CARNEGIE MELLON UNIVERSITY

PH.D. THESIS DEFENSE

Computational Design of Morphing Looped Graph Structures

Author:

Jianzhe GU

Thesis Committee:
Lining Yao (Chair)
Nikolas Martelaro
Alexandra Ion
Ding Zhao

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

Dec. 16, 2024

Technical Report Number: CMU-HCII-24-112

CARNEGIE MELLON UNIVERSITY

Abstract

Human-Computer Interaction Institute

Computational Design of Morphing Looped Graph Structures

by Jianzhe GU

Over the past few decades, robotics has undergone a transformative evolution, yet creating machines capable of dramatic volumetric shape changes while maintaining high structural stability and precision remains elusive. Among various approaches, truss robots and tensegrity robots - which we term looped-graph structures (LGSs) - offer unique advantages through their architecture of nodes and length-changeable edges. These structures leverage their graph topology with loops, where edges can circle back to their starting nodes, to achieve extensive degrees of freedom and distinctive shape-changing capabilities while maintaining structural stability through their distributed network.

LGSs face two fundamental challenges that have limited their practical implementation. First, as their complexity grows exponentially with size, they encounter the Curse of Dimensionality (CoD), making both physical fabrication and control system design increasingly intractable. Second, the discrete nature of their graph topology and categorical parameters like actuator grouping assignments make traditional continuous optimization approaches ineffective, particularly as the design space grows geometrically with robot size.

We present a systematic progression of solutions that addresses both challenges. To tackle the CoD in physical implementation, we introduce an actuator grouping mechanism inspired by biological muscle synergy, where complex movements emerge from coordinated muscle groups rather than individual control. This enables the fabrication of truss robots with over 100 actuators controlled by just a few modules, dramatically reducing system complexity while maintaining shape-changing capabilities. Building on this foundation, we develop an interactive editor with real-time simulation that bridges the gap between conceptual design and physical implementation.

For complex structures capable of multiple tasks, we first implement a customized genetic algorithm that navigates the discrete graph space while respecting connectivity constraints. However, as truss complexity scales up, this discrete optimization becomes inefficient in the geometrically increasing search space. To overcome these limitations and enable topology generation, we develop two complementary approaches using variational auto-encoders (VAE) that transform the discrete design space into continuous latent representations for more efficient optimization. We introduce a novel truss grammar that represents designs through sequential tokens and parameters, enabling translation between discrete structures and continuous latent spaces. Our graph attention network approach achieves 99.925% accuracy in reconstructing actuator groupings, while our long short-term memory network successfully generates complete truss topologies and parameters, creating the first end-to-end framework for optimizing both discrete topology and continuous parameters of truss robots.

Through demonstrations ranging from quadrupedal locomotion to shape-shifting structures, we show that our framework enables the practical implementation of complex morphing LGSs. By bridging the gap between theoretical capability and physical realization, this work establishes foundations for a new generation of adaptive robots that can reshape themselves to meet diverse task requirements.

Acknowledgements

This research journey has been simultaneously challenging and enriching, filled with challenges, discoveries, and meaningful collaborations. It has been enriched by the guidance, support, and friendship of numerous individuals who have contributed to both my research and personal interests.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Lining Yao, and co-advisor, Professor Ding Zhao. Their patience, inspiration, and knowledge have been instrumental in shaping this work. They have consistently demonstrated both ambitious vision and attention to detail, while providing unwavering support under various conditions.

I am particularly grateful to my thesis committee members: Professor Lining Yao (Chair), Professor Nikolas Martelaro, Professor Alexandra Ion, and Professor Ding Zhao. Their diverse perspectives and constructive feedback have significantly strengthened this research.

I would like to thank the Human-Computer Interaction Institute at Carnegie Mellon University, particularly Department Head Scott Hudson and Professor Geoff Kaufman, for fostering an environment conducive to interdisciplinary research.

I gratefully acknowledge the financial support that made this Ph.D. research possible. This work was supported by the U.S. Department of Defense through the Multidisciplinary University Research Initiatives (MURI) program, the National Science Foundation (NSF), and Honda Research Institute.

The administrative staff have been crucial in navigating this journey. Special thanks to Queenie Kravitz for her invaluable suggestions and support at critical moments, and to Leah Buffington, Becky Wang, Eric Davison, Lindsay Olshenske, and Nick Hernandez for their assistance.

I have been fortunate to work with many talented classmates, collaborators, and colleagues. I would particularly like to thank: Guanyun Wang, Humphrey Yang, Lea Albaugh, Stella Shen, Dinesh K Patel, Yuyu Lin, Ke Zhong, Michael Vinciguerra, Shuhong Wang, Danli Luo, Youngwook Do, Yibo Fu, Jenny Hu, Qiuyu Lu, Harshika Jain, Kexin Lu, Emily Guan, Sunniva Liu, Tate Johnson, Tucker Rae-Grant, Ziwen Ye, Fang Qin, Cathy Mengying Fang, Chris Harrison, Vidya Narayanan, James McCann, Victoria Webster-Wood, Qiang Cui, Tingyu Cheng, Haiqing Xu, Di Wu, Melinda Chen, Jack Forman, Taylor Tabb, Koya Narumi, Sijia Wang, Haiqing Xu, Zeyu Yan, Kuanren Qian, Yuxuan Yu, Haolin Liu, David Jourdan, Matthew McGehee, Lingyun Sun, Jiaji Li, Yue Yang, Hengrong Ni, Tianyu Yu, Advait Wadhwani, Andreea Danielescu, Lydia Yang, Aditi Maheshwari, Byoungkwon An, Yu Chen, Xiaoqian Li, Cheng Yao, Fangtian Ying, Zhi Yu, Tianying Chen, Amber Horvath, Andrew Kuznetsov, Danny Weitekamp, Huy Nguyen, Hyeonsu Kang, Jason Wu, Michael Xieyang Liu, Tianshi Li, Chris Atkeson, Youda Huang, Haiyi Zhu, Robert Kraut, Roberta Klatzky, Maxwell Abbott, Vivian Wong, Clara Hsu, Ryan Patel, Felipe Martinez, and Aniket (Niki) Kittur.

I am grateful for the collaborative relationships with researchers at other institutions: Guanyun Wang (Zhejiang University), Ye Tao (Zhejiang University City College), David E. Breen (Drexel University), Sam Kriegman and Wei Chen (Northwestern University), Teng Zhang (Syracuse University), Ido Levin and Eran Sharon (Hebrew University), and Lifeng Zhu (Dongnan University), Josiah Hester(Geogia Tech Univerisity).

Personally and most importantly, I want to express my profound gratitude to my family - my parents, whose unconditional love and sacrifice have made this journey possible, and my extended family, whose encouragement from afar has kept me grounded. I am deeply thankful to my girlfriend, Sophie Kang, for her unwavering support and understanding throughout this journey. I am also grateful to my friends who have made the United States feel like

home: June Song, Yibo Fu, Jeff Chao, Dingkun Guo, Ziqi Wang, Cone, Zhitong Cui, Eddie Huang, Eden Ritchie, and Bill Kenny.

This thesis builds upon the work of countless researchers in robotics, computer science, and related fields. While I cannot name them all, I am grateful to stand on the shoulders of giants.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nature offers countless examples of creatures that dramatically transform their morphology to adapt to their environment and tasks. From caterpillars that inch through narrow spaces to octopus that squeeze through tiny openings, this adaptability enables survival in diverse conditions. Inspired by these remarkable capabilities, robotics researchers have long sought to create machines that can similarly modify their shape and structure. However, achieving such versatile morphological adaptation while maintaining structural integrity and control precision remains a significant challenge in robotics. Recent studies have demonstrated that a robot's morphology fundamentally determines its functional capabilities [1]. A robot that can actively modify its shape gains remarkable advantages: it can compress itself to navigate confined spaces, conform to objects for better manipulation, and even alter its appearance to facilitate more natural human-robot interaction. These capabilities become particularly crucial in scenarios where environmental conditions or task requirements can change dramatically, from search and rescue operations in collapsed buildings to adaptive furniture that responds to user needs.

The unique adaptability of living organisms extends far beyond simple compressions or expansions. Lizards like the frilled-neck lizard dramatically modify their throat structure, expanding dewlaps to intimidate predators or attract mates [2]. The magnificent frigatebird can inflate its gular pouch to nearly its body size during courtship displays, transforming its appearance for social signaling [3]. Perhaps most remarkably, octopi can modulate their entire body shape fluidly, not only for squeezing through constraints but also for complex behaviors like tool manipulation and environmental exploration [4]. These adaptations represent more than just changes in size or shape - they demonstrate fundamental alterations in body volume and structure that serve diverse functions from defense and social communication to hunting and survival. Notably, these transformations often involve body regions that defy simple categorization into traditional anatomical parts like limbs or appendages. Instead, they represent integrated, whole-body adaptations that blur the distinction between structural and functional elements.

The quest for morphological adaptability has led to several distinct approaches in robot design. Each category offers unique advantages while facing specific limitations in achieving versatile shape transformation (Fig. 1.1). **Limbed robots**, which mirror the anatomical structure of animals and humans [5, 6], utilize a central body with branching limbs [7, 8]. These robots can be equipped with diverse end-effectors such as wheels, grippers, or sensors, and some feature variable limb lengths [9, 10] or morphing wheels [11]. While effective for many tasks, their tree-like topology [12–14] constrains their shape-changing capabilities. They can move limbs but cannot fundamentally alter their volume or overall structure. **Continuum robots**, inspired by spineless organisms like octopi and worms [15, 16], represent another approach. These robots employ soft, continuously deformable materials that provide virtually infinite degrees of freedom in movement and shape adaptation. Much like how an

octopus can squeeze through openings barely larger than its eye, these robots can theoretically achieve remarkable flexibility. However, this flexibility comes at a cost: they often lack mechanical stability and load-bearing capacity. Moreover, their reliance on complex material properties presents significant challenges in fabrication and precise control, particularly for larger structures or extended operations.

A robot's morphology fundamentally determines not just its physical capabilities but also its potential for meaningful interaction with humans and environments. Shape-changing robots have emerged as a powerful paradigm in human-computer interaction (HCI) and human-robot interaction (HRI), where their adaptability serves diverse functions beyond traditional robotics tasks. In HCI, shape-changing interfaces enable dynamic physical affordances, convey information through form changes, and adapt to different usage contexts. For instance, a single adaptive structure might transform from a flat surface for display to a curved form for ergonomic interaction, or reconfigure itself to provide tactile feedback or physical constraints. In HRI scenarios, a robot's ability to modify its shape can significantly enhance its social presence and functionality - from adjusting its posture to match the social context, to modifying its form factor to better suit different interaction scenarios. This expansion beyond conventional robotic applications has driven the development of various shape-changing systems, each addressing different scales and interaction requirements.

At room and furniture scales, shape-changing robots demonstrate remarkable potential for adapting physical spaces to user needs. Recent advances have produced systems capable of dramatic volumetric transformations while maintaining practical utility. InflatableBots [17] and RoomShift [18] enable large-scale encountered-type haptics in virtual reality through different approaches - the former using inflatable structures and the latter employing furniture-moving swarm robots. These systems can dynamically reconfigure spaces, creating physical affordances that match virtual environments. Similarly, LiftTiles [19] demonstrates how modular, reconfigurable actuators can construct room-scale shape-changing interfaces, enabling adaptive furniture and environmental-scale haptic feedback. More permanent installations like the Constraint-Driven Robotic Surfaces [20] show how shape-changing walls can morph into functional furniture elements, while systems like TRANSFORM [21] and inFORM [22] create dynamic physical affordances through arrays of actuated elements. These implementations reveal a crucial insight: effective shape-changing systems must balance transformation capabilities with structural stability and practical utility.

At desktop and personal scales, shape-changing interfaces demonstrate even greater diversity in their transformation capabilities and interaction modalities. Systems like ChainFORM [23] and LineFORM [24] showcase how linear, reconfigurable structures can serve multiple functions - morphing from information displays to wearable devices, or transitioning between interactive tools and aesthetic elements. PneUI [25] introduces pneumatically actuated interfaces that can adapt their form for different contexts, from shape-changing mobile devices to responsive wearables that modify their structure based on user needs. More specialized systems like MorphIO [26] demonstrate entirely soft sensing and actuation modules that enable programmable shape changes through tangible interaction, while ShapeBots [27] shows how swarms of small, shape-changing robots can collectively form larger structures for various interaction scenarios. Even seemingly simple interfaces like Dynablock [28] and the Fluidic Computation Kit [29] push the boundaries of what's possible, enabling dynamic 3D shape formation and mechanical logic-driven transformations respectively. These systems highlight a critical advantage of smaller-scale shape-changing interfaces: their ability to seamlessly transition between different functional states while maintaining close physical interaction with users.

