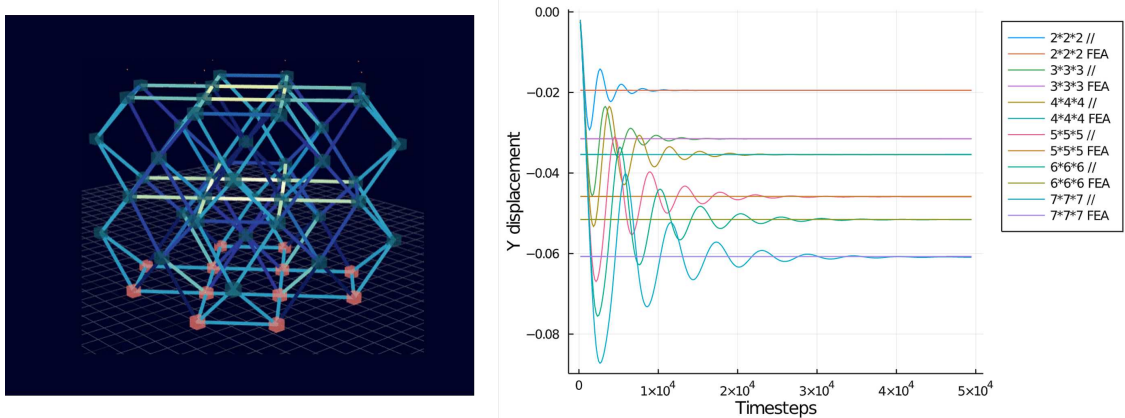


Figure 2.3: FEA vs MetaVoxels Validation for Linear Static Analysis

The same test case was used to test the convergence of MetaVoxels' dynamic solver to static FEA's solution. The graph above shows that given enough time steps (because of damping and propagation effects) the dynamic model converges to the linear FEA model solution (see Figure 2.3).

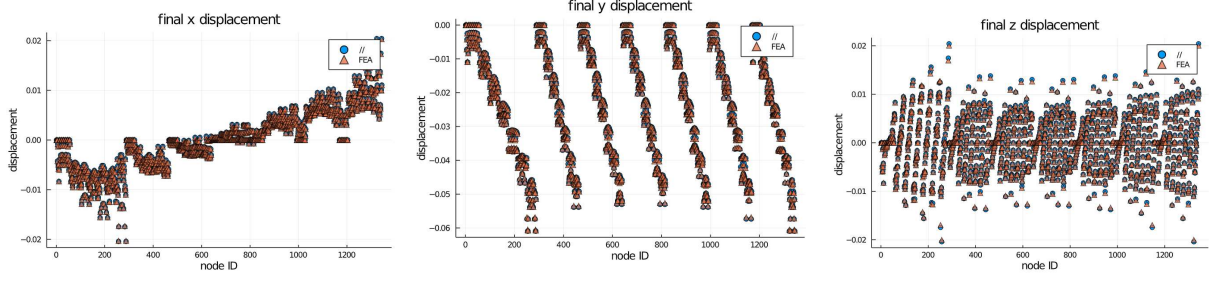


Figure 2.4: Final Node Displacement for the 7 by 7 by 7 Lattice - FEA vs Parallel Dynamic Solver

Figure 2.4 shows the final node displacement in the x , y , and z directions for the 7 by 7 by 7 lattice. One can see how accurate the simulation is for all 1400 nodes. The convergence rate is based on three physical parameters: the size (depth) of the structure, Young's modulus (the stiffer, the faster it converges), and the mass of the nodes. That's because, to ensure stability, the time step and damping factor calculation are based on the Young's modulus and node mass.

Even though many time steps are needed for the convergence, the simulation speed is still orders of magnitude faster than the corresponding FEA analysis for large structures. That is because:

1. The dynamic model does not need to assemble or invert a large stiffness matrix for the displacement calculation, which is usually the bottleneck of FEA solvers as these steps are very hard to parallelize (there are ways to parallelize them using graph coloring techniques, but they are still very inefficient for non-sparse structures).
2. All calculations are highly parallelized, and one can use the GPU for computation.
3. The bottleneck in the GPU computation is getting the arrays from the CPU to the GPU, but once it is there, the marginal cost of computing more time steps is nearly negligible.
4. Now that the dynamic model is implemented, it is very easy to incorporate nonlinear and large deformation/analysis.

2.5 Hierarchical Modeling

Building on the dynamic parallel solver, I introduce new types of hierarchical particles with increased degrees of freedom. I will explain the workflow that transitions from thousands of elements (FEA/DEM) to hundreds of elements (Beam Theory) to just tens of elements (Homogenization and hierarchy). This approach uses homogenization theory to numerically calculate the elasticity tensor from voxel units, pre-calculated from more detailed analysis.

Using these properties, for coarse simulations, we can assume that the material is on a cubic lattice, with Euler-Bernoulli beams of thickness equal to the size of the voxel. In addition to the 6 degrees of freedom, each particle has other properties like Young's modulus,

Poisson ratio, and coefficient of thermal expansion. However, global properties are still defined by local interactions (see Figure 2.5).

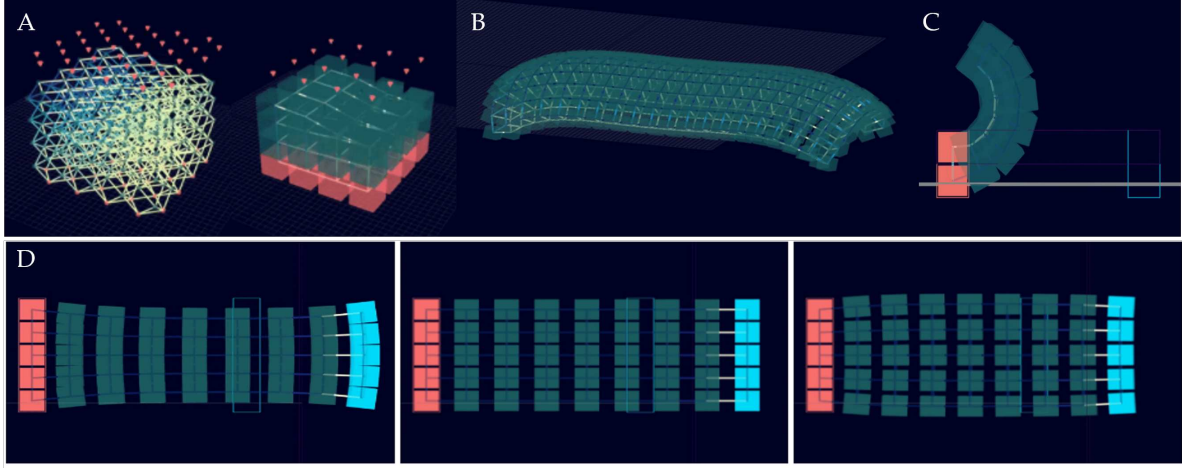


Figure 2.5: **Simulation Examples.** (A) Multi-material Detailed vs Hierarchical Simulation. (B) Thermal Expansion and Laminate Structures. (C) Tendon Simulation. (D) Applying tension to positive vs zero vs negative Poisson ratio structures.

2.6 Multi-Scale Hierarchical Modeling

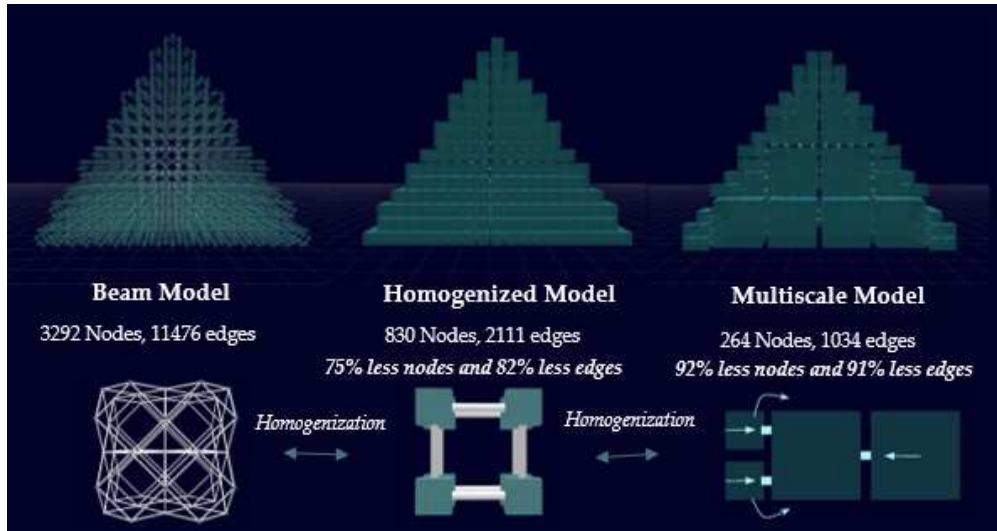


Figure 2.6: Multiscale Hierarchical Modeling

To speed up the simulation even further, multiscale particle simulation was implemented, where one can simulate particles with different sizes in the same simulation, using bigger particles in more homogeneous materials and smaller particles when more accuracy is needed. In the multiscale particle implementation, two important points had to be adjusted to the force

rules: first the non-axial forces on the nodes, and second the new hierarchical representation with nodes having parents and children (see Figure 2.6).

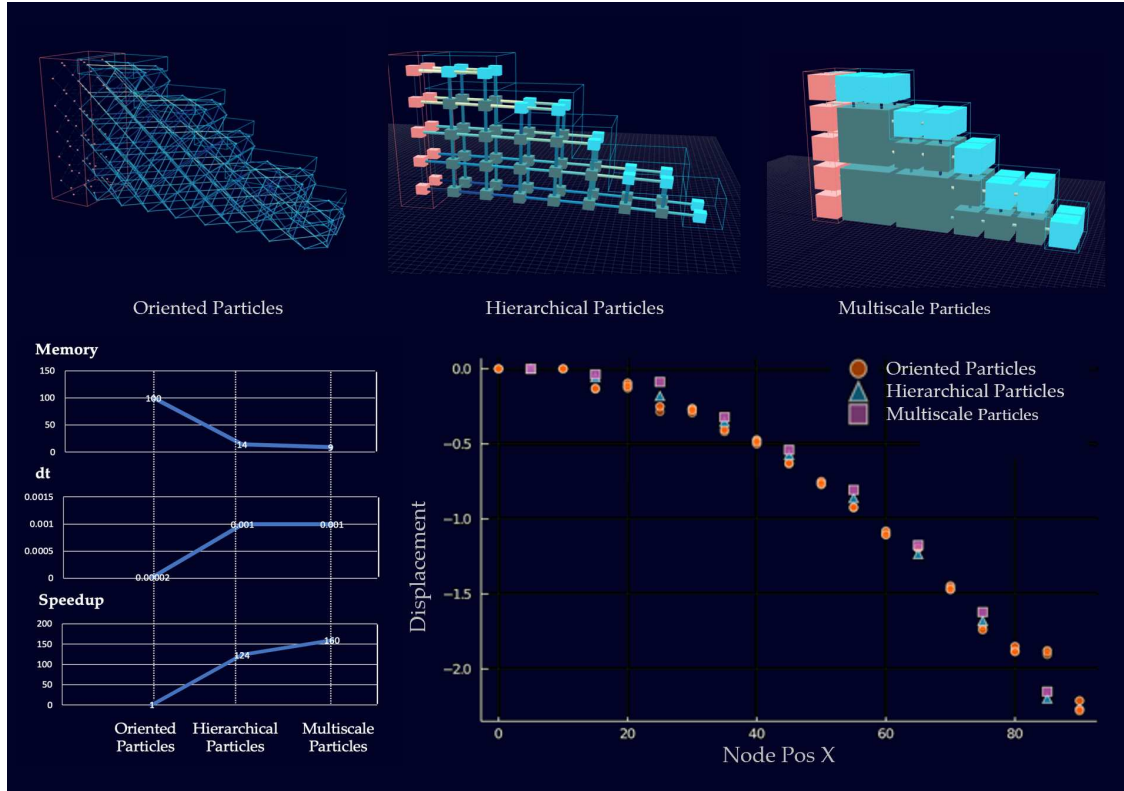


Figure 2.7: Comparative Study between Oriented, Hierarchical, and Multiscale Particle Simulation (Cantilever)

A case study (cantilever) was conducted to compare the accuracy of different hierarchical representations. An important point to note is that, in comparison to beam modeling, hierarchical modeling will take much less time to simulate, not only because it has a smaller number of nodes/edges, but also because one can take much bigger time steps. Since the timestep size is directly proportional to the particle mass and inversely proportional to the slenderness ratio of the beam, one can take much bigger time steps in hierarchical modeling while the simulation remains stable (130x speedup) with minimal loss of accuracy (see Figure 2.7).

2.6.1 Case Study: Morphing Robotic Lattices Modeling

MetaVoxels was also used as a fast prototyping tool to model the morphing behavior (twisting and bending) of discrete lattices [30]. This behavior is very hard to model using traditional implicit SOA simulation tools given the large non-linear deformation as well as how hard to integrate collision models (with the floor) in the simulation. There is qualitative agreement with the deformation of the physical experiments, but more quantitative studies are underway to tune the hierarchical particle stiffness to have a better quantitative agreement (see Figure 2.8).

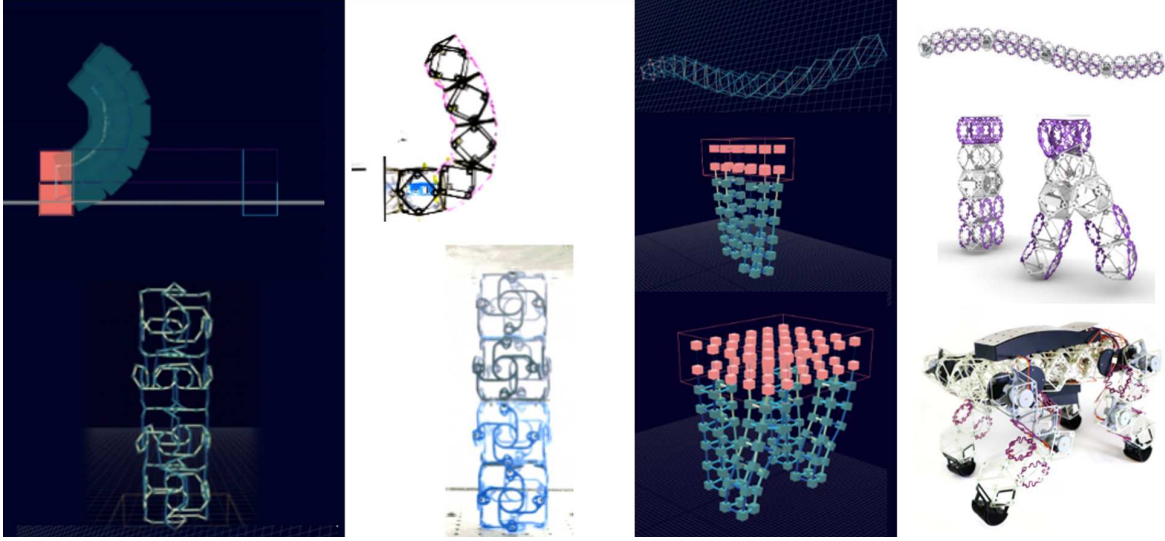


Figure 2.8: Morphing Robotic Lattices - MetaVoxels Hierarchical Modeling vs Robots

2.7 Fluid-Solid Interaction

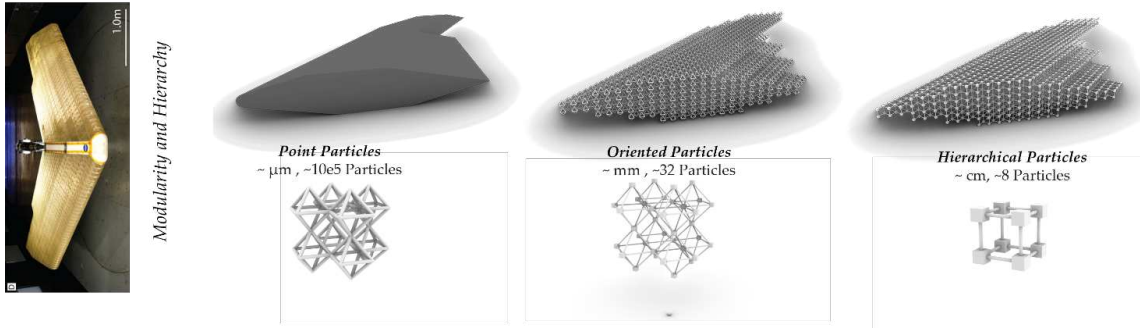


Figure 2.9: MADCAT Representations

2.7.1 Case Study: MADCAT

Finally, as a case study, I used the NASA MADCAT morphing wing [31] [9], which the team had great difficulty simulating when designing / optimizing it when using SOA tools (see Figure 2.9). Figure 2.10 and Table 2.2 show the quasi-static simulation of a uniform load using different hierarchical representations.

Coarse	Adaptive/Multiscale	Fine
345 nodes and 743 edges Time: 1.79 seconds	1520 nodes and 3953 edges Time: 8.12 seconds	2491 nodes and 6411 edges Time: 15.03 seconds

Table 2.2: MADCAT Comparison (Coarse vs Adaptive/Multiscale vs Fine)

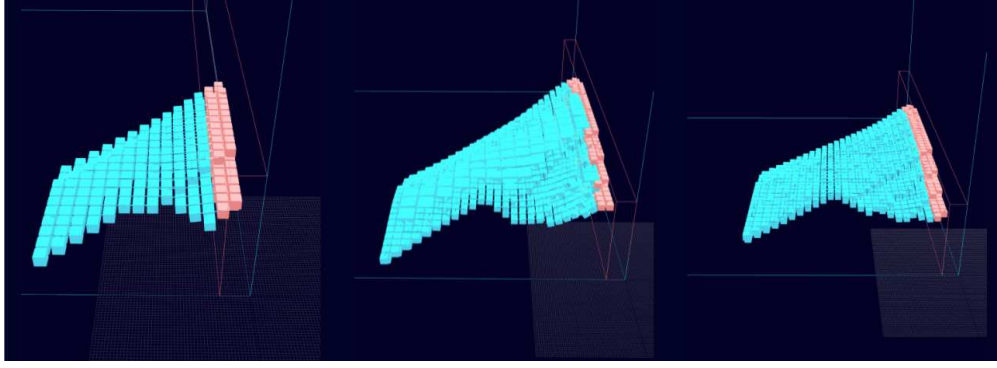


Figure 2.10: MADCAT Comparison (Coarse vs Adaptive/Multiscale vs Fine)

As a last step for the MADCAT case study, instead of just using uniform loading like the last case study, I implemented 3D Lattice Boltzmann method (LBM) to emulate the wind load on the wing. 3D LBM was implemented on the GPU and full fluid-structure interaction was implemented as explained in Figure 2.11.

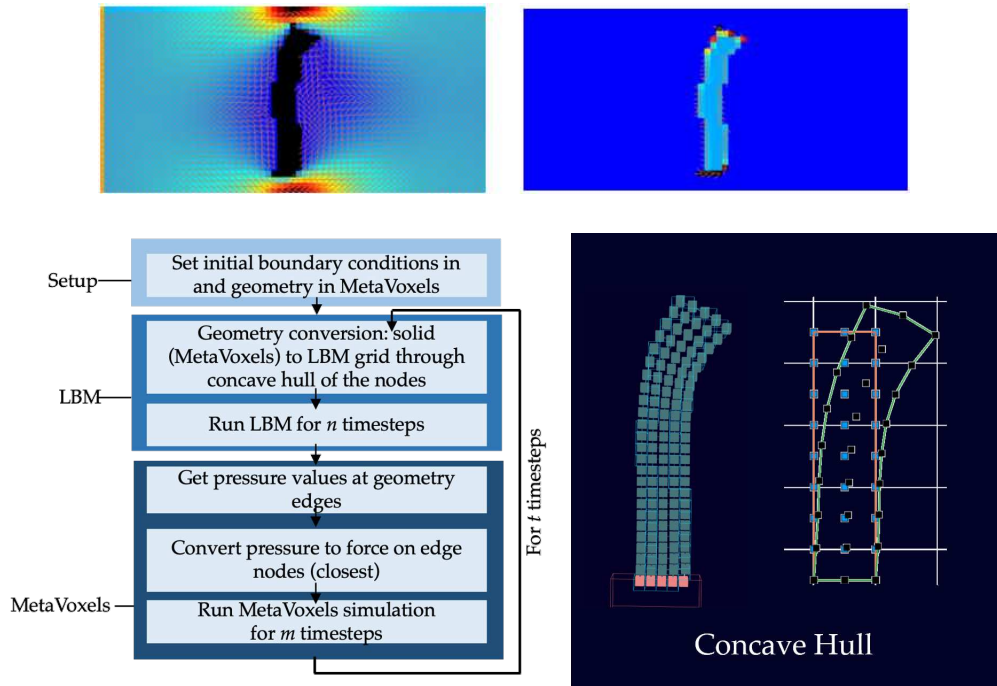


Figure 2.11: Multiphysics (Fluid-Structure Interaction) Workflow

Figure 2.12 shows an exaggerated wind load on the wing to show not only the effect of the wind load but also the effect of the wing on the fluid stream, creating a fully dynamic simulation.

This represents an initial step toward advanced fluid-structure interaction simulation. While further validation is needed, it stands as one of the few tools capable of performing full 3D fluid-structure interaction simulations at this scale in real-time or faster. This capability makes it valuable for low to medium-fidelity simulation applications.

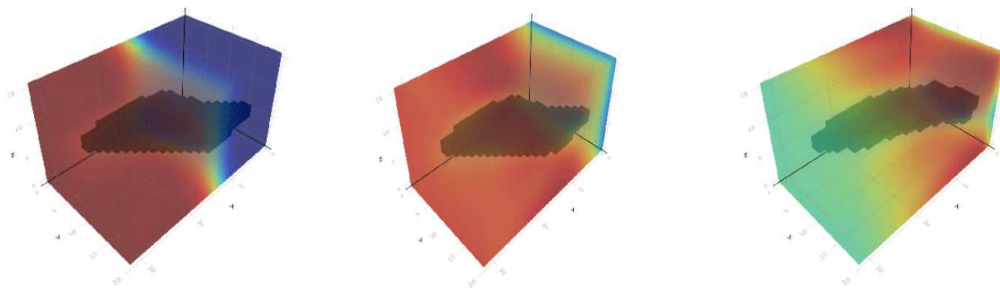


Figure 2.12: MADCAT Simulation - Exaggerated Loads - MetaVoxels and LBM Simulation

Chapter 3

Inverse Design of Macro-scale Discrete Mechanical Metamaterial

3.1 Introduction

Now that we have established how to simulate these complex digital material systems, the next step is to determine how to design and optimize them. Designing materials based on global properties has been a perennial challenge. Using metamaterials, local cellular structures could be designed to control global properties to surpass continuum material properties. Inspired by nature (wood, bones, and foams), research showed advancement in metamaterial design in many applications on many scales [32], [33], [34].

Inverse design methods including topology optimization have been used to either design the micro or macro structures of metamaterial to achieve target goals [35], [36], [37], [38]. Homogenization theory is used to translate between scales [35] [39] and different concurrent optimization methods [40] are used to hierarchically design the structures [37]. These methods are used to efficiently optimize structures in a scalable manner [41].

However, when designing such structures, it's important to include fabrication constraints in the optimization process, which is often left to the post-processing phase leading to suboptimal structures. For microstructure optimization, lots of research has been implemented to limit and cluster microstructures or create connectivity constraints to make sure they can be fabricated [42]. Then, additive manufacturing is nearly always used to fabricate the nano and micro metamaterial [34]. Even for research that suggests the use of topology optimization for large-scale mechanical metamaterial [41], additive manufacturing/3D printing is still suggested as the fabrication method, even though 3D printing is not a scalable fabrication solution for large-scale mechanical metamaterial. That is because of the grave limitations in terms of cost, build volume, throughput, and constrained/poor compatible material performance. Moreover, even if one can scale the build volume, there is a high tradeoff between scale, precision, and cost. Since large, precise machines are expensive to build, there are significant challenges to creating high-quality, low-cost macroscale (>1 m) mechanical metamaterials.

To address this problem, discrete metamaterial has been introduced as a scalable way to fabricate macroscale mechanical metamaterial. High performance detailed parts [4] are mass-produced with inexpensive efficient injection molding and 2D processes and could be

later robotically assembled in a high-throughput manner to build large-scale structures [43], [44], [45]. If the mechanical connections have been properly designed to be stronger than the local beams, the local beam properties govern the global lattice behavior, continuum behavior is achieved, and the discrete structures behave like they were produced monolithically.

In addition, the connections are designed to be reversible, which is important in case of failure, maintenance, or when designing reconfigurable structures for earth or space applications [6]. This approach has been proven to be scalable for thousands of pieces [11][9]. By changing the geometrical properties of these discrete parts, structures with different distinct global behaviors can be developed [6] and since the connection is modular, by combining different parts, one can design heterogeneous structures that have rich nonisometric properties.

However, in all of these examples, a tedious manual forward design process has been used to design either the units or the large-scale structures. Even if these structures have record-breaking properties, they are not specifically optimized for specific applications, and, especially for heterogeneous structures, there is no way to know the optimum placement for each type of microstructure.

Therefore, in this chapter, I will introduce an inverse design workflow to design discrete mechanical metamaterial. The fabrication process and its constraints are inherent in the concurrent micro-macro-optimization. I first introduce a top-down approach where I start with the macrostructure design requirements and use hierarchical clustering to derive a family of optimized 2D (for 2D processes) or 3D parts that are assembled to create an optimized heterogeneous structure. Next, I generalize the previous method by introducing a bottom-up approach where I use an inverse design workflow to create a universal library of parts that could be used in any macrostructure; along with an automated workflow for macrostructure optimization to determine where each part should be placed. A comparative experimental analysis will be presented to understand the benefits and costs of each.

3.2 Results

3.2.1 Inverse Design Workflow

To ensure that the optimized design of the mechanical metamaterial is able to meet the manufacturing and assembly constraints of the mechanical metamaterial assembled discretely, the discrete material construction system introduced in [6] - cuboctahedra voxels connected to the face (cubocts) - has been used where each voxel is discretized into 6 faces. There are two types of mechanical connections: inner voxel joints that connect the faces to assemble a voxel, and intervoxel joints that connect neighboring voxels (see Figure 3.1 A).

To guarantee an efficient search process, it is important to find the correct design representation for the micro- and macrostructure geometry. This representation should ensure a low-dimensional design space while remaining expressive enough to represent a wide range of high-resolution geometries. Neural implicit fields using Fourier projections were therefore chosen as the geometry representation micro- and macrostructure. Unlike discrete 3D geometric representations (meshes, voxels), which contain a finite amount of information about the geometry/signal, implicit neural representations parameterize a continuous differentiable signal using a neural network. These representations have been widely used for 3D scene

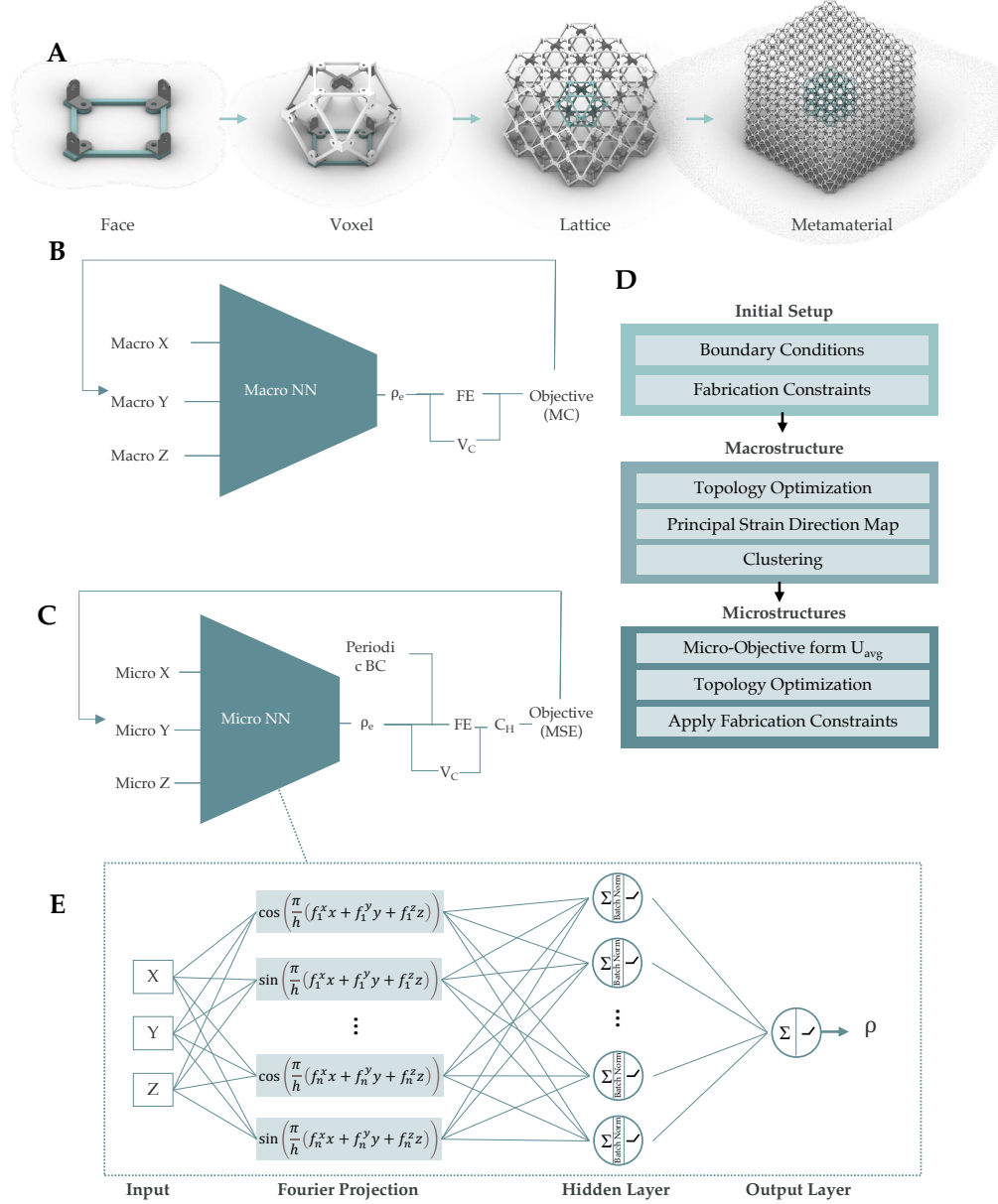


Figure 3.1: **Inverse Design Workflow Overview.** (A) Discrete Metamaterial. (B) Macrostructure Optimization Workflow. (C) Microstructure Optimization Workflow. (D) Multiscale Optimization Workflow. (E) Neural Implicit Field with Fourier Features.

reconstruction and compression [46]. For encoding high-frequency, high-resolution data, simple periodic feature transformations such as Fourier feature mappings help maintain a low-dimensional design space [47]. This approach has been applied in topology optimization for length-scale control, ensuring that optimized designs meet manufacturing and functional requirements with minimal post-processing [48].

Therefore, a workflow using neural implicit fields, combined with a differentiable FEA simulator and embedded fabrication constraints, all running efficiently on the GPU, is introduced for the multiscale topology optimization of discrete mechanical metamaterial (see Figure 3.1 B-C).

3.2.2 Bottom-Up Optimization

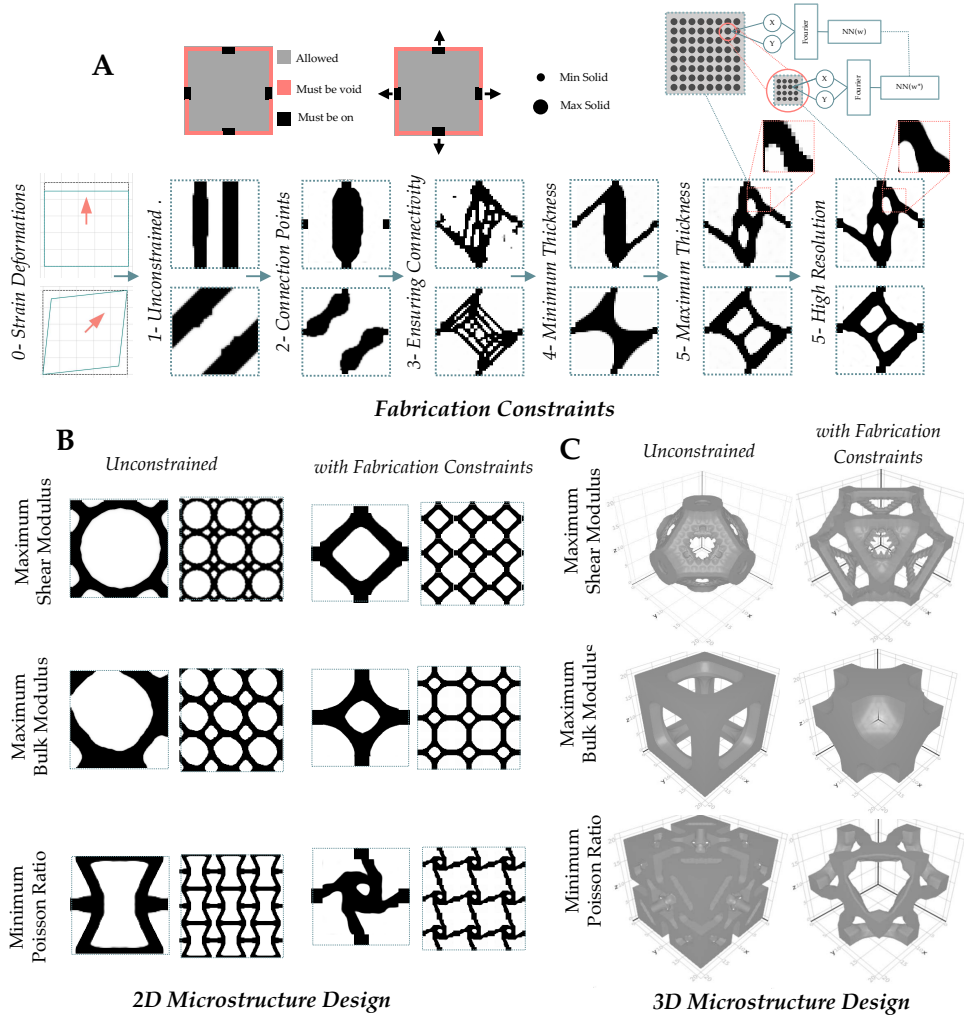


Figure 3.2: **Bottom-Up Optimization.** (A) Fabrication Constraints. (B) 2D Microstructure Design. (C) 3D Microstructure Design.