The role of morphological adaptation becomes particularly crucial in social robotics, where a robot's physical form directly influences human perception and interaction. Studies in social robot morphology [30, 31] reveal how different physical forms evoke distinct

social responses - from the level of trust users place in the robot to their willingness to engage in various interaction scenarios. This understanding has led to innovations in dynamic morphological expression, exemplified by research in texture-changing robot skin [32–34]. These systems demonstrate how subtle changes in surface morphology can convey emotional states and social signals, much like the social displays observed in nature. Beyond simple aesthetic changes, morphological adaptation in social robots can serve functional purposes - a robot might appear more approachable during casual interactions, then reconfigure to provide physical assistance when needed [35]. The ability to transition between different morphological states becomes particularly valuable in healthcare and social care settings, where robots must adapt their appearance and functionality to suit various care scenarios while maintaining appropriate social presence. This dynamic adaptation capability represents a significant advance over traditional fixed-morphology robots, enabling more nuanced and context-appropriate human-robot interactions.

Among various shape-changing structures such as linear pin arrays [36], inflatable bladders [37, 38], and origami [39], meshes and trusses have been widely adopted for their advantages in modularity, reconfigurability, and high volume-weight ratio. **Looped Graph Robots (LGRs)**, encompassing both truss robots and tensegrity robots, emerge as a promising alternative that combines structural stability with shape-changing capabilities. Truss robots consist of joints connected by rigid linear actuators whose varying lengths enable complex shape transformations [40]. These Variable Geometry Trusses (VGTs) can achieve diverse transformations including rotation, twisting, linear scaling, and volumetric changes. This flexibility enables a wide range of capabilities: from locomotion [41, 42] and manipulation [43, 44] to specialized morphological adaptations [45–48]. The unique advantage of LGRs lies in their graph topology with loops, where edges can circle back to their starting nodes. This topology provides both structural stability and extensive degrees of freedom, allowing for dramatic shape changes while maintaining load-bearing capacity. Tensegrity robots [49] further extend this concept by combining cables and rods in a prestressed network, offering exceptional strength-to-weight ratios and natural impact absorption capabilities. Both types of LGRs share a crucial feature: their sparse internal volume allows for more extensive shape changes compared to solid-body alternatives, while their discrete structure enables more precise control than continuous soft robots.

1.2 Background

The quest for morphological adaptability has led to several distinct approaches in robot design. Each category offers unique advantages while facing specific limitations in achieving versatile shape transformation (Figure 1.1).

Limbed robots, which mirror the anatomical structure of animals and humans [5, 6], utilize a central body with branching limbs [7, 8]. These robots can be equipped with diverse end-effectors such as wheels, grippers, or sensors, and some feature variable limb lengths [9, 10] or morphing wheels [11]. While effective for many tasks, their tree-like topology [12–14] constrains their shape-changing capabilities - they can move limbs but cannot fundamentally alter their volume or overall structure.

Continuum robots, inspired by spineless organisms like octopi and worms [15, 16], represent another approach. These robots employ soft, continuously deformable materials that provide virtually infinite degrees of freedom in movement and shape adaptation. Much like how an octopus can squeeze through openings barely larger than its eye, these robots can theoretically achieve remarkable flexibility. However, this flexibility comes at a cost: they often lack mechanical stability and load-bearing capacity. Moreover, their reliance on

complex material properties presents significant challenges in fabrication and precise control, particularly for larger structures or extended operations.

Cubic robots take a different approach, building from repeated modular units. Multi-material voxel robots [50–53] achieve shape changes by selectively activating specific voxels with different material properties. While this approach offers good shape approximation capabilities, it faces fundamental limitations in scalability and precision due to the inherent constraints of solid volumes and the complex interactions between connected voxels. Magnetic self-reconfigurable cubic robots [54, 55] enable voxelated shape reassembly through magnetic connections but sacrifice structural integrity and continuous motion capabilities.

Looped Graph Robots (LGRs), encompassing both truss robots and tensegrity robots, emerge as a promising alternative that combines structural stability with shape-changing capabilities. Truss robots consist of joints connected by rigid linear actuators whose varying lengths enable complex shape transformations [40]. These Variable Geometry Trusses (VGTs) can achieve diverse transformations including rotation, twisting, linear scaling, and volumetric changes. This flexibility enables a wide range of capabilities: from locomotion [41, 42] and manipulation [43, 44] to specialized morphological adaptations [45–48].

The unique advantage of LGRs lies in their graph topology with loops, where edges can circle back to their starting nodes. This topology provides both structural stability and extensive degrees of freedom, allowing for dramatic shape changes while maintaining load-bearing capacity. Tensegrity robots [49] further extend this concept by combining cables and rods in a prestressed network, offering exceptional strength-to-weight ratios and natural impact absorption capabilities. Both types of LGRs share a crucial feature: their sparse internal volume allows for more extensive shape changes compared to solid-body alternatives, while their discrete structure enables more precise control than continuous soft robots. These characteristics make LGRs particularly suitable for applications ranging from adaptive furniture to search-and-rescue robots that must navigate varying terrain while maintaining structural integrity.

The final category, **Looped Graph Robots (LGRs)**, represents a distinct approach to morphological adaptation through its unique topology of interconnected nodes and edges. This category encompasses two main types: **truss robots** and **tensegrity robots**.

Truss robots, also known as variable geometry trusses (VGTs), consist of joints (nodes) and rigid linear actuators (edges) arranged in tetrahedral or octahedral structures [40]. By varying the lengths of their actuator beams, these robots can achieve diverse transformations including rotation, twisting, linear movement, and volumetric scaling. This flexibility enables VGTs to perform a wide range of tasks: from traditional robotic functions like locomotion [41, 42] and manipulation [43, 44], to specialized activities requiring complex morphological adaptations [45–48]. However, despite their advantages in degrees-of-freedom (DOFs) and versatility, current VGTs face a fundamental challenge: their control systems scale exponentially with the number of beams [40]. As a result, existing physical implementations are limited to either simple structures with few tetrahedral units [43, 56, 57] or larger structures where only a small subset of beams are actuable [45].

Tensegrity robots [49] offer a complementary approach within the LGR family. These robots combine rigid rods with adjustable cables, achieving dramatic shape changes through the coordinated length adjustments of these components. While this design offers advantages in weight reduction and impact absorption, tensegrity robots share similar challenges with VGTs: they feature discrete looped-graph topologies and must maintain complex rules of self-balance, making their design and control optimization particularly challenging.

The defining characteristic of LGRs is their graph topology with loops, where edges can circle back to their starting nodes. This architecture provides three key advantages: structural strength derived from the looped graph structure, extensive degrees of freedom due to the length variability of each edge, and distinctive volumetric shape-changing capabilities. These

features make LGRs particularly suitable for applications ranging from search-and-rescue robots that must adapt to challenging environments, to companion robots with customizable shapes, to dynamic entertainment structures. However, realizing these applications requires overcoming significant challenges in both design optimization and control.

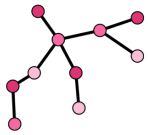
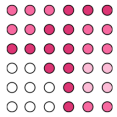
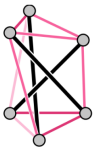
Topology				
Adaptability & Shape Change	While versatile in movement, they may struggle in narrow spaces due to rigid limb structures.	Highly adaptable to confined spaces but may lack the structural strength for diverse tasks.	Limited adaptability in shape and environment due to their uniform, grid-like structure.	Dramatic shape changes, adapting to various environments, e.g. terrains and narrow spaces.
Dynamic Shape for Expression	More functional than expressive, with limited dynamic shape capabilities.	Can be expressive due to flexibility, but less dramatic in shape change compared to LGRs.	Primarily functional; limited in dynamic and expressive shape capabilities.	Capable of expressive dynamic shapes, suitable for both functional and aesthetic applications.
Load and Impact Distribution	Good load-bearing but can have stress points at joints.	Limited load-bearing capacity, primarily due to flexible nature.	Consistent load distribution but limited by the uniformity of the structure.	Efficiently distribute load and impact, enhancing durability and functionality.
Redundancy and Longevity	Less redundancy; damage to a limb can significantly impact performance.	Some redundancy due to flexibility, but overall longevity can be a concern.	Modular nature can offer redundancy, but overall longevity depends on individual module durability.	High redundancy in structure contributes to longer operational life and resilience.
Modularity and Actuation Variety	Modularity is possible but limited by the need for strong, precise	Limited in terms of actuation variety due to the focus on continuous, flexible structures.	High modularity but often restricted to specific types of actuators that fit the grid system.	Allow for a wide range of actuation mechanisms, enhancing modularity and customization.
Simulation and Optimization	Simulation can be complex due to the many moving parts and joints.	Modeling and simulation can be challenging due to the continuous deformation and flexibility.	Relatively easier to simulate due to uniformity, but optimization can be limited by the grid constraints.	Easier to simulate and optimize due to their unique structure, allowing for quicker design iterations.

FIGURE 1.1: Examples, topologies and features of limbed robots [5–8], continuum robots [15, 16], cubic robots [58, 59], and looped graph robots [40, 49].

LGRs derive their unique capabilities from their graph topology with loops, where edges can circle back to their starting nodes. This architecture offers three fundamental advantages: structural strength through the distribution of forces across the looped structure, extensive degrees of freedom from independently variable edge lengths, and distinctive volumetric shape-changing capabilities that surpass those of traditional robotics approaches. These characteristics enable LGRs to achieve complex transformations while maintaining structural integrity - a combination rarely found in other morphing robot designs.

Both truss and tensegrity robots within the LGR family demonstrate remarkable versatility through their large degrees of freedom (DoF). This extensive DoF enables precise local adjustments for manipulation tasks while supporting dramatic whole-body transformations. The key to achieving such versatility lies in the robots' discrete structure - each edge can be independently controlled, creating a highly adaptable system that can be precisely tuned for specific tasks.

This discrete, edge-based architecture offers another crucial advantage: computational tractability. Unlike continuum robots that require complex material modeling, or voxel-based robots that demand intensive computational resources for their solid volumes, LGRs present relatively straightforward simulation models. Their mechanical behavior, while sophisticated, follows clear principles based on the interactions of discrete elements. This simulability makes LGRs particularly suitable for computational tasks like design optimization, motion generation, and performance analysis.

The modular nature of LGRs extends these advantages into practical benefits. By using a sparse network of edges and nodes rather than solid volumes, these robots achieve remarkably high volume-to-weight and volume-to-mass ratios. This efficient use of material

creates structures that are not only lightweight but also highly functional. The open, lattice-like architecture serves multiple purposes: it provides natural mounting points for sensors and actuators, allows for easy access to internal components, and enables straightforward assembly and maintenance. When transportation or storage is needed, these robots can be efficiently disassembled and later reconstructed without compromising their functionality.

These fundamental characteristics - extensive DoF, computational tractability, and modular design - combine to enable a diverse range of applications. In locomotion tasks, LGRs can dynamically adjust their gait and body shape to navigate varying terrains, from urban environments to disaster sites. Their precise control and load-bearing capabilities make them suitable for practical applications like adaptive furniture or reconfigurable workspace elements. As animated sculptures, they can create engaging kinetic displays through controlled, dynamic transformations. Perhaps most significantly, they can actively adapt to environmental challenges, such as flattening to pass through narrow spaces or adjusting their contact geometry for optimal friction and stability.

1.3 Challenge: Curse of Dimensionality and Discreteness in LGR

Despite their theoretical potential for extensive degrees of freedom and dramatic shape changes, the practical implementation of LGRs remains constrained. Current physical realizations of truss robots are limited to either simple topologies with fewer than 40 beams, or more complex structures that exist only in simulation [60]. These limitations stem from two fundamental challenges: the Curse of Dimensionality (CoD) and the discrete nature of graph structures (Figure 1.2).

Despite their advantages in degrees-of-freedom (DOFs) and versatility, current VGTs face scalability issues due to the complexity of their control systems, which scale exponentially with the number of beams [40]. This complexity results in physical VGT implementations being limited to either a few tetrahedral units [43, 56, 57] or structures where only a few beams are actuable [45], restricting their achievable motions. Given the same size of the elements, beam lengths and cable lengths, scaling up the volume of an LGR leads to a cubic increase in the number of elements. This curse of dimensionality manifests in multiple critical ways:

First, the hardware complexity grows exponentially. For truss robots, the assembly effort increases cubically with size. The challenge of individually controlling each actuator becomes even more daunting - while wireless communication and on-body control systems could theoretically make the structure untethered, the associated costs become prohibitive, and the increasing total weight begins to negate the structure's traditionally advantageous volume-to-weight ratio.