Figure 3.2 A shows the fabrication constraints introduced in the optimization process, making sure (1) the voxel faces are only connected using the inner and intervoxel connections, (2) each face constitutes one connected geometry, and (3) constraints for the minimum and maximum thickness of each beam features that guarantee that the discrete structure behaves like a monolithic structure are implemented.

$$\begin{aligned}
& \underset{\phi}{\text{minimize}} && f(C^H(\phi, D^{(i)})) \\
& \text{subject to} && K(\phi)d^{(i)} = F(\phi)^{(i)} \quad \forall i \\
& && g(C^H(\phi, D^{(i)})) \geq g_{min} \\
& && \sum_{e \in \Omega} \rho_e v_e \leq V_{max} \\
& && \phi_{min} \leq \phi_n \leq \phi_{max} \quad \forall n \in \Omega \\
& && d^{(i)} \text{ is } \Omega - \text{periodic}
\end{aligned}$$

• Where:

- f is the objective function (such as minimum compliance, maximum stiffness, etc.)
- ϕ represents the design variables (neural implicit field parameters)
- $K(\phi)$ is the global stiffness matrix derived from the design
- C^H is the homogenized constitutive matrix
- g represents constraints such as square symmetry or isotropy requirements
- $d^{(i)}$ are the displacements associated with applied strain fields
- $F^{(i)}$ are the force vectors associated with the strain fields
- V_{max} is the maximum allowed volume fraction

For the first fabrication constraint, this is implemented modifying the design space, insuring after each optimization epoch that any neighboring voxels faces are only connecting through the prescribed 4 connections. Then to ensure that each face is made form only one connected geometry, small dummy forces that are normal to the connections are added to the boundary conditions. Finally, the minimum and maximum thickness for the 2D and 3D connections can be tuned using the Fourier mapping parameters. Figure 3.7 A & B polar diagram of elastic anisotropy show the effect of the discretization on an optimized microstructure.

Figure 3.2 B shows the optimization results before and after introducing all fabrication constraints for the 2D microstructures. These optimizations target different objective functions including maximum shear modulus, bulk modulus, and minimum Poisson's ratio. The same workflow is extended to design 3D discrete mechanical metamaterial. Figure 3.2 C shows optimized 3D microstructures with various target objectives.

3.2.3 Top-Down Optimization

2D Optimization

Next, I will introduce a top-down approach for the optimization of digital material macro and micro structures. Figure 3.1 C shows a diagram of the workflow. I start by discretizing the structure and setting the boundary conditions and the objective function for the macrostructure, and optimizing the macrostructure using the following objective function.

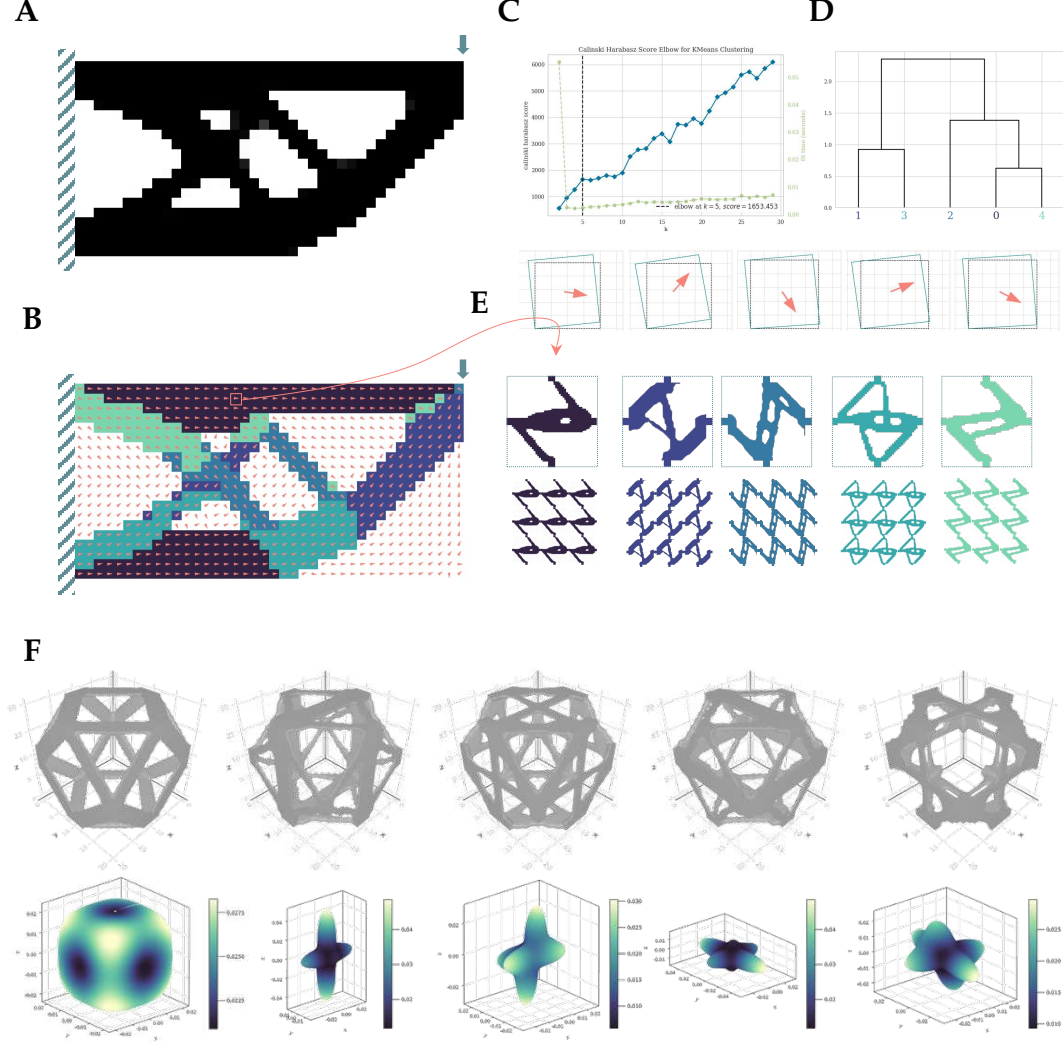


Figure 3.3: **Top-Down 2D Optimization.** (A) Optimized Macrostructure. (B) Strain Direction and Clustering. (C) K means Elbow Plot. (D) Hierarchical Clustering. (E) Optimized Microstructures. (F) Anisotropic Properties from 3D units made of 2D Pieces.

$$\begin{aligned}
 & \text{minimize} && \mathbf{u}^\top K(\rho) \mathbf{u} \\
 & \text{subject to} && K(\rho) \mathbf{u} = \mathbf{f}, \\
 & && \sum_e \rho_e v_e \leq V^*
 \end{aligned} \tag{3.1}$$

Next, I was able to get the principle strain deformation for each element, and get the following vector: and, using K means clustering, I was able to cluster the elements that have

similar principle strain deformation by minimizing the following distance:

$$dist = \sum_{r=1}^{N^c} \sum_{\mathbf{x}_i \in \Omega_r} \|\mathbf{x}_i - \boldsymbol{\mu}_r\|_2^2$$

The sample data used in the direction-based cluster analysis for element i can be expressed as [49] (where α_i denotes the chosen principal stress direction of element i):

$$\mathbf{x}_i = [\cos(2\alpha_i), \sin(2\alpha_i)]$$

By plotting the elbow plot as well hierarchical clustering I can choose the appropriate number of clusters. Next, for each cluster using the average strain vector (deformation U_e of nodes) for each group, I can optimize the microstructures using the same workflow introduced in the previous section. Figure 3.3 A-E shows the workflow implemented to optimize microstructures for a 2D cantilever.

3D Optimization from 3D Faces

The same workflow can be extended to 3D, by grouping different 3D elements using their 3D principal strain directions and implementing the fabrication constraints in Figure 3.7 B. Figure 3.4) shows the workflow implemented for the multi-scale optimization of a 3D cantilever.

3D Optimization from 2D Faces

Additionally, I can extend the workflow for the inverse design of 3D macroscale metamaterial constructed from 2D pieces. Creating 3D metamaterials from 2D pieces using high-throughput fabrication processes like laser cutting and stamping offers significant advantages over traditional 3D printing methods. These 2D manufacturing techniques enable rapid mass production of precision components at a fraction of the cost and time required for additive manufacturing. Although 3D printing faces severe limitations in build volume, material options, and production speed, 2D fabrication processes can quickly produce thousands of identical parts that can be robotically assembled into large-scale structures. This approach allows for efficient scaling to structures that exceed one meter in size while maintaining high precision and structural integrity. Additionally, using 2D manufacturing techniques enables flat packing for transportation, simplifies quality control through standardized parts, and facilitates easier maintenance and reconfiguration since components can be individually replaced when damaged. The modular nature of these assembled structures also permits the creation of heterogeneous systems with varied mechanical properties throughout the structure.

Figure 3.5 A demonstrates that microstructures constructed from 2D faces can still achieve highly anisotropic properties. The multiscale optimization workflow can be modified by extracting 2D deformation maps in the XY, XZ, and YZ directions, then applying the same clustering approach, before optimizing the 2D microstructures. This approach allows us to create complex, high-performing 3D metamaterials using a small library of 2D pieces, offering a scalable and affordable method for designing and fabricating macroscale metamaterial structures.

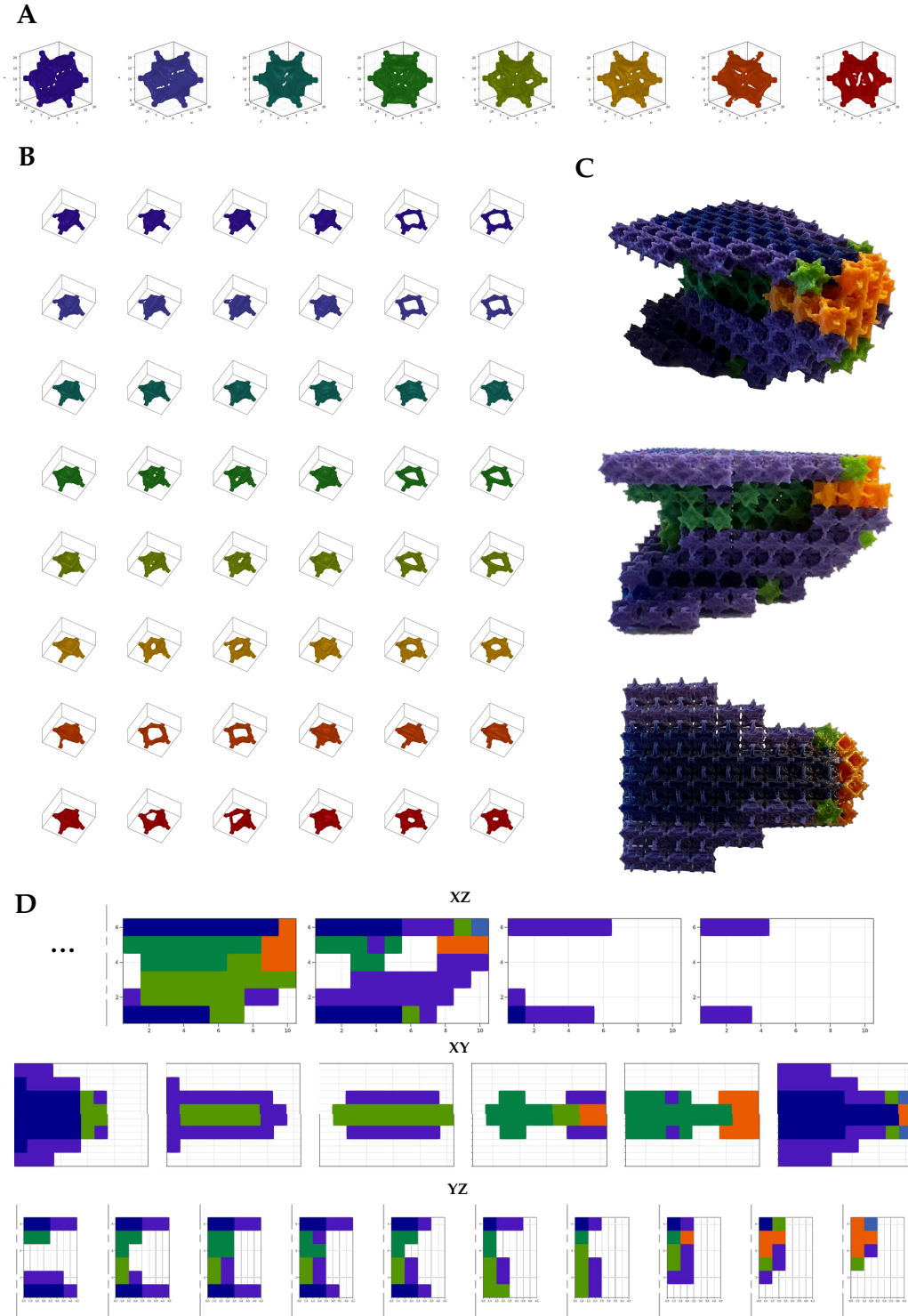


Figure 3.4: **Top-Down 3D Optimization.** (A) Optimized Microstructures. (B) 3D Faces. (C) Optimized Macrostructure (3D Printed). (D) Clustering Cross Sections.

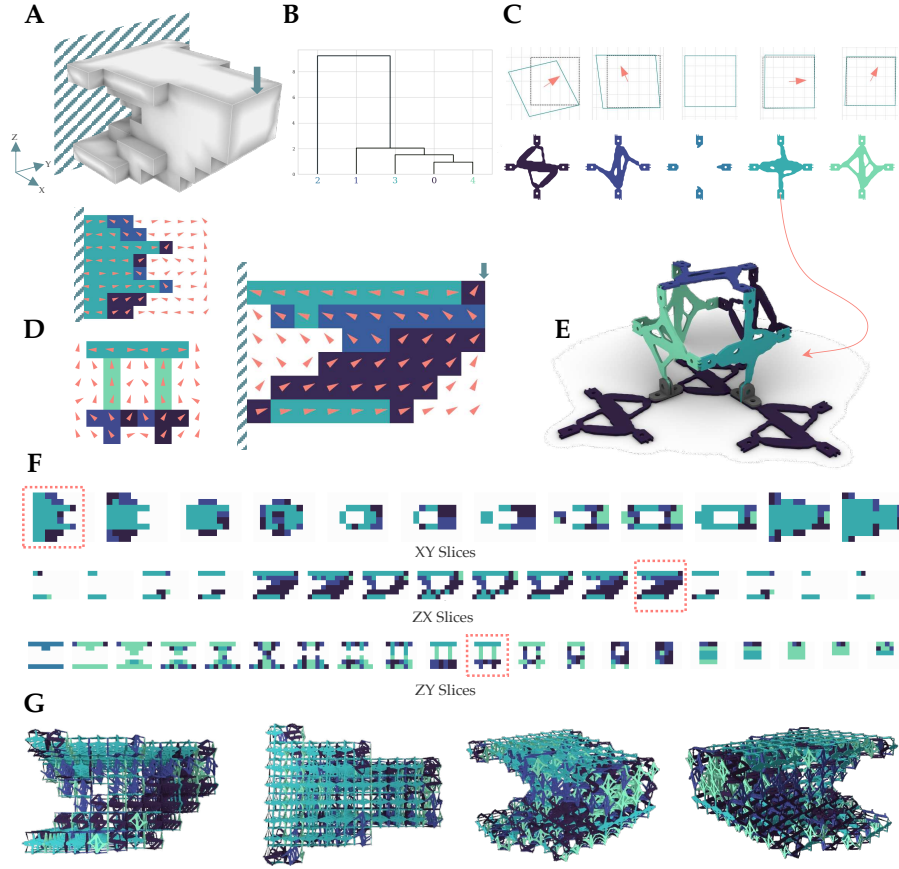


Figure 3.5: **3D Top-Down Optimization from 2D Pieces.** (A) Input Macrostructure. (B) Hierarchical Clustering. (C) Optimized Microstructures. (D) Strain Directions. (E) 3D Microstructure Example. (F) Clustering Cross Sections. (G) Optimized Macrostructure.

2D Material Library

Furthermore, I extended the workflow to design a universal 2D library of pieces using different strain directions that could be used to build any macroscale structure. This approach eliminates the need to redo microscale optimization for each new macroscale structure, significantly reducing computational costs and design time. By analyzing a wide spectrum of possible deformation modes and creating optimized pieces for each mode, I developed a comprehensive material library that covers most practical applications. When designing a new macroscale structure, one can simply map the local strain fields to the closest matching microstructure in our library, allowing for rapid deployment of optimized designs without repeating the computationally intensive microscale optimization process.

First, I use n strain directions with an increment of 11.25 degrees. In order to translate the principal strain direction to the strain vector that I can use for optimization, I use CMAES to find a strain vector field with a unit norm that will result in the desired principal direction (see Figure 3.6 A). Then by rotating and mirroring these 2D faces I can cover all principal directions (see Figure 3.6 B). Figure 3.6 C shows how this approach has been used in the multiscale optimization of a 3D cantilever.

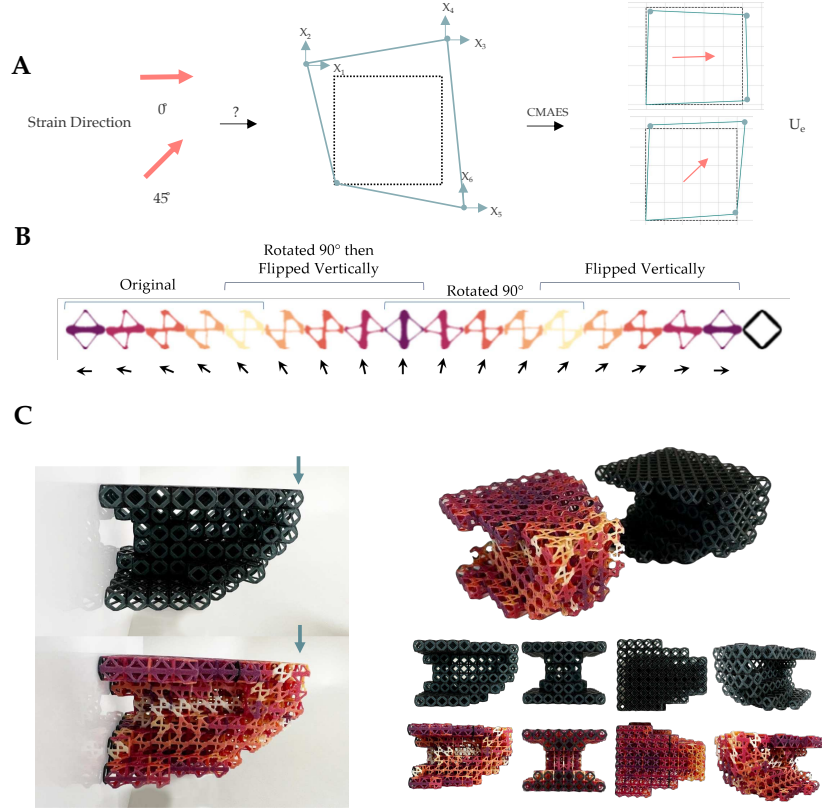


Figure 3.6: **2D Material Library.** (A) Unit Strain Field Discovery Workflow from Strain Direction. (B) 2D Material Library. (C) 3D Printed 3D Optimized Microstructure Vs Cuboct (base case).

3.2.4 Fabrication and Testing

To test the introduced workflow, we fabricated a standard cuboctahedron lattice as well as an optimized lattice for a vertical compression load. In each case, we assembled a 3 by 3 by 3 block of the material using 2D faces laser-cut from 1.5 mm thick acetal sheets, riveted together using 3D printed brackets made from nylon with chopped carbon fiber (Markforged Onyx filament), as shown in Figure 3.8 B. The geometry of the joints is relatively simple, consisting of interconnected L-brackets. Because the interconnection primarily assists in the fabrication workflow, and is not structurally active, we could switch to, for example, off-the-shelf metal L-brackets to extend this workflow to more rigid materials, without needing to redesign the connection or fabrication system.

To optimize the structure against a vertical compression load, I applied the boundary conditions and extracted 2D deformation maps in the XY, XZ, and YZ directions. I then replaced the cuboctahedron faces in the standard lattice with appropriate faces from the universal material library based on the principal strain directions.

The standard and optimized cuboctahedron geometries were compared using compression testing on an Instron 5985 machine (see Figure 3.8 C). The voxels were assembled into three-by-three blocks, which were then compressed at a rate of 15mm/min to a distance

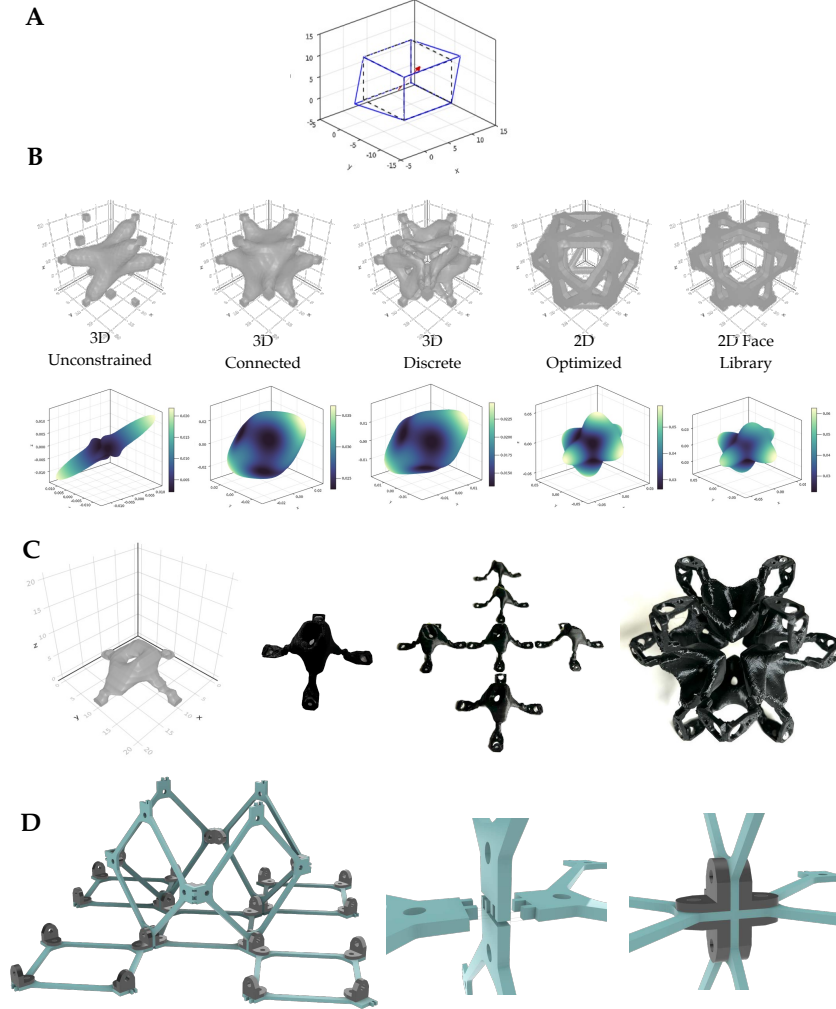


Figure 3.7: **Effect of Discretization.** (A) 3D Unit Strain Field and Direction. (B) Comparison of Different Discretizations Strategies on Unit Performance. (C) 3D Voxel Fabrication Details. (D) 3D Voxel from 2D Pieces Fabrication Details.

of 60mm. The measured side length of both blocks is 0.2215 m, giving a cross-sectional area of 0.04906 m^2 and total volume of 0.01087 m^3 . The weight of the standard geometry is 0.3423 kg, while the weight of the optimized version is 0.4094 kg. The bulk engineering modulus of the material is then found by linear fit to the elastic regime of the material's response, while the maximum compressive strength of the material is taken as the maximum stress in the material. The heavier weight of the optimized version reflects a higher minimum beam thickness used during the optimization process. Though we would expect this to result in a higher strength and modulus from the outset, the experimental results show a more substantial increase than could be accounted for in increased minimum beam thickness alone.

This testing verifies that the optimized geometry outperforms the standard geometry in terms of stiffness and strength. The standard geometry's bulk modulus is 0.329MPa, and the optimized geometry's is 1.916MPa, representing a 5.8x increase. To account for the weight

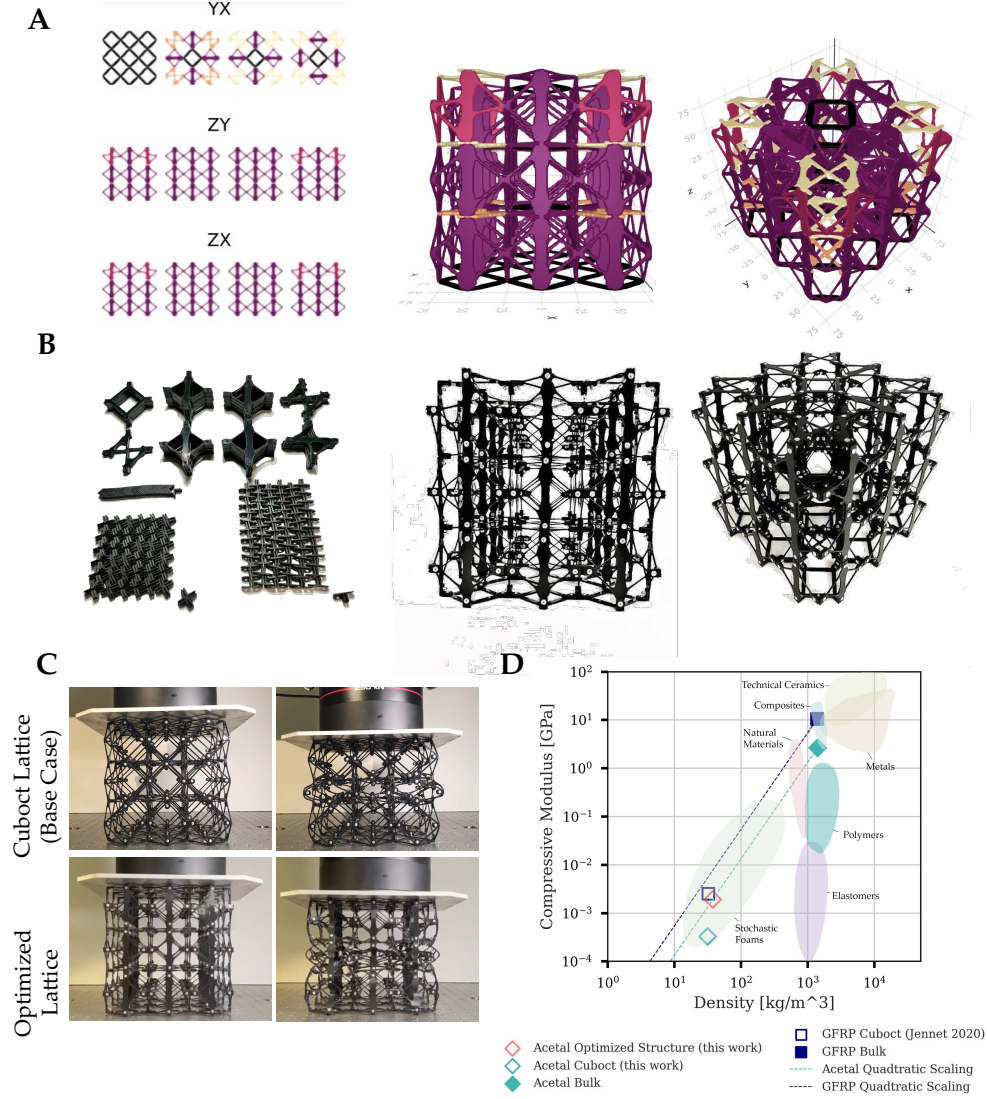


Figure 3.8: Fabrication and Testing. (A) 3 by 3 by 3 Lattice Optimized Under Compressive Loading. (B) Fabricated 2D Pieces (Cuboct base Case vs Optimized Lattice). (C) Instron Testing. (D) Compressive modulus [GPa] versus density kg/m^3 for the fabricated geometries in this work, showing an approximately 5x improve in specific modulus for the optimized geometry as compared to the standard cuboctahedron, as well as better geometric performance as compared to the injection molded cuboctahedron lattice in [3].

differences, we can consider the specific modulus instead. For the standard geometry, it is $0.0104 \text{ MPa}/(kg/m^3)$, and for the optimized geometry, it is 0.0508 , representing a 4.88x increase. For the standard geometry, this $12.32kPa$, and for the optimized geometry, this is $38.16kPa$, representing a 3.097x increase.

3.3 Discussion

In this chapter, I present an automated workflow for designing large-scale 3D discrete metamaterials constructed from 2D pieces. The approach enables efficient fabrication of complex mechanical structures with customized properties. Comparative analysis of elastic anisotropy maps between full 3D optimization and the discrete approach demonstrates that the method achieves comparable performance with only minimal property degradation, despite the constraints introduced by discretization into voxels and faces.

The 2D-based approach offers significant practical advantages over conventional 3D manufacturing methods. These include efficient flat-packing for transportation, exploitation of symmetries to reduce the number of unique pieces required, and simplified assembly processes. The 2D manufacturing approach is substantially more time and cost-efficient compared to alternatives. This will result in dramatic improvements in production speed compared to 3D printing and lower tooling costs than injection molding for small to medium production runs.

While I primarily demonstrated minimum compliance optimization, the framework extends naturally to diverse objective functions. For example, I successfully applied this methodology to design 3D mechanical inverters with optimized force transmission characteristics. The versatility of the approach enables applications in soft robotics, including customized actuators and mechanisms with programmable deformation responses. Additionally, the modular nature of the discrete metamaterials facilitates reconfiguration, allowing adaptive structures that can be modified for different functional requirements. Although manual assembly was used in our prototypes, the standardized connection system is well-suited for robotic assembly [43] [44], opening pathways to scalable, automated production of large metamaterial structures.

Chapter 4

Learning Morphogenesis: Differentiable Physical Growth

4.1 Introduction

Once optimal structures are identified, the next challenge lies in translating these static designs into feasible fabrication and growth sequences. The biological world is built from growth; from simple distributed rules emerges a complex form. In engineering parlance, biological growth provides a process in which design and manufacturing are coupled — the rules that encode how cells propagate and die to determine the structure, while the processes of birth and death themselves yield embodiment. It is a system in which concept and realization are unified. By contrast, the process of human engineering is historically decoupled; designs are conceived, typically in a manufacturing-aware way, and then fabricated; therefore, I aim to create centralized additive and subtractive manufacturing processes (such as CNC milling or 3D printing) that can realize almost any design. This sequential and centralized design-and-manufacturing workflow produces constructs that lack some of the innate capability of their organic counterparts. Namely, unlike the built world, the biological world is a) resilient, b) refinable, and c) inherently manufacturable. Biological processes have been of interest since the early days of computing, harkening back to Conway’s Game of Life, but developing rules for growing interesting and functional structures has remained a difficult problem. I present a method for learning growth rules using feedback from physics to grow functionally performant static and dynamic structures that realize the listed benefits of biological growth. Our approach marries mathematical analysis with biological processes to quickly learn distributed growth rules atop neural cellular automata (NCA) architectures. Our approach has biomimetic origins; just as life sprouts diversity, robustness, and capability through morphogenesis, so too does our artificial life through computational morphogenesis. In presenting strategies for computational design in approaches inspired by cell-based growth, such study also allows learning about the nature of biological growth itself. In order for a biological cell to grow into a complex organism, rules must be encoded through its genetic code; this code has evolved through billions of years to generate the types of complex multicellular organisms we see today. In NCA, that code is represented through neural function. Other than the way in which genes and neural networks are translated to function,

a key differentiator between biological systems and ours is the timescales upon which design rules are synthesized — whereas evolution uses survival of the fittest to optimize life over many geological eons, NCA optimizes over just hundreds of deep learning epochs. Contrasted with their classical discrete counterparts, such as Conway’s and Wolfram’s automata, NCA exploits the differentiability of deep neural networks for rapid adaptation; combining these differentiable building blocks with a differentiable model of the world [50] creates an end-to-end process that quickly learns optimal growth policies that are tolerant to noise and defects. Because NCA learns from noisy mini-batches of data over the course of a growth process, they can grow structures locally and have a memory of how to grow from incomplete structures. These algorithmic design choices provide resilience and a certifiably structurally stable recipe for manufacturing by organic or inorganic growth-based processes. Because NCA is built atop visual deep learning architectures, they are amenable to high-dimensional input and output spaces, allowing them to reason about and design intricate multi-material structures. Because the NCA learns from its own exploration of simulated worlds, it can learn free from the need for curated datasets. Atop the design acceleration provided by leveraging analytical system and model gradients, our parallelizable deep-learning-based approach offers significant hardware acceleration, making it highly practical.

4.2 Results

I demonstrate the approach using two problems: the construction of cellular structures and the morphing of soft robots, demonstrating its applicability to the built static and dynamic world, and providing insight into how biology evolves to learn optimal growth policies.

4.2.1 NCA Architecture

In both problems, differentiable physics simulation (differentiable Finite element analysis (FEA) and material point method (MPM)) is used to evaluate the objective function. The error is back-propagated to update the NCA (convolutional neural network) weights and optimize the policy. The assumption is that the structures/robots are discretized into cells, and each has enough memory to hold the NCA weights and its state – 16 floating point values/channels, some have physical meanings like density, mass, material properties, and the rest of hidden values that help with the NCA training [50].

As for the NCA weights, Figure 4.1 shows the simple NCA (one layer of CNN with 128 kernels), where the memory needed is $(128 * \text{the size of the neighborhood})$. Each cell can only communicate and share its state with either its immediate von Neumann neighbors (4 for 2D and 6 for 3D) or Moore neighbors (8 for 2D and 26 for 3D) and can do computation to update the NCA state. All examples presented in the following sections were trained using this simple NCA architecture.