The second manifestation of CoD appears in the design and optimization space. As the number of edges in a structure increases linearly, both the possible combinations of actuator groupings and the complexity of control strategies grow geometrically. For example, a linear increase in the volume of an LGR typically leads to a cubically increased search space for combinatorial optimizers.

The other fundamental challenge stems from the discrete nature of the graph structure of LGRs. While graphs offer advantages in sparsity for storage and compatibility with computer data structures, their discrete nature creates significant optimization challenges. Unlike continuum robots that are inherently suitable for continuous optimization, or cubic robots that can be optimized in a continuous space and then discretized to a regular grid structure, the non-regular and non-Euclidean graph structure of LGRs poses unique difficulties. This

discrete nature proves particularly challenging when combined with actuator grouping strategies, limiting optimization approaches to heuristics-based combinatorial methods that often prove less efficient than continuous optimization techniques.

These challenges - the curse of dimensionality and the discrete nature of graph structures - become particularly acute as we attempt to scale LGRs to more complex tasks and larger sizes. The optimization becomes exponentially more difficult as the search space grows larger with the robot's complexity, often increasing rapidly in size. This makes finding optimal configurations, groupings, and balancing various design parameters an increasingly challenging task. The situation becomes even more complex when dealing with tensegrity robots, which must maintain additional constraints for structural stability.

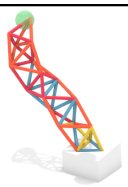
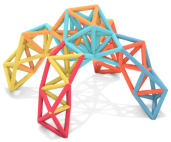
Example				
# of linear actuators	7	18	150	317
# of combinations given three equal size groups	10^2	10^4	3.2 * 10^32	3 * 10^57
# of combinations given three equal size groups w/o connectivity	2178	3* 10^8	3* 10^71	...

FIGURE 1.2: Illustration of the curse of dimensionality (CoD) challenge in the fabrication and design of looped graph robots. As the number of edges increases linearly, the required number of actuable beams and the combination of beam grouping increase geometrically.

1.4 Thesis Overview

The looped graph structure offers immense potential in robotics through its high degrees of freedom and shape-morphing capabilities. These structures uniquely combine scalability, modularity, and lightweight design with the ability to achieve dramatic shape transformations. However, two fundamental challenges - the Curse of Dimensionality (CoD) and the inherent discreteness of these structures - currently limit both their fabrication and computational capabilities.

This thesis presents a systematic progression of solutions to unlock the full potential of LGRs, focusing particularly on truss robots. Our approach evolves from practical hardware solutions to increasingly sophisticated computational methods, each building upon the insights of the previous:

1.4.1 Overcoming CoD in Fabrication through Synergy-Inspired Actuator Grouping

We first address the challenge of CoD in fabrication and controller complexity through a novel actuator grouping mechanism. Drawing inspiration from muscle synergy in biological systems, our approach mirrors how complex movements in nature emerge from coordinated muscle groups rather than individual muscle control. In biological systems, this synergy

mechanism enables sophisticated behaviors like walking or jumping without conscious control of individual muscles, significantly reducing the neural pathway complexity [61, 62]. Details in PneuMesh: Actuator Grouping in Truss Robot.

Translating this biological principle to robotics, we introduce a method to group actuators through customizable connection joints. Each subset of actuators can be controlled simultaneously by a single control module, significantly reducing the redundancy in control mechanisms while maintaining the system's expressiveness. This innovation represents a fundamental shift in truss robot design - rather than scaling control complexity linearly with the number of actuators, our approach enables exponentially more complex structures with only linear increases in control complexity.

The practical implementation of this concept involves carefully designed joint structures that can selectively channel actuation signals across multiple beams. This design not only simplifies the control architecture but also reduces the physical complexity of wiring and power distribution. By allowing a single control signal to coordinate multiple actuators, we achieve sophisticated morphological changes with dramatically reduced hardware overhead. This breakthrough makes the fabrication of complex LGRs more feasible, enabling the construction of truss robots with greater degrees of freedom [63].

However, this solution introduces new challenges in the design space: determining optimal topology for specific tasks, efficiently allocating actuator groupings, and ensuring that grouped actuators can effectively coordinate to achieve desired motions. These challenges motivated our subsequent developments in design tools and optimization methods.

1.4.2 Interactive Editor and Simulator for Forward Design

To support human designers in navigating these complex design spaces, we developed an interactive editor and simulator that bridges the gap between conceptual design and physical implementation. This tool enables real-time visualization and iterative design in a virtual environment, providing immediate feedback on the feasibility and effectiveness of different design choices. Details in PneuMesh: Actuator Grouping in Truss Robot.

Our simulator implements a highly-damped dynamical model that balances computational efficiency with physical accuracy. The model captures essential physical behaviors including length constraints, gravity, ground collision, and friction, while using significant damping to approximate quasi-static behavior. This design choice enables rapid evaluation of designs while maintaining sufficient accuracy for predicting physical performance. The editor interface provides intuitive controls for defining actuator groups, setting control parameters, and visualizing resulting movements.

Through extensive workshop demonstrations, we validated the tool's effectiveness across various applications, from simple shape-morphing structures to complex adaptive furniture designs. The workshops revealed both the potential and limitations of human-driven design. While designers could effectively create basic locomotion patterns and simple transformations, more complex requirements - such as precise shape-matching or multi-objective optimization - proved challenging for human intuition alone. For instance, designing a truss robot to achieve specific target shapes while maintaining structural stability across all configurations often led to suboptimal solutions when relying solely on human judgment.

These insights into the limitations of human design capabilities, particularly when facing tasks requiring precise shape control (like expressive helmets) or complex manipulation sequences, highlighted the need for automated optimization approaches. The gap between human intuition and optimal solutions became particularly apparent in scenarios requiring simultaneous optimization of multiple objectives or precise coordination of numerous actuator groups.

1.4.3 Search for Optimal Solutions in Discrete Space

The need for automated optimization brought the challenges of CoD and discreteness into sharp focus. The expansive search space created by CoD, combined with the discreteness and connectivity constraints of actuator grouping, complicated the use of traditional continuous optimizers. Our solution emerged in the form of a custom genetic algorithm (GA) specifically tailored to the unique constraints and objectives of truss robot design. Details in Muscle Synergy Inspired Evolution of Actuator Network.

Our GA implementation incorporates several key innovations. First, we developed custom operators for mutation and crossover that explicitly respect connectivity constraints while exploring the design space. These operators ensure that all generated solutions maintain physical feasibility - a crucial consideration for practical implementation. Second, we introduced a multi-objective evaluation framework that simultaneously considers factors such as structural stability, motion capability, and control complexity.

To handle the specific requirements of actuator grouping, we implemented a hierarchical optimization structure. This approach first optimizes the high-level topology and grouping assignments before refining individual actuator parameters within these groups. This hierarchical strategy helps manage the combinatorial explosion of possible designs while maintaining solution quality.

The effectiveness of this approach was demonstrated across various test cases, from simple locomotion tasks to complex shape-morphing sequences. However, as we scaled to larger structures and more demanding applications, we began to encounter the fundamental limitations of discrete optimization approaches. The efficiency of genetic algorithms diminished as the search space grew, particularly when dealing with hundreds of actuators or complex sequences of shape transformations.

1.4.4 Reinforcement Learning for Closed-Loop Control

While genetic algorithm optimization enables effective open-loop control sequences, physical implementation revealed limitations when facing environmental disturbances and uncertain conditions. Building on our actuator grouping mechanism, we explored reinforcement learning as a means to enable closed-loop control while maintaining the computational advantages of grouped actuation.

Our approach frames the control problem as a reinforcement learning task where an agent learns policies for dynamically selecting actuator group states based on real-time feedback. The state space incorporates both geometric configuration of the truss and translation-invariant environmental information, with the action space defined by the binary activation patterns of actuator groups. To develop these policies, we implemented a two-stage optimization process: first using genetic algorithms to discover effective grouping strategies, then training control policies through Proximal Policy Optimization (PPO).

This hierarchical approach demonstrated improved adaptability in physical experiments. For instance, in locomotion tasks, the closed-loop controller maintained consistent performance across multiple cycles where open-loop control showed increasing deviation. However, this work also revealed new challenges, particularly in developing efficient state representations and reward functions that generalize across different truss configurations.

The insights gained from implementing closed-loop control, especially regarding the interplay between physical structure and control policy, informed our subsequent investigation into neural network approaches for comprehensive optimization of truss topology and parameters. The complete implementation details and experimental results are presented in Chapter Closed-loop Control of Truss Robot using Reinforcement Learning.

1.4.5 Continuous Representation and Optimization of Topology and Parameters

Despite the effectiveness of our discrete optimizer, scaling up LGRs with increasing complexity revealed fundamental limitations of genetic algorithms. Not only does their efficiency diminish in larger search spaces, but they also cannot generate novel topologies better suited for specific tasks. This limitation becomes particularly acute when dealing with multi-objective scenarios, where unconventional morphologies might outperform human-designed structures. Details in Truss Topology and Parameter Generation with Variational Auto-encoders.

To address these challenges, we developed two complementary approaches using variational auto-encoders (VAE). Our first solution leverages graph attention networks (GAT) to optimize actuator grouping connectivity within fixed topologies. The GAT architecture's natural ability to process graph structures proved particularly effective in capturing both local edge relationships and global structural patterns, enabling shape optimization while maintaining physical connectivity constraints.

We then introduced a more comprehensive solution using long short-term memory (LSTM) networks, coupled with a novel truss grammar that can represent loop-containing structures through sequential tokens. This approach enables simultaneous optimization of both topology and continuous parameters - a significant advance over traditional tree-based grammars that cannot handle the complex loop structures inherent in truss robots. The LSTM-VAE learns to reconstruct complete truss designs while managing the interplay between topological structure and geometric properties.

The resulting architecture enables the generation of entirely novel truss robot designs that meet specified performance objectives, with the continuous latent space allowing efficient exploration through gradient-based optimization. Our experiments demonstrate successful generation of actuator groupings for shape deformation and, more significantly, complete truss designs that can transform between multiple target shapes. To our knowledge, this represents the first work enabling continuous optimization and generation of both truss topology and parameters.

1.5 Key Contributions and Insights

1.5.1 Biological Inspiration for Control Complexity Reduction

Our work demonstrates how biological principles can inform robotics design beyond mere morphological mimicry. The muscle synergy principle, where complex movements emerge from coordinated muscle groups rather than individual control [61, 62], provided both theoretical foundation and practical direction for our actuator grouping mechanism. This biological insight led to a fundamental advance in truss robot design - achieving exponentially more complex structures with only linear increases in control complexity. The success of this approach suggests broader applications of biological control principles in robotics design.

1.5.2 Hierarchical Approach to Design Optimization

Through developing increasingly sophisticated optimization methods, we discovered the effectiveness of a hierarchical approach to truss robot design. Beginning with discrete optimization through genetic algorithms, then progressing to continuous latent space optimization via neural networks, each method revealed different aspects of the design space. The genetic algorithm proved effective for optimizing within fixed topologies but highlighted the need for topology generation. This led to our VAE approaches, which showed that separating

topology generation from parameter optimization allows for more efficient exploration of the design space. This progression suggests that complex robotics design problems may benefit from similar hierarchical decomposition.

1.5.3 Novel Capabilities in Volumetric Shape Change

Our work reveals unique advantages of LGRs in achieving dynamic volumetric shape changes, distinct from other morphing robot approaches. Unlike inflatable structures that permit only preset transformations [64], or limbed robots lacking internal volume, LGRs enable controlled, reversible shape changes with precise timing. Their sparse network structure overcomes the resolution and expansion limitations faced by cubic robots [53], while maintaining structural integrity throughout transformation. This capability opens new possibilities for adaptive robotics in confined spaces and dynamic environments.

1.5.4 Fundamental Trade-offs in Complex Robot Design

Our most significant insight concerns the relationship between control complexity and functional capability. Our experiments reveal that performance improvements diminish beyond a certain number of control groups, suggesting an optimal balance point that varies with task complexity. This finding challenges the common assumption that more degrees of freedom necessarily lead to better performance, instead pointing toward the importance of strategic actuation grouping. This principle could inform the design of other complex robotic systems where control complexity poses significant challenges.

Chapter 2

Background

Recent advances in robotics have driven the development of diverse morphological approaches, each offering unique advantages and limitations. To contextualize the significance of Looped Graph Robots (LGRs), we first examine the broader landscape of robotic systems, followed by an analysis of static truss and tensegrity structures, and conclude with an overview of their dynamic counterparts and optimization approaches.

2.1 Non-LGR Morphological Approaches

2.1.1 Limbed Robots

Limbed robots, which mirror the tree-structured anatomy of biological organisms, represent the most prevalent category of robotic systems. These robots feature appendages branching from a central body, with each limb potentially serving distinct functions. Notable implementations include robots with variable limb lengths [9, 10], morphing wheels [11], and traditional legged systems [5, 6]. The topology of these robots follows a tree structure, where each node represents either the body or a limb, characterized by both discrete attributes (e.g., wheel vs. gripper) and continuous parameters (e.g., length, motion range) [7, 8]. While this architecture excels in specific tasks, it inherently limits volumetric adaptability since limb movements do not alter the robot's core volume.

2.1.2 Continuum Robots

Inspired by organisms lacking rigid skeletons, continuum robots employ soft materials to achieve continuous deformation. These systems offer nearly infinite degrees of freedom in their movement and environmental response [15, 16]. However, they face significant challenges in mechanical stability and strength. Their fabrication, typically relying on specialized 3D printing processes, presents scalability issues. Additionally, the non-linear physical models governing their behavior make accurate simulation particularly challenging, especially for complex geometries or extended temporal sequences.

2.1.3 Cubic Robots

Cubic or voxel-based robots represent another significant category, characterized by their regular, repeating units. Multi-material voxel robots [50–53] achieve shape transformation through selective activation of specific voxels. However, their solid volume nature and non-linear inter-voxel interactions pose challenges for scalability and precision in real-world applications. Magnetic self-reconfigurable cubic robots [54] enable voxelated shape reassembly through magnetic connections but struggle with structural integrity and continuous motion capabilities.

2.2 Static Structures: Foundation for Dynamic Systems

2.2.1 Truss Structures

Traditional truss structures, composed of rigid rods and linkages forming tetrahedral and octahedral configurations, provide foundational insights for dynamic systems. Their key advantages include structural stability, high volume-to-weight ratios, and efficient material utilization [40, 65, 66]. These properties facilitate easy transportation, assembly, and disassembly while maintaining structural integrity. The inherent modularity of truss designs allows for straightforward integration of additional components and functionalities.

2.2.2 Tensegrity Structures

Static tensegrity structures, utilizing a combination of rigid rods and tensioned cables, demonstrate unique mechanical properties. Their design ensures that rigid components never directly connect, instead relying on a network of cables under tension to maintain structural integrity. This architecture results in exceptional volume-to-weight ratios and impact absorption capabilities. While these structures excel in certain applications, their reliance on cables can limit load-bearing capacity and impose specific material requirements. The complexity of maintaining self-balancing tension adds another layer of algorithmic challenges.

2.3 Dynamic Systems: Truss and Tensegrity Robots

2.3.1 Truss Robots

Truss robots represent a dynamic evolution of static truss principles, replacing traditional rigid rods with linear actuators capable of length variation. The fundamental tetrahedral and octahedral topology of these systems ensures that shape changes remain deterministic as actuator lengths modify. This predictability stems from the inherent geometric constraints of the structure, allowing precise control over morphological transformations.

The capabilities of truss robots extend far beyond simple structural support. In the realm of manipulation and locomotion, these systems demonstrate remarkable versatility. Research by Spinos et al. [57] and Liu et al. [42] has shown how truss robots can achieve complex maneuvers through coordinated actuator movements. Their high degree of freedom enables them to execute intricate shape transformations while maintaining structural stability, a crucial feature for advanced robotic applications.

Environmental adaptation represents another significant capability of truss robots. Zagal et al. [67] demonstrated how these systems can navigate through irregular terrain and confined spaces by modifying their morphology in response to environmental challenges. This adaptability proves particularly valuable in exploration and rescue operations, where the ability to change shape can mean the difference between success and failure in accessing restricted areas.

The structural integrity inherited from traditional truss designs enables these robots to manage substantial loads while maintaining their shape-changing capabilities. Usevitch et al. [68] explored this dual capacity, showing how truss robots can distribute loads effectively across their structure while still executing complex movements. However, as Kovacs et al. [69] noted, these systems face a significant challenge in scalability. The control system complexity increases proportionally with structural complexity, as each additional actuator typically requires independent control.

2.3.2 Tensegrity Robots

Tensegrity robots advance the principles of static tensegrity structures by incorporating active elements in both their rigid components and cables. This dynamic adaptation of the tensegrity principle creates systems with unique capabilities and characteristics. These robots maintain the fundamental tensegrity requirement that rigid components never directly connect, instead relying on a network of cables under tension to maintain structural integrity.

The distributed tension network in tensegrity robots provides natural shock absorption capabilities, as demonstrated by Caluwaerts et al. [70] in their work on space exploration applications. This inherent resilience makes tensegrity robots particularly well-suited for navigation across rough terrain or environments where impact resistance is crucial. The structure dissipates forces throughout its network of tensions and compressions, rather than concentrating them at specific points.

The biomimetic properties of tensegrity robots, explored by Lessard et al. [71], reveal interesting parallels with biological systems. These similarities lead to more natural movement patterns and improved adaptability to various environmental conditions. The integration of rigid and flexible elements mirrors biological structures, suggesting potential applications in bio-inspired robotics and medical devices.

Deployability represents another significant advantage of tensegrity robots. Sabelhaus et al. [72] investigated how these structures can efficiently collapse and expand while maintaining their structural integrity. This capability proves particularly valuable in applications where transportation or storage space is limited, such as space exploration missions or portable rescue equipment.

The application spectrum of tensegrity robots continues to expand. Bruce et al. [73] demonstrated their potential in planetary exploration, where the combination of structural resilience and adaptability proves invaluable. In the medical field, Liu et al. [74] explored applications in rehabilitation devices, leveraging the unique combination of rigidity and flexibility inherent in tensegrity structures. These robots can provide controlled support while maintaining compliance, making them suitable for direct human interaction.

The development of tensegrity robots faces distinct challenges in control system design and fabrication. The interdependent nature of tensions and compressions within the structure requires sophisticated control algorithms to maintain stability during shape changes. Additionally, the selection and integration of appropriate materials for both rigid components and tensioned elements remain active areas of research, as these choices significantly impact the robot's performance and capabilities.

2.4 Neural Network Approaches for Robot Design

The optimization of robot designs, particularly those with graph-based structures, has seen significant advancement through neural network approaches. Traditional optimization methods often struggle with the discrete nature of robotic structures, leading researchers to explore techniques that can bridge discrete and continuous representations. Variational Autoencoders (VAEs) have emerged as a powerful tool in this domain, offering the ability to transform discrete structural representations into continuous latent spaces suitable for optimization. This approach has proven particularly valuable in molecular design and robot morphology generation [75].

Graph Neural Networks (GNNs) and Graph Attention Networks (GAT) have demonstrated particular promise in processing graph-structured data, with Kim et al. [76] showing their effectiveness in condensing complex kinematic structures into lower-dimensional representations. These networks excel at capturing both local connectivity patterns and global structural features, making them well-suited for robotic design optimization. The integration

of GAN-based approaches with evolutionary algorithms, as demonstrated by Hu et al. [77], has further expanded the possibilities for generating and optimizing modular robot designs.

2.5 Grammar-Based Robot Representation

The challenge of representing complex robotic structures in a form suitable for computational optimization has led to the development of various grammar-based approaches. RoboGrammar, introduced by Zhao et al. [7], demonstrates how discrete representation combined with grammar-based procedural generation can effectively guide the search process for optimal robot designs. This approach has proven particularly effective for tree-structured robots, where components branch from a central body in a hierarchical manner.

However, the representation of robots with cyclic or looped structures presents unique challenges not addressed by traditional tree-based grammars. While researchers have successfully applied grammar-based approaches to limbed robots [8, 78] and CAD models [79], these methods typically rely on having a unique root node from which the structure can grow. The extension of grammar-based representations to handle looped structures, while maintaining the advantages of sequential token representation, remains an active area of research.

The success of transformer-based approaches in processing these grammar representations, as shown by Gupta et al. [78], suggests promising directions for future development. These methods offer the potential to learn universal controllers for modular robot design spaces, while maintaining the ability to handle complex topological constraints and symmetry requirements.

Chapter 3

PneuMesh: Actuator Grouping in Truss Robot

From transoceanic bridges to large-scale installations, truss structures have been known for their structural stability and shape complexity. In addition to the advantages of static trusses, truss structures have a large degree of freedom to change shape when equipped with rotatable joints and retractable beams. However, it is difficult to design a complex motion and build a control system for large numbers of trusses. In this paper, we present PneuMesh, a novel truss-based shape-changing system that is easy to design and build but still able to achieve a range of tasks. PneuMesh accomplishes this by introducing an air channel connection strategy and reconfigurable constraint design that drastically decreases the number of control units without losing the complexity of shape-changing. We develop a design tool with real-time simulation to assist users in designing the shape and motion of truss-based shape-changing robots and devices. A design session with seven participants demonstrates that PneuMesh empowers users to design and build truss structures with a wide range of shapes and various functional motions.

3.1 Introduction

Among various shape-changing structures such as linear pin arrays [36], inflatable bladders [37, 38], and origami [39], meshes and trusses have been widely adopted for their advantages in modularity, reconfigurability, and high volume-weight ratio. Mesh or truss-based shape-changing structures have been seen as modeling toolkits computationally augmented for both sensing [80] and actuation [81, 82]. Mesh or truss-based structures have been used to construct dynamic [69] and static [83] artifacts and devices through different actuation and morphing techniques as well. Beyond HCI, truss structures are also commonly used in industry and architecture due to their structural stability and modularity [84–86]. A truss typically consists of multiple triangular units constructed from straight beams whose ends are connected at joints. By replacing passive beams with linear actuators that can change length independently, researchers enable a truss structure to locomote, change shape drastically, or manipulate objects [87]. Truss devices do not have specific morphology or motions: they can be assembled into arbitrary shapes, and each part of their bodies can be an actuator. By changing the body shape with a large number of actuators, truss devices can execute various motions, including linear or volumetric scaling, rotation, twisting, and adapting to different environments.

Despite their versatility and adaptiveness, truss devices suffer from an increasingly scaled complexity of the control system. As each beam is controlled independently, the number of control units (e.g. air tubings, wires, motors) is proportionally scaled with regard to the complexity of shape and motion. To reduce the control complexity, researchers use a small number of beams [68], or set most of the beams as passive units [69]. However, these design

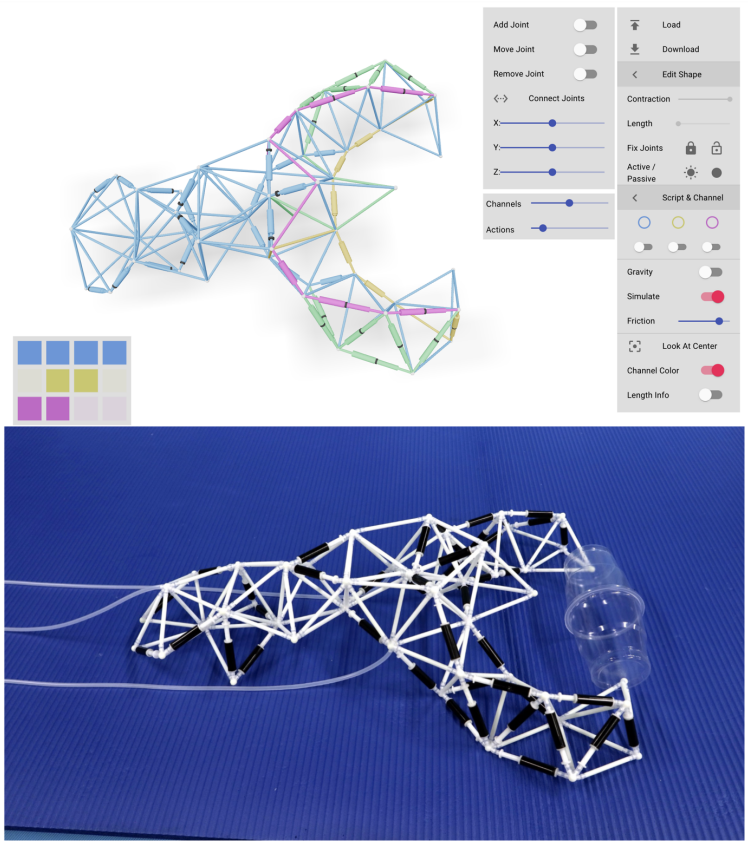


FIGURE 3.1: PneuMesh consists of a computation design platform and a set of novel hardware design improvements for Truss-based morphing and locomotion robots.

strategies often compromise the resolution of the shape (i.e., the number of the truss units), or the complexity of the motion. In this paper, we seek an approach that can give us a certain degree of control over every individual beam but still keep a relatively simple physical setup.