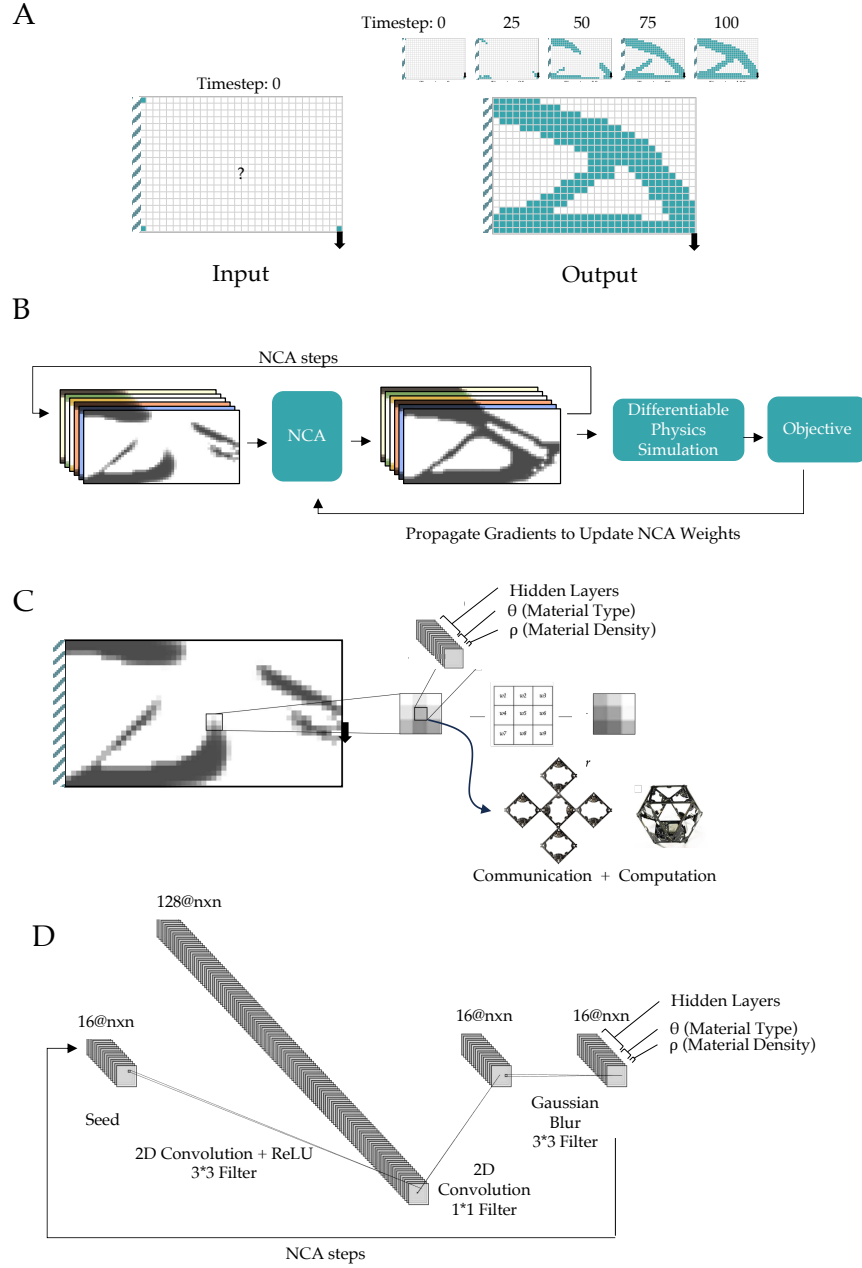


Figure 4.1: **Neural Cellular Automata (NCA) Overview.** (A) Problem Description. (B) NCA Workflow. (C) NCA Implementation. (D) NCA Architecture.

4.2.2 Static Structures

Assumptions and Hardware

In the first problem, given boundary conditions (supports, loads, and volume constraint), I concurrently find the optimal cellular structure's shape based on an objective function (e.g., minimal compliance or maximum thermal diffusion) and its construction sequence (see Figure 4.1 A).

For the base case, I consider a 2D design domain with two cell types: passive and active. The passive cells serve a structural function, while only the active cells can perform computation and communication with their Neumann neighbors. I begin with a grid of sacrificial active cells (matching the design domain size) and program them to perform one NCA computation synchronously per step. When the first state value (ranging from 0 to 1) exceeds a threshold of 0.8, I place a corresponding passive voxel/cell in front of the active cell. Figure 4.2 A illustrates a learned NCA rule growing a cantilever structure under these assumptions.

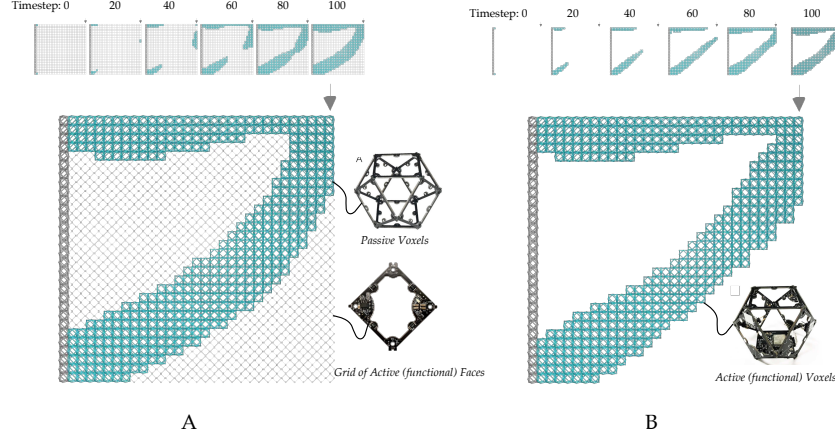


Figure 4.2: **Synchronous vs Asynchronous NCA Growth.** (A) Synchronous update using a grid of active functional voxel faces and passive 3D voxels. (B) Asynchronous update using active functional 3D voxels.

In the next case, I relax the assumptions and consider only one type of cell: structural cells that can also perform computation. By training the NCA to update each cell's state only if its density or one of its neighbors' densities exceeds a threshold (0.9), we ensure that cells grow exclusively from the seeds, creating continuous growth. Figure 4.2 B shows an example of a trained NCA growth rule for the 2D cantilever. The final shape is very similar to the base case, but the construction sequence differs.

Examples

Figure 4.3 shows training the different synchronous NCA to grow different structures using different boundary conditions. In all examples, the objective function (minimal compliance) was evaluated at the 100th time step, using a volume constraint (40 percent of total volume).

$$\begin{aligned} \min_{\rho} c(\rho) &= \mathbf{U}^T \mathbf{K}(\rho) \mathbf{U} \\ \text{subject to: } V(\rho) &= \sum_{i=1}^n \rho_i v_i \leq V^* \\ 0 \leq \rho_i &\leq 1, \quad i = 1, 2, \dots, n \end{aligned}$$

- where:

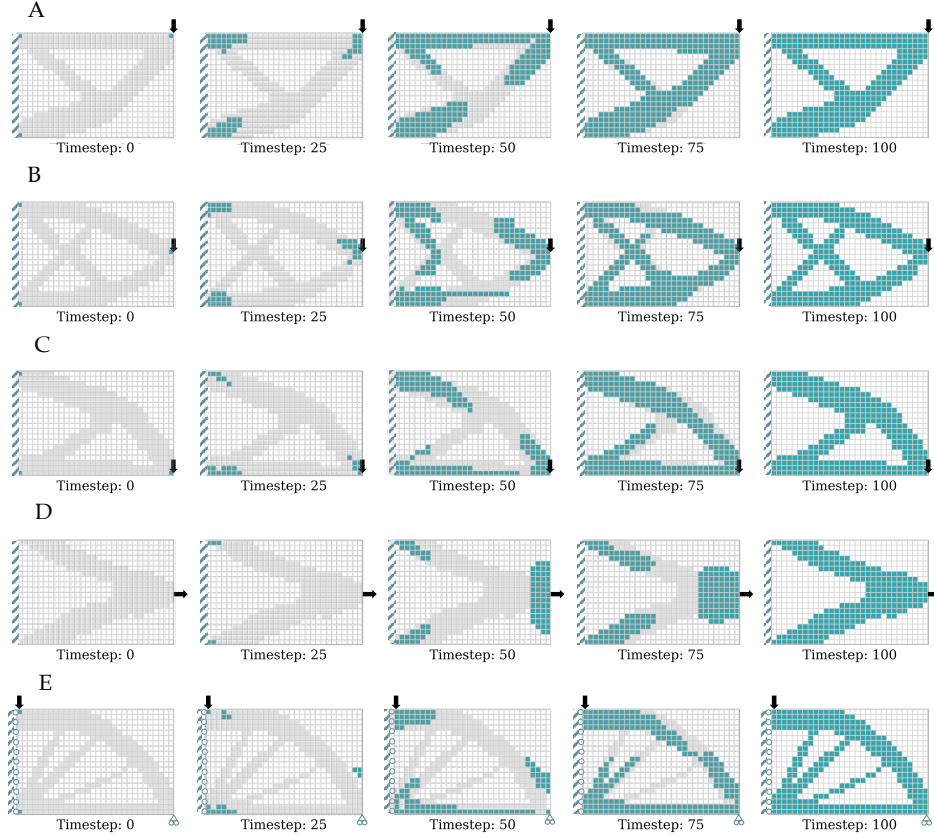


Figure 4.3: **Examples of Different Boundary Conditions.** (A) Top-Cantilever. (B) Mid-Cantilever. (C) Bottom-Cantilever. (D) Horizontal Force. (E) Messerschmitt-Bölkow-Blohm (MBB) Beam.

- ρ is the material density
- $\mathbf{U}$ is the displacement vector
- $\mathbf{K}(\rho)$ is the global stiffness matrix
- v_i is the volume of element i
- V^* is the maximum allowable volume
- n is the total number of elements

The SIMP (Solid Isotropic Material with Penalization) approach was used to drive the densities toward either 0 or 1. Examples took around 1600 epochs to train and converge to the final shape (see Figure 4.4). The same method can be extended to train the growth of 3D structures or other objective functions beyond structural stiffness, such as maximizing thermal conductivity (see Figure 4.5).

Stability and Self-Healing

I then extended the training to ensure the robustness of the grown structures. First, I wanted to verify that if the NCA continued running after 100 timesteps, the grown structure would

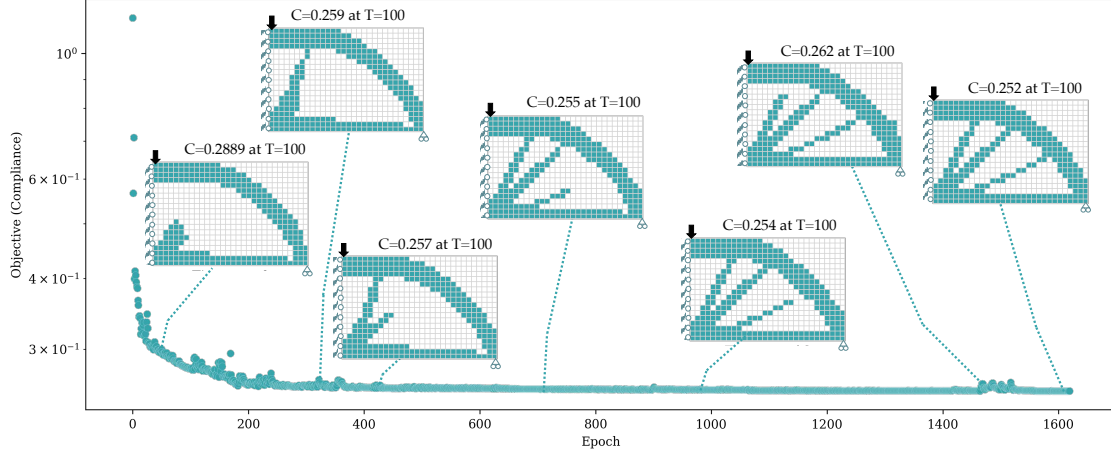


Figure 4.4: **NCA Training Progress Example.** NCA Training progress for the MBB Example showing objective function value (compliance) at final timestep.

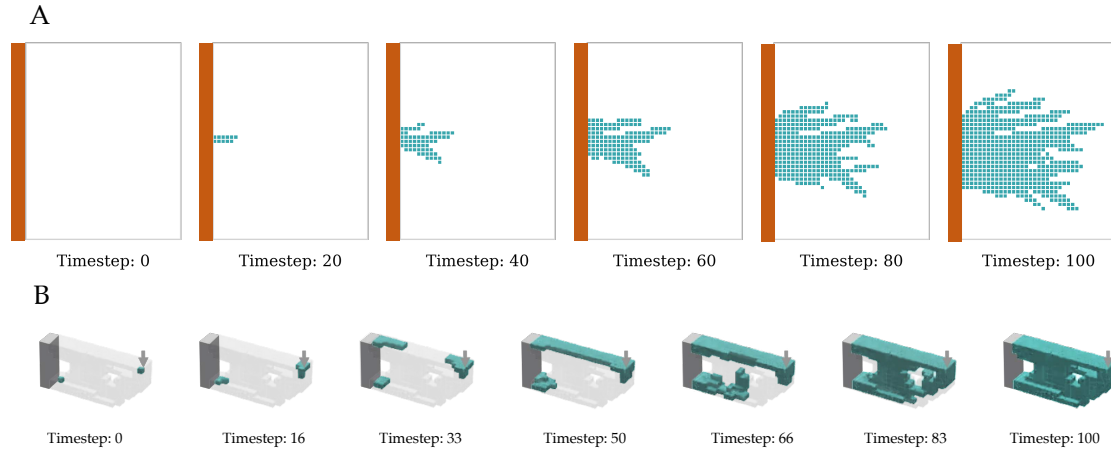


Figure 4.5: **Extension to Different Physics and Dimensions.** (A) Heat Sink example (optimization to maximize thermal diffusion). (B) Extension to 3D (3D Cantilever Problem).

remain stable. I used the approach introduced in [50] where, in some of channels, during training, instead of using the starting seed as the initial step, I would provide the final structure as the first step and evaluate after 100 time steps. This ensures that the NCA rules can not only grow structures from the starting seed but also maintain the final structure geometry. As shown in Figure 4.6 A, this approach demonstrates that a stable growth rule can be sustained beyond the number of timesteps it was trained on.

Moreover, this method can be extended to provide growth rules for structures that can heal themselves if damaged. By introducing random damage to the structure and using this as the starting point—as shown in Figure 4.6 B—we can observe how the structure recovers, "heals," or regrows the damaged parts.

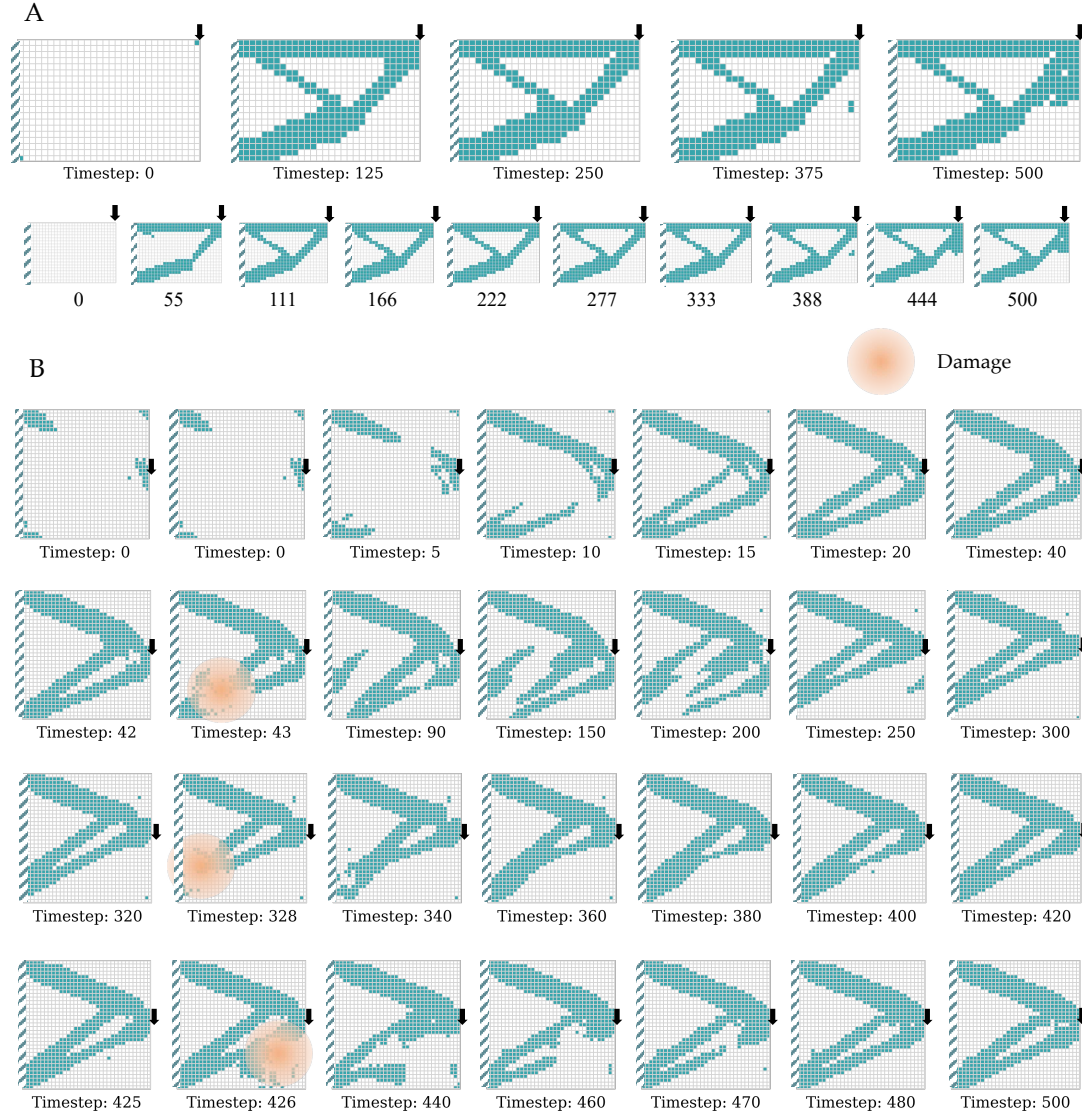


Figure 4.6: **NCA Stability and Resilience.** (A) Training NCA for the structure to be stable beyond its growth. (B) Training NCA to be resilient and self-heal in case of damage.

Hierarchy

The NCA is not only able to train the growth of structures with one type of building block, but since it's possible to have building blocks (microstructures) with different geometries (and hence different mechanical properties), I trained the NCA to tell us both where to place the units and which units to place (from a library of units with different geometries and their homogenized properties (compliance matrices)). Figure 4.7 A,B,C shows how the trained NCA grew different structures when given different microstructure libraries under the same boundary conditions. It also demonstrates different structures produced with the same microstructure library but different boundary conditions (see Figure 4.7 C,D,E).

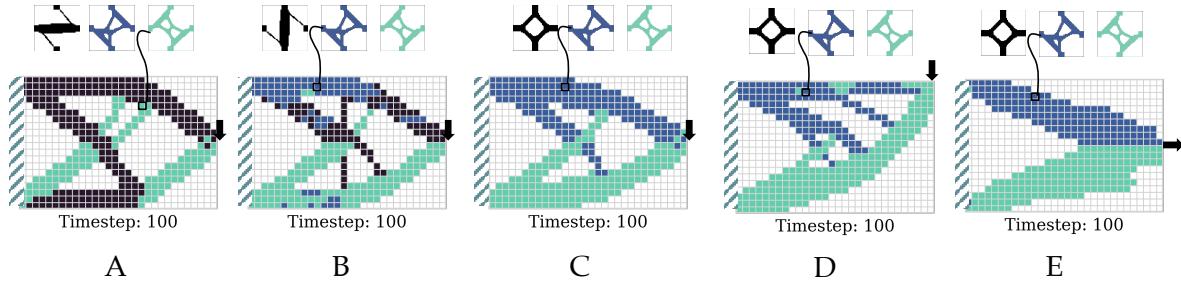


Figure 4.7: **Hierarchical Growth with Multiple Microstructures.** (A - B - C) Same Boundary Conditions, Different Microstructures. (C - D - E) Same Microstructures, Different Boundary Conditions.

Generalization

In its current form, the NCA needs to be trained from scratch each time there's a new boundary condition (domain size, support, and loading condition). I explored how to make the approach generalizable by training for a family of boundary conditions in parallel (in this case, changing load location). During training, each channel had a different boundary condition and starting seed location. This enabled us to train an NCA growth policy that is generalizable—one that can grow an optimal structure when given a new, unseen boundary condition (see Figure 4.8 A).

Controlled Growth and Reconfiguration

Next, I aimed not only to generate the growth sequence but to design it based on different rules and boundary conditions. This could be accomplished using either the synchronous or asynchronous growth strategies mentioned earlier. Figure 4.9 A-B demonstrates the asynchronous growth strategy, showing how the same boundary condition can produce different growth sequences depending on the starting seed location. In Figure 4.9 C, I trained an NCA with an additional constraint—cells cannot be removed once placed. This was implemented by modifying the rule so that when a cell's density exceeds the threshold of 0.8, it stops updating its state and only maintains communication capabilities.

We can take this further by not only training the growth of structures but also their reconfiguration when boundary conditions change over time. Figure 4.8 B and C show different NCAs that are trained to grow and then reconfigure structures—one capable of moving a load upward (see Figure 4.8 B) and another forming a reconfigurable bridge (see Figure 4.8 C).

4.2.3 Dynamic Structures

Reconfigurable Soft Robots

I then wanted to extend the workflow to be used not only for the reconfiguration of static structures, but also dynamic structures—in particular, soft robots that are able to morph their shape to move through obstacles or to accomplish different objectives.

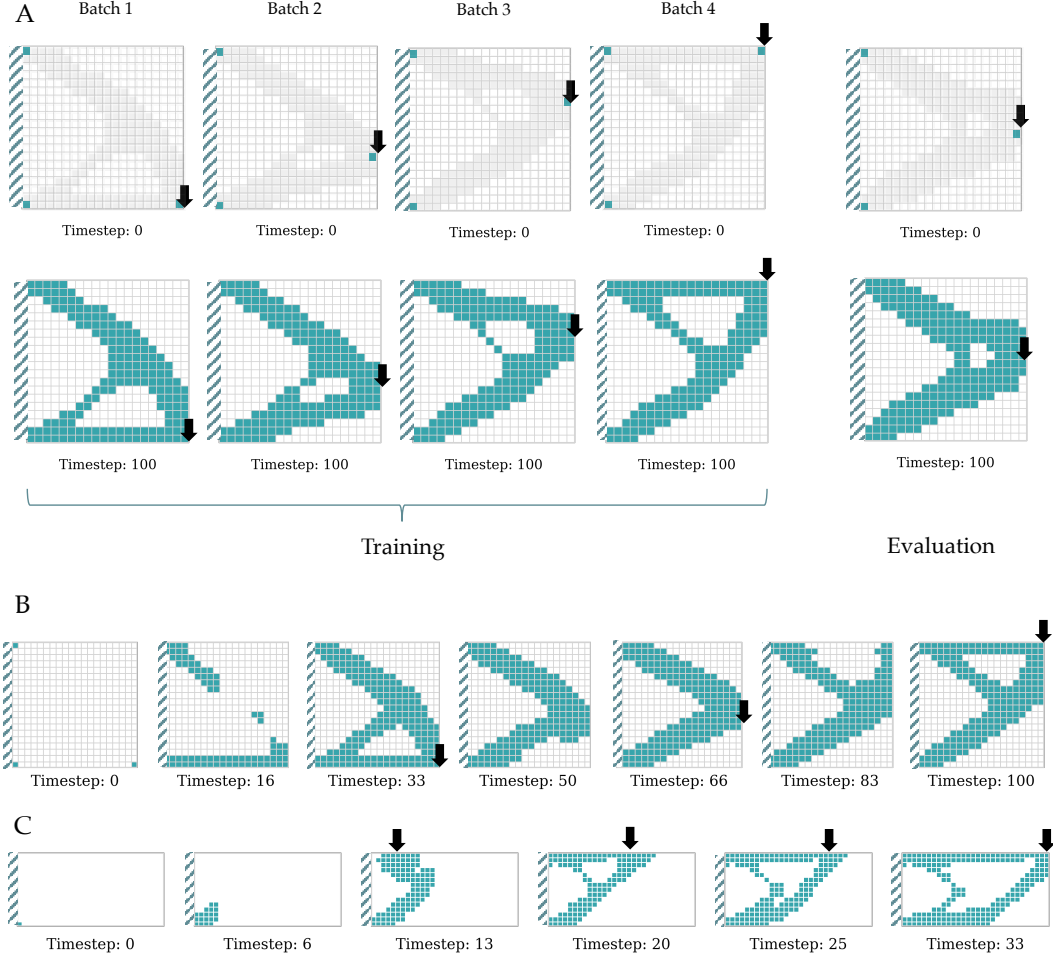


Figure 4.8: **Generalization to Changing Boundary Conditions.** (A) Training Generalization to new unseen boundary conditions. (B - C) Training against changing boundary conditions to design shape reconfiguration (using synchronous (B) and asynchronous (C) rule update).

Building upon the demonstrated capabilities of Neural Cellular Automata (NCA) for static structure growth and reconfiguration, I now extend this framework to dynamic, continuously morphing systems. Contemporary soft robotics research has predominantly employed predefined shape transformations or discrete reconfiguration states, with limited exploration of continuous growth-based morphological adaptation during locomotion. This research addresses a critical gap in the literature, as previous studies have not comprehensively investigated geometric representations that simultaneously facilitate growth, reconfiguration, and locomotion in soft robots.

Unlike conventional approaches that separate the design of robot morphology from its control system, the NCA-based framework unifies these aspects through a single differentiable pipeline. This integration allows soft robots to not only navigate through complex environments by changing their shape but also to fundamentally transform their morphology in response to varying objectives. The key innovation lies in representing both the physical

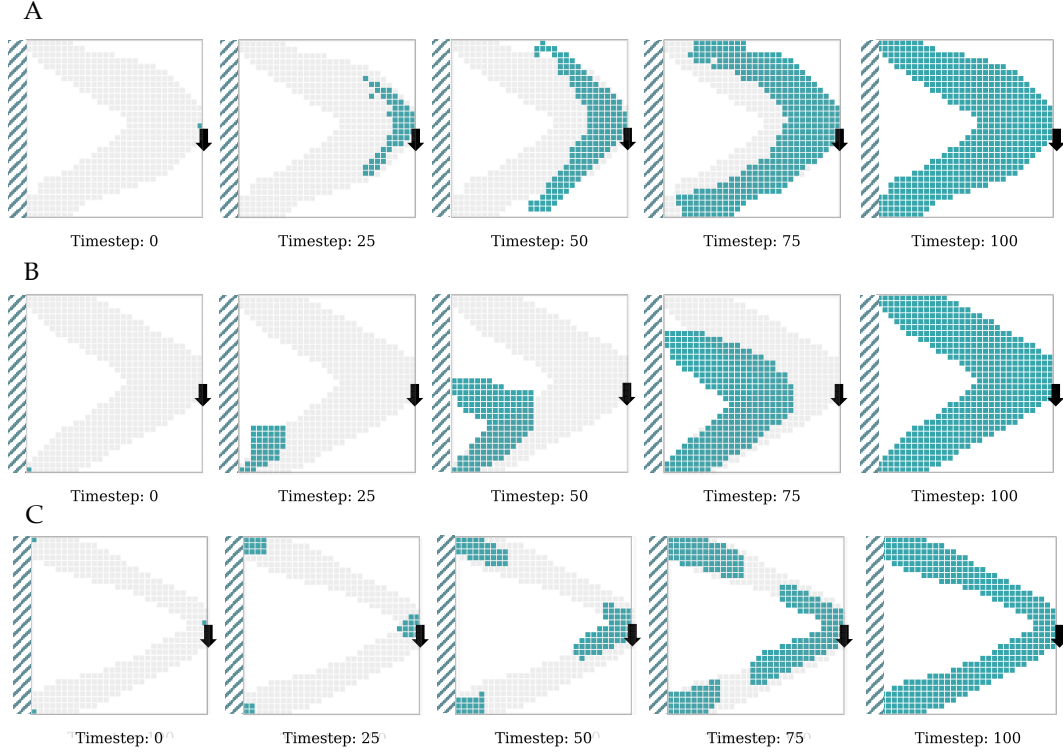


Figure 4.9: **Design of Targeted Growth.** (A-B) Growth of the same shape but different starting seed location using asynchronous neighbor rule. (C) Grow-only rule where voxels are not allowed to be removed.

structure and its temporal evolution through the same cellular architecture, enabling robots to "grow" new appendages, modify their mass distribution, or transform their entire body plan while maintaining functional continuity.

This approach builds upon recent advances in soft robotics [51] [52] [53] but fundamentally differs by introducing growth as a continuous control parameter rather than a discrete design variable. Furthermore, while previous work in reconfigurable robots [54] [55] has explored modular transformations, none have leveraged the emergent properties of cellular automata to achieve smooth, continuous morphological adaptation during locomotion.

Workflow

Our workflow uses the NCA architecture as Figure 4.1 but with a different representation, as each cell in the NCA represents 8 by 8 particles in the MPM simulation (see Figure 4.10). The NCA timesteps are distinct from the MPM simulation steps—after each NCA step, which represents a change in morphology, the simulation runs n timesteps. Finally, if the state of the NCA cell that represents the density/mass of the particles falls below a threshold, I "remove" the masses by changing their location to a faraway position. This prevents the simulation from becoming unstable, which typically happens when particles masses becomes too small.

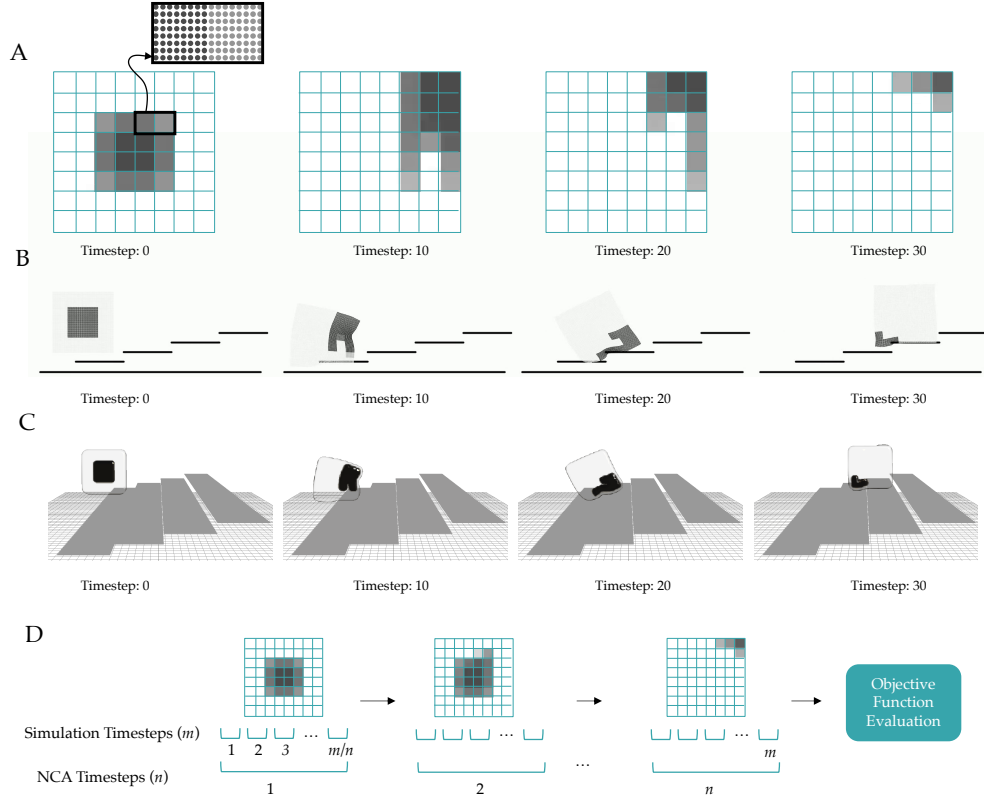


Figure 4.10: **Soft Robot NCA Reconfiguration Workflow.** (A) NCA cells density state map. (B) MPM particle simulation translation. (C) Soft Robot Visualization. (D) NCA workflow for the dynamic simulations.

Examples

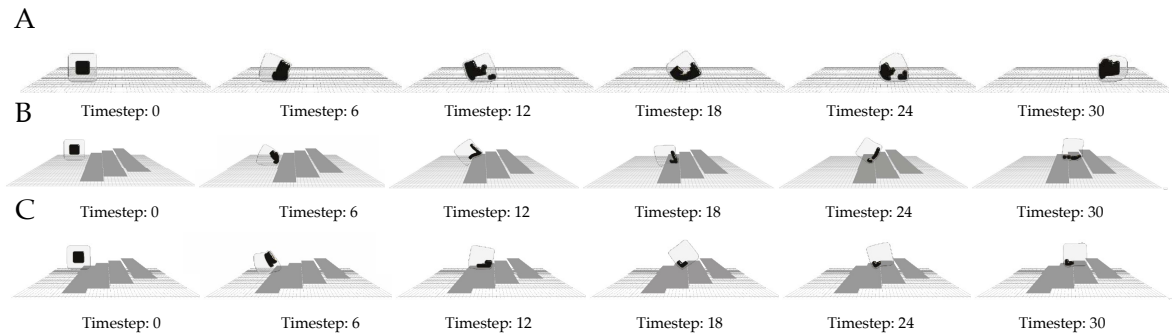


Figure 4.11: **Soft Robot Mass Change.** (A) Learning to reconfigure center of mass to move forward. (B - C) Learning to reconfigure center of mass to climb obstacles.

Our first set of experiments are training soft robots that are able to change their center of mass, to move forward to climb different obstacles (see Figure 4.11).

I then extended the simulation to train soft objects that are able to fully reconfigure and grow or remove limbs completely to move forward or navigate through obstacles (see Figure

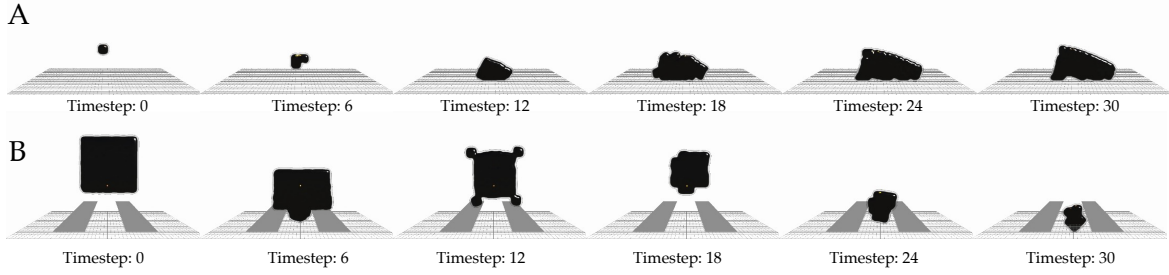


Figure 4.12: **Soft Robot Reconfiguration.** (A) Growing Limbs to go forward. (B) Shrinking to avoid obstacles.

4.12).

Generalization

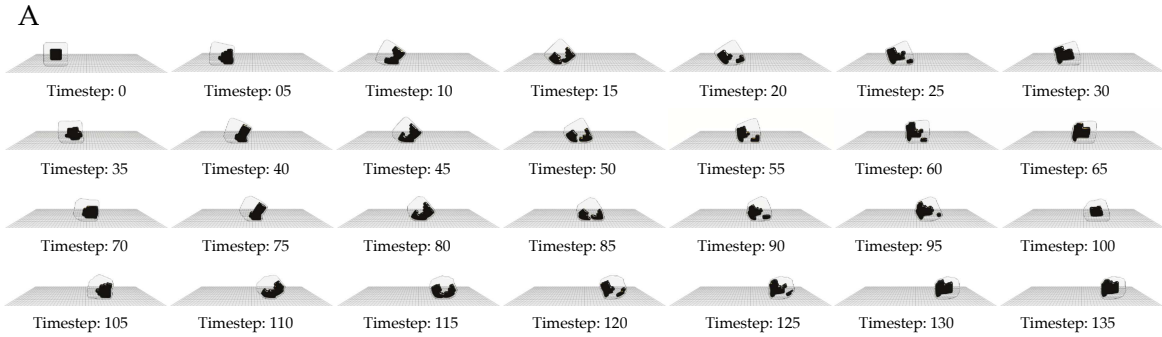


Figure 4.13: **Generalization: Learning Walking Gait.**

One of the limitations occurred when I extended the MPM simulation timesteps was that the gradient of the objective function in relation to the NCA weights became no longer useful due to the vanishing/exploding gradients effect. Therefore, I explored how to train for a generalizable NCA reconfiguration strategy that could represent a "walking gait" for the robots. I accomplished this by not only training for the NCA reconfiguration to maximize the objective function, but also adding another constraint where the starting and ending geometry of the robot are the same, as well as the particles' position and velocity. Figure 4.13 demonstrates the generalizable walking gait that could run beyond the number of timesteps it was trained on.