We propose PneuMesh, a pneumatic truss-based shape-changing system that can achieve multiple tasks with a small number of control units (Figure 3.1). PneuMesh is composed of adjustable pneumatic linear actuators (beams) with air channels inside, multi-way joints that direct air to separate channels, and airflow valves controlling each channel. We introduce 1) a partial connection strategy of dividing the entire truss structure into multiple *channel compartments*, with each compartment containing multiple beams that share an airflow valve and connect through algorithmically generated joints, and 2) a replaceable stopper structure that controls the minimum beam length under negative air pressure by restricting contraction. The interconnected beams within one compartment are actuated simultaneously yet have individually reconfigurable contraction ratios. By carefully designing the connection strategy, we allow users to actuate a large number of beams (up to more than 118) with only a few air channel controllers (fewer than or equal to four in this paper), but still create rich motions through the combinations of stopper locations and airflow signals. For example, a six-leg walker can achieve forward motion plus left and right turns with only two air ports and two tubings (Figure 3.2). Trussformer [69] showed a similar design with 12 individually controllable linear actuator units with correspondingly higher wiring and control complexity, cost, and weight.

As the number of beams and time span increase, the search space of the stopper positions, channel connectivity, and control signals scales up proportionally. Thus, we build an online

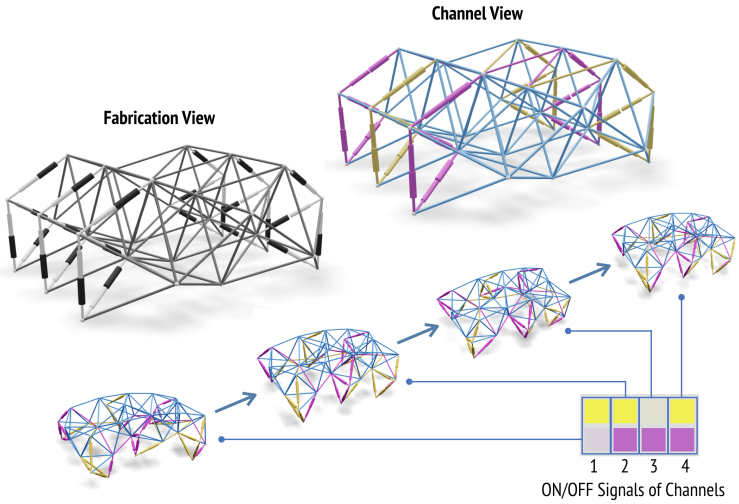


FIGURE 3.2: A six-leg walker with only two channels compartments can move ahead effectively, designed in the PneuMesh platform. In contrast, most typical six-legged robots have more control units (12 linear actuators in TrussFormer [69])

design tool that allows users to edit and simulate the device motions without try-and-error in the physical world.

Our main contributions are as follows:

- A multifunctional truss-based shape-changing system with small numbers of air channels as the control. The system can act on various motions through adjusting beam contraction ratio, channel configuration, and airflow control signals.
- A computational design tool that assists users to edit, simulate and export the shape-changing device.
- Digital designs and physical prototypes that demonstrate the design potential of the system.

3.2 PneuMesh System

We present PneuMesh, a pneumatic truss-based shape-changing system that can achieve multiple tasks with a small number of control units. To change beam lengths on the same channel, we first introduce a passive stopper structure that stops the contraction of the beam at a specific length. The stopper can be manually replaced to adjust the contraction ratio of the beam without any controller modules. Second, to reduce the complexity of the control system of traditional truss devices without losing shape-changing capacity, we introduce the partial connection strategy: each air channel connects a portion of the beams through multi-way joints, where the interconnected beams are actuated simultaneously but independently from beams on other channels. The combination of air channel connection, stopper locations and air flow signals enable PneuMesh to achieve multiple complex motions with a limited number of air channels. Below we will describe the four main design features in more details:

(1) Adjustable Contraction Ratio of Active Beams. Our linear actuators are constructed with a shell, a piston, an adapter, and a stopper structure that can be relocated on the piston (Figure 3.3a). Each stopper can be manually plugged in to one of three grooves on the 3D

printed piston such that the contraction of the piston will be stopped at the stopper location (Figure 3.3b). In short, when being inflated, every beam actuator will extend to the same full length. When deflated, they will contract by a certain percentage based on the constraint position. Based on our design and measurement, the actuator can be set to one of four different contraction ratios under deflation as shown in Figure 3.3c. Figure 3.3d, e show the designed and fabricated single actuator unit, respectively. We demonstrate how the adjustable contraction ratio of each beam can be leveraged to transform a tetrahedron in Figure 3.4a, b.

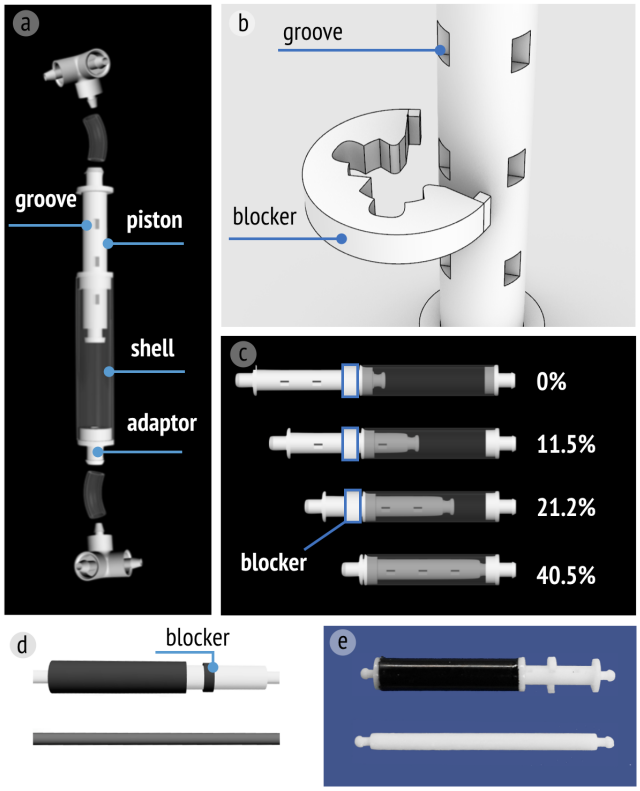


FIGURE 3.3: (a) Each pneumatic linear actuator is composed of a shell, a customized piston, and a tubing adaptor. Flexible rubber tubing connects the adapter to the multiway joints. (b) The stopper structure can be placed onto any of the three grooves on the piston to control the contraction ratio in (c). (d) The physically fabricated structure. (e) The computer model of the actuator unit.

(2) Passive Beams. To accommodate complex geometry while simplifying the fabrication tasks, we allow users to convert actuation (active) beams that have been set with full contraction into geometrically equivalent passive beams, Figure 3.4c.

(3) Reconfigurable Channel Compartment. We design 3D printable multi-way joints each allowing different channels to go through separately.

(4) Control Signals. Since we allow reconfigurable channel compartments, each compartment can be actuated separately. Users can design a set of motions or transformations sequentially by merely changing the control signals which change the state (ON or OFF) of each compartment (Figure 3.5).

Through 3D printable multi-way joints, our system allows multiple channel compartment design. Users can individually actuate each compartment, or combine multiple compartments at the same time. Users can also manually adjust the contraction of every single beam with

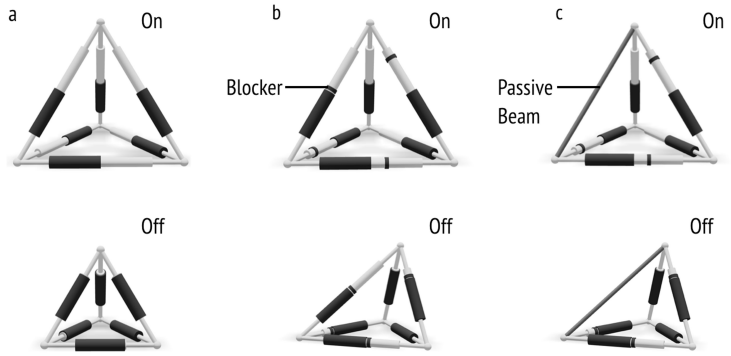


FIGURE 3.4: Examples of different design variables enabled by the platform. (a) A tetrahedron with no blockers (top) uniformly contracts after deflation. (b) A tetrahedron with blockers deforms into an irregular shape. (c) An actuator with full contraction is replaced with a passive beam and keeps the same transformation. This is our strategy of keeping the desired transformation behaviors as designed while decreasing the weight of the structure.

the adjustable actuator design. With the combination of the channel connection and stopper replacement, users can explore various geometries and shape-changing behaviors.

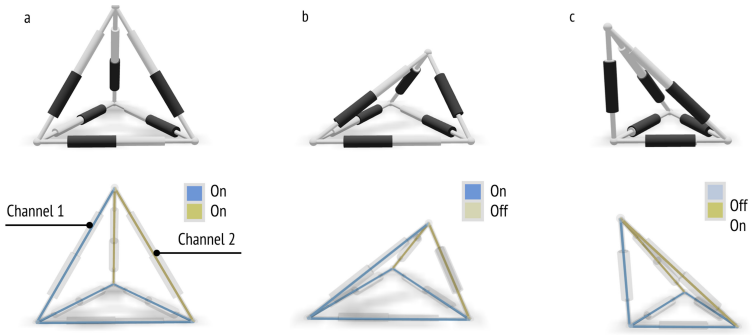


FIGURE 3.5: Partial connection strategy. A tetrahedron is composed of beams from two channel compartments. (a) Both channels are actuated. (b) Channel 2 is deflated. (c) Channel 1 is deflated.

3.3 User Workflow

To assist novice users to design PneuMesh artifacts, we developed an online design tool that allows editing and simulating the PneuMesh setting, as well as exporting 3D printable joint models and assembly instructions. We will go through the user's workflow in the bellow section.

Step 1: Shape design. Users can build arbitrary truss structures by using five interactive tools iteratively (Figure 3.6). The shape design starts with a single tetrahedron under the deflated status. Users can 1) click "Add Beam" to add one actuator with a new joint onto an existing joint, 2) drag the new joint to change its location, 3) select multiple joints connect an isolated joint to other joints with new beams (the joint position would automatically adjust such that beams are of the same minimum length, 4) undo/redo edits with "ctrl-z" and remove a joint and all connected beams with "Remove Joint", and finally 5) fix the positions of joints by selecting joints and clicking "Fix", which is useful when creating stationary applications.

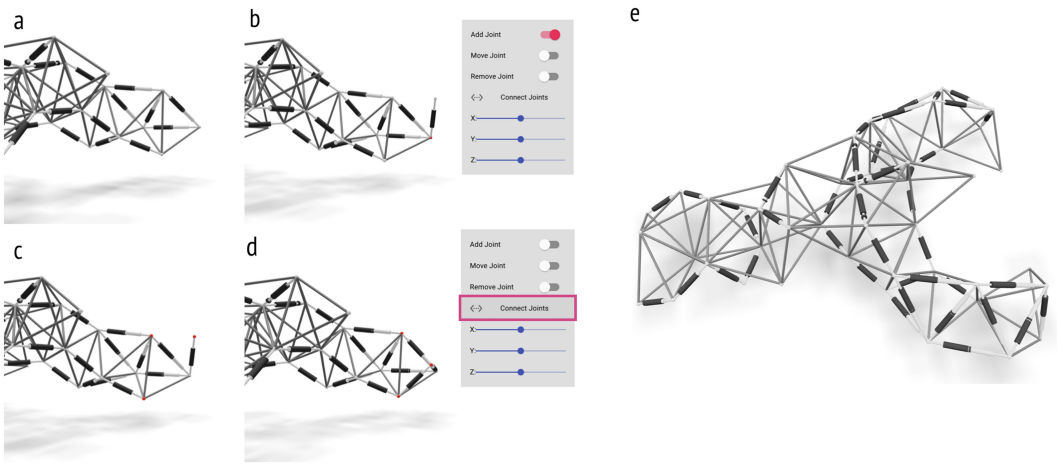


FIGURE 3.6: Shape editing function. (a-b) Choose "Add Joint" and click a joint to append a beam. (c-d) Choose multiple joints and click "Connect Joints" to add beams between each pair of selected joints. Iteratively doing (a-d) and eventually finish a shape design, e.g., a lobster (e).