4.3 Recap

In this chapter, I have demonstrated a novel approach to designing and controlling structures using Neural Cellular Automata (NCA). The framework unifies design and manufacturing processes through biologically-inspired growth mechanisms, allowing for the emergence of complex forms from simple distributed rules. I have shown how this approach can be applied

to static structures—enabling self-healing, hierarchical organization, and reconfiguration—as well as dynamic soft robots capable of continuous morphological adaptation. The results demonstrate that NCA-based systems can not only generate optimal structures for given boundary conditions but can also adapt these structures in response to changing environments or objectives.

By extending the NCA framework to dynamic structures, I not only advance the field of biologically-inspired robotics but also provide new insights into how natural organisms might optimize their morphology in real-time to meet changing environmental demands. Future work will focus on physical implementation of these systems, exploring multi-material fabrication techniques that can realize the continuous growth and reconfiguration capabilities demonstrated in simulation. I believe this unified approach to growth-based design and control opens new possibilities for adaptive architecture, reconfigurable machines, and resilient engineering systems that can grow, heal, and evolve like biological organisms.

Chapter 5

Self-Replicating Hierarchical Modular Robotic Swarms

5.1 Introduction

Designing growth sequences inevitably raises the question of fabrication: how do we physically realize these structures in the real world, especially at large scales? Construction applications are an important motivation for the development of robotic swarms, however the practice of large-scale automated fabrication has largely focused on the use of gantries, which are limited by costs that can scale super-linearly in the machine size [56], while offering poorer performance relative to their smaller counterparts [57].

Biological growth spans many more orders of magnitude than a gantry, from ribosomes assembling amino acids into proteins, up to organelles, cells, organs, organisms, and populations. This dynamic range is attributable to two capabilities. First, self-replication—ribosomes can recursively make more ribosomes [58], and second, hierarchy — at each scale, smaller building blocks are used to assemble larger constructs which in turn are used as the building blocks for the next level up [59]. Both offer exponential speed-ups over a serial process. The goal of this work is to apply these insights to robotic assembly and investigate how swarms can construct recursively and hierarchically, in order to mimic nature, where the assembly throughput increases with scaling.

Prior work has demonstrated thousand robot swarms for coordinated shape generation [60] and package sorting and delivery for online retail [61]. Swarm systems have also been developed for fabrication and construction. Teams of wheeled robot arms can collectively 3D print a single object larger than the workspace for any individual stationary robot [62].

Other more customized robots can extrude a composite tube adjacent to teammates, resulting in a dynamic, multi-agent system [63]. Termite-inspired, wheeled robots can locomote on, build, and reconfigure 2.5D brick-based structures [64]. These examples do offer parallelization, but not hierarchy or self-replication.

The motivation behind modular robotic systems is to achieve versatility, robustness, and efficiency through composition of basic part types [65, 66]. Self-reconfiguration has been demonstrated with actuated modules capable of detaching and moving relative to their neighbors then reattaching to form a newly configured robot [67]. Self-reproduction

has been demonstrated [68], but the complexity of a single module can be prohibitive to scaling and implementation. Separate actuation, structure, and energy modules have been demonstrated experimentally with customized motor and electronics systems housed within modular capsules [69]. All of these examples have been limited to laboratory demonstrations; for scalability, I consider here combined material-robot systems, where the robots and the material they build with are designed simultaneously to be indistinguishable.

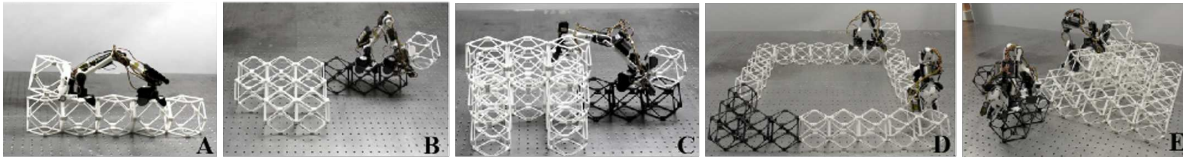


Figure 5.1: Relative Robot: Material-Robot System.

Our approach is based on the reversible assembly of discrete lattice building blocks, where global precision is determined by local geometry, errors can be incrementally detected and corrected, dissimilar materials can be joined, and parts can be disassembled for reconfiguration and re-use. This has resulted in macro-scale, ultralight lattice materials with record-setting mechanical properties [4] and mass-producibility [5], as well as heterogeneous mechanical [6] and robotic properties [7]. Applications include structural systems for morphing wings [8] and ultralight aerostructures [9], statically reconfigurable infrastructure [10], and designs for large scale space structures [11].

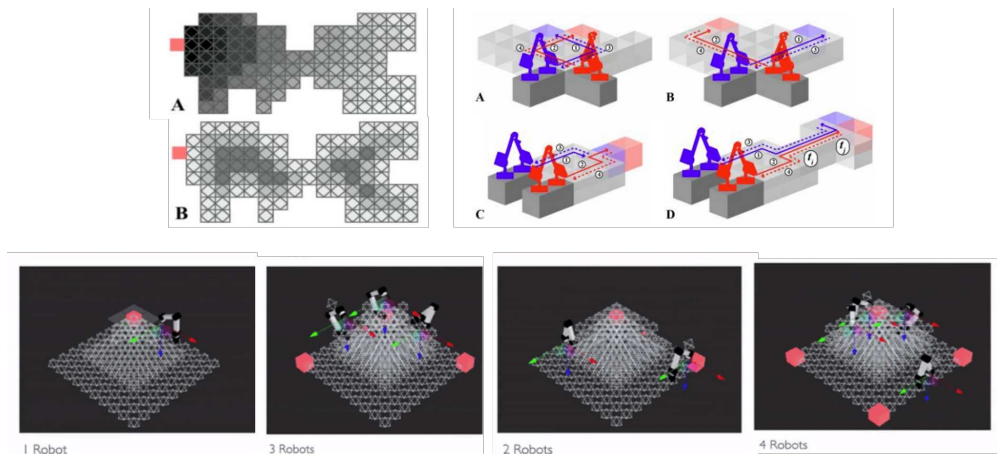


Figure 5.2: Swarm Assembly.

The modular, incremental nature of a discrete material system lends itself to automation. Custom end effectors have been used to demonstrate their automated assembly using a traditional gantry system [70]. For larger applications, mobile robotic platforms, designed to interface with the discrete lattice building blocks [8], have been demonstrated in serial assembly of 1D, 2D, and 3D cellular structures [43] (see Figure 5.1 & 5.2). These relative carrier robots are designed to register to the lattice, enabling large-scale construction without typical penalties for metrology systems, and teams of robots can use the lattice to coordinate their motion. Smaller scale, discrete robotic systems have been demonstrated through the

addition of degrees of freedom and modular actuation components [71]. These systems offer an alternative to the traditional division between monolithic robots (which are capable but not flexible) and modular robots (which are flexible but less capable), however, their numbers and sizes have been fixed. This is a lost opportunity, as these systems have lower performance and throughput with scaling because of the increase in travel time and time wasted in collision avoidance (see Figure 5.3).

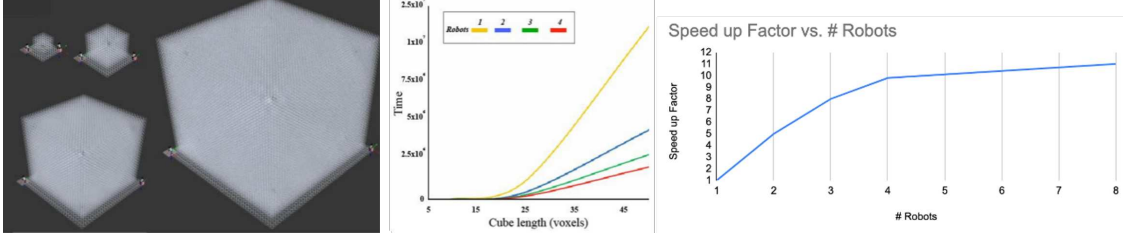


Figure 5.3: Swarm Assembly Scaling and Limitations

In this chapter, I seek to address these shortcomings through the introduction of a discrete material-robot system that is capable of serial, recursive (making more robots), and hierarchical (making larger robots) assembly. I first implement a comparative scaling study of different assembly approaches and compare it to self-assembly throughput in biological systems. We then introduce a hardware system with the required capabilities. Finally, I built upon the algorithm introduced in [43] for serial and swarm assembly and I present a new scalable algorithmic approach for robotic self-replication, shape discretization, path planning and task allocation for recursive and hierarchical assembly.

5.2 Results

5.2.1 Scaling Robotic Construction

In order to evaluate the scaling benefit of recursive and hierarchical swarm assembly strategies in comparison to traditional serial monolithic construction techniques, I will consider the task of assembling a cube of side length 2^N voxels of size d ($N = 1, 2, 3, 4, 5, 6$) from a material pickup station with an automatic voxel feeder located at $x = y = z = 0$ (Figure 5.4 and detailed derivation in Figure 5.5 and its following detailed description¹).

¹To build a X by Y by Z cuboid from a pickup station that is located at $x = y = z = 0$ (see Figure 5.5 A1), a carrier robot that takes 1 timestep to travel 1 voxel will take a total construction time:

$$T = 2 \sum_{z=1}^Z \sum_{y=1}^Y \sum_{x=1}^X x + y + z = XYZ^2 + YZX^2 + ZXY^2 + 3XYZ \quad (S1)$$

When $X = Y = Z = 2^N$:

$$T = 3(2^{4N} + 2^{3N}) \quad (S2)$$

If we have 2 robots, we will assume each robot will assemble a $\frac{X}{2}$ by Y by Z (see Figure 5.5 A2) cuboid hence

$$T = \frac{XYZ^2}{2} + \frac{XZY^2}{2} + \frac{Z Y X^2}{4} + \frac{3XYZ}{2} = \frac{5}{4} \cdot 2^{4N} + \frac{3}{2} \cdot 2^{3N} \quad (S3)$$

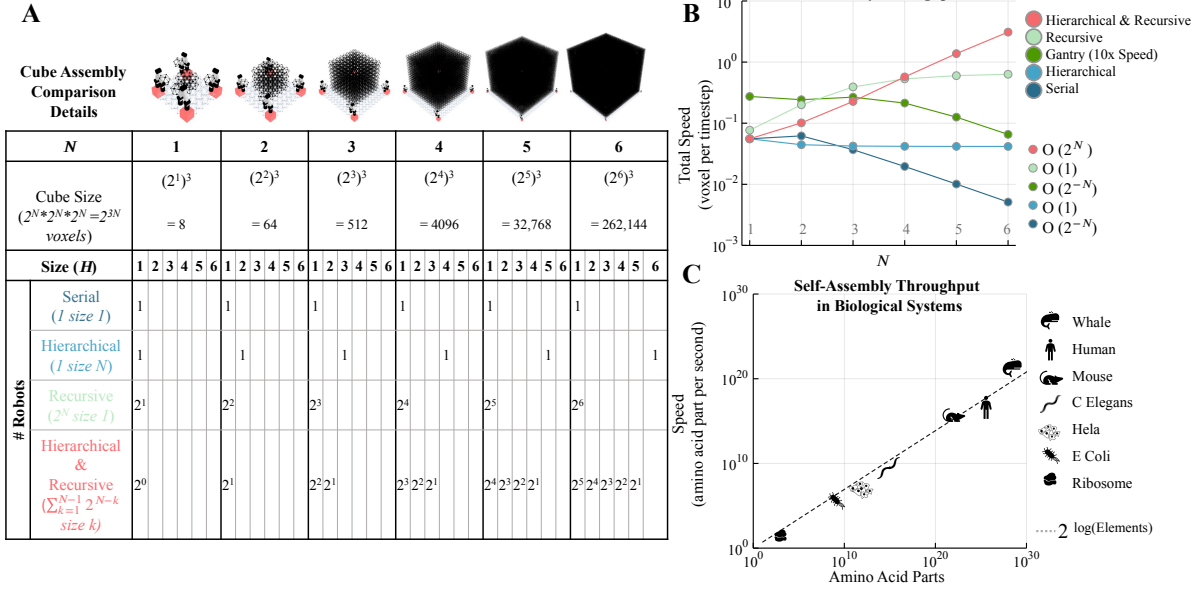


Figure 5.4: **Comparative Analysis of Different Robotic Assembly Strategies vs Nature.** (A) Recursive and hierarchical construction study for building a cubes of side length 2^N with size $N = 1, 2, 3, 5, 6$. (B) Assembly throughput scaling trend of different swarm configurations. (C) Throughput estimates of ribosomal assembly in biological systems [72].

I will start by a base case, a gantry system (similar to [70]); that assembles the voxels layer by layer and travels with the speed of $S \cdot d$ per second. This gantry will take $t_{x,y,z} = \frac{2d}{S}(\max(x, y) + z)$ seconds to place a voxel at location x, y, z (in order to avoid collision, we assume the gantry will move in the z direction first then move in the x and y directions simultaneously); and the total construction time for the cube will be $\sum_{z=1}^{2^N} \sum_{y=1}^{2^N} \sum_{x=1}^{2^N} t_{x,y,z}$ seconds. A much simpler carrier robot (similar to [43]), will move much slower (by a factor of $F \leq 1$) than the gantry system, and will only travel in orthogonal directions, therefore it will take $t_{x,y,z} = \frac{2dF}{S}(x + y + z)$ to place one voxel and the total construction time will be $\sum_{z=1}^{2^N} \sum_{y=1}^{2^N} \sum_{x=1}^{2^N} t_{x,y,z} = \frac{3dF}{S}(2^{4N} + 2^{3N})$ seconds. Even though the gantry system is much faster than the carrier robot, the construction time for both systems is still $O(2^{4N})$. Moreover,

In general for R robots, each will assemble a $\frac{2 \cdot X}{R}$ by $\frac{Y}{2}$ by Z (see Figure 5.5 A4) cuboid, hence:

$$T = \frac{2XYZ^2}{2R} + \frac{2XZY^2}{4R} + \frac{4YZX^2}{2R^2} + \frac{3 \cdot 2XYZ}{2R} = \left(\frac{2}{R^2} + \frac{3}{2R}\right)2^{4N} + \frac{3}{R}2^{3N} \quad (S4)$$

For recursive construction, $R = 2^N$, substituting in the equation S4 $T = \frac{3}{2}2^{3N} + 5 \cdot 2^{2N}$. It is important to note that the maximum number of robots based on the will be $2 \cdot 2^N$, hence $T = \frac{3}{4}2^{3N} + 2 \cdot 2^{2N}$ (still $O(2^{3N})$). For hierarchical construction, robot of size H contains $2 \cdot 3^H$ functional voxels. If it takes 72 timesteps to build a robot of size 1 (substituting $N=1$ in equation S1). The self-assembly time for a robot of size H $\sum_{i=1}^H 2^{i-1} \cdot 2^{3(H-i)} \cdot 72$ and when it takes $\sum_{i=1}^N 2^{i-1} \cdot 2^{3(N-i)} \cdot 72$ timesteps to build a cube with side length 2^N . 2^{i-1} is the slow down factor, and $2^{3(N-i)}$ is the number of cubes/stock needed to build a cube of size N . Finally for hierarchical and recursive construction the self assembly time is $\sum_{i=1}^{N-1} \frac{2^{i-1} \cdot 2^{3(N-1-i)} \cdot 12}{2^{N-1-i}}$ and construction time is $\sum_{i=1}^{N-1} 2^{i-1} \cdot 416$ (416 is the result of substituting $N = 2$ in equation S2).

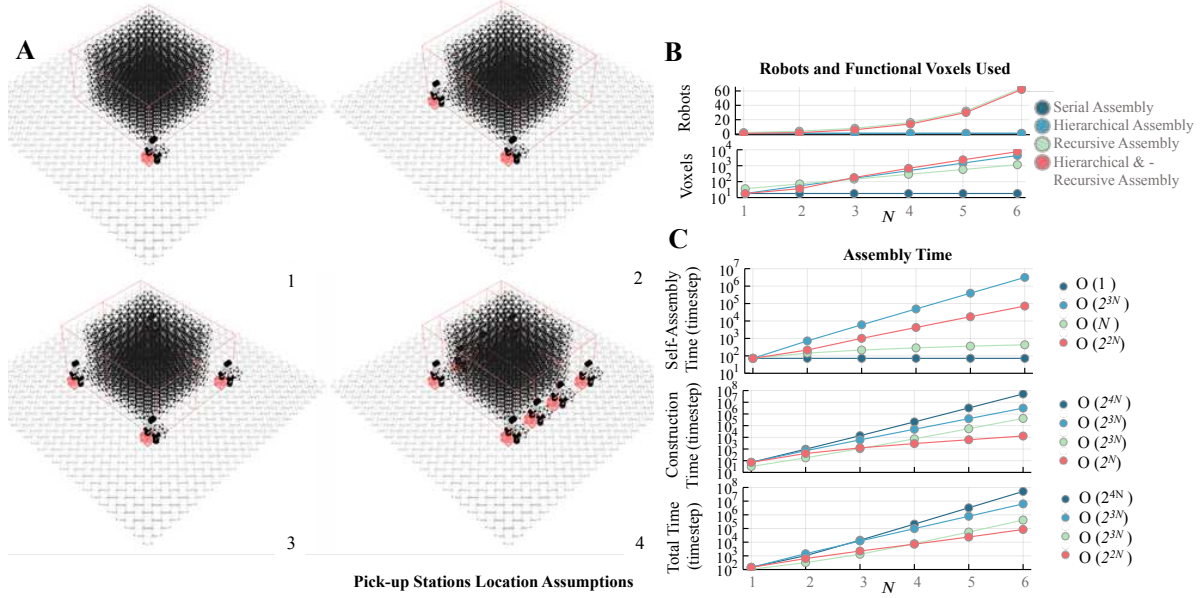


Figure 5.5: **Detailed Assumptions for Comparative Cube Construction Study.** (A) Assumptions for Pickup Location and Building Boundary (1) 1 Robot, (2) 2^1 Robots, (3) 2^2 Robots, (D) 2^3 Robots. (B) Number of Robots and Functional Voxels Used, (C) Self-Assembly, Construction and Total Time.

as N increases, the cost of installing a gantry system that is bigger than the construction site increases greatly while its speed remains constant.

If we however introduce a carrier robot that is not only able to build the target structure, but build identical robots from a modular material building blocks, that means it can recursively build a swarm of robot builders to parallelize the construction. If we assume the number of robots R is equal to (2^N) , and following the assumptions for the pickup stations in figure 5.5, R hence the construction time will be reduced $O(2^{3N})$ and the robot self-assembly time is on the factor of $O(N)$.

The introduction of self-replicating robots will first increase the assembly throughput (total voxels placed divided by total assembly time (cube and self-assembly time)) as N increases. However, a problem arises when N increases, as the carrier robots will travel a long distance from pickup stations to the target voxel placement location (specially in the z direction). That means that efficiency of the system and the throughput of the robots will remain constant (Figure 5.4 B).

Therefore, it is important to introduce modular robots that not only can build identical robots, but can also build larger robots in a hierarchical manner. These larger robots (size H) are able to carry hierarchical building blocks comprised of $2^{3(H-1)}$ voxels, and the small carrier robot can now assemble these blocks near the pickup station. Using this strategy, the bigger robot will take only one trip to the far away target location instead of $2^{3(H-1)}$ trips. If there is only one hierarchical robot of size $H = N$ to build a cube of size 2^{3N} voxels then the total assembly time will be $O(2^{3N})$ and the assembly throughput will remain constant when N increases. If we however apply both recursive and hierarchical assembly strategies, a mix of carrier robots of different sizes having the same number of robotic parts as the

hierarchical (see Figure 5.5(B) for details), the total assembly time is reduced to be $O(2^{2N})$ and the assembly throughput is on the order $O(2^N)$.

Comparing the assembly throughput of the different robotic assembly strategies to self-assembly throughput estimates of ribosomal assembly in biological systems (Figure 5.4(C) [72]) we will find that the self-assembly throughput increases with the number of elements (E), and is of order $O(2^{\log(E)})$. Only by applying self-replication and hierarchy in our robotic construction were we able to reach a similar trend, in comparison to serial assembly where throughput decreased with the number of elements. This proves that self-replication and hierarchy in robotic construction should be integral factors in developing the next generation of robots for large-scale manufacturing.

5.2.2 Material-Robot System

The hardware developed is an extension of the system presented by Jenett et. al [43]. This previous system consisted of passive structural lattice voxels which form a substrate for locomotion of purpose-built inchworm robots capable of rearranging and placing additional voxels. The combination of voxels and robots forms a material-robot system, enabling the precise assembly of large structures with simple robots through localized registration of the underlying lattice. Our system, photographed in Figure 5.6, tightens the connection of the material-robot system by creating a modular robotic toolkit that utilizes active lattice voxels as the primary structural building blocks. Combining these active voxels with actuators, control, and power enables unique behaviors including robotic self-replication and hierarchical robot assembly. A single robot tasked with building a large structure can create a heterogeneous swarm of constructors tailored to maximize material throughput for a given target geometry.

Active voxels that form the basis of the structural-robotic system are a cuboctahedral unit cell (Cuboct) that offers superior structural properties such as high stiffness at low density [6] with geometric characteristics suitable for robotic assembly and fabrication [43]. Specifically, the flat Cuboct faces enable fully-constrained connections between single-voxel pairs with four points of contact, as well as a useful decomposition for voxel fabrication. The voxels are constructed from six sides formed by laminating a printed circuit board (PCB) substrate with an acetal face sheet as shown in Figure 5.6(A). These sides then combine to form the full unit cell in Figure 5.6(B) with soldered intravoxel electrical connections, Figure 5.6(C) and epoxy reinforcement. The fully assembled voxel forms a 101.6 mm (4 inch) cube and weighs 65 grams, resulting in a lattice with an effective density of 0.062 g/cm^3 .

Figure 5.6(D) shows a detailed view of one of four corners involved in a face-to-face inter-voxel connection. A pair of 4.7625 mm x 3.175 mm (3/16" x 1/8") magnets of opposite polarity create an orientation independent structural connection while the 6-pin spring-connector creates a hermaphroditic interface for the three electrical circuits routed through each voxel face: power, ground and a single serial communication line. Each face-to-face connection can transmit 8 Amps at 10 V and 50 N of tensile force. Additional details of the complete fabrication process, from bare PCB to completed voxel are provided in Figure 5.7.

The structural-robotic system is then completed with additional active elements shown in Figures 5.6(E)-(H). Figure 5.6(E) shows a combined control and power voxel which incorporates an ESP32 based microcontroller with a 7.4 volt lithium polymer battery pack. Two rotary actuators, the elbow with rotation parallel to the plane of attachment, and the

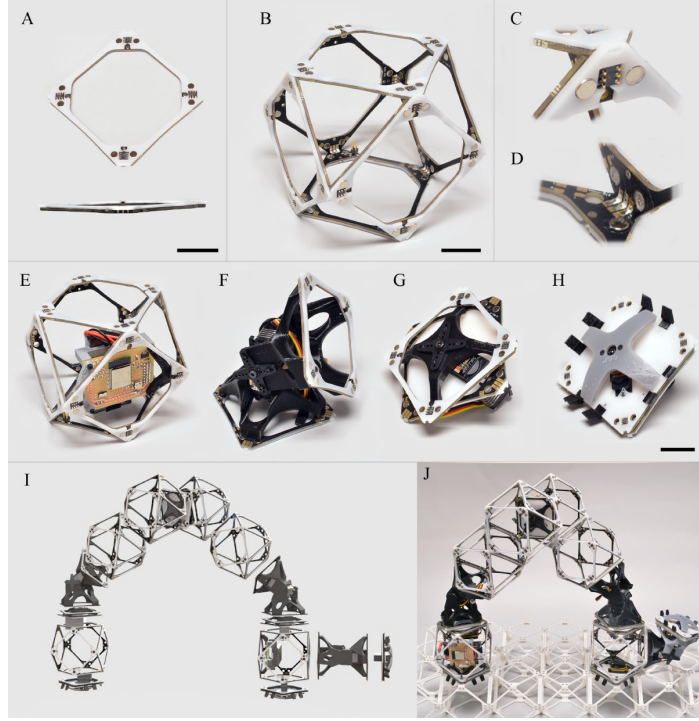


Figure 5.6: **Modular Building Blocks and Assembler.** (A) Voxel face consisting of laminated PCB and acetal overlay (B) Assembled voxel (C) Detail view of hermaphroditic magnetic and electrical connections (D) Detail view of internal voxel electrical connections (E) Control and power voxel (F) Elbow servo-actuated joint (G) Wrist servo-actuated joint (H) Servo-actuated gripper, (I) Exploded view rendering of assembler robot, (J) Photograph of assembler robot on passive lattice. (25 mm scale bars)

wrist with rotation perpendicular to the attachment plane, are shown in Figures 5.6(F) and (G) respectively. Finally, Figure 5.6(H) shows a gripper designed to hold lattice elements for positioning, locomotion and assembly. Each of these actuators incorporates an ATTiny412 microcontroller board for interfacing with the ESP32 serial commands. Details of the actuator assemblies are provided in Figure 5.8²; and a complete bill of materials including weights and costs is provided in Table 5.1.

5.2.3 Robot Self-Replication and Hierarchy

The simplest carrier robot, shown in Figures 5.6(I) and (J) is a discrete voxel analog of the purpose-built relative robot [43]. It uses inchworm locomotion, which registers to the underlying voxel lattice structure using two grippers for navigation. An additional elbow

²Figure 5.8 shows each of the active modular robotic modules broken down into component pieces. Each actuator incorporates voxel faces on both sides for attaching to the lattice. One acetal overlay in the wrist actuator and gripper are modified to mount with the Hitec servos as shown in Figures 5.8(A) and (C). The wrist and elbow actuators use FDM printed PLA for major structural components, while the gripper rotary actuator is printed using SLA for smoother surface finish. All three actuators use ATTiny based circuit boards to interface the PWM controlled servos with the voxel serial bus.

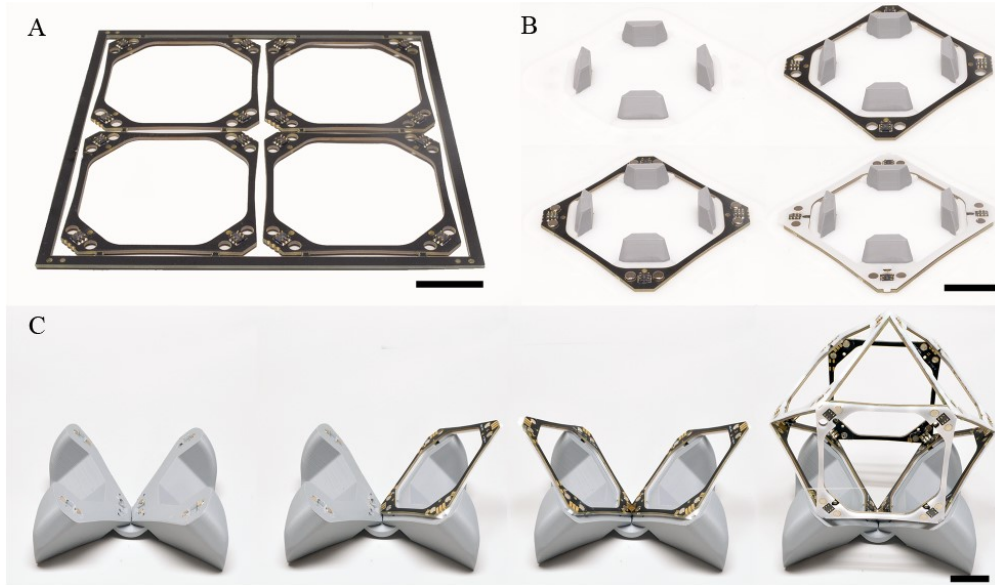


Figure 5.7: **Voxel manufacturing.** (A) Panelized circuit board after attaching smd connectors, (B) Tooling and lamination of voxel side, (C) Tooling and assembly of sides to form complete voxel (25 mm scale bars)

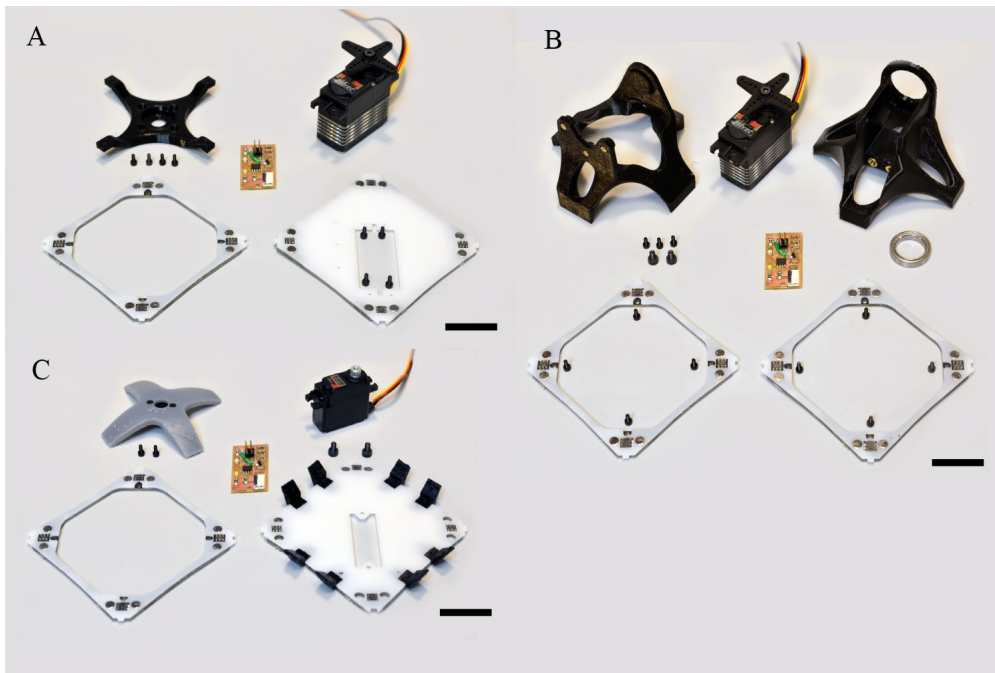


Figure 5.8: **Actuator Components.** (A) Wrist actuator, (B) Elbow Actuator, (C) Gripper (25 mm scale bars)

joint with a gripper allows the robot to manipulate voxels to reconfigure the lattice structure. This robot is capable of carrying a single voxel or actuator at a speed of 1 voxel per time step. While this design forms the basic unit of robotic assembler, the modular system and

lattice compatible connections enable both the construction of additional assemblers by a base assembler, self-replication, as well as the production of larger robots, forming a hierarchy of assemblers.

The algorithm for the robot reproduction is constrained by two requirements. First, the new robot must be free to move after assembly. While this requirement seems trivial, it precludes building flat on the lattice substrate and subsequently lifting the completed robot into position due to the nature of the magnetic connections. To overcome this, child carrier robots are built out from a base consisting of a control voxel and gripper which is capable of anchoring to the lattice. Second, only three components of the robotic construction kit may be directly manipulated by the gripper: base voxels, control and power voxels, and elbow joints. Both the wrist and gripper modules are designed to attach directly to a free voxel face. To accomplish this a carrier robot first picks up one of the first three part types, then maneuvers to attach wrists and grippers to the base part before placing in the assembly in a process we call accessorizing. The grippers have electrical interfaces so that actuators, including elbows, wrists and grippers, may be controlled by the assembler. This self-replication process is shown in Figure 5.9(A), with detailed steps outlined in Algorithm 1.

Algorithm 1 Self-replication Sequence

- 1: **Pick up** control and power voxel (c_1)
 - 2: **Accessorize** with gripper
 - 3: **Anchor** to substrate with gripper (g_1)
 - 4: **Pick up** elbow actuator (e_1)
 - 5: **Accessorize** with wrist actuator (w_1)
 - 6: **Attach** to control and power voxel (c_1) via wrist actuator (w_1)
 - 7: **Pick up** voxel (v_1)
 - 8: **Accessorize** with wrist actuator (w_2)
 - 9: **Attach** to elbow actuator (e_1) via voxel (v_1)
 - 10: **Pick up** voxel (v_2)
 - 11: **Attach** to wrist actuator (w_2)
 - 12: **Pick up** voxel (v_3)
 - 13: **Accessorize** with gripper (g_2)
 - 14: **Anchor** to substrate with gripper (g_2)
 - 15: **Pick up** elbow actuator (e_2)
 - 16: **Accessorize** with wrist actuator (w_3)
 - 17: **Attach** to voxel (v_3) via wrist actuator (w_3)
 - 18: Control voxel (c_1) **moves** voxel (v_2) to **connect** to elbow actuator (e_2)
 - 19: Child Bill-E now capable of lattice locomotion
 - 20: **Pick up** elbow (e_3)
 - 21: **Accessorize** with gripper (g_3)
 - 22: **Attach** elbow (e_3) to voxel (v_3)
-

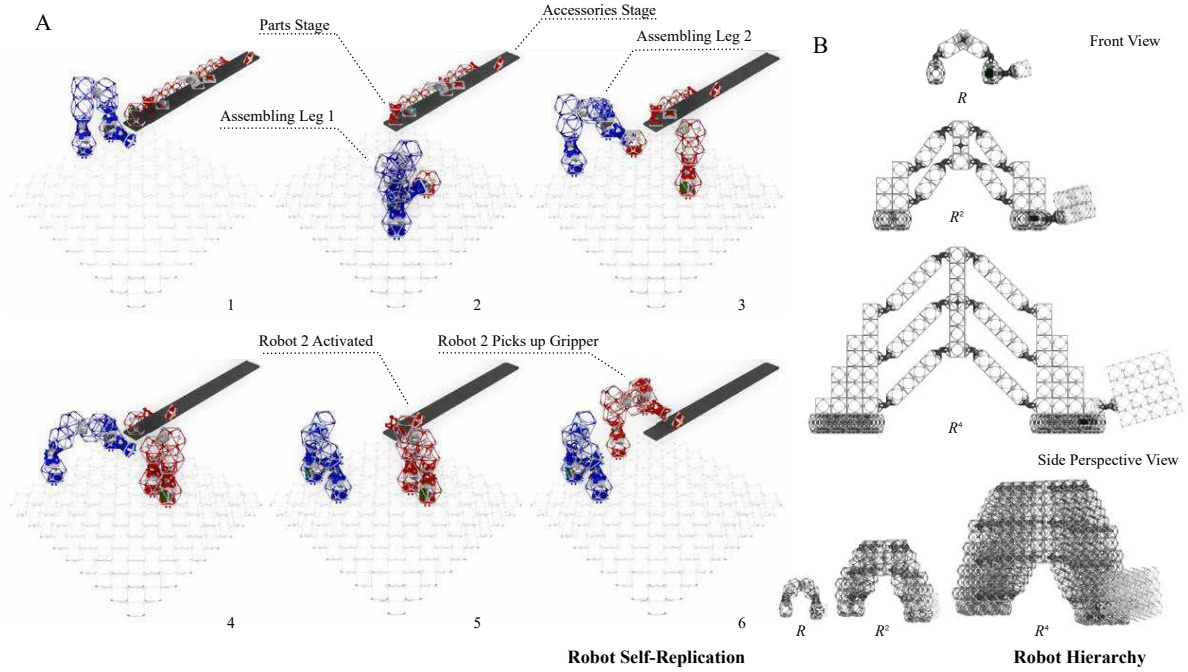


Figure 5.9: **Robot Self-replication and Hierarchy.** (A) Image stills in sequential order (1-6) of robot self-replication simulation according to the assembly sequence in Algorithm 1. (B) Proposed design for carrier robots of size R ($H = 1$), R^2 ($H = 2$) and R^4 ($H = 3$) that can carry hierarchical cubic building blocks of size 1, 2 and 4 respectively.

Conceptual drawings of a family of assembler robots, designed to carry voxel cubes of side lengths one two and four respectively, are shown in Figure 5.9(B) and 5.10. The larger designs maintain the inchworm architecture, while employing four-bar linkage assemblies to enable multiple actuators to act in parallel on a single degree of freedom. These early studies indicate that the number of actuators scales with a factor of 2 when H increases, while the total number of robotic elements scales with a factor of 2.5, lower than the geometric scaling of 3 due to tailored sparsity of the larger lattice structures. Additional gains in actuation and structural efficiency may be possible by exploring additional morphologies beyond the inchworm configuration.

5.2.4 Inverse Problem: From Structure to Swarm

Using the proposed material-robot system, I will next present algorithms that determine the optimal swarm configuration and path planning needed to build a target structure. First, I will discuss the baseline assumptions behind the path planning and simulation of the self-replicative and hierarchical swarm construction model. I am building on the assumptions made in [43] where we assumed a centralized path planning and control strategy. The centralized control system is responsible for (1) processing and discretizing input target geometry into building blocks, (2) determining the building blocks build sequence, and (3) the task allocation and path planning based on the number of available robots. The carrier robots, on the other side, are intended to have minimal communication and computational

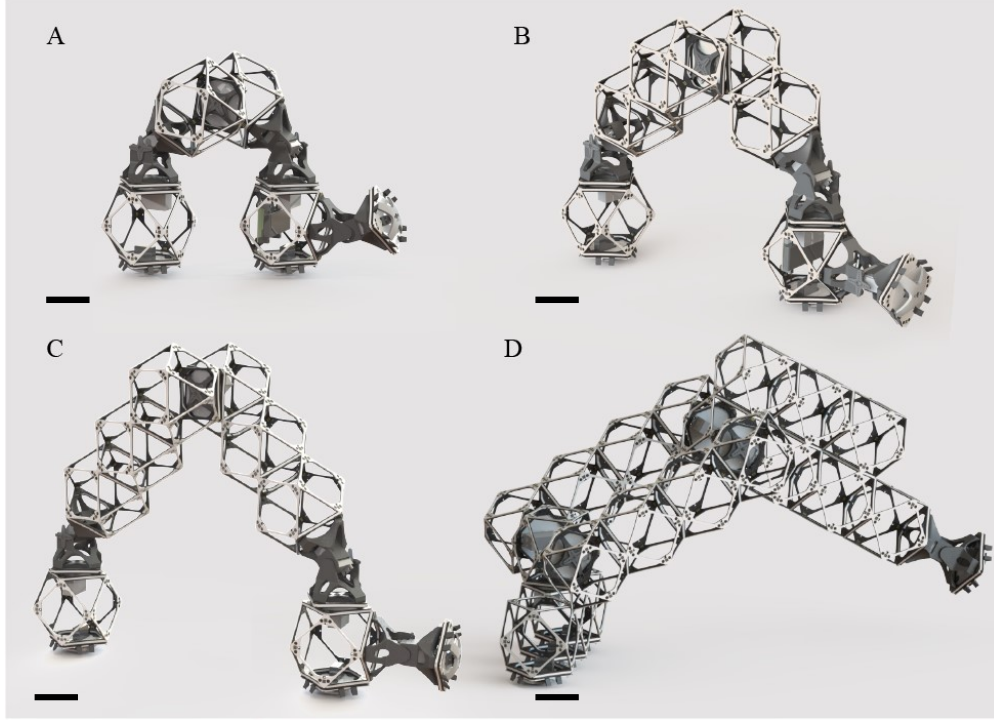


Figure 5.10: **Hierarchical Robots Design Space.** Example of carrier robots showing the design degrees of freedom (using the discrete robotic toolkit), either changing length or member stiffness. Example of carrier robots able to travel **(A)** one voxel per step, **(B)** 2 voxels per step, **(C)** 3 voxels per step, **(D)** or carry more passive or functional voxels (concept art showing stiffness can get added programmatically). (50 mm scale bars).

requirements. They can only communicate with the centralized system at the pickup stations and are designed to receive and conduct one task at a time. A task entails picking up a building block from the pickup station in the form of a list of high-level step instructions towards the drop-off location (step forward, right, left, up, down, concave or convex turn) [8], as well as the steps needed to go back to the same or a closer pickup location. The carrier robot only stores the mapping for the high level steps to robot position, hence, memory requirement of the carrier robots depend on the size of the target build structure, i.e. based on the furthest voxel expected to be placed. In order to make the system more robust and resilient to uncertainties and disturbances, the robots could be equipped with tactile sensors that ensure that the instructed step is valid (the voxel exists and is empty) before the robot places its leg to take a step [43]. If there is a discrepancy, an emergency protocol is initiated where the robot takes the reverse steps to go back to the pickup station to notify the central system; or in case this scenario is interrupted it will stay in place and fire an emergency signal to the central system for other task-specific robots to fix the problem.

The presented algorithm assumes synchronous time stepping, where each robot spends S seconds (one timestep) to either take a step or pick up/drop off a voxel. For larger hierarchical robots, because of actuation limits, their speed is set to 2^{H-1} timesteps per step. The centralized system has a spatial-temporal Markov model of the system, where it predicts and stores the state of the system (voxel occupancy and robot locations) at each timestep in

the future based only on the current state (for simplicity). For each building block in the queue to be built, the centralized system greedily finds the shortest path from the pickup station to the target drop-off location, based on an A* tree search algorithm, avoiding collision with other robots that already have tasks assigned. The carrier robots' clocks are synced with the central system when they return to the pickup station, which also function as charging stations. Based on the expected fault tolerance T in the system synchronicity as well as the expected maximum steps taken, the central system adjusts the collision avoidance strategy introducing a safety factor where it makes sure, based on the spatial-temporal model, the robots do not collide within $\pm T$ timesteps.

The algorithm is intended to be flexible, as the number of robots in the system and pickup stations is not assumed to be fixed. During the construction, one can add/remove a robot, or a pickup station as needed, and the centralized robot will adjust its spatial-temporal model based on the new changes. Verification of the structural integrity of the lattice while being built (with imposed static and dynamic loads of robots) will be left for future work, with numerous existing precedents of solutions to this problem [70]. However, this verification will not alter the relative construction time and the speedup that recursive and hierarchical assembly offer.

Building Sequence

The centralized system's first step is to automatically generate the build sequence given target shape. I present a scalable adaptive shape compiler that discretizes the input geometry into ordered hierarchical building blocks. The target built shape is input from the user in the form of a connected mesh, and the compiler generates its signed distance field (SDF), which represents the closest signed distance from any point in the 3D space to the surface of the mesh. We used the Gilbert-Johnson-Keerthi method [73] to calculate the distance field using Minkowski difference for convex meshes, and the approach presented in [74] for non-convex meshes. I used octree space subdivision to calculate the adaptively sampled distance field (ASDF) [75], which subdivides the mesh into variably sized building blocks, that are larger at the center of the mesh and smaller near the surface. Minimum and maximum building block sizes are prescribed based on the target build resolution and maximum desired carrier robot size.

The next step is to determine the optimum sequence to assemble the building blocks, while avoiding any deadlocks in the construction. The target is to divide the building blocks into bins, whereas all blocks that belong to the same bin can be built in parallel (see Figure 5.11). Since all the pickup stations are assumed to be located at the ground plane of the construction site, the structure is set to be built layer by layer, from bottom to top, which restricts the family of structures that can be built to ones that do not have underhangs. Before starting the construction, the robots build a substrate layer with an offset around the built area. Next, for each layer, the larger building blocks are assembled first since they inherently have higher SDF values. The building blocks with the same size are then grouped into bins according to their centers' SDF value (with a tolerance that is proportional to the building block size, see Figure 5.11(B)). Figure 5.11(C) visualizes the build sequence tree for an example cone mesh (radius 10 voxels and height 16 voxels).

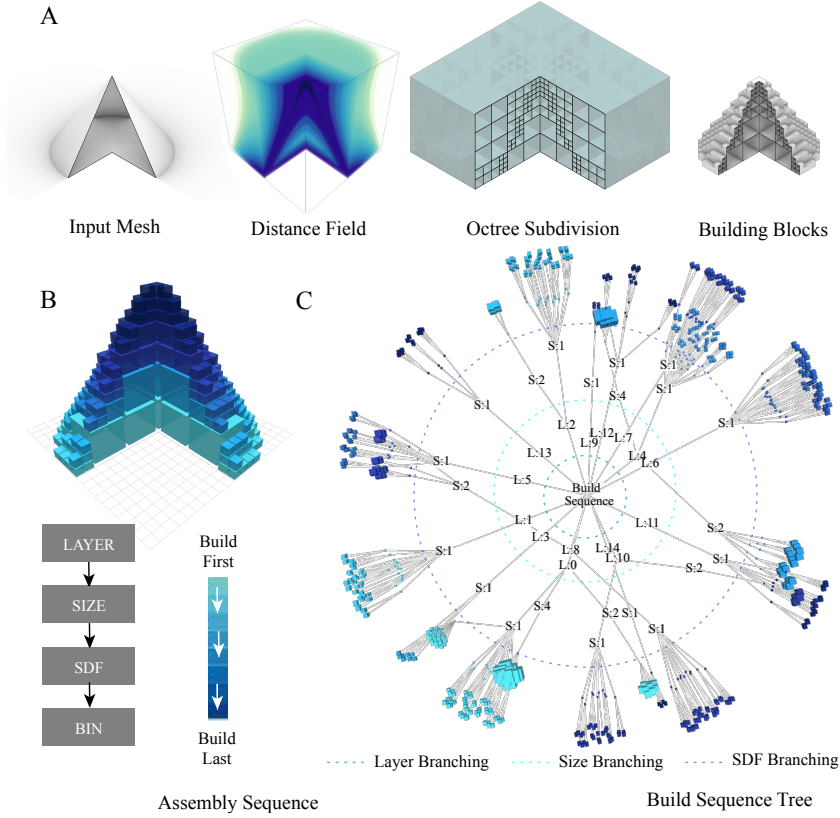


Figure 5.11: **Adaptive Shape Compiler.** (A) Hierarchical shape discretization steps using adaptively sampled distance fields (ASDF). The distance field is colored based on the distance to the input mesh surface. (B) The hierarchical building blocks are colored according to the assembly sequence. The blocks are ordered into bins first according to their layer number (L_0 to L_{14}), then their size (S_1 , S_2 , and S_4), and finally their signed distance function (SDF) value, (C) Visualization of the final build sequence tree, all blocks that belong to the same bin can be built in parallel.

Swarm Configuration and Task Allocation

Now that the adaptive shape compiler generated the build sequence tree, the central control system can search for the optimal swarm configuration: the number of carrier robots in each size that is able to build the structure in a minimum number of timesteps. Inspired by a biological analogy, where a carrier robot has the option to either to build the structure (build), build another robot (reproduce), or build a bigger robot (evolve), I generate the tree of all possible alternatives, given a finite amount of robotic building blocks stock (see Figure 5.12). Next, I traverse the tree searching for the configuration that will take the minimum amount of timesteps to be built (time to build both the robots and the structure). Once the optimal number of robots is calculated, the central system sends out instructions for the first carrier robot to assemble the rest of the robots in a recursive manner.

When going beyond homogeneous cubes as target built structures, it is important to note that the target shape morphology and resolution greatly affects the benefit of hierarchical and recursive construction strategies. Figure 5.13 shows contour plots of the swarm specific

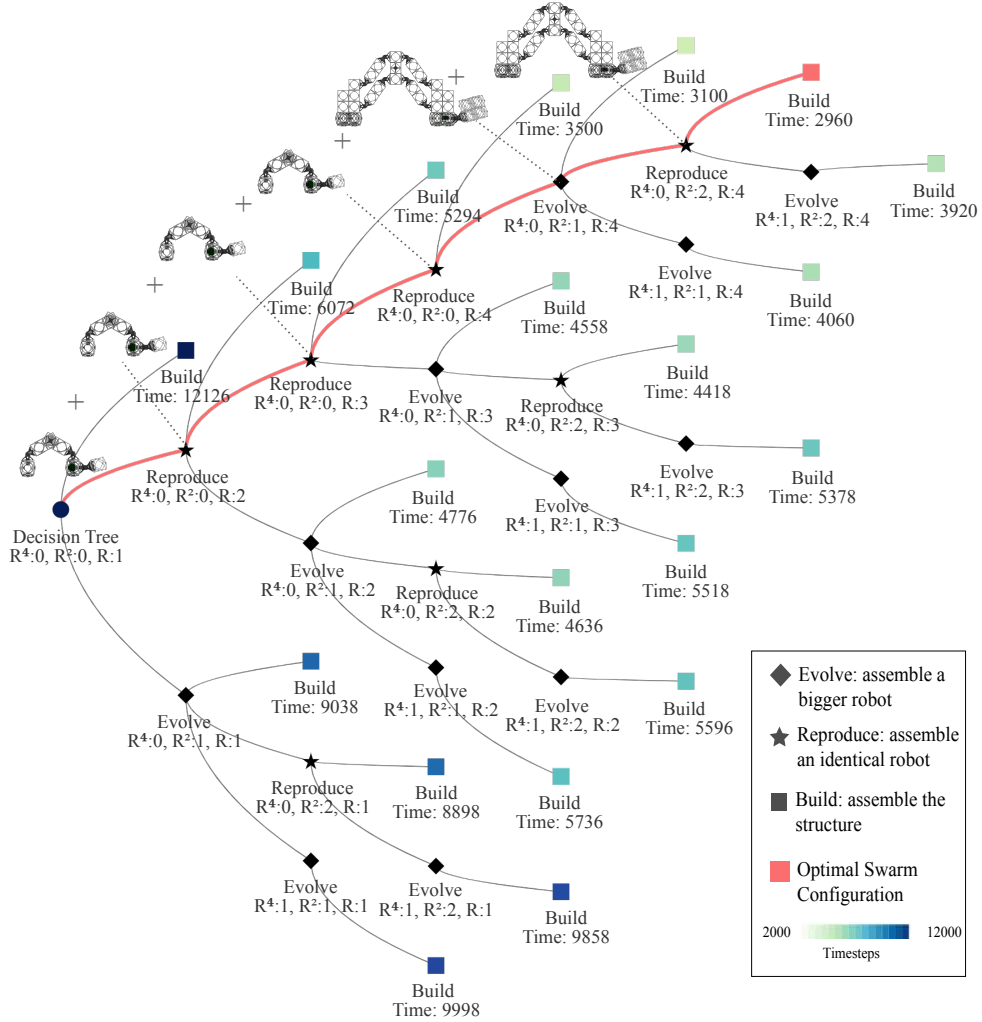


Figure 5.12: **Robot Decision Tree.** To determine the optimum swarm configuration (number of robots and their sizes), each robot can either build an identical robot (reproduce), a bigger robot (evolve), or start construction (build). I then traverse the tree searching for the optimal configuration (smallest build time given a maximum number of robots). This is the tree generated to build the example cone in Figure 5.11(B), with a maximum of 4, 2, 1 robots of sizes R ($H = 1$), R^2 ($H = 2$) and R^4 ($H = 3$) respectively.

speedup for a range of heterogeneous swarms building the reference cone at four different resolutions (smoothness scale from 0 (no octree subdivisions, the final voxelized structure is just the bounding cube) to 100 (very fine subdivisions where the final voxelized structure is identical to the input mesh)). We define the specific speedup as the construction speedup (compared construction with one robot of each size) divided by the effective number of robots (total number of robotic parts in the system). The swarm configurations with the highest specific speedup are the ones that are most optimal in terms of construction throughput per unit assembly robot.

At high resolutions the most efficient swarms employ only a single R^4 ($H = 3$) robot as more of the build time is consumed placing detailed features. Conversely, the swarms

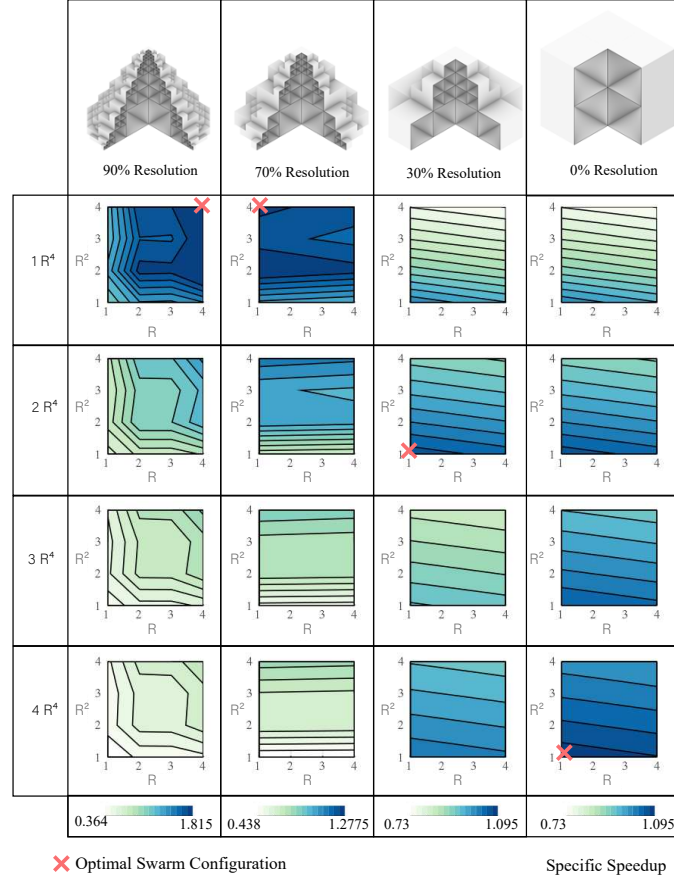


Figure 5.13: **Specific Speedup and Optimum Configuration.** The specific speedup, a measure of construction efficiency, is calculated (for the target shape (cone) with different resolutions using 1, 2, 3, and 4 robots of size R ($H = 1$), R^2 ($H = 2$) and R^4 ($H = 3$)) by dividing the construction speedup (compared to construction with 1 robot of each size) by the effective number of robots (total number of robotic parts in the system). The dark blue color suggests a higher specific speedup.

with the highest specific speedup for the coarse geometries consist of only a single R and R^2 ($H = 2$) robot with multiple of the largest R^4 ($H = 3$) robots. In all cases the optimal swarms outperform a single assembler robot in throughput per assembler by between 9 and 80%. Absolute construction speedup is shown in Figure 5.14 where we see that larger swarms always build the target structure faster, though at the cost of decreased efficiency in terms of speedup per robot³.

³Figure 5.14 shows the total construction time needed to build a cone with different resolution using a variety of swarm configurations. For the high-resolution structure, changing the number robots of size R and R^2 (self-replication) greatly alters the build time, while adding more R^4 robots (hierarchy) has minimal effect on the total build time. For the 70% resolution structure, the total construction time of the intermediate is insensitive to adding robots of size R , but decreases with additional robots of size R^2 and R^4 . For the two coarsest structures, the benefit of adding R^4 robots greatly exceeds the benefit of adding additional

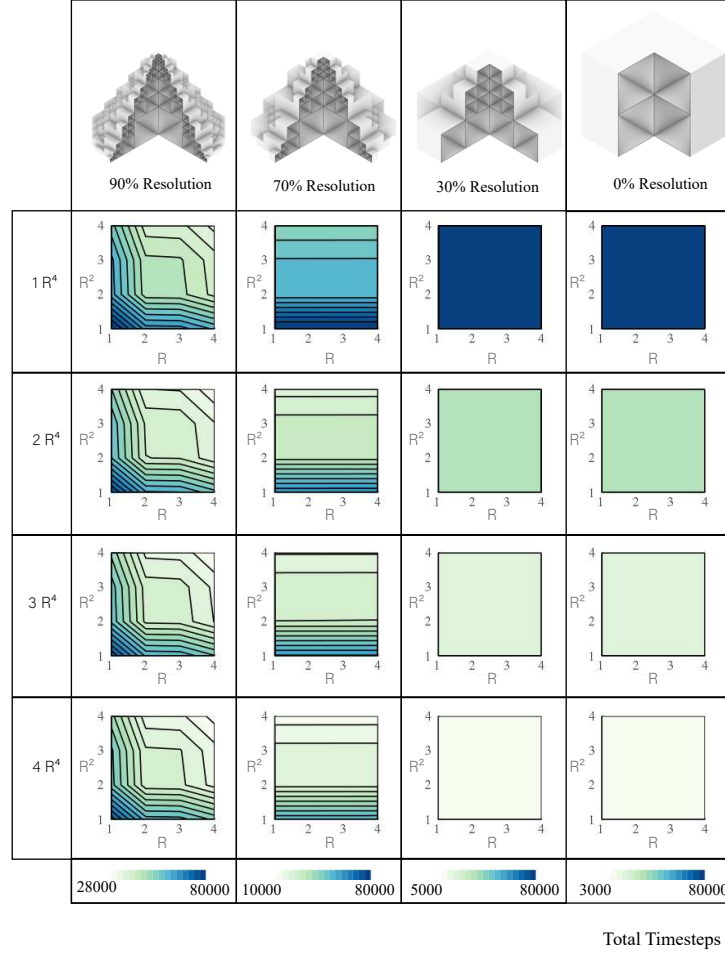


Figure 5.14: **Effects of Structure Resolution on Total Construction Time..** Total construction time needed to build a target shape (cone) with different resolutions using 1, 2, 3 and 4 robots of size R , R^2 and R^4 .

Finally, now that the robots are built, the construction of the target structure begins. A pseudo-code for the task allocation and planning strategy is presented in Algorithm 2. The assembly happens in a hierarchical manner, where smaller robots build stock in situ (beside the pickup station) for the bigger robots to place at their final destination (see Figure 5.15). In order to test the presented algorithm, I have developed a simulation environment (details in Materials and Methods section). A full simulation of the swarm construction of the cone is illustrated in Figure 5.11(B).

robots of size R and R^2 . Finally, even though the coarse resolution structure's volume is larger than the fine resolution one, it can be built in a faster time using the same number of robots because the robots have higher throughput, because they take leverage of hierarchical construction. This stresses on the importance of the reconfigurability of robotic construction swarms and their ability to self-assemble to optimum configurations based on the size and morphology of the target built structure.

Algorithm 2 Task Allocation

```
1: for each bin  $B$  in the Building Sequence Tree do
2:   while  $B \subset$  building block  $b$  do
3:      $S \leftarrow$  size of the building block  $b$ 
4:     if there is robot same size  $S$  in the system then
5:       if there is available stock same size  $S$  then
6:         if robot  $R$  same size  $S$  in at any pickup station then
7:           Assign  $R$  to pick-up stock and place it at target location
8:           Remove  $b$  from bin  $B$ 
9:         end if
10:      else
11:        Add stock (8 size  $S - 1$  blocks) in front of the bin  $B$  to be built in situ
12:      end if
13:    else
14:      Divide  $b$  into 8 size  $S - 1$  blocks & add to  $B$  to build at new target locations
15:      Remove  $b$  from bin  $B$ 
16:    end if
17:  end while
18: end for
```

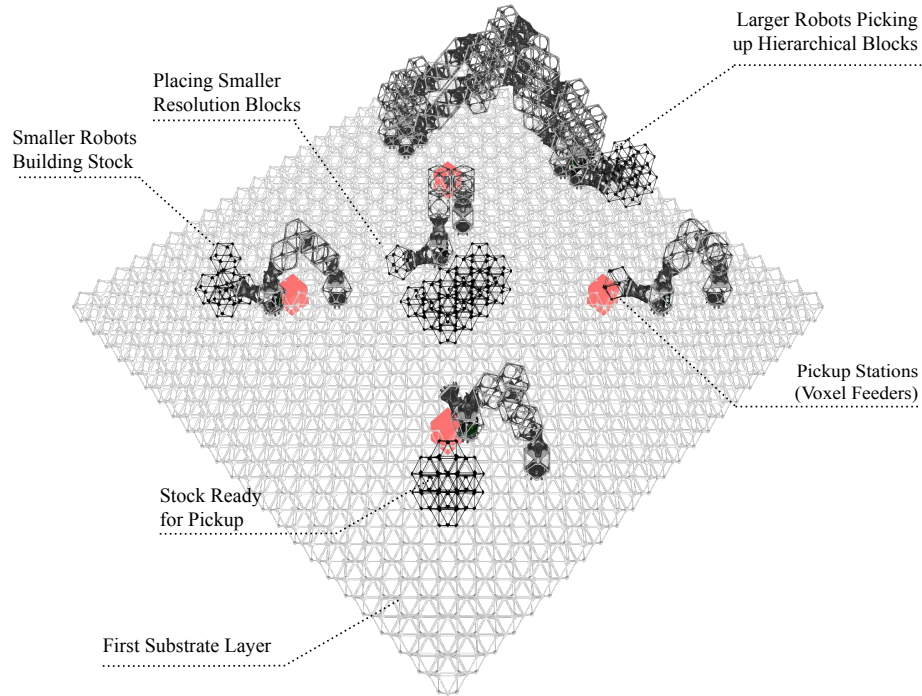


Figure 5.15: **Hierarchical Construction.** Example construction site where smaller robots build hierarchical building blocks near the pickup locations for the bigger robots to pick up and place them in the final destination.

5.3 Methods

5.3.1 Hardware Fabrication and Components

The material robot system employs a variety of off-the-shelf and custom hardware. A complete bill of materials is provided in Table 5.1. Figure 5.7 illustrates the assembly process for a single voxel face and a complete voxel. Voxel circuit boards were printed by PCBWay custom prototyping using 1.6mm FR4 substrate and 1oz copper layers. Acetal faces were lasercut using a Trotec Speedy-100 Flexx and laminated to voxel circuit boards using Loctite SF-770 primer and Loctite 401 adhesive. Figure 5.8 shows disassembled actuator assemblies. The elbow and wrist actuators use Hitec D950TW servo motors while the gripper is actuated by a smaller Hitec HS-5087MH servo. Elbow and wrist components produced by fused deposition modeling of PLA using Prusa MK3 printers. The gripper rotary actuator was 3D printed using stereolithography of Formlabs Tough1500 resin and the Form 3 printer. The ESP32 control board and servo serial interface boards were milled on a Roland MDX-20 milling machine.

5.3.2 Voxel Manufacturing

The voxel manufacturing process is shown in Figure 5.7. The PCBs arrive from the fabrication house in a 2x2 panelized board. The molex connectors are soldered to this board as shown in Figure 5.8(A) before separating the boards. Figure 5.7(B) shows the voxel side lamination process. Tooling was built to locate the PCB and ensure the magnets were inserted with the correct polarity. With the PCB and magnets in place, the acetal face is activated with Loctite SF770 before bonding with Loctite 401 cyanoacrylate. Figure 5.7(C) shows the assembly of the individual sides into a completed voxel. The magnetic jig is used to hold pairs of sides at a right angle during soldering.

5.3.3 Simulation Environment

I have developed a dedicated robotic swarm simulation environment, a digital twin, to test the presented material robot system. The simulation tool was coded in JavaScript and Three.js (<https://threejs.org/>) was used for 3d visualization. First, each component of the discrete robotic toolkit CAD file is imported and assembled in place based on the robot's parameters (leg and arm length and robot width). An interactive environment was created where one can explore these parameters and visualize the robot design space. Second, I derived the inverse kinematics model for the carrier robot (dependent on the robot parameters) to get the prescribed sequence of motor angles for each of the high-level robot steps. The robot then can go from one location to another when given an array of these steps. Finally, another layer of abstraction was added where the robot can be prescribed to pick up a voxel from a pick-up location at x_p, y_p, z_p to another target location at x_t, y_t, z_t , and the list of high-level steps needed is automatically calculated. Using these abstractions, the self-replication demo (Algorithm 1) was implemented.

As for the hierarchical path planning and assembly, the list of robots with their sizes and initial positions is prescribed as an input as well as the location for the voxel pickup

Table 5.1: Robot Material System Bill of Materials.

Voxel Face			
<i>Component</i>	<i>Weight [g]</i>	<i>Price [USD]</i>	<i>QTY</i>
Voxel PCB 1.6mm 1oz Cu	4.9	\$1.89	1
Molex 788641001 Connector	0.04	\$0.18	4
3/32" Diameter x 1/8" N48 Magnets	0.41	\$0.12	4
Acetal Overlay, 1.6mm	4.2	\$0.40	1
Totals	10.9	\$3.49	10
Voxel			
<i>Component</i>	<i>Weight [g]</i>	<i>Price [USD]</i>	<i>QTY</i>
Voxel Face	10.9	\$3.49	6
Totals	65.4	\$20.94	6
Elbow Actuator			
<i>Component</i>	<i>Weight [g]</i>	<i>Price [USD]</i>	<i>QTY</i>
Voxel Face	10.9	\$3.49	2
Hitec D950TW Servo	68	\$110.49	1
ATTiny412 Servo Serial Board	2.8	\$0.45	1
FDM Active Arm	23.4	\$0.98	1
FDM Passive Arm	26.7	\$1.12	1
Thin section bearing	3.5	\$1.89	1
m3 Heat Set Insert	0.475	\$0.13	2
m2 Heat Set Insert	0.18	\$0.14	8
m3x5 SHCS	0.65	\$0.09	2
m2x5 SHCS	0.225	\$0.14	8
Totals	151.69	\$124.55	27
Wrist Actuator			
<i>Component</i>	<i>Weight [g]</i>	<i>Price [USD]</i>	<i>QTY</i>
Voxel Face	10.9	\$3.49	2
Voxel Face (Wrist)	16.3	\$3.49	1
Hitec D-950TW Servo	68	\$110.49	1
ATTiny412 Servo Serial Board	2.8	\$0.45	1
FDM Active Arm	7	\$0.15	1
m2 Heat Set Insert	0.18	\$0.14	4
m2x5 SHCS	0.225	\$0.14	8
Totals	118.42	\$123.22	18
Gripper			
<i>Component</i>	<i>Weight [g]</i>	<i>Price [USD]</i>	<i>QTY</i>
Voxel Face	10.9	\$3.49	1
Voxel Face (Gripper)	16.4	\$3.49	1
Hitec HS-5087MH Servo	21.9	\$35.27	1
ATTiny412 Servo Serial Board	2.8	\$0.45	1
SLA Gripper	7.2	\$1.05	1
FDM Locating Bosses	0.52	\$0.02	8
m3x5 SHCS	0.65	\$0.09	2
m2x5 SHCS	0.225	\$0.14	2
Totals	65.11	\$44.37	17
Control And Power Unit			
<i>Component</i>	<i>Weight [g]</i>	<i>Price [USD]</i>	<i>QTY</i>
ESP32 WROOM-32D PCB	13.9	\$3.80	1
7.4V, 1 Ah Li Polymer Battery	54	\$12.99	1
FDM board support	1.69	\$0.04	2
FDM battery support	2.69	\$0.06	2
m2 Heat Set Insert	0.18	\$0.14	6
m2x5 SHCS	0.225	\$0.14	6
Totals	79.09	\$18.66	18

stations. Given an input mesh of the target shape to be built, the adaptive shape compiler outputs the building sequence tree which is a nested ordered list of bins ordered according to their layer number, block size and SDF value. Each bin is a unordered list of blocks

with their corresponding x_t, y_t, z_t location. Finally, the planning and assembly of the path is implemented according to the Algorithm 2, and an abstracted CAD model was used for the representation of the robot.

5.4 Discussion

I have introduced a material-robot system that extends the concept of robotic swarms by being able to construct serially, recursively (making more robots), and hierarchically (making larger robots). This freedom in component composition is supported by algorithms for compiling shapes into swarms for constructing them and for planning their paths, and I evaluated their scaling through simulation showing the geometry dependence of these heterogeneous swarms. The hardware presented is an instantiation of a collective robotic system capable of recursive and hierarchical construction. It is used as an exemplary embodiment to do the scaling calculations/analysis presented, and it serves as a guideline for the requirements needed for future development. Future work will combine these contributions at scale. The system shown uses magnetic connections between the voxels. These are sufficient to demonstrate robotic assembly and locomotion, but joints with improved stiffness and strength will be required for continuum metamaterial behavior in higher-performance structures [6]. For ease of automation recent work has demonstrated that this can be achieved with a captive, interlocking, androgynous fastener [76], and other possible solutions include electro-permanent magnets [77] and bistable latches [78].

For actuation we used commercially available digital servo motors, with an average net force output of roughly 57 N/kg and upper torque limits of around 10 N · m [79]. This granularity limits the system scalability, which could be addressed through the serial and parallel addition of one-bit actuators [71]. Beyond self-replication and hierarchy, this system also enables a large design space in robot morphology. Preliminary work has shown a range of kinds of locomotion through distributed deformations rather than rigid joints. Robot design tools, such as those for 3D printed robotic creatures [22, 80], will be needed to simulate and optimize over these expanded degrees of freedom.

The algorithms presented are centralized; scalable compilers [81] and decentralized control strategies [82] will be needed as the system size grows. These algorithms provided constructive demonstrations but are not shown to be optimal; more complex path planning and collision avoidance strategies could be implemented to decrease the construction time [45], and exploring the number and location of pickup stations is an important design parameter that greatly affects the robots' throughput [43]. Finally, simulation to determine stress and deflection of the structure being built under static and dynamic robot loading can be pre-computed or monitored in real time for larger structures [83].

These swarms have a range of potential applications in areas that today either require substantial capital investments for fixed infrastructure or are infeasible; candidates include seismic metamaterials [84], automobile assembly lines [85], aircraft subsystems [86] and airframes [87].