Step 2: Channel compartment configuration. Users can assign each beam with a channel compartment to connect by selecting the beams and corresponding compartments (Figure 3.7). A channel validation algorithm will reject edits causing discontinuous channels (Figure 3.7 c). Users can change the air port locations of each channel by selecting a joint and the corresponding air channel of the port.

Step 3: Stopper locating and length setting. Users can change minimum beam lengths as well as the PneuMesh shape at fully contracted status by either setting a beam as passive by clicking "Active/Passive", or changing the contraction ratio of an active beam. For passive beams, users can further change the static length of the beam. For active beams, a stopper will be visualized on the corresponding position (Figure 3.8).

Step 4: Airflow signal editing. The signal editor is in the left bottom corner of the screen. Users can first change the number of time frames of the signal, with each time frame lasting for 2.5 seconds (500 simulation time steps). Users can then toggle the inflation/deflation status of each channel at every time frame. The PneuMesh will repeat the control signals according to the script (Figure 3.9).

Step 5: Simulate. If "Simulate" is toggled on, a mass-spring-based numerical simulator will run in the background and animate the PneuMesh structure. The simulator includes the tensile energy of beams, bending energy of joints, the air volume in the channels. The terrain mode includes gravity, ground, and friction. Real-time simulation can be run at any step during the design process, allowing interactive design.

Step 6: Export Fabrication Instructions. The "Export" function uses an air channel generation algorithm to create 3D printable models of multiway joints incorporating our optimized channel geometries (Figure 3.10). The air channels are routed to prevent them from colliding or intersecting. To enable assembly without it “being a puzzle,” a number is shown on each printed joint model as well as in the design tool. Users can assemble the PneuMesh referring to the numbers on the joints and the positions of stoppers.

Step 7: Physical Assembly and Control. Figure 3.11 show the fabricated and assembled structure of a lobster. By manually switching the blockers of the yellow channel, the lobster can be converted between two modes. By receiving the actuation signal for different channels, the lobster can enact 1) grabbing plus slowly moving (Figure 3.11 a) in the first mode or 2) quickly moving in the other mode (Figure 3.11 b).

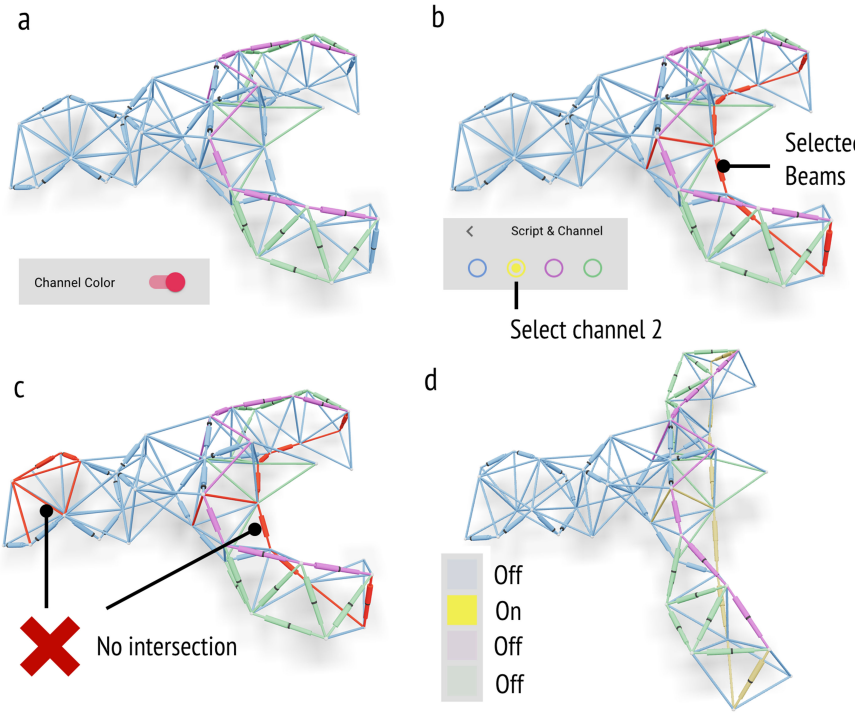


FIGURE 3.7: Channel setting and validation. (a) Check "Channel Color" to visualize the channel compartment assignment. Beams of the same color belong to the same compartment and will be actuated by a single air channel. (b) Choose beams and assign them with corresponding channel compartments by clicking the compartment icons. (c) Beams belonging to the same channel compartment must be connected through neighboring joints. (d) Users can then switch on the control signal of each channel channel compartment and simulate the transformation.

3.4 Demonstration of Design Space

Although the basic building block is simply a length-changing beam, when a large number of such beams assemble and move programmatically under a temporal sequence, the design space increases exponentially. As Figure 3.12 shows, PneuMesh can potentially be used to design structures for locomotion, manipulation, and shape transformation. The shape change behavior can also act as a way to interact with the environment, people, or other objects nearby.

Pill-bug We showed a basic crawling robot designed to move forward with three control channels (Figure 3.13). The bug uses crawling motion and gait to locomote on the ground.

Turtle The turtle shows the capability of switching locomotion behavior by merely changing the control signals Figure 3.14. In this case, neither the contraction ratio of the beams nor the initial geometry (physical assembly) has been changed. Figure 3.14 b also shows additional sensors, such as an IR sensor, that can be attached to the robot to make it interact with the human hand.

Fox We implemented the fox to show the capabilities of switching transformation behaviors by changing both the contraction ratio of selective beams and the control signals. While Figure 3.15 a shows the fox is moving forward, Figure 3.15 b shows that the fox can transform shape by bending down its head to travel through a constrained space.

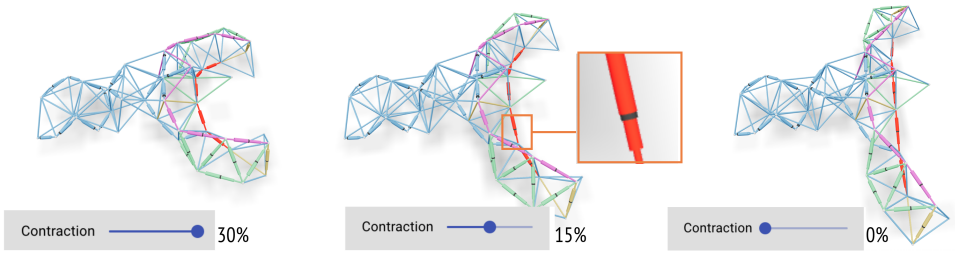


FIGURE 3.8: Beam contraction ratio. By selecting actuators and moving the slider of "Contraction", users can change the contraction ratio. The location of the stopper is shown in black.

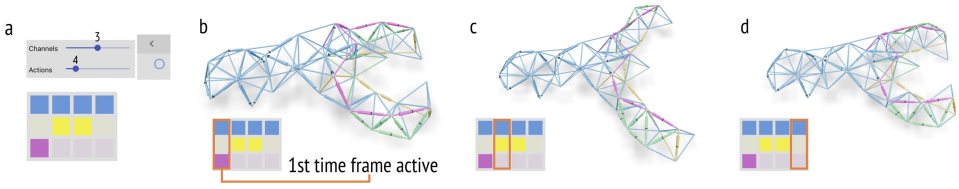


FIGURE 3.9: Airflow control signals. Users can edit the control signal panel to control the inflation/deflation status of each compartment at each time frame. The colors of the squares correspond to the color of channel compartments. Each column indicates a time frame.

3.5 Qualitative Design Session

To validate the usability of PneuMesh, we conducted a qualitative design session with seven participants. In the design session, users were asked to create shape-changing devices with the PneuMesh design tool and share their feedback to reveal existing problems or show future improvements.

Participants. We recruited seven participants(two males, aged 19-26). Four of them were students majoring in industry designs, one in architecture, one in material science, and one in art. Four participants reported to have basic CAD design experience and three were proficient in CAD modeling and design. During each session, each participant created one to four designs except for P7 who didn't finish the design within the given time.

Process. The design session took 2 hours. Participants first went through a tutorial to understand the basic knowledge of the shape-changing truss and the challenges when building large shapes. They were then asked to build three basic shape-changing structures (twisting column, bending strip, expanding sphere) to get familiar with the tool and its functions. Next, users were guided to build several basic shapes and get familiar with the tool. After that, they were asked to build their own design. Following the design session, participants went through a semi-structured interview. Questions included: "What problems did you find?","Could you create the design you had in mind?", "Without PneuMesh, how would you design the device you want?".

Overall, participants responded positively to PneuMesh's design tool: "I appreciate the tool being intuitive and easy to use, but can still create many complicated designs" (P3). "Pretty interesting project that gives the user easy access to the design of a complex interface." (P6). "I really enjoy the pipeline of interactively editing and seeing the result in the real-time, it made my life easier in designing shape-changing interface". (P1). They found the tool "easy and fun to learn" (P2). They thought "the color-coding of different channels and the correspondence between the channel color and the control script block color gives an

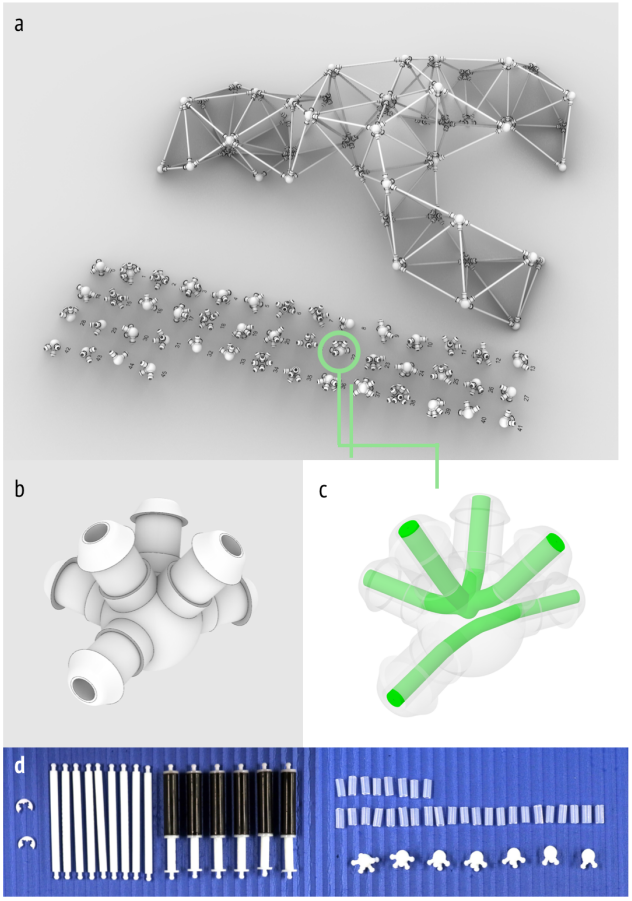


FIGURE 3.10: (a-c) Multi-way joints generated for additive manufacturing. (d) Hardware components are fabricated before the final assembly.

intuitive visual cue that helps me to understand the channel connection strategy and improve my design." (P4)

Learning curve. Participants improved greatly during the design session. In under two hours, P1 was able to finish five designs of increasing shape and motion complexity. (Figure 3.16 P1). Participants started to understand "how to build and allocate actuating beams if I want to build a twisting beam or a bending flower after using the tool for a while" (P2).

Design space. Participants appreciated the large design space of the system and the design tool that lowers the barrier to the PneuMesh system. To build the same design without PneuMesh, P4 would "first use some plasticine and sticks to prototype the shape and then use Arduino with motor and linkage system to implement." But they could imagine that would require "tedious and repetitive design iterations" (P4). P1 mentioned that without the editing tool, they would use a traditional CAD tool to sketch and iterate on the shape, but it would be non-intuitive and error-prone as the structure scales up and the number of units increases.