5.5 Recap

The introduction of self-replication and hierarchy into robotic swarms challenges the historical correlation between large objects needing large machines to make them. The ability of this system to span both scale and number offers the potential for exponential increase in capacity when it is needed, and, just as importantly, an exponential reduction when it is not. Sharing the same underlying principles, this promises to bring the dynamic range of biological growth into manufacturing processes.

Chapter 6

Shape-Control Optimization of Reconfigurable Digital Material Robots

6.1 Introduction

Beyond fabrication, I next explore how these robotic agents can be embedded within structures to enable reconfiguration over their lifetime. This chapter expands the scope of inverse design from static structures to dynamic systems that can adapt their morphology and behavior in response to environmental or programmatic changes.

Modular reconfigurable robots offer lots of advantages in contrast to traditional monolithic robots. They are viewed to be more versatile, adaptable to different tasks if needed, simpler hence faster to manufacture, cheaper and more cost-effective, and more robust as modules can be replaced easily in case of maintenance or damage [88], [65], [89]. Modular self-reconfigurable robots are categorized into different categories based on their geometrical features and arrangements [89], mobile robots, chain-types, lattice-type, and hybrid structures where the modules could either assembled into lattices or chains, similar to the reconfigurable modular system I am introducing here. These robots' reconfiguration could be categorized into manually reconfigurable and self-reconfigurable systems [65]. The self-reconfigurable systems are usually costly/not scalable as each module is very complex and assumed to be able to navigate on or around the structure and dock/attach to it. In my approach to self-reconfiguration, I assume the existence of an assembler robot [43], [44] that is made of the same simple modules it hence capable of self-replication [44].

The co-optimization of the shape (body) and control (brain) of modular robots has been a long-standing goal, to create artificial life [90]. This problem is viewed to be intractable because of the high dimensionality of the design space and ways to approach the problem. It requires finding an efficient, yet expressive, design representation for the robots and their controls, as well as appropriate search methods that balance exploration and exploitation.

The choice of modular, higher-level components, and limited architectures has been widely used because, even though it constricts design freedom, it expedites convergence as the design space becomes more tractable. Grammars (linear grammars [90], [91], graph grammars [91], [92], [93], spatial grammars [94]) have been used as representations to express design rules and inherently include physical and fabrication constraints. As for the control, different

methods have been used, either evolving centralized controllers [92], distributed controls, fixed manual controls [94], [95], joint control [96], or reinforcement learning [97].

As for the search methods, evolutionary algorithms strategies are widely used [98], [99] but are resource-extensive, do not guarantee optimality (as they often lead to local minima), and their results are not reproducible with subsequent runs. Recent approaches use graph search and deep learning for learning and predicting heuristics for an efficient search process to reduce the number of expensive evaluations/simulations [92], [95].

Finally, recent studies showed that the search method is not the only factor that affects the quality of the search, but also the timing of the search/evolution process. In most of the cases mentioned above, the development / search process is performed before the start of the evaluation process, but recent findings [100], [101] show that continuing the development process where the robot reconfigures while interacting with the environment can result in more efficient robot designs.

6.2 Results

6.2.1 Modular Robot Design Tool

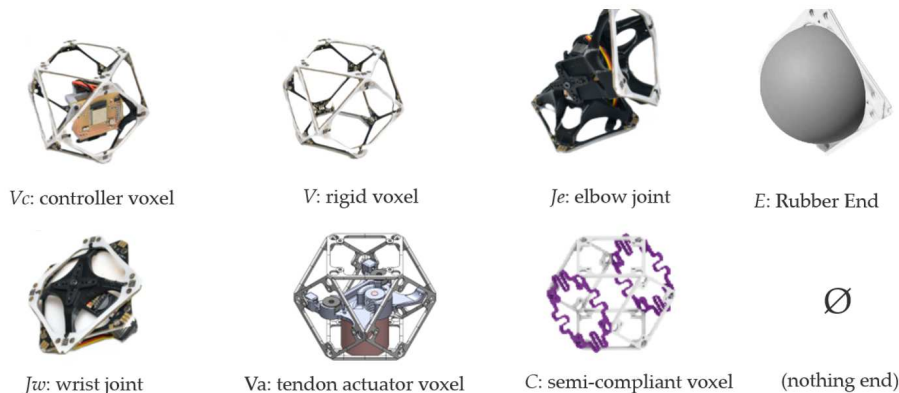


Figure 6.1: Discrete Robotic Toolset

I introduce a fully automated workflow for the generation and reconfiguration of modular robots based on high-level tasks. I suggest a framework to search for the optimum robot shape-control given a task. I started by developing a robot shape grammar based on the discrete robotic toolkit introduced in [44] (see Figure 6.1).

I first developed a robot design environment in NVIDIA Omniverse using Isaac Sim [102] where the user can design any robot and interact with it/test different control strategies (see Figure 6.2). What is unique about this is the realism between the design representation tool and the real robot, and there is no abstraction happening as the underlying physics description of each joint, as well as the limits, masses, and forces match reality which makes the transition from sim2Real a straightforward process (see Figure 6.3).

Figure 6.4 shows a large family of robots that could be quickly built using the design tool. This modular design tool represents a significant advancement in robotics research, as it

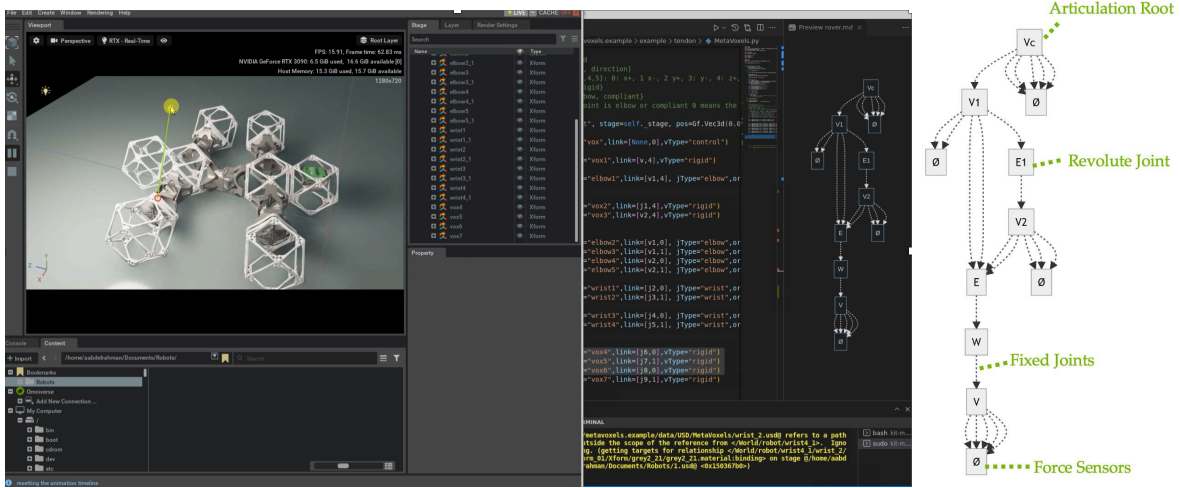


Figure 6.2: Modular Robot Design Tool [102]

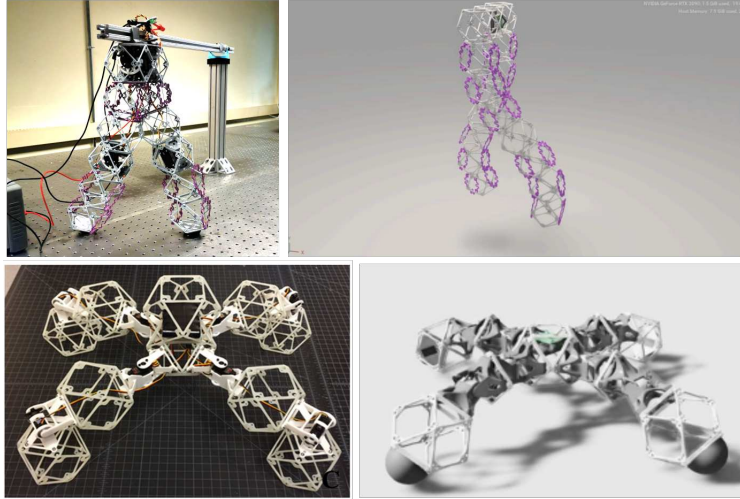


Figure 6.3: Design Tool sim2Real

enables rapid prototyping and iteration without the time and resource constraints of physical construction. The intuitive interface of the tool allows researchers to experiment with various morphologies and configurations, accelerating the discovery of optimal designs for specific tasks.

These designs are ready to be used in the Isaac Gym workflow [103] for reinforcement learning, creating a seamless pipeline from conception to trained control policies. The ability to quickly generate diverse robot designs also facilitates comparative studies and evolutionary approaches that would be prohibitively expensive with physical prototypes. The Isaac Gym environment provides a GPU-accelerated physics simulation that allows for parallelized training across thousands of robot instances simultaneously, dramatically reducing the time required to train robot control policies (see Figure 6.5).

The next step is to not only optimize the control of the robot but also morphology, specifically for reconfiguration (see Figure 6.6). This integrated approach is crucial as it

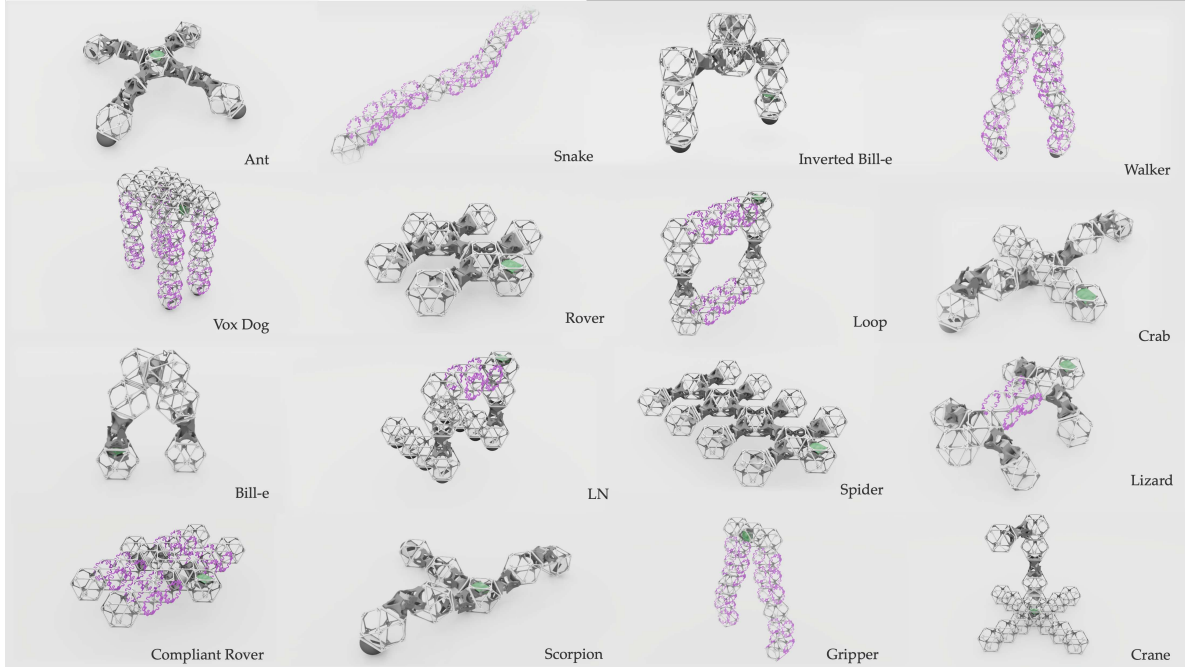


Figure 6.4: Robot Zoo

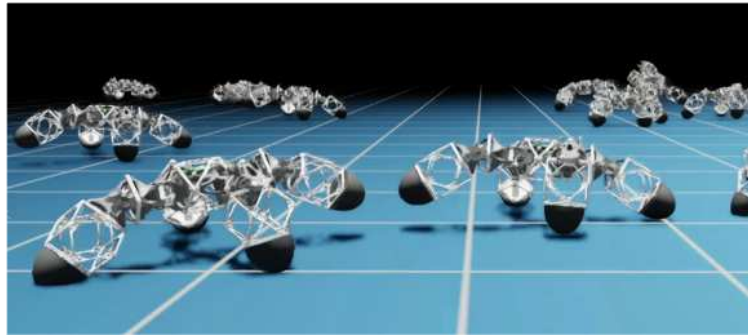


Figure 6.5: Robot Control Reinforcement Learning

enables task-specific adaptation where different physical structures can be optimized for particular requirements. It allows for environmental adaptation as robots that can reconfigure while interacting with their environment achieve more efficient designs than those optimized beforehand [100], [101]. Furthermore, morphology optimization increases versatility over traditional monolithic robots, helps overcome the high-dimensional design space limitations that have traditionally made this problem intractable, and facilitates easier maintenance and damage recovery through module replacement [88], [65], [89].

6.2.2 Shape-Control Optimization

The following section outlines a methodology currently under development for shape control in modular robotic systems. While final results are still forthcoming, the framework establishes a foundation for constructing physical robots and exploring novel, reconfigurable morphologies

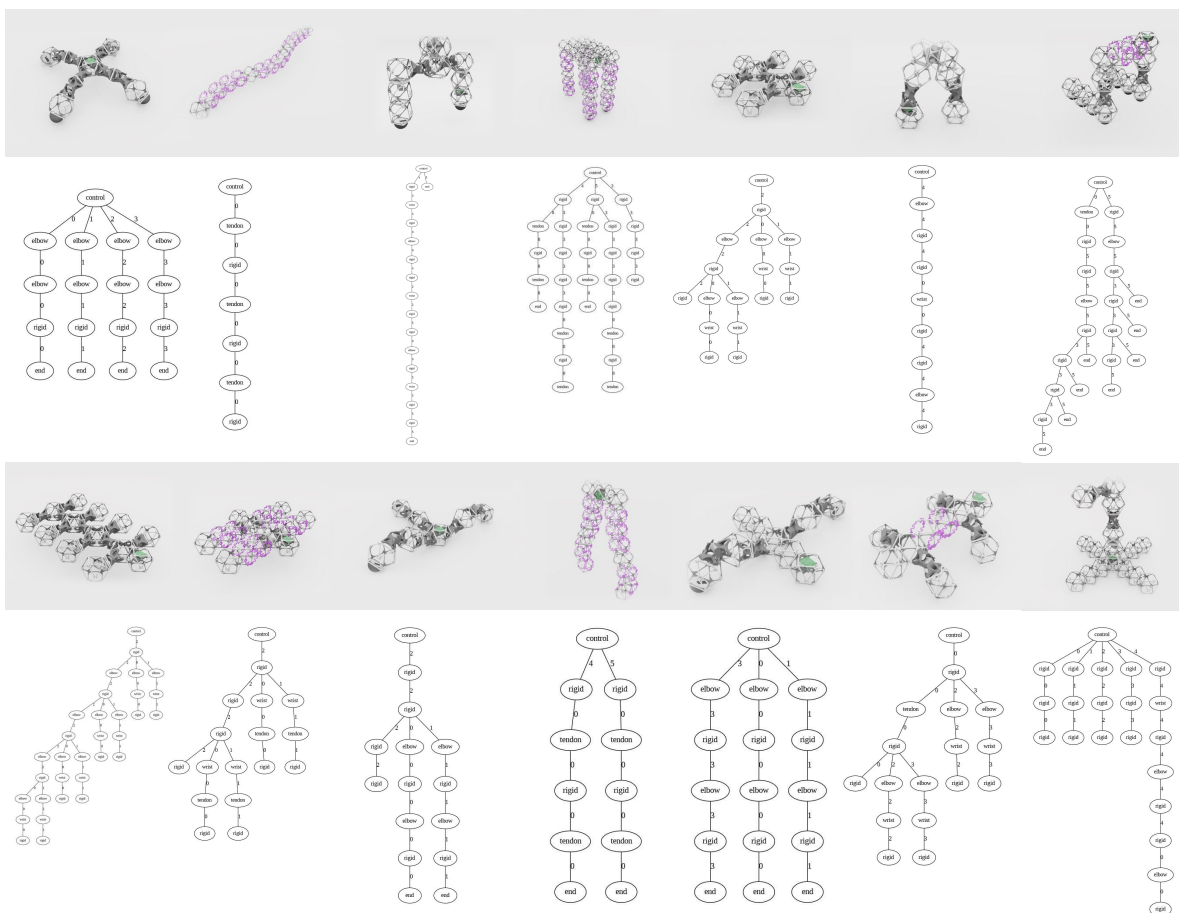


Figure 6.6: Robot Zoo and Graphs.

(see Figure 6.7). This approach aims to enable dynamic adaptation of structure and function through integrated design and control strategies.

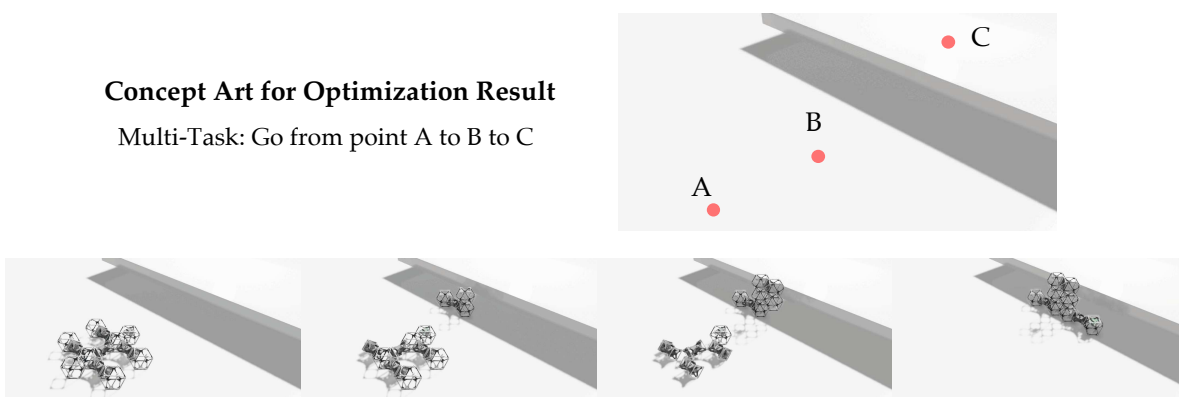


Figure 6.7: Multi-Task Robot Reconfiguration Strategy.

What makes this approach novel is the discretization of control parameters, making the search more tractable and interpretable. I also introduce hierarchy in the discovery of new morphologies by first optimizing subcomponents—such as legs performing specific gaits—that

can be reused to design more complex structures. This makes the search much more efficient and leverages symmetry, a principle widely observed in nature (see Figure 6.8).

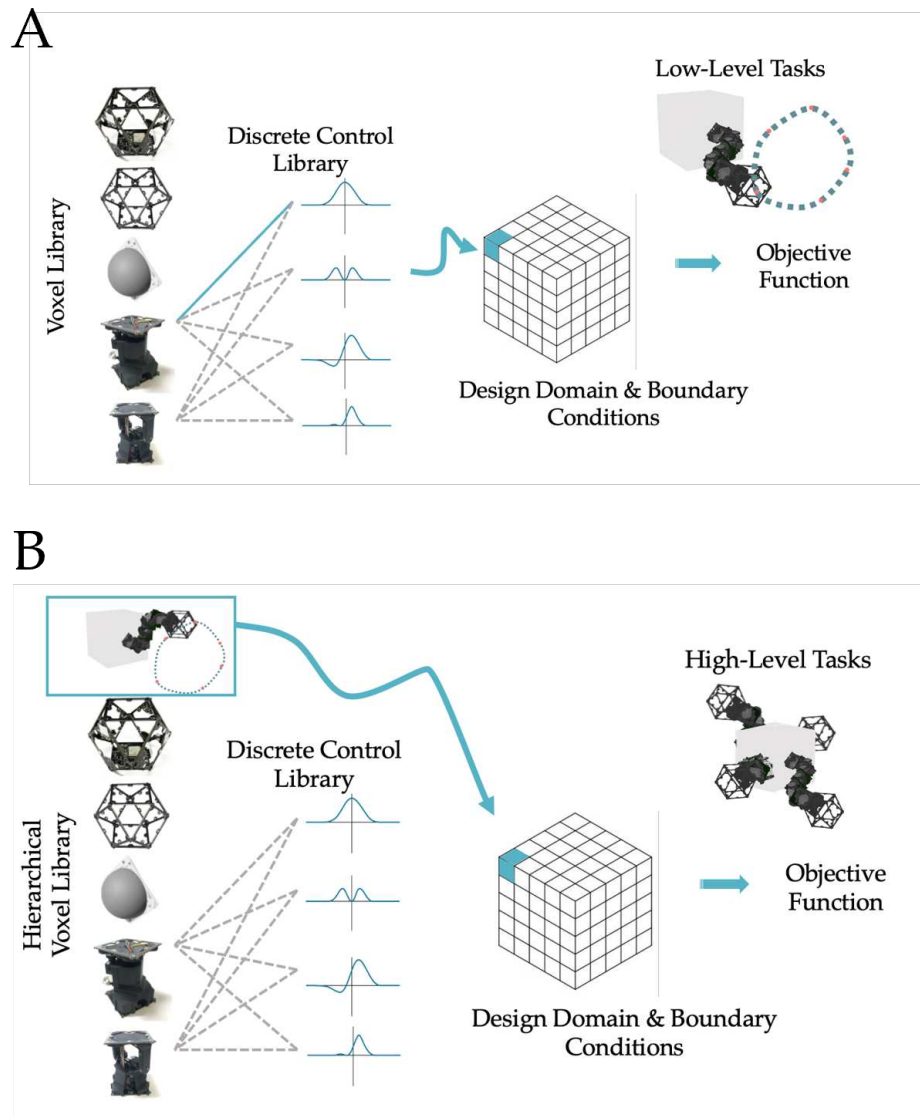


Figure 6.8: Shape-Control Hierarchical Optimization Workflow.

To build a robot, each timestep involves choosing one of four actions: starting a robot, adding a voxel, removing a voxel, or terminating the process. The first action must always be to start a robot, which places a control voxel (limited to one per robot). When adding a voxel, you must specify both a neighboring voxel and the free face for attachment. You also need to select the voxel type (cuboct, wrist joint, elbow joint, or end). For joint types, you must choose the control signal shape and attributes from a finite list of discrete options. Voxel removal is restricted to those connected to at most one neighbor. The final possible action is to terminate, which completes the building or reconfiguration process (see Figure 6.9).

I assume a finite number of voxels of each type. Maskable PPO [104] is used to search

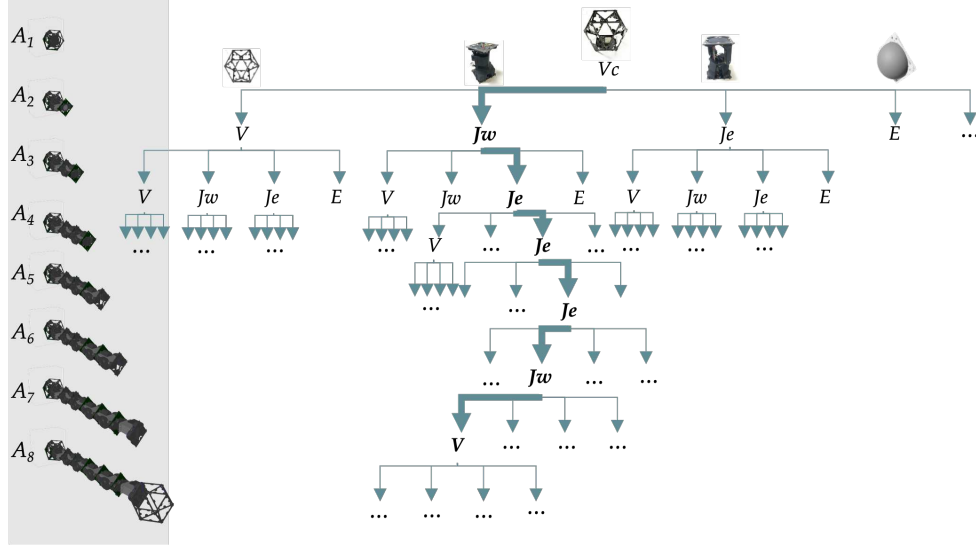


Figure 6.9: Tree Search.

for optimal policies for new tasks, whether for growth or reconfiguration. My evaluation approach differs from standard reinforcement learning—I simulate after intermediate steps and collect rewards immediately. This provides a heuristic for measuring the importance of each incremental step, potentially accelerating the RL workflow and improving policy quality (see Figure 6.10).

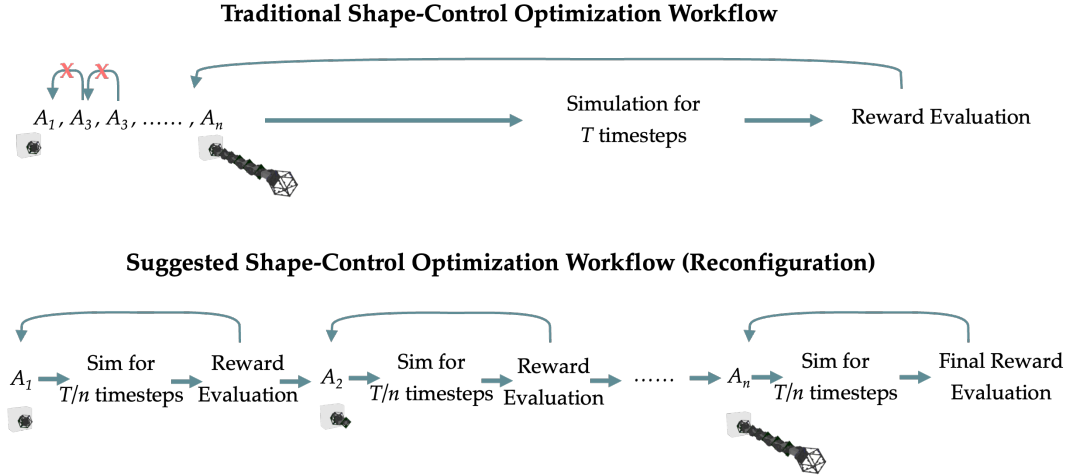


Figure 6.10: Integrated RL and Simulation Workflow.

I plan to use the modular robotic system introduced in [105] to automatically build the newly designed robot and then deploy the control strategy on the real robot. The workflow is already set up, and despite using an open-loop feedforward control strategy, there is good alignment between the simulator and the real robot (see Figure 6.11).



Figure 6.11: Sim2real: Control.

6.3 Recap

This chapter presents a novel framework for the design and control optimization of modular reconfigurable robots, enabling task-specific reconfiguration and environmental adaptation. By introducing a discrete approach to both morphology and control parameters, I have established a more tractable search space that maintains interpretability while enabling complex behaviors. The hierarchical discovery method, which optimizes subcomponents before integrating them into more complex structures, significantly improves search efficiency and leverages natural principles like symmetry. While comprehensive results are still being finalized, the initial demonstrations show promising results, as well as good alignment between simulation and physical implementation.

Chapter 7

Design Integration: Physical Computing Design Tool

7.1 Introduction

In this chapter, I extend the design scope from structural morphology to embedded intelligence, integrating both the body (mechanical structure) and brain (electronic logic) within a unified design process. In [106] we introduced DICE (Discrete Integrated Circuit Electronics) (see Figure 7.1. Rather than separately developing chips, packages, boards, blades, and systems, DICE spans these scales in a direct-write process with the three-dimensional assembly of computational building blocks. To design and fabricate new systems with DICE we had to create a new design tool/workflow.

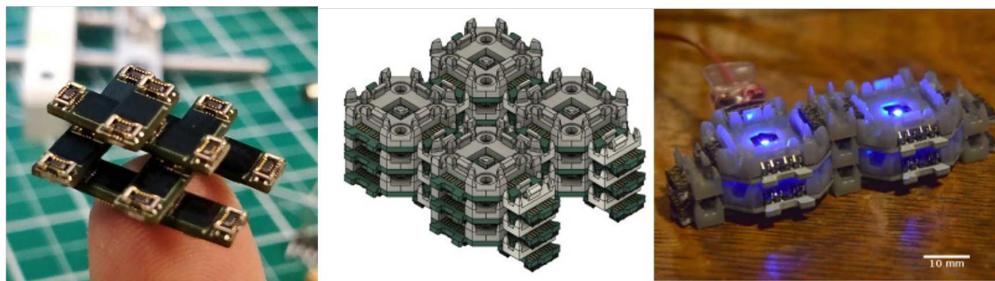


Figure 7.1: Tiny and Meso DICE Parts [106].

Traditional electronics design workflows follow a sequential linear process that starts with design, then analysis and simulation, and finally, system fabrication and testing. Often, these stages are executed independently from each other, using different tools, making it difficult to translate the feedback from the simulation or testing stages into viable design amendments. This adds considerable inefficiency to an inherently iterative design workflow.

7.2 Results

The Physical Computing Design Tool [107] is an integrated design tool for the design, simulation, optimization, and fabrication of reconfigurable computing systems.

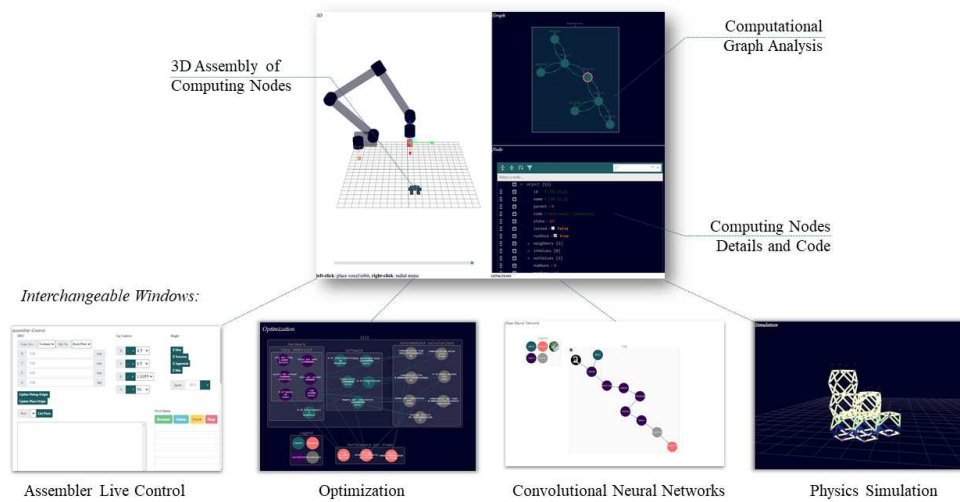


Figure 7.2: Physical Computing Design Tool [107].

The following steps explain in detail the proposed design workflow:

1. **Design:** The first step is to place the DICE nodes and visualize the 3D assembly of the computing system.
2. **Programming:** The user then selects individual nodes to specify details, such as processor type, connected neighbors, program, number of operations, maximum token size, and so forth. During these steps, a connectivity graph is generated in real time, showing the average computation and communication cost of each node based on these pre-programmed node details. In the future, these numbers could be automatically calculated by pre-compiling and profiling code for each node.
3. **Analysis:** By treating the computation graph as a flow network, graphical algorithms are used to find and highlight computation and communication bottlenecks. Based on this analysis, design recommendations are generated and displayed, including suggestions for different processors, extra nodes, or alternative routing.
4. **Optimization:** If the user wants to further optimize the reconfigurable computing system, the performance projection graph tool visualizes the effect of each hardware and software decision on the projected overall system performance. A generative probabilistic programming model is used to draw samples from these inputs and project the cost, energy use, and speed of the system. Using classic inference techniques, one can infer the optimal hardware configuration for a given software workflow. In the future, this could be extended to additionally perform hardware architecture discovery.

5. **Simulation:** the simulation window allows the user to visualize the state and output of the DICE computation. Currently, this shows the output of a simulated computation provided on the host computer prior to physical assembly. In the future, this framework will be used to visualize data from the DICE network.
6. **Fabrication:** In parallel with the previous steps, the real-time assembly control window provides a simple mechanism for controlling the mechanical DICE assembler. Using this feature, systems can be fabricated, programmed, and tested in real-time to quickly pull in feedback and amend the system architecture accordingly.

This integrated design workflow paves the way for the design of discrete integrated circuits, but also reconfigurable computing systems that can change and evolve while execution as needed.

7.2.1 Voxel Invention Kit

The physical computing design tool was also used for the design and analysis of structures made using the Voxel Invention Kit (VIK) [108]. VIK uses reconfigurable metamaterial building blocks (termed functional voxels) that assemble into high-stiffness, lightweight structures with integrated electronics (see Figure 7.3). New functionality has been added to change the voxel type (five types: rigid/base voxels, microcontroller voxels, sensing voxels, display voxels, and load voxels (where loads/forces are applied)).

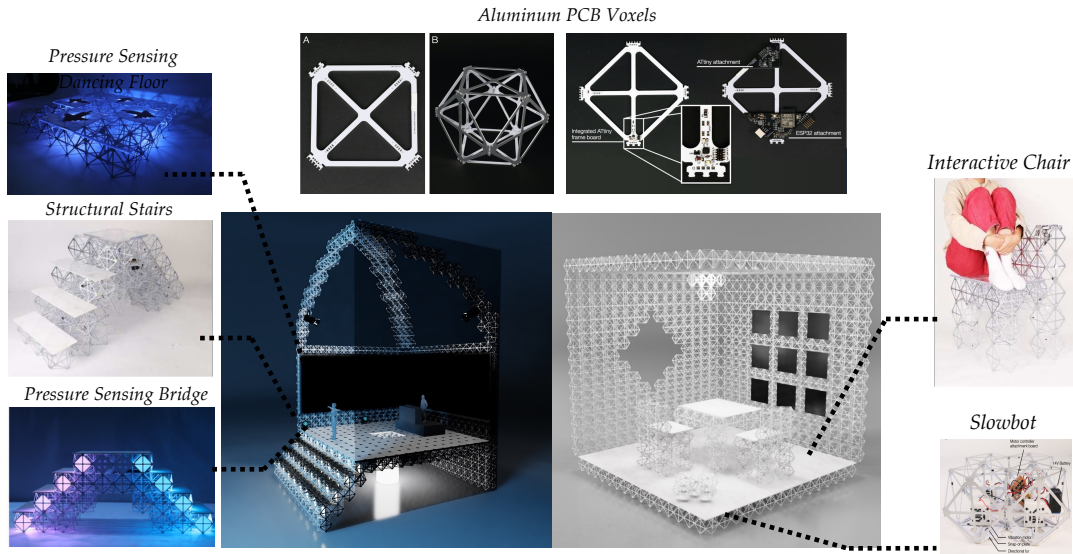


Figure 7.3: The Voxel Invention Kit (VIK) [108].

Then the user can run a structural mechanics simulation (using MetaVoxels), an animation of the structure's deflection (exaggerated 10x) appears, and the beams are colored based on their stress value to identify potential points of failure. Finally, a summary of the results (minimum and maximum displacement, minimum and maximum stress values, and safety factor) is calculated and displayed at the bottom footer.

In the unique interface, the user can make informed design decisions by having instant feedback and visualization of the structure's assembly and simulation in a simple integrated design tool without going back and forth between different interfaces. Importantly, the user does not need to understand any of what was just described. Voxels are assembled like Minecraft, and the simulation window helps users understand the flow of mechanical forces in their system without needing intuition or experience in this area (see Figure 7.4).

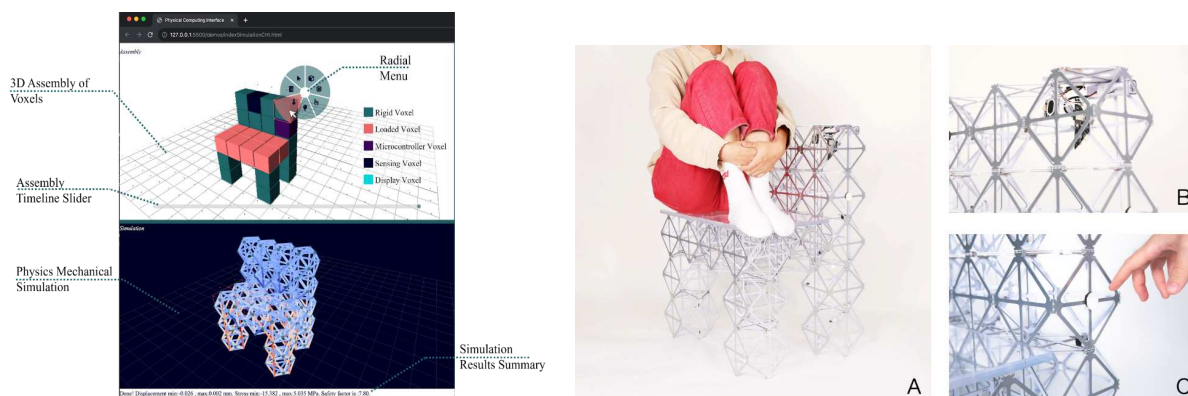


Figure 7.4: Using Physical Computing Design Tool to Design a Chair using the Voxel Invention Kit.

7.3 Co-Design of Mechanical and Electrical Components

The physical computing design tool unlocks the ability to be able to design the mechanical components (body) of systems but also electronic components (brain) which makes it possible to design multiscale digital material robots. Looking forward, this unified approach lays the foundation for future autonomous systems that are not only reconfigurable in shape but also in logic, paving the way for robotic systems that can adapt their morphology and behavior in real time based on embedded sensing and computation.

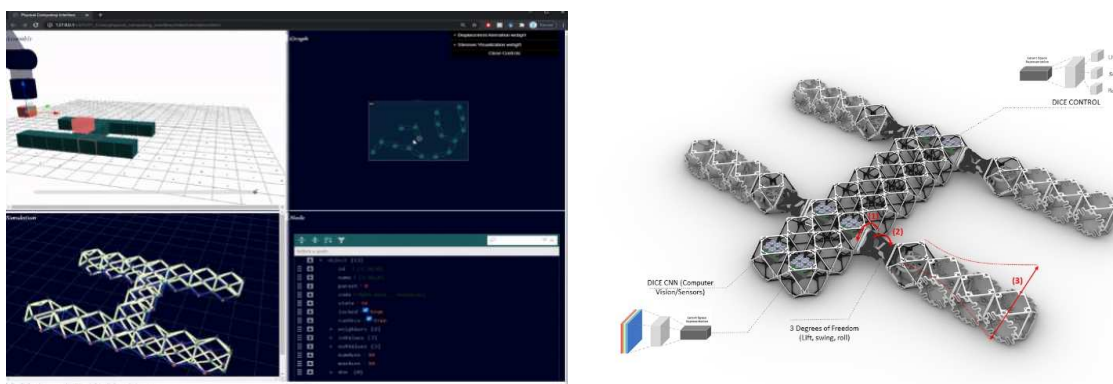


Figure 7.5: Using Physical Computing Design Tool to Design the Body and Brain of Digital Material Robots.

Chapter 8

Case Study: How can Anyone Design and Build a Pavilion in One Day?

8.1 Overview

In the previous chapters, I have presented methods for simulation, optimization, fabrication, reconfiguration, and control of material-robot systems made from digital material. In this final section, I will illustrate the practical integration of these methods through a detailed case study. The central inquiry guiding this study is: "How can **anyone** design and build a pavilion in **one day**?"

While the case study (see Figure 8.1) specifically addresses pavilion construction, the underlying methods and principles are broadly relevant due to the discrete, modular and adaptable nature of digital materials. Voxel-based construction simplifies the design process, allowing individuals without specialized technical knowledge to efficiently design complex structures. The modularity of digital materials enables rapid customization and scalability, making this approach applicable beyond architecture to areas such as furniture design, robotics, automotive, and aerospace engineering.

8.2 Design: Text to Viable Design

The first section addresses the "anyone" component of the question: how can someone with no prior design or construction background design a pavilion (or a car, plane, etc.)? With the growing popularity of large language models, one can now easily obtain image inspiration for any design using simple text prompts. Using the text prompt "Pavilion in Dubai," I first generated an inspirational image, followed by a multi-view generated hallucination. When relying solely on the large reconstruction model (LRM), the resulting high-resolution textured 3D model revealed significant structural flaws: columns failed to reach the floor, and under self-weight simulation, the pavilion would collapse. These issues stem from the fact that LRM models weren't trained on large-scale structures like buildings or complex multi-object scenes.

Therefore, I introduce a workflow to incorporate fabrication and physical constraints into 3D models generated by generative AI (see Figure 8.1). This approach ensures that generated objects satisfy functional and aesthetic requirements and optimizes the geometric

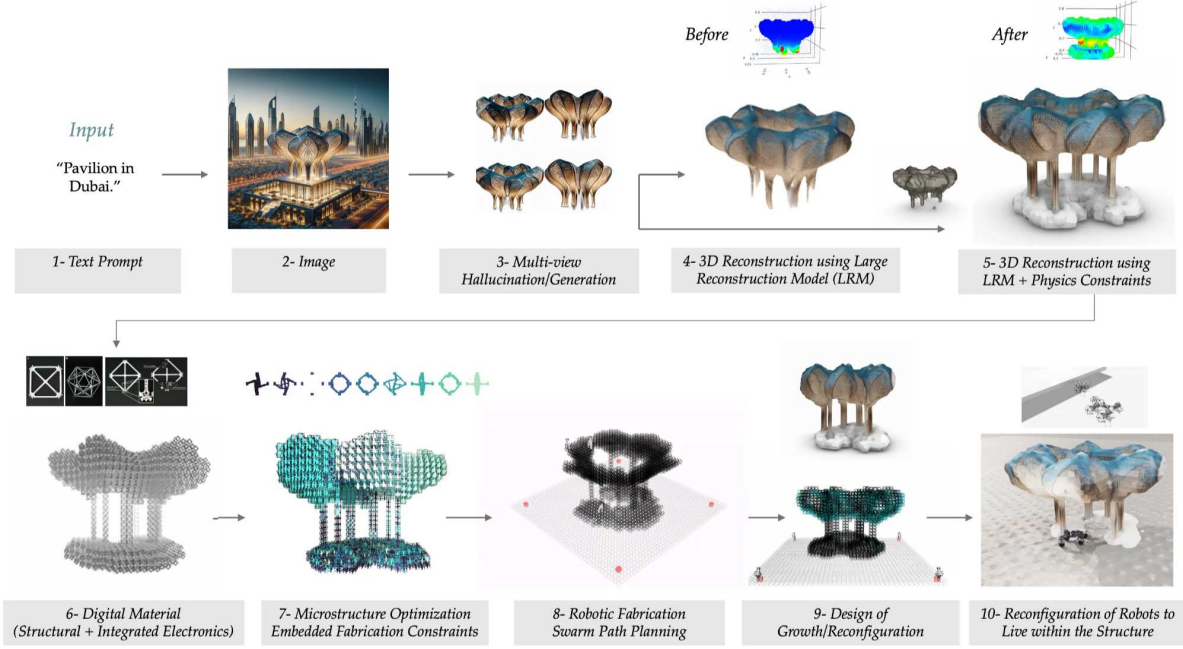


Figure 8.1: Discrete Inverse Design Overview.

features based on these requirements. This is achieved through an automated process that detects boundary conditions and affordances, enabling accurate evaluation of 3D objects across diverse topologies.

In my proposed workflow, users specify both aesthetic and functional requirements for 3D objects using only text prompts as input. I employ an off-the-shelf diffusion model [109] to generate an image from the text prompt. Building on the workflow introduced in [110], I leverage a multiview diffusion model and a sparse-view large reconstruction model to generate a high-quality 3D model from a single image. Once the 3D model is generated, I optimize the 3D neural implicit field against two objective functions: (1) a physical loss function to enforce functional constraints and (2) an aesthetic loss function to maintain visual fidelity to the multi-view generated images.

Using my workflow, the system automatically inferred that the objective function should minimize compliance (i.e., displacement under self-weight). For physics-based evaluation and gradient computation, I use a differentiable physics simulator based on the Material Point Method (MPM) [111], which provides gradients of the objective function with respect to the parameters of the 3D implicit neural field.

The neural implicit field is optimized to minimize the object’s compliance (i.e., deflection under load) while maintaining a volume constraint that prevents drastic alterations to the original geometry —allowing only essential modifications. The physics constraints corrected the model by generating structural supports, such as legs and a base, to prevent sliding and buckling. The workflow preserved the textures to match the multi-view reference images, ensuring a balance between structural integrity and aesthetic fidelity.



*Building four walls and a roof with a length of 10 m, width of 5 m, and height of 3 m.

Figure 8.2: Large Scale Digital Fabrication Methods Comparison (data from [112]).

8.3 Digital Material as a Sustainable Alternative in AEC Industry

Now that we have a pavilion design, if it were a small object, we would simply 3D print it. However, can we digitally fabricate a large object like a pavilion or building?

Digital fabrication methods for large-scale structures primarily include extrusion-based processes, such as 3D concrete printing (3DCP), and assembly-based techniques like precast modular reinforced concrete (MPC), cold-formed steel (CFS), and traditional stick framing (see Figure 8.2). Although 3DCP offers design flexibility, its practical limitations include high embodied carbon (47,131 kg CO₂ for a small building) and significant machinery investment, reducing sustainability and scalability. Modular precast concrete and framing methods like CFS and stick framing demonstrate improved sustainability compared to 3DCP but suffer from higher labor costs and extended construction times, ranging from approximately 20 to 65 hours, with high reliance on skilled labor [112].

In contrast, voxel-based construction methods uniquely enable both high material efficiency and rapid assembly. This approach significantly reduces material mass, achieving lower embodied carbon levels (e.g., 12,501 kg CO₂ for a wood-based voxel system). Furthermore, voxel-based methods leverage automation, drastically cutting construction time to as little as

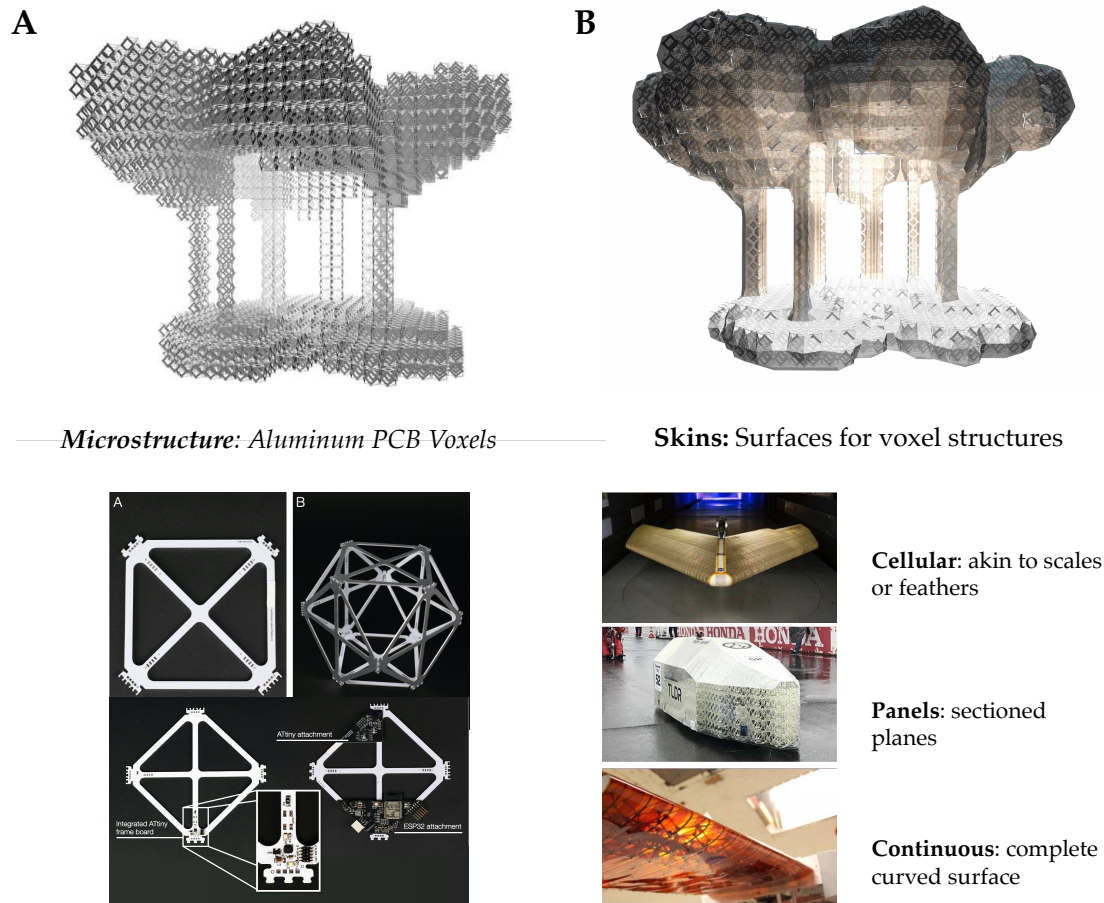


Figure 8.3: Digital Material.

10.64 hours with increased robot numbers. This makes voxel-based construction the only viable solution capable of meeting the ambitious target of designing and building a pavilion within a single day. The inherent modularity, robotic compatibility, and error-correcting assembly features of voxel systems establish their clear quantitative advantage in speed, sustainability, and scalability compared to other digital fabrication methods [112].

The Voxel Invention Kit (VIK) [108] further enhances the capabilities of voxel-based construction by providing a versatile, reconfigurable system suitable for rapid prototyping and scalable design. The kit comprises modular voxels that integrate structural and electronic functionalities, enabling users to quickly assemble and modify complex structures without extensive technical expertise. The simplicity and flexibility of VIK facilitate iterative design processes, significantly accelerating prototyping and experimentation.

By leveraging the inherent modularity of voxels, VIK allows for dynamic reconfiguration of structures, enabling quick adaptation to changing requirements or conditions (see Figure 8.3). This feature is particularly advantageous in scenarios requiring frequent modifications, such as temporary installations, adaptable architectural environments, or disaster relief shelters. Additionally, the integration of electronics within the voxel modules allows for embedded sensing and control, with need to care about wiring, further expanding the functional capabilities of the assembled structures and supporting advanced applications

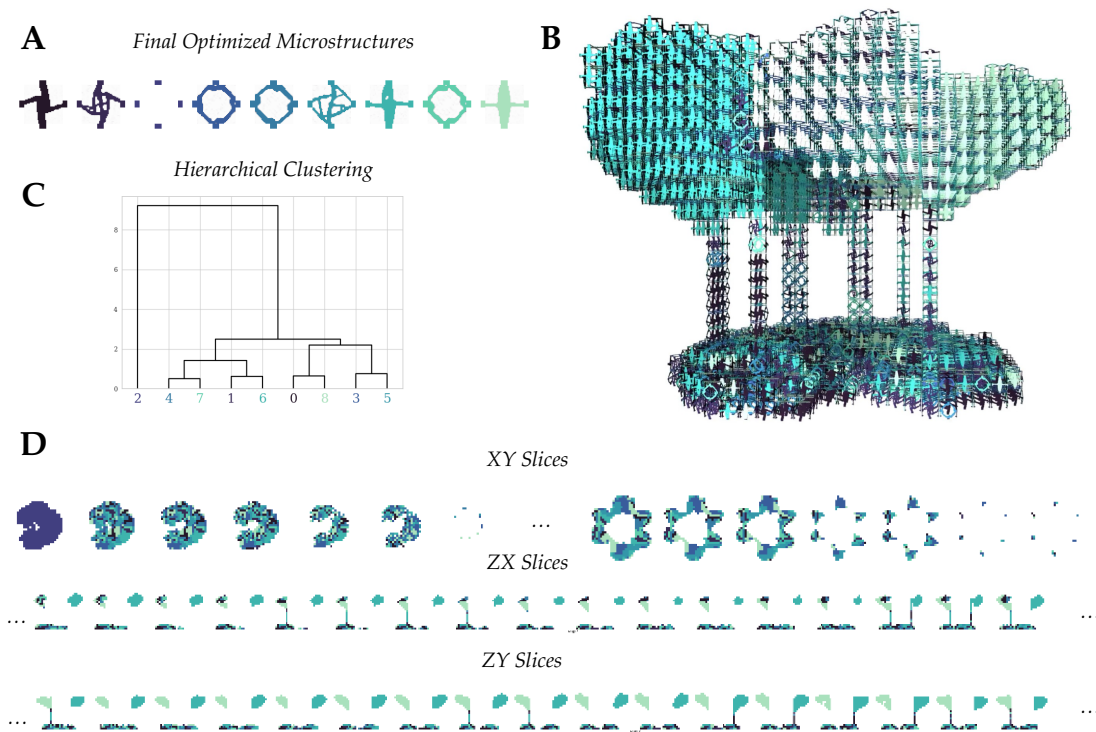


Figure 8.4: Optimized Microstructures.

such as responsive environments and smart infrastructure. Regarding the skin, based on previous research [9] [3], there are numerous design approaches available, including discrete cellular elements (such as scales and panels) or even continuous elements, with the selection depending on weather conditions and other functional requirements.

8.4 Microstructure Optimization

Even though the VIK elements are high performance (low density and stiff), we can further use the workflow introduced in chapter 3 to optimize the microstructures. The workflow is highly scalable and can generate a list of optimized microstructures within seconds, specifically tailored for the pavilion to achieve even greater stiffness using the same amount of materials (see Figure 8.4).

8.5 Swarm Robotic Construction

Next question: are we really able to build it in one day? The relative robots introduced in chapter 5 are ideal for implementing digital material construction in the AEC industry. Unlike other methods, they don't require building structures larger than the building itself to build it, and the process can be highly parallelized. Assuming each robot takes 3 seconds per action, two robots can construct a $5\text{m} \times 5\text{m} \times 3\text{m}$ room in 8 hours. This efficiency can

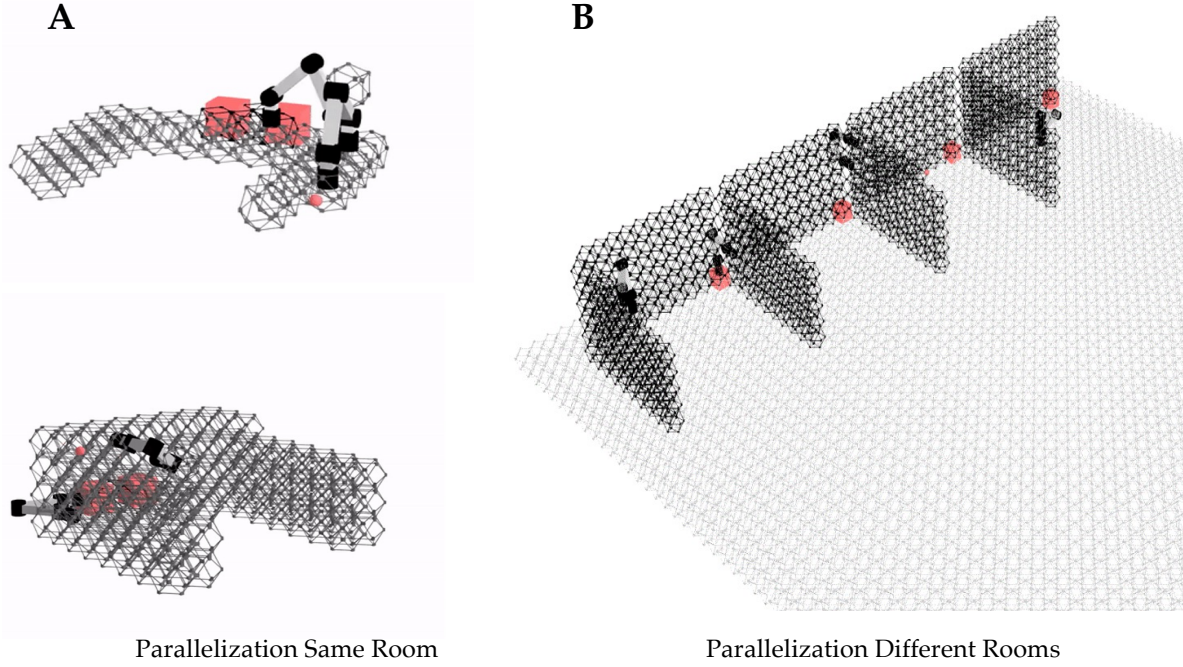


Figure 8.5: Swarm Robotic Construction.

be further enhanced by deploying multiple robots to build other rooms simultaneously (see Figure 8.5).

For the pavilion, using the path planning algorithm introduced in [43] [44], if we deploy 4 robots similar to the configuration in Figure 8.6, each would take approximately 29,131 timesteps to complete its assigned section. This would require a total of approximately 24.27 hours to complete the pavilion.

8.6 Design of Reconfiguration

Beyond static construction, we can leverage the methodologies presented in chapter 4 to design through time, approaching the pavilion as a dynamic entity capable of reconfiguration. Similarly, we will approach the design of the built environment as a program describing how different components of the environment are built, actuated, and reconfigured rather than as a static description of elements in space. The natural environment is far from static, and the objects we design must be the same. They should be responsive to external triggers along different timescales: seconds (human touch/interaction), minutes (structural loads), hours (thermal lighting conditions), days (change of activities), seasons (weather), or even years (function).

Figure 8.7 shows an example of a trained NCA that can describe the reconfiguration of the pavilion through time. This advanced model captures how the structure can adapt to changing requirements or environmental conditions. The relative robots that were used to build the

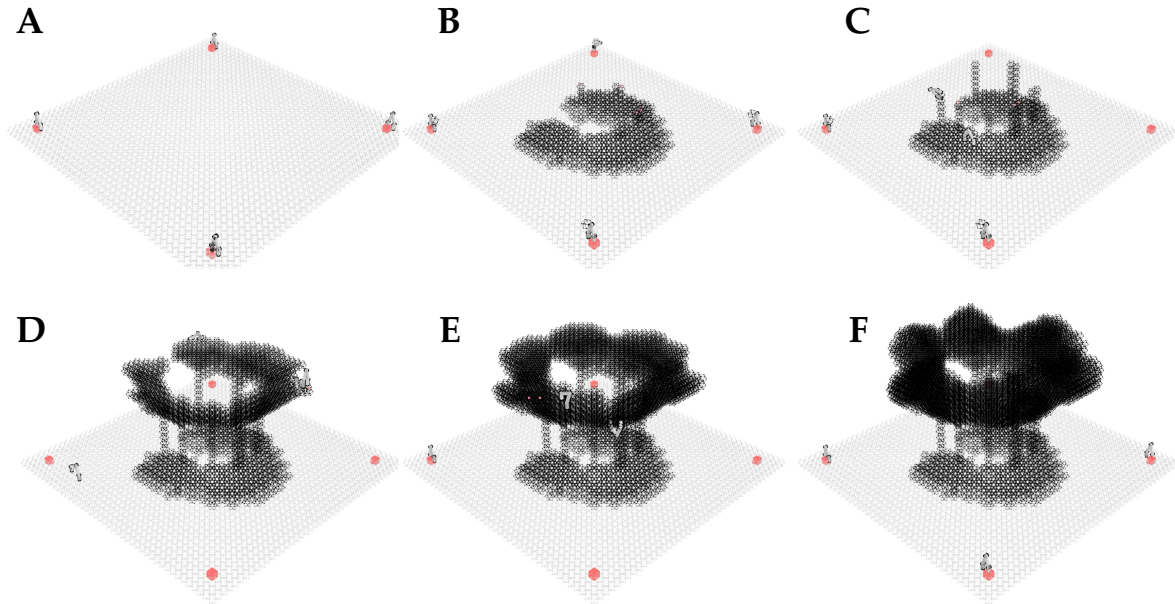


Figure 8.6: Pavilion Swarm Robotic Construction.

structure can also be utilized for reconfiguration tasks. This dual-purpose functionality demonstrates the versatility of the robotic system, allowing for both initial construction and subsequent modifications without requiring additional specialized equipment.

Furthermore, relative robots can be reconfigured for secondary purposes during the building lifecycle (see Figure 8.8). These versatile robots can transition from construction agents to maintenance workers, performing tasks such as structural monitoring, cleaning building facades, repairing damaged components, and even upgrading building systems as technology evolves. Additionally, they can be programmed to reconfigure interior spaces based on changing occupant needs, transforming living or working environments without requiring extensive demolition or reconstruction. Furthermore, the relative can be later reconfigured to perform other tasks during the building lifecycle (see Figure 8.8).

The modular nature of these robots represents a paradigm shift in the way we approach building life cycles. Their adaptability reduces the need for specialized equipment for different building phases, significantly lowering operational costs and environmental impact. This versatility is particularly valuable in remote or resource-constrained environments where importing specialized equipment is challenging. Moreover, the ability to repurpose construction robots for ongoing building management creates a continuous feedback loop between the physical structure and its maintenance systems, enabling truly responsive architecture that can evolve over time rather than remaining static after initial construction.

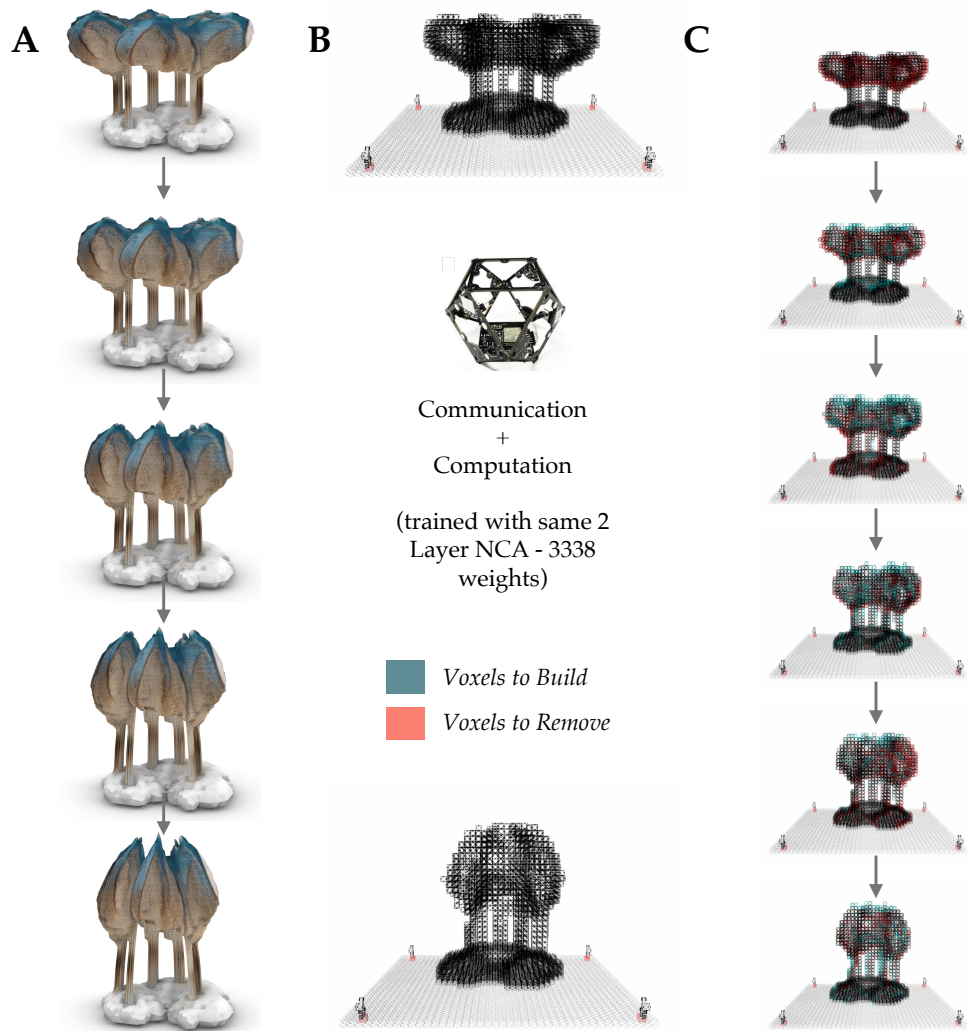


Figure 8.7: NCA Reconfiguration.

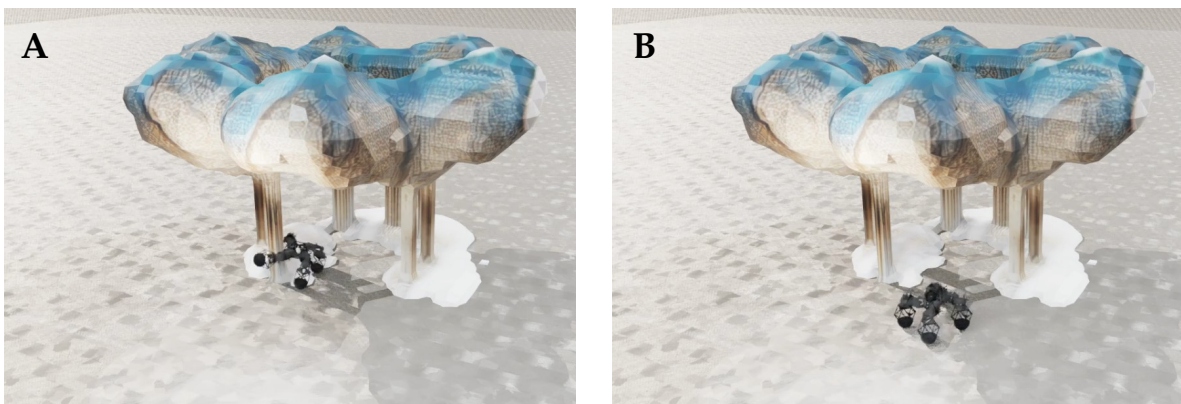


Figure 8.8: Ant Reconfiguration.

8.7 Recap

This case study presents a novel workflow that integrates generative AI with physics constraints to produce functional and manufacturable 3D models. By incorporating automatic boundary condition detection and optimizing neural implicit fields based on both aesthetic and structural requirements, my approach ensures that generated designs are not only creative alternatives but also physically viable. The results demonstrate the potential to combine AI-driven creativity with engineering rigor, paving the way for more reliable generative design solutions in various applications.

Chapter 9

Conclusion

In this dissertation, I have introduced the concept of inverse discrete design as a novel paradigm to bridge the gap between digital modeling and physical realization across multiple scales and disciplines. This approach leverages interdisciplinary perspectives, where the same tools and workflows can be used to design complex environments that concurrently model and fabricate electronics, furniture, buildings, cars, planes, and cities.

9.1 Summary of Contributions

This thesis contributes significant innovations through the development of novel simulation frameworks, optimization techniques, and fabrication methodologies. These key advances include:

- A differentiable multi-scale multi-physics simulation tool for both static and dynamic mechanical metamaterial simulation.
- Multi-scale topology optimization workflow for the inverse design of discrete mechanical metamaterials.
- Target-based optimization workflow for the growth and reconfiguration of cellular structures and soft robots.
- Advanced path planning algorithms—swarm, recursive, and hierarchical—for scalable robotic swarm assembly.
- Shape-control optimization workflow for discrete modular robots.
- Comprehensive design tool to integrate physical computing design with fabrication processes; for the co-design of mechanical and electrical components.
- Workflow for text-to-3D generation integrating physics simulation for design validation.
- End-to-end inverse design workflow for electromechanical systems that transforms text specifications into designs and path planning for both construction and reconfiguration throughout their lifecycles.

9.2 Open-Source Tools

1. MetaVoxels: Hierarchical Simulation of Metamaterial: www.cba.mit.edu/amiraa/metavoxels
2. Inverse Design of Discrete Metamaterial: www.cba.mit.edu/amiraa/Inverse-Design-of-Discrete-Metamaterial
3. Recursive and Hierarchical Robotic Swarm Assembly: www.cba.mit.edu/amiraa/self-replicating-hierarchical-robotic-swarms
4. NVIDIA Omniverse MetaVoxels Designer: www.cba.mit.edu/amiraa/nvidia-omniverse-metavoxels
5. Physical Computing Design Tools: www.cba.mit.edu/amiraa/physical-computing-design-tools

9.3 Impact

This could have significant implications:

1. **Environmental Impact:** The approach significantly reduces embodied carbon through optimized interdisciplinary digital material structures. By designing components that can be efficiently reused, reconfigured, and repurposed throughout their lifecycle, we minimize waste and resource consumption. This system enables precise material allocation only where structurally necessary, potentially resulting in a significant reduction in material usage compared to conventional methods.
2. **Economic Impact:** The proposed methodology revolutionizes the construction industry by enabling high-throughput fabrication and streamlined onsite assembly. This transformation could reduce construction timelines while lowering labor costs through automated processes. Additionally, the standardization of digital components creates new markets for specialized fabrication services and material suppliers.
3. **Social Impact:** By democratizing interdisciplinary design and fabrication, this approach empowers communities previously excluded from the design process. Open-source platforms and accessible tools allow diverse stakeholders to participate in creating their built environment, fostering greater inclusivity and representation in design outcomes. Educational initiatives built around these methodologies could help bridge technological divides and create pathways to high-skilled employment in emerging digital fabrication sectors.

9.4 Future Directions

Potential research directions enabled by this dissertation include:

1. **Voxels Capabilities (Bottom-Up Approach):** One potential direction involves expanding voxel capabilities by enhancing flexibility across different lattice structures and multiscale hierarchical grids. Transitioning from static assumptions, shape-changing voxels could be developed, dynamically tuning stiffness and geometry to create significant global structural adaptations. The incorporation of force-sensing capabilities presents an opportunity for developing real-time optimization methods, enabling structures to adapt dynamically to changing external conditions. Additionally, exploring sustainable and compostable materials for voxel fabrication emerges as a promising direction to minimize environmental impacts.
2. **Applications and System Integration (Top-Down Approach):** Another significant avenue involves application-driven research inspired by biological systems, focusing particularly on the design of semi-rigid and semi-soft robotic structures with directional compliance. Building upon existing research in reconfigurable structures and adaptive robot swarms, future work could focus on robust systems capable of operating under unpredictable or evolving boundary conditions. The advancement of "Reconfigurable AI," involving three-dimensional physical computing systems that dynamically evolve hardware configurations based on new tasks or discoveries, is another exciting possibility. Additionally, developing large-scale structures with distributed actuation and active damping capabilities suitable for extreme environments, such as space applications, presents intriguing potential.
3. **Integrated Design Tools:** This introduced workflow could be improved by enhancing the physical-digital integration by embedding error detection and correction capabilities directly into physical prototypes, facilitating real-time feedback loops with digital modeling platforms. Developing declarative workflows tailored for non-experts, incorporating physical guarantees, and employing interpretable machine learning techniques could further democratize access to these technologies. Moreover, reimagining geometric representations by integrating material, physical, and fabrication properties into cohesive design frameworks presents a valuable research opportunity. Continuing the development of dynamic frameworks to model the growth, evolution, and reconfiguration of physical systems enables adaptive designs that respond to changing conditions over time. Finally, integrating the human aspect in the loop will be important to make sure that the automated workflows find optimized, yet creative solutions.