Feedback and improvements. Participants also pointed out existing problems and gave suggestions for new features in the tool. One main complaint was the lack of common CAD functions, for example, mirroring and trimming (P7, P3, P2). Users mentioned that the process of building a large number of beams was tedious, and suggested a tool that automatically converts mesh into tetrahedrons(P7, P5). They also mentioned that it would be helpful to give a library of basic shapes to save time, such as twisting units, elongation units, and expansion units. Finally, participants proposed some inverse kinematics algorithms and optimization methods that could automatically generate the channel connection by assigning

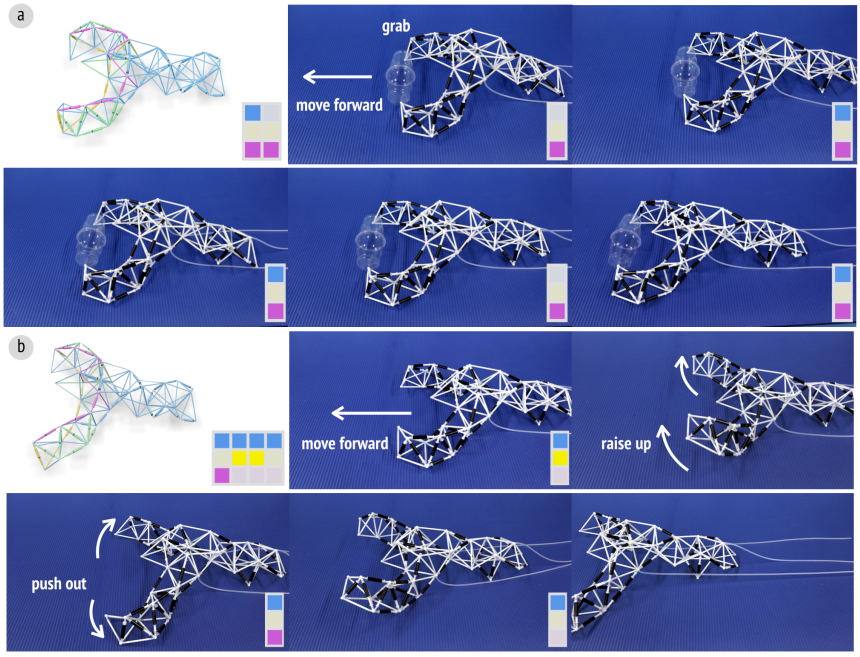


FIGURE 3.11: (a) With the top-left configuration, The lobster grabs two cups and slowly moves forward (12 cycles, 47 seconds). (b) The lobster quickly moves forward (5 cycles, 22 seconds).

a target shape or target motion.

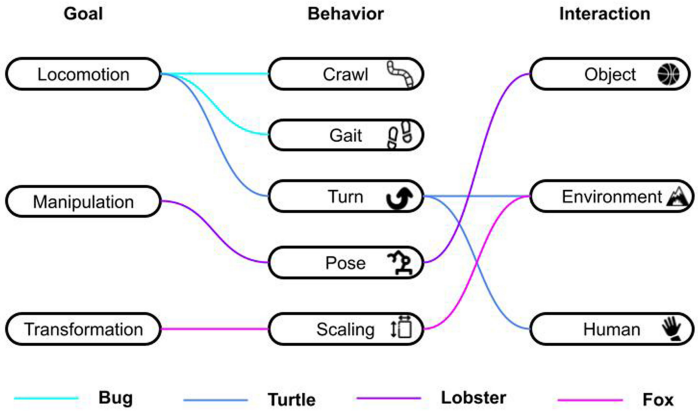


FIGURE 3.12: The design space of PneuMesh for different goals, behaviors, and interactions. Lines with the same color indicate the design space of one of the four shapes (bug, turtle, lobster, and fox).

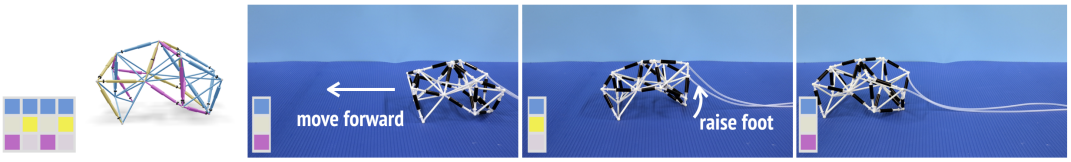


FIGURE 3.13: A crawling pillbug robot moving forward.

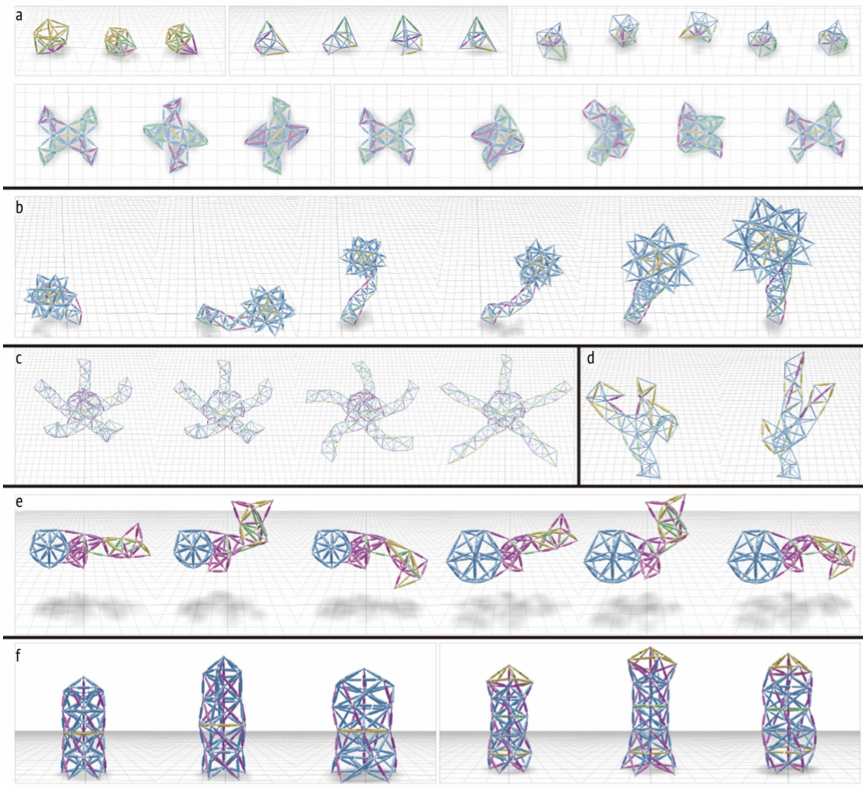


FIGURE 3.16: Designs created by six participants with the PneuMesh editing tool. P1 designed three self-rolling polygons and two dancing robots; P2 and P3 designed two flowers; P4 designed a claw; P5 designed a globefish; P6 designed two adjustable pillars.

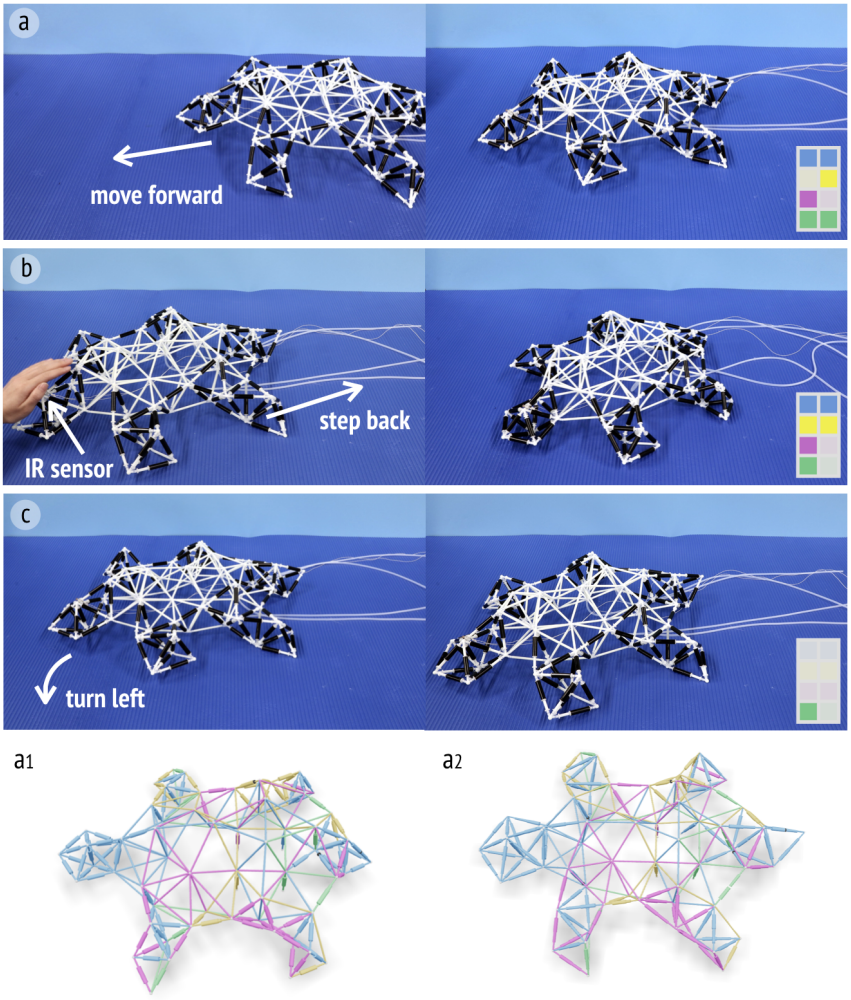


FIGURE 3.14: The turtle switches locomotion modes (a - c) by the changes of control signals.

Participants' designs. Figure 3.16 shows a series of designs by participants. We observed that users like to build things that are related to their life or work. For example, P6 is an architect who built Figure 3.16 P6 which represents an interactive column that changes height and diameter and can twist. It turned out that users who want to build a motion prefer changing the channel connection, while users who want to build specific shapes have a higher chance to change the contraction ratio. The reason might be that channel connection change gives a large change in the global shape and motion but might miss the shape detail, while contraction ratio change enables users to better control the shape details. We also noticed that participants like to use the tool to build bio-inspired objects that have organic shapes and irregular motions, such as flowers (Figure 3.16 P2, P3), globefish (Figure 3.16 P5) and claws (Figure 3.16 P4).

3.6 Characterization

To determine the parameters of the simulation, we measured essential mechanical parameters of the basic units with a constant-pressure pump with positive air pressure at 7.5 ± 0.3 psi and negative pressure at -7.5 ± 0.3 psi.

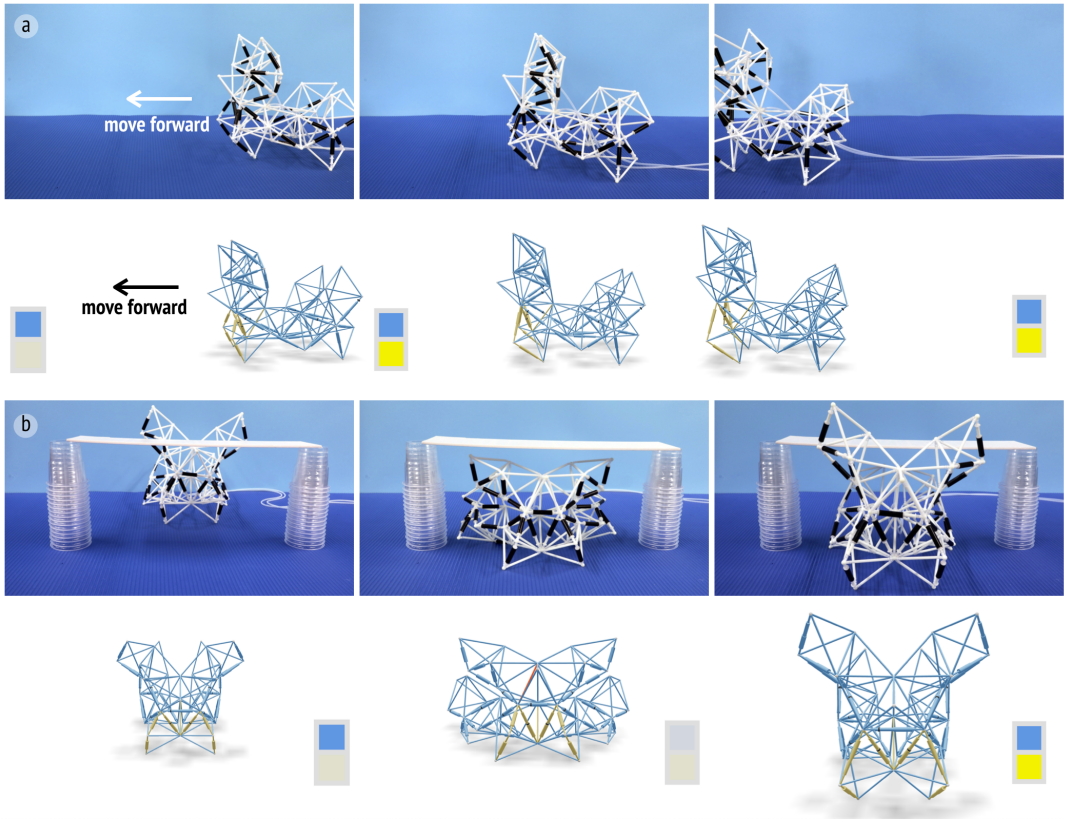


FIGURE 3.15: The fox switches geometry and locomotion by changing control signals. (a) Moving ahead. (b) Lowering the head and traveling through a constrained space.