References

- [1] U. Nations. *The Impact of Digital Technologies*. en. Publisher: United Nations. URL: <https://www.un.org/en/un75/impact-digital-technologies> (visited on 01/10/2024).
- [2] I. E. Sutherland. “Sketchpad: a man-machine graphical communication system”. In: *Proceedings of the May 21-23, 1963, spring joint computer conference*. AFIPS ’63 (Spring). New York, NY, USA: Association for Computing Machinery, May 1963, pp. 329–346. ISBN: 978-1-4503-7880-2. DOI: [10.1145/1461551.1461591](https://doi.org/10.1145/1461551.1461591). URL: <https://dl.acm.org/doi/10.1145/1461551.1461591> (visited on 01/10/2024).
- [3] B. E. Jenett. “Discrete Mechanical Metamaterials”. en. PhD Thesis. Massachusetts Institute of Technology, 2020.
- [4] K. C. Cheung and N. Gershenfeld. “Reversibly assembled cellular composite materials”. In: *science* 341.6151 (2013). Publisher: American Association for the Advancement of Science, pp. 1219–1221.
- [5] C. E. Gregg, J. H. Kim, and K. C. Cheung. “Ultra-Light and Scalable Composite Lattice Materials”. In: *Advanced Engineering Materials* 20.9 (2018). Publisher: Wiley Online Library, p. 1800213.
- [6] B. Jenett, C. Cameron, F. Tournomousis, A. P. Rubio, M. Ochalek, and N. Gershenfeld. “Discretely assembled mechanical metamaterials”. en. In: *Science Advances* 6.47 (Nov. 2020), eabc9943. ISSN: 2375-2548. DOI: [10.1126/sciadv.abc9943](https://doi.org/10.1126/sciadv.abc9943). URL: <https://www.science.org/doi/10.1126/sciadv.abc9943> (visited on 01/04/2024).
- [7] W. Langford, A. Ghassaei, and N. Gershenfeld. “Automated assembly of electronic digital materials”. In: *International Manufacturing Science and Engineering Conference*. Vol. 49903. American Society of Mechanical Engineers, 2016, V002T01A013.
- [8] B. Jenett, S. Calisch, D. Cellucci, N. Cramer, N. Gershenfeld, S. Swei, and K. C. Cheung. “Digital morphing wing: active wing shaping concept using composite lattice-based cellular structures”. In: *Soft robotics* 4.1 (2017). Publisher: Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, pp. 33–48.
- [9] N. B. Cramer, D. W. Cellucci, O. B. Formoso, C. E. Gregg, B. E. Jenett, J. H. Kim, M. Lendraitis, S. S. Swei, G. T. Trinh, K. V. Trinh, et al. “Elastic shape morphing of ultralight structures by programmable assembly”. In: *Smart Materials and Structures* 28.5 (2019). Publisher: IOP Publishing, p. 055006.

- [10] B. Jenett, D. Cellucci, C. Gregg, and K. Cheung. “Meso-scale digital materials: modular, reconfigurable, lattice-based structures”. In: *International Manufacturing Science and Engineering Conference*. Vol. 49903. American Society of Mechanical Engineers, 2016, V002T01A018.
- [11] B. Jenett, C. Gregg, D. Cellucci, and K. Cheung. “Design of multifunctional hierarchical space structures”. en. In: *IEEE Aerospace Conference Proceedings*. 2017.
- [12] J. Hiller. “Digital materials: voxel design, rapid assembly, structural properties, and design methods”. In: (2011).
- [13] H. A. Simon. “The Science of Design: Creating the Artificial”. In: *Design Issues* 4.1/2 (1988). Publisher: The MIT Press, pp. 67–82. ISSN: 0747-9360. DOI: [10.2307/1511391](https://doi.org/10.2307/1511391). URL: <https://www.jstor.org/stable/1511391> (visited on 01/09/2024).
- [14] *State of the Art in Methods and Representations for Fabrication-Aware Design - Bermano - 2017 - Computer Graphics Forum - Wiley Online Library*. URL: <https://onlinelibrary.wiley.com/doi/full/10.1111/cgf.13146> (visited on 01/09/2024).
- [15] H. Simon. “Anecdotes-a very early expert system”. en. In: *IEEE Annals of the History of Computing* 15.3 (1993), pp. 64–68. ISSN: 1058-6180. DOI: [10.1109/85.222851](https://doi.org/10.1109/85.222851). URL: <http://ieeexplore.ieee.org/document/222851/> (visited on 01/09/2024).
- [16] J. Cagan, M. I. Campbell, S. Finger, and T. Tomiyama. “A framework for computational design synthesis: model and applications”. In: (2005).
- [17] M. I. Campbell and K. Shea. “Computational design synthesis”. In: *AI EDAM* 28.3 (2014). Publisher: Cambridge University Press, pp. 207–208.
- [18] M. Van Diepen. “Computational Design Synthesis for the Co-Design of Morphology and Actuation of Soft Robots for Locomotion”. en. Accepted: 2020-11-24T13:52:13Z. Doctoral Thesis. ETH Zurich, 2020. DOI: [10.3929/ethz-b-000452781](https://doi.org/10.3929/ethz-b-000452781). URL: <https://www.research-collection.ethz.ch/handle/20.500.11850/452781> (visited on 01/09/2024).
- [19] M. van Diepen and K. Shea. “Co-Design of the Morphology and Actuation of Soft Robots for Locomotion”. In: *Journal of Mechanical Design* 144.083305 (June 2022). ISSN: 1050-0472. DOI: [10.1115/1.4054522](https://doi.org/10.1115/1.4054522). URL: <https://doi.org/10.1115/1.4054522> (visited on 01/15/2024).
- [20] A. M. Mehta, J. DelPreto, B. Shaya, and D. Rus. “Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Sept. 2014, pp. 2892–2897. DOI: [10.1109/IROS.2014.6942960](https://doi.org/10.1109/IROS.2014.6942960). URL: <https://ieeexplore.ieee.org/document/6942960> (visited on 01/15/2024).
- [21] A. Mehta, N. Bezzo, P. Gebhard, B. An, V. Kumar, I. Lee, and D. L. Rus. *A Design Environment for the Rapid Specification and Fabrication of Printable Robots*. en. Accepted: 2021-12-22T20:52:23Z ISSN: 1610-7438 Publication Title: MIT web domain. Springer International Publishing, Nov. 2015. URL: <https://dspace.mit.edu/handle/1721.1/137209.2> (visited on 01/05/2024).

- [22] V. Megaro, B. Thomaszewski, M. Nitti, O. Hilliges, M. Gross, and S. Coros. “Interactive design of 3d-printable robotic creatures”. en. In: *ACM Transactions on Graphics* (2015).
- [23] A. Schulz, C. Sung, A. Spielberg, W. Zhao, R. Cheng, E. Grinspun, D. Rus, and W. Matusik. “Interactive robogami: An end-to-end system for design of robots with ground locomotion”. en. In: *The International Journal of Robotics Research* 36.10 (Sept. 2017). Publisher: SAGE Publications Ltd STM, pp. 1131–1147. ISSN: 0278-3649. DOI: [10.1177/0278364917723465](https://doi.org/10.1177/0278364917723465). URL: <https://doi.org/10.1177/0278364917723465> (visited on 01/04/2024).
- [24] J. Hiller and H. Lipson. “Automatic Design and Manufacture of Soft Robots”. In: *IEEE Transactions on Robotics* 28.2 (Apr. 2012). Conference Name: IEEE Transactions on Robotics, pp. 457–466. ISSN: 1941-0468. DOI: [10.1109/TRO.2011.2172702](https://ieeexplore.ieee.org/document/6096440). URL: <https://ieeexplore.ieee.org/document/6096440> (visited on 01/08/2024).
- [25] A. P. Ghassaei. “Rapid design and simulation of functional digital materials”. PhD Thesis. Massachusetts Institute of Technology, 2016.
- [26] A. Abdel-Rahman. *Amira Abdel-Rahman / MetaVoxels · GitLab*. en. May 2020. URL: <https://www.cba.mit.edu/amiraa/metavoxels> (visited on 01/05/2024).
- [27] E. Strand, F. Tournomousis, and N. Gershenfeld. “Mesoscale material modeling with memoryless isotropic point particles”. In: *Journal of Computational Science* 75 (Jan. 2024), p. 102198. ISSN: 1877-7503. DOI: [10.1016/j.jocs.2023.102198](https://www.sciencedirect.com/science/article/pii/S1877750323002582). URL: <https://www.sciencedirect.com/science/article/pii/S1877750323002582> (visited on 01/04/2024).
- [28] J. Hiller and H. Lipson. “Dynamic Simulation of Soft Multimaterial 3D-Printed Objects”. en. In: *Soft Robotics* 1.1 (Mar. 2014), pp. 88–101. ISSN: 2169-5172, 2169-5180. DOI: [10.1089/soro.2013.0010](https://www.liebertpub.com/doi/10.1089/soro.2013.0010). URL: <https://www.liebertpub.com/doi/10.1089/soro.2013.0010> (visited on 12/28/2024).
- [29] G. Sossou, F. Demoly, H. Belkebir, H. J. Qi, S. Gomes, and G. Montavon. “Design for 4D printing: A voxel-based modeling and simulation of smart materials”. In: *Materials & Design* 175 (Aug. 2019), p. 107798. ISSN: 0264-1275. DOI: [10.1016/j.matdes.2019.107798](https://www.sciencedirect.com/science/article/pii/S0264127519302357). URL: <https://www.sciencedirect.com/science/article/pii/S0264127519302357> (visited on 03/19/2025).
- [30] A. Parra Rubio, D. Fan, B. Jenett, J. Del Águila Ferrandis, F. Tournomousis, A. Abdel-Rahman, D. Preiss, J. Zemánek, M. Triantafyllou, and N. Gershenfeld. “Modular Morphing Lattices for Large-Scale Underwater Continuum Robotic Structures”. en. In: *Soft Robotics* 10.4 (Aug. 2023), pp. 724–736. ISSN: 2169-5172, 2169-5180. DOI: [10.1089/soro.2022.0117](https://www.liebertpub.com/doi/10.1089/soro.2022.0117). URL: <https://www.liebertpub.com/doi/10.1089/soro.2022.0117> (visited on 01/04/2024).
- [31] *What is MADCAT? - NASA*. en-US. Running Time: 41 Section: Aeronautics. Apr. 2019. URL: <https://www.nasa.gov/aeronautics/what-is-madcat/> (visited on 01/11/2024).
- [32] T. Schaedler, A. J. Jacobsen, A. Torrents, A. Sorensen, J. Lian, J. Greer, L. Valdevit, and W. Carter. “Ultralight Metallic Microlattices”. In: *Science* 334 (2011), pp. 962–965. DOI: [10.1126/science.1211649](https://www.semanticscholar.org/paper/7f659db74b67cab16a2a9156875aacb365cd05d3). URL: <https://www.semanticscholar.org/paper/7f659db74b67cab16a2a9156875aacb365cd05d3>.

- [33] X. Zheng et al. “Ultralight, ultrastiff mechanical metamaterials”. In: *Science* 344 (2014), pp. 1373–1377. DOI: [10.1126/science.1252291](https://doi.org/10.1126/science.1252291). URL: <https://www.semanticscholar.org/paper/696d97cd51c6de4d954e43fa90c3faed69d610fd>.
- [34] L. Meza, S. Das, and J. Greer. “Strong, lightweight, and recoverable three-dimensional ceramic nanolattices”. In: *Science* 345 (2014), pp. 1322–1326. DOI: [10.1126/science.1255908](https://doi.org/10.1126/science.1255908). URL: <https://www.semanticscholar.org/paper/b75412504a18c1cb538413d05439193cea524f15>.
- [35] G. Dong, Y. Tang, and Y. F. Zhao. “A 149 line homogenization code for three-dimensional cellular materials written in matlab”. In: *Journal of Engineering Materials and Technology* 141.1 (2019). Publisher: American Society of Mechanical Engineers Digital Collection.
- [36] L. Makatura, B. Wang, Y.-L. Chen, B. Deng, C. Wojtan, B. Bickel, and W. Matusik. “Procedural Metamaterials: A Unified Procedural Graph for Metamaterial Design”. In: *ACM Transactions on Graphics* 42.5 (July 2023), 168:1–168:19. ISSN: 0730-0301. DOI: [10.1145/3605389](https://doi.org/10.1145/3605389). URL: <https://dl.acm.org/doi/10.1145/3605389> (visited on 01/11/2024).
- [37] B. Zhu, M. Skouras, D. Chen, and W. Matusik. “Two-scale topology optimization with microstructures”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017). Publisher: ACM New York, NY, USA, p. 1.
- [38] B. Deng, A. Zareei, X. Ding, J. C. Weaver, C. H. Rycroft, and K. Bertoldi. “Inverse Design of Mechanical Metamaterials with Target Nonlinear Response via a Neural Accelerated Evolution Strategy”. en. In: *Advanced Materials* 34.41 (2022). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.202206238>, p. 2206238. ISSN: 1521-4095. DOI: [10.1002/adma.202206238](https://doi.org/10.1002/adma.202206238). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202206238> (visited on 01/11/2024).
- [39] E. Andreassen and C. S. Andreasen. “How to determine composite material properties using numerical homogenization”. In: *Computational Materials Science* 83 (2014). Publisher: Elsevier, pp. 488–495.
- [40] J. Gao, Z. Luo, L. Xia, and L. Gao. “Concurrent topology optimization of multiscale composite structures in Matlab”. en. In: *Structural and Multidisciplinary Optimization* 60.6 (Dec. 2019), pp. 2621–2651. ISSN: 1615-1488. DOI: [10.1007/s00158-019-02323-6](https://doi.org/10.1007/s00158-019-02323-6). URL: <https://doi.org/10.1007/s00158-019-02323-6> (visited on 01/04/2024).
- [41] N. Aage, E. Andreassen, B. S. Lazarov, and O. Sigmund. “Giga-voxel computational morphogenesis for structural design”. eng. In: *Nature* 550.7674 (Oct. 2017), pp. 84–86. ISSN: 1476-4687. DOI: [10.1038/nature23911](https://doi.org/10.1038/nature23911).
- [42] J. Panetta, Q. Zhou, L. Malomo, N. Pietroni, P. Cignoni, and D. Zorin. “Elastic textures for additive fabrication”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015). Publisher: ACM New York, NY, USA, pp. 1–12.
- [43] B. Jenett, A. Abdel-Rahman, K. Cheung, and N. Gershenfeld. “Material–robot system for assembly of discrete cellular structures”. In: *IEEE Robotics and Automation Letters* 4.4 (2019). Publisher: IEEE, pp. 4019–4026.

- [44] A. Abdel-Rahman, C. Cameron, B. Jenett, M. Smith, and N. Gershenfeld. “Self-replicating hierarchical modular robotic swarms”. en. In: *Communications Engineering* 1.1 (Nov. 2022), p. 35. ISSN: 2731-3395. DOI: [10.1038/s44172-022-00034-3](https://doi.org/10.1038/s44172-022-00034-3). URL: <https://www.nature.com/articles/s44172-022-00034-3> (visited on 01/04/2024).
- [45] A. Costa, A. Abdel-Rahman, B. Jenett, N. Gershenfeld, I. Kostitsyna, and K. Cheung. “Algorithmic approaches to reconfigurable assembly systems”. In: *2019 IEEE Aerospace Conference*. IEEE, 2019, pp. 1–8.
- [46] D. Grattarola and P. Vanderghenst. “Generalised Implicit Neural Representations”. en. In.
- [47] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*. arXiv:2006.10739 [cs]. June 2020. DOI: [10.48550/arXiv.2006.10739](https://doi.org/10.48550/arXiv.2006.10739). URL: <http://arxiv.org/abs/2006.10739> (visited on 01/04/2024).
- [48] A. Chandrasekhar and K. Suresh. *Length Scale Control in Topology Optimization using Fourier Enhanced Neural Networks*. arXiv:2109.01861 [cs]. Sept. 2021. DOI: [10.48550/arXiv.2109.01861](https://doi.org/10.48550/arXiv.2109.01861). URL: <http://arxiv.org/abs/2109.01861> (visited on 01/04/2024).
- [49] Z. Qiu, Q. Li, S. Liu, and R. Xu. “Clustering-based concurrent topology optimization with macrostructure, components, and materials”. en. In: *Structural and Multidisciplinary Optimization* 63.3 (Mar. 2021), pp. 1243–1263. ISSN: 1615-1488. DOI: [10.1007/s00158-020-02755-5](https://doi.org/10.1007/s00158-020-02755-5). URL: <https://doi.org/10.1007/s00158-020-02755-5> (visited on 07/31/2025).
- [50] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin. “Growing Neural Cellular Automata”. en. In: *Distill* 5.2 (Feb. 2020), e23. ISSN: 2476-0757. DOI: [10.23915/distill.00023](https://doi.org/10.23915/distill.00023). URL: <https://distill.pub/2020/growing-ca> (visited on 01/04/2024).
- [51] M. Hammond, V. Cichella, and C. Lamuta. “Bioinspired Soft Robotics: state of the art, challenges, and future directions”. In: *Current Robotics Reports* 4.3 (Sept. 2023). arXiv:2312.12312 [cs], pp. 65–80. ISSN: 2662-4087. DOI: [10.1007/s43154-023-00102-2](https://doi.org/10.1007/s43154-023-00102-2). URL: <http://arxiv.org/abs/2312.12312> (visited on 07/31/2025).
- [52] *Sustainable Morphing Matter: Design and Engineering Practices - Patel - 2023 - Advanced Materials Technologies - Wiley Online Library*. URL: <https://advanced-onlinelibrary-wiley-com.libproxy.mit.edu/doi/10.1002/admt.202300678> (visited on 07/31/2025).
- [53] *Spider-Inspired, Fully 3D-Printed Micro-Hydraulics for Tiny, Soft Robotics - Smith - 2023 - Advanced Functional Materials - Wiley Online Library*. URL: <https://advanced-onlinelibrary-wiley-com.libproxy.mit.edu/doi/10.1002/adfm.202207435> (visited on 07/31/2025).

- [54] Q. Wang, C. Pan, Y. Zhang, L. Peng, Z. Chen, C. Majidi, and L. Jiang. “Magnetoactive liquid-solid phase transitional matter”. English. In: *Matter* 6.3 (Mar. 2023). Publisher: Elsevier, pp. 855–872. ISSN: 2590-2393, 2590-2385. DOI: [10.1016/j.matt.2022.12.003](https://doi.org/10.1016/j.matt.2022.12.003). URL: [https://www.cell.com/matter/abstract/S2590-2385\(22\)00693-2](https://www.cell.com/matter/abstract/S2590-2385(22)00693-2) (visited on 01/22/2024).
- [55] Y. Meng, Y. Zhang, and Y. Jin. “Autonomous self-reconfiguration of modular robots by evolving a hierarchical mechanochemical model”. In: *IEEE Computational Intelligence Magazine* 6.1 (2011). Publisher: IEEE, pp. 43–54.
- [56] *Masterprint*. 2020. URL: <https://en.machinetools.camozzi.com/products/additive-manufacturing/all-products/masterprint-5x-kl> (visited on 09/30/2020).
- [57] “Nanoscribe GmbH”. pt. In: *Nanoscribe Technology*. nanoscribe.de/en/technology/ (2017).
- [58] E. Thomson, S. Ferreira-Cerca, and E. Hurt. “Eukaryotic ribosome biogenesis at a glance”. In: *Journal of cell science* 126.21 (2013). Publisher: The Company of Biologists Bidder Building, 140 Cowley Road, Cambridge, CB4 ..., pp. 4815–4821.
- [59] “Particle Sciences. Protein Structure”. es. In: *Tech Br* 8 (2009), pp. 1–2.
- [60] M. Rubenstein, A. Cornejo, and R. Nagpal. “Programmable self-assembly in a thousand-robot swarm”. In: *Science* 345.6198 (2014). Publisher: American Association for the Advancement of Science, pp. 795–799.
- [61] P. R. Wurman, R. D’Andrea, and M. Mountz. “Coordinating hundreds of cooperative, autonomous vehicles in warehouses”. In: *AI magazine* 29.1 (2008), pp. 9–9.
- [62] X. Zhang, M. Li, J. H. Lim, Y. Weng, Y. W. D. Tay, H. Pham, and Q.-C. Pham. “Large-scale 3D printing by a team of mobile robots”. In: *Automation in Construction* 95 (2018). Publisher: Elsevier, pp. 98–106.
- [63] M. Kayser, L. Cai, S. Falcone, C. Bader, N. Inglessis, B. Darweesh, and N. Oxman. “FIBERBOTS: an autonomous swarm-based robotic system for digital fabrication of fiber-based composites”. In: *Construction Robotics* 2.1 (2018). Publisher: Springer, pp. 67–79.
- [64] J. Werfel, K. Petersen, and R. Nagpal. “Designing collective behavior in a termite-inspired robot construction team”. In: *Science* 343.6172 (2014). Publisher: American Association for the Advancement of Science, pp. 754–758.
- [65] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. “Modular self-reconfigurable robot systems [grand challenges of robotics]”. In: *IEEE Robotics & Automation Magazine* 14.1 (2007). Publisher: IEEE, pp. 43–52.
- [66] M. Yim, D. G. Duff, and K. D. Roufas. “PolyBot: a modular reconfigurable robot”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE, 2000, pp. 514–520.
- [67] D. Rus, Z. Butler, K. Kotay, and M. Vona. “Self-reconfiguring robots”. In: *Communications of the ACM* 45.3 (2002). Publisher: ACM New York, NY, USA, pp. 39–45.

- [68] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson. “Self-reproducing machines”. In: *Nature* 435.7039 (2005). Publisher: Nature Publishing Group, pp. 163–164.
- [69] J. Neubert and H. Lipson. “Soldercubes: a self-soldering self-reconfiguring modular robot system”. In: *Autonomous Robots* 40.1 (2016). Publisher: Springer, pp. 139–158.
- [70] G. Trinh, D. Cellucci, B. Jenett, S. Nowak, S. Hu, M. O’Connor, G. Copplestone, and K. Cheung. “Robotically Assembled Aerospace Structures: Digital Material Assembly using a Gantry-Type Assembler”. en. In: *IEEE Aerospace Conference Proceedings*. 2017.
- [71] W. Langford and N. Gershenfeld. “Discretely assembled walking machines”. de. In: *J Micro-Bio Robot* (2020).
- [72] W. K. Langford. “Discrete robotic construction”. PhD Thesis. Massachusetts Institute of Technology, 2019.
- [73] S. Cameron. “Enhancing GJK: Computing minimum and penetration distances between convex polyhedra”. en. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 1997.
- [74] A. Roosing. “Fast Distance Fields for Fluid Dynamics Mesh Generation on Graphics Hardware”. en. In: *Commun Comput Phys* (2019).
- [75] S. Frisken, R. Perry, A. Rockwood, and T. Jones. “Adaptively sampled distance fields: A general representation of shape for computer graphics”. en. In: *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*. 2000.
- [76] O. Formoso, C. Gregg, G. Trinh, A. Rogg, and K. Cheung. “Androgynous Fasteners for Robotic Structural Assembly”. en. In: *IEEE Aerospace Conference Proceedings*. 2020.
- [77] A. Knaian. *Electropermanent magnetic connectors and actuators: devices and their application in programmable matter*. en. MIT, 2010.
- [78] B. Weight, S. Lyon, L. Howell, and S. Wait. “Two-position micro latching mechanism requiring a Single Actuator”. en. In: *Proceedings of the ASME Design Engineering Technical Conference*. 2002.
- [79] J. Marden and L. Allen. “Molecules, muscles, and machines: Universal performance characteristics of motors”. pt. In: *Proc Natl Acad Sci U S A* (2002).
- [80] R. Desai, Y. Yuan, and S. Coros. “Computational abstractions for interactive design of robotic devices”. en. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2017.
- [81] Y. Deng, Y. Hua, N. Napp, and K. Petersen. “A Compiler for Scalable Construction by the TERMES Robot Collective”. en. In: *Rob Auton Syst* (2019).
- [82] E. Niehs, A. Schmidt, C. Scheffer, D. Biediger, M. Yannuzzi, B. Jenett, A. Abdel-Rahman, K. Cheung, A. Becker, and S. Fekete. *Recognition and Reconfiguration of Lattice-Based Cellular Structures by Simple Robots*. en. 2020.

- [83] N. Melenbrink, P. Kassabian, A. Menges, and J. Werfel. “Towards force-aware robot collectives for on-site construction”. en. In: *Disciplines and Disruption - Proceedings Catalog of the 37th Annual Conference of the Association for Computer Aided Design in Architecture, ACADIA 2017*. 2017.
- [84] S. Brûlé, S. Enoch, and S. Guenneau. *Emergence of seismic metamaterials: Current state and future perspectives*. en. Section A: Physics Letters, 2020.
- [85] R. Bogue. “DaimlerChrysler installs new robot-based flexible assembly line”. In: *Industrial Robot: An International Journal* (2008). Publisher: Emerald Group Publishing Limited.
- [86] R. Holden, P. Haworth, I. Kendrick, and A. Smith. *Automated riveting cell for A320 wing panels with improved throughput and reliability (SA2)*. en. In: SAE Technical Papers, 2007.
- [87] J. Markish and K. Willcox. “Value-Based Multidisciplinary Techniques for Commercial Aircraft System Design”. fr. In: *AIAA J* (2008).
- [88] A. Brunete, A. Ranganath, S. Segovia, J. P. de Frutos, M. Hernando, and E. Gambao. “Current trends in reconfigurable modular robots design”. In: *International Journal of Advanced Robotic Systems* 14.3 (2017). Publisher: SAGE Publications Sage UK: London, England, p. 1729881417710457.
- [89] J. Seo, J. Paik, and M. Yim. “Modular Reconfigurable Robotics”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019). _eprint: <https://doi.org/10.1146/annurev-control-053018-023834>, pp. 63–88. DOI: [10.1146/annurev-control-053018-023834](https://doi.org/10.1146/annurev-control-053018-023834). URL: <https://doi.org/10.1146/annurev-control-053018-023834> (visited on 01/08/2024).
- [90] H. Lipson and J. B. Pollack. “Automatic design and manufacture of robotic lifeforms”. en. In: *Nature* 406.6799 (Aug. 2000). Number: 6799 Publisher: Nature Publishing Group, pp. 974–978. ISSN: 1476-4687. DOI: [10.1038/35023115](https://doi.org/10.1038/35023115). URL: <https://www.nature.com/articles/35023115> (visited on 01/04/2024).
- [91] J. B. Pollack, G. S. Hornby, H. Lipson, and P. Funes. “Computer Creativity in the Automatic Design of Robots”. en. In: *Leonardo* 36.2 (Apr. 2003), pp. 115–121. ISSN: 0024-094X, 1530-9282. DOI: [10.1162/002409403321554170](https://doi.org/10.1162/002409403321554170). URL: <https://direct.mit.edu/leon/article/36/2/115-121/44357> (visited on 01/08/2024).
- [92] A. Zhao, J. Xu, M. Konakovic-Lukovic, J. Hughes, A. Spielberg, D. Rus, and W. Matusik. “RoboGrammar: Graph Grammar for Terrain-Optimized Robot Design”. In: *SIGGRAPH Asia*. 2020.
- [93] L. C. Schmidt, H. Shetty, and S. C. Chase. “A graph grammar approach for structure synthesis of mechanisms”. In: *J. Mech. Des.* 122.4 (2000), pp. 371–376.
- [94] M. Van Diepen and K. Shea. “A spatial grammar method for the computational design synthesis of virtual soft locomotion robots”. In: *Journal of Mechanical Design* 141.10 (2019). Publisher: American Society of Mechanical Engineers Digital Collection.

- [95] S. Ha, S. Coros, A. Alspach, J. M. Bern, J. Kim, and K. Yamane. “Computational Design of Robotic Devices From High-Level Motion Specifications”. In: *IEEE Transactions on Robotics* 34.5 (Oct. 2018). Conference Name: IEEE Transactions on Robotics, pp. 1240–1251. ISSN: 1941-0468. DOI: [10.1109/TRO.2018.2830419](https://doi.org/10.1109/TRO.2018.2830419). URL: <https://ieeexplore.ieee.org/document/8395009> (visited on 01/08/2024).
- [96] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros. *Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity*. arXiv:1902.05546 [cs, stat]. Nov. 2019. DOI: [10.48550/arXiv.1902.05546](https://doi.org/10.48550/arXiv.1902.05546). URL: <http://arxiv.org/abs/1902.05546> (visited on 01/05/2024).
- [97] D. Ha. “Reinforcement Learning for Improving Agent Design”. In: *Artificial Life* 25.4 (Nov. 2019). arXiv:1810.03779 [cs, stat], pp. 352–365. ISSN: 1064-5462, 1530-9185. DOI: [10.1162/artl_a_00301](https://doi.org/10.1162/artl_a_00301). URL: <http://arxiv.org/abs/1810.03779> (visited on 01/09/2024).
- [98] K. Sims. “Evolving virtual creatures”. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 1994, pp. 15–22.
- [99] K. Sims. “Evolving 3D Morphology and Behavior by Competition”. en. In: 1.4.
- [100] G. Nadizar, E. Medvet, and K. Miras. “On the Schedule for Morphological Development of Evolved Modular Soft Robots”. en. In: *Genetic Programming*. Ed. by E. Medvet, G. Pappa, and B. Xue. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 146–161. ISBN: 978-3-031-02056-8. DOI: [10.1007/978-3-031-02056-8_10](https://doi.org/10.1007/978-3-031-02056-8_10).
- [101] S. Kriegman, N. Cheney, and J. Bongard. “How morphological development can guide evolution”. en. In: *Scientific Reports* 8.1 (Sept. 2018). Number: 1 Publisher: Nature Publishing Group, p. 13934. ISSN: 2045-2322. DOI: [10.1038/s41598-018-31868-7](https://doi.org/10.1038/s41598-018-31868-7). URL: <https://www.nature.com/articles/s41598-018-31868-7> (visited on 01/08/2024).
- [102] A. Abdel-Rahman. *Amira Abdel-Rahman / NVIDIA Omniverse MetaVoxels · GitLab*. en. Nov. 2022. URL: <https://www.cba.mit.edu/amiraa/nvidia-omniverse-metavoxels> (visited on 07/31/2025).
- [103] *Isaac Gym Benchmark Environments*. original-date: 2021-08-27T02:36:51Z. Jan. 2024. URL: <https://github.com/NVIDIA-Omniverse/IsaacGymEnvs> (visited on 01/05/2024).
- [104] S. Huang and S. Ontañón. “A Closer Look at Invalid Action Masking in Policy Gradient Algorithms”. In: *The International FLAIRS Conference Proceedings* 35 (May 2022). arXiv:2006.14171 [cs]. ISSN: 2334-0762. DOI: [10.32473/flairs.v35i.130584](https://doi.org/10.32473/flairs.v35i.130584). URL: <http://arxiv.org/abs/2006.14171> (visited on 08/01/2025).
- [105] M. Smith, A. Abdel-Rahman, and N. Gershenfeld. “Self-Reconfigurable Robots for Collaborative Discrete Lattice Assembly”. en. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan: IEEE, May 2024, pp. 3624–3631. ISBN: 9798350384574. DOI: [10.1109/ICRA57147.2024.10609866](https://doi.org/10.1109/ICRA57147.2024.10609866). URL: <https://ieeexplore.ieee.org/document/10609866/> (visited on 07/31/2025).

- [106] Z. Fredin, J. Zemanek, C. Blackburn, E. Strand, A. Abdel-Rahman, P. Rowles, and N. Gershenfeld. “Discrete Integrated Circuit Electronics (DICE)”. en. In: *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. Waltham, MA, USA: IEEE, Sept. 2020, pp. 1–8. ISBN: 978-1-72819-219-2. DOI: [10.1109/HPEC43674.2020.9286236](https://doi.org/10.1109/HPEC43674.2020.9286236). URL: <https://ieeexplore.ieee.org/document/9286236/> (visited on 08/01/2025).
- [107] A. Abdel-Rahman. *Amira Abdel-Rahman / Physical Computing Design Tools · GitLab*. en. Sept. 2024. URL: <https://gitlab.cba.mit.edu/amiraa/physical-computing-design-tools> (visited on 08/01/2025).
- [108] M. Smith, J. Forman, A. Abdel-Rahman, S. Wang, and N. Gershenfeld. “Voxel Invention Kit: Reconfigurable Building Blocks for Prototyping Interactive Electronic Structures”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. CHI ’25. New York, NY, USA: Association for Computing Machinery, Apr. 2025, pp. 1–15. ISBN: 9798400713941. DOI: [10.1145/3706598.3713948](https://doi.org/10.1145/3706598.3713948). URL: <https://dl.acm.org/doi/10.1145/3706598.3713948> (visited on 07/31/2025).
- [109] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. “Zero-Shot Text-to-Image Generation”. en. In: *Proceedings of the 38th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2021, pp. 8821–8831. URL: <https://proceedings.mlr.press/v139/ramesh21a.html> (visited on 07/31/2025).
- [110] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan. *InstantMesh: Efficient 3D Mesh Generation from a Single Image with Sparse-view Large Reconstruction Models*. arXiv:2404.07191 [cs]. Apr. 2024. DOI: [10.48550/arXiv.2404.07191](https://doi.org/10.48550/arXiv.2404.07191). URL: <http://arxiv.org/abs/2404.07191> (visited on 07/31/2025).
- [111] Z. Zong, X. Li, M. Li, M. M. Chiaramonte, W. Matusik, E. Grinspun, K. Carlberg, C. Jiang, and P. Y. Chen. “Neural Stress Fields for Reduced-order Elastoplasticity and Fracture”. en. In: *SIGGRAPH Asia 2023 Conference Papers*. Sydney NSW Australia: ACM, Dec. 2023, pp. 1–11. ISBN: 9798400703157. DOI: [10.1145/3610548.3618207](https://doi.org/10.1145/3610548.3618207). URL: <https://dl.acm.org/doi/10.1145/3610548.3618207> (visited on 07/31/2025).
- [112] Miana Smith, Paul Richard, and Neil Gershenfeld. “Sustainability and Economics of Discretely Assembled Architected Lattices for Construction Automation”. In: *(under preparation)*. 2025.