Theoretically, the actuating force from the air pressure is constantly $f_a = p \cdot s$, where p is the air pressure and s is the inner cross section area of the linear actuator. If the loading force expanding (if the pressure is inner negative) or compressing (if the inner pressure is positive) the actuator is larger or equal to the actuating force, the actuating status of the actuator will be switched. To verify this theory, we first fixed the end of the linear actuators with negative pressure, applied an expanding/tension force with a thrust meter, slowly increasing the force (Figure 3.17 a). We then fixed the body of the linear actuators with positive pressure, applied a compressing force along the axis, and slowly increased the force (Figure 3.17 b). The results aligned with our assumption. We did 18 tests on three linear actuators, including nine with positive pressure and nine with negative pressure. The average maximum tension force for negative pressure is 9.2 ± 0.03 N and maximum compression force for positive pressure is 8.3 ± 0.03N.

The joint is composed of multi-way joints and flexible rubber tubings. The joints and the tubings are firmly connected by the friction and tension of the tubing. Each joint has a maximum and minimum bending angle and maximum shear friction. We first measure the maximum bending angle of the joints (Figure 3.17 c). We did 15 tests on five 3-way joints. We first fixed one side of the beam on the table, and slowly changed the angle of the free beam until it broke. We recorded the average maximum bending angle as 91.2 ± 0.2°. We then measure the maximum shear friction of the tube, over which the joint will detach (Figure 3.17 d). We did nine tests on three tubings and recorded the average maximum shear force is 3.42 ± 0.03 N with a standard deviation of 0.42N. The measured bending force (Figure 3.17 c) is smaller than 0.1 N at a radius of 51 mm, which is ignorable compared to the shear friction and is therefore left out of the simulation.

3.7 Implementation

3.7.1 Characterization

To determine the parameters of the simulation, we measured essential mechanical parameters of the basic units with a constant-pressure pump with positive air pressure at 7.5 ± 0.3 psi and negative pressure at -7.5 ± 0.3 psi.

Theoretically, the actuating force from the air pressure is constantly $f_a = p \cdot s$, where p is the air pressure and s is the inner cross section area of the linear actuator. If the loading force expanding (if the pressure is inner negative) or compressing (if the inner pressure is positive) the actuator is larger or equal to the actuating force, the actuating status of the actuator will be switched. To verify this theory, we first fixed the end of the linear actuators with negative pressure, applied an expanding/tension force with a thrust meter, slowly increasing the force (Figure 3.17 a). We then fixed the body of the linear actuators with positive pressure, applied a compressing force along the axis, and slowly increased the force (Figure 3.17 b). The results aligned with our assumption. We did 18 tests on three linear actuators, including nine with positive pressure and nine with negative pressure. The average maximum tension force for negative pressure is 9.2 ± 0.03 N and maximum compression force for positive pressure is 8.3 ± 0.03N.

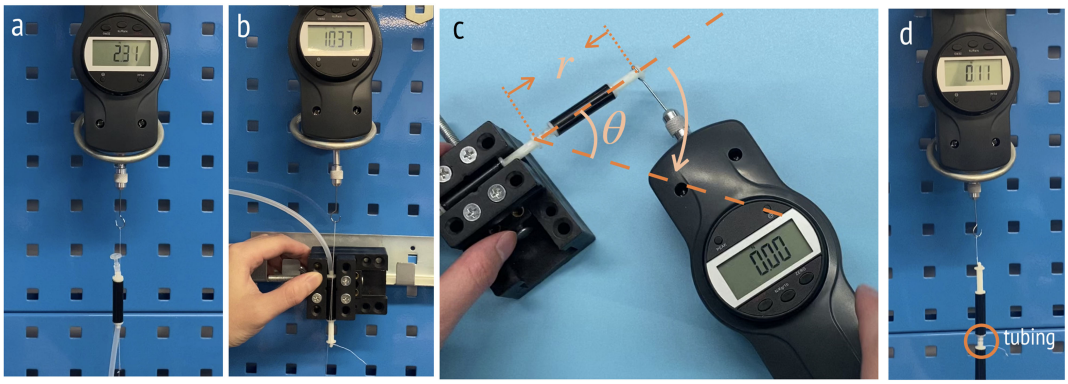


FIGURE 3.17: (a) The threshold tension force of the actuator through a tension test, (b) the threshold compressing force of the actuator through a compression test, (c) measurement of the maximum and minimum bending angles, and the bending force to deform the rubber tubing, and (d) measurement of the shear friction between the rubber tubing and joints.

The joint is composed of multi-way joints and flexible rubber tubings. The joints and the tubings are firmly connected by the friction and tension of the tubing. Each joint has a maximum and minimum bending angle and maximum shear friction. We first measure the maximum bending angle of the joints (Figure 3.17 c). We did 15 tests on five 3-way joints. We first fixed one side of the beam on the table, and slowly changed the angle of the free beam until it broke. We recorded the average maximum bending angle as 91.2 ± 0.2°. We then measure the maximum shear friction of the tube, over which the joint will detach (Figure 3.17 d). We did nine tests on three tubings and recorded the average maximum shear force is 3.42 ± 0.03 N with a standard deviation of 0.42N. The measured bending force (Figure 3.17 c) is smaller than 0.1 N at a radius of 51 mm, which is ignorable compared to the shear friction and is therefore left out of the simulation.

3.7.2 Simulation

We used JavaScript to develop a position-based dynamics model for PneuMesh structures. The PneuMesh structure uses rigid plastic for the beams and joints and flexible rubber tubing to connect the beams and joints. The self-deformation is driven by the pneumatic actuation force along the axial direction of actuators as well as the bending resistance force from the tubing. However, due to the softness of the tubing and the structural stability of triangles, we consider the self-deformation only governed by the axial actuation force. In addition to the axial actuation force, the simulation also includes the ground contact and friction.

To simplify the model, we consider the beams as rigid bodies and joints as point masses. During simulation, we update the target length of each beam based on the actuation speed and update the joint positions following the length constraint. To achieve real-time performance, we allow deviation from the length constraint and use the beam strain energy as a soft penalty. We compute forces applied to joints by minimizing the beam strain energy and updated velocity and displacement of each joint by forward Euler integration.

Target Length. At every time step, each actuator has a target length l_g based on the volume inside the beam and the blocker position. We define $l_g = k_a l_M + (1-k_a)(l_M \sigma_M, l_M - d_b)$, where k_a equals to 1 if the channel is under inflation or 0 otherwise, l_M is the maximum length of actuators, and σ_M is the maximum contraction ratio of actuators. d_b is the beam displacement which equals to $t_a \cdot v_b$, where t_a is the time of the current actuation, v_b is the expanding/contraction speed of beams. For passive beams, σ_M and d_b are constantly 0.

Beam Strain Energy. To keep the length of the beams approximating their target lengths, we introduce the strain energy $E_s = k_s(l-l_g)^2$, where k_s is the weight of the energy. In the experiment, we use a rubbery mat as the ground, which gives enough friction and stabilizes the device. Accordingly, we assume that the ground absorbs all impact by setting the z-component of velocity to 0 for the joints moving against the ground. We use Coulomb's model of friction with the static friction factor as 0.72 and the dynamic friction factor as 0.36.

Durability validation. Tubing detachment and the inversion of actuators' actuation are two factors that cause the most instability. The tubing is the weakest part of the PneuMesh structure, and it is fixed to adapter structures by friction and internal tension. The structure may break down due to the detachment of the tubing rather than other parts. Although the two parts of an actuator are considered to be rigid bodies, the movement between the two parts might be inverted due to the load on the beam (e.g. a tall structure gives a huge load on the beams at the bottom). Specifically, an inflating actuator can be compressed and a deflated actuator can be stretched. We only consider the former case because, under large tension, the tubing will be detached before the latter case happens. At the end of each time step, we validate the force applied on the beam and the tube. If they exceed the threshold value we found through characterization, the tool throws a warning. Similarly, the tool evaluates the bending angle of each beam attached to the joint and throws a warning once any angle exceeds the threshold angle.

Directional surface. We found that adding a directional friction surface 3D-printed with PLA, which only provides friction in one defined direction, on the bottom of the mesh makes the locomotion more efficient. To simulate it, for each joint with a directional surface, we calculate a forward direction by optimizing a vector that minimizes the angle difference between the vector and all the neighboring edges compared to the initial setting.

3.7.3 Joint Generation

A joint is composed of ports connecting beams to a joint sphere, and air tunnel networks inside the sphere. The air tunnel network consists of multiple air tunnels intersecting at

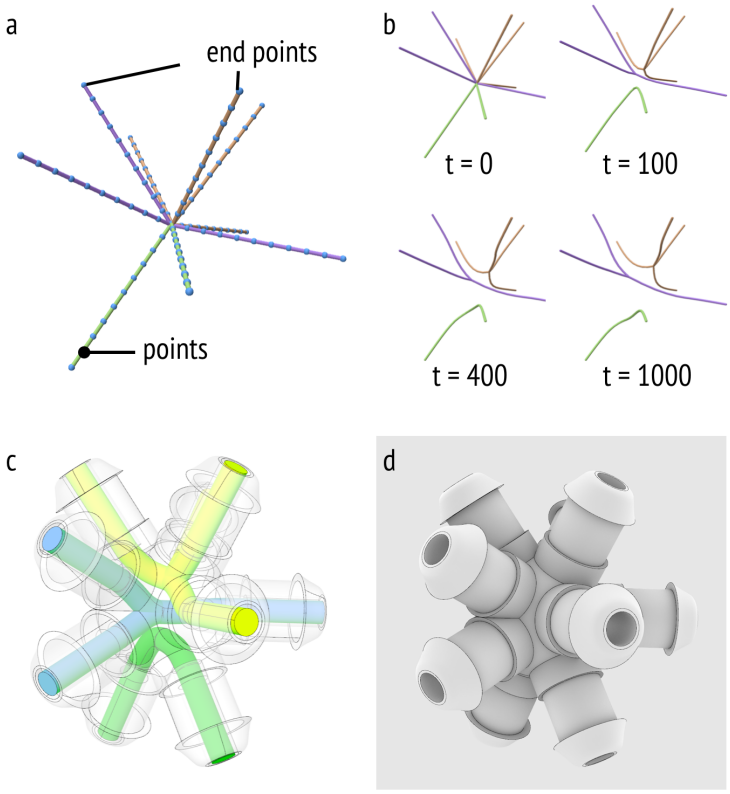


FIGURE 3.18: Joint generation. (a) The tunnels are initialized as straight lines sharing nodes in the middle, with endpoints fixed. (b) The optimization pushes channels apart, where t represents time steps. (c,d) Grasshopper generates the 3D model with separated channels for 3D printing.

one point, where each air channel connects one beam on the joint. Each air tunnel network connects all the beams belonging to the same channel. In other words, if beams on the joint belong to the same channel, only one air tunnel network will be generated, where all the air tunnels will simply intersect in the center and remain straight. Otherwise, multiple air tunnel networks will coexist inside the joint sphere and must be separated from each other. There are a few requirements to ensure joint quality. First, there is a minimum distance between tunnels from different networks to ensure isolation. Second, a larger radius of the channel gives rise to a larger airflow. Third, a shorter and straighter tunnel has a higher printing success rate and larger airflow. We first initialize all the air tunnels straight and connect them to the center point of the sphere. We then discretize each air tunnel into N points. Air tunnels in the same network have a shared point in the center of the sphere. We then optimize the nodes' locations by minimizing a weighted sum of tensile energy and repulsive energy $P = \text{argmin}_P k_e E_e + k_r E_r$, where P is the set of all point positions (Figure 3.18 a,b).

Tensile energy. We set tensile energy to every neighboring pair of points in the same network as $E_e = d^2$, where d is the distance between the points and is initialized as 0.05mm. Minimizing this energy gives relatively short and smooth channels, which have a higher fabrication success rate.

Repulsive energy. We set repulsive energy to every pair of points belonging to different networks as $E_r = -1/d, d < d_m$, $Er = 0, d \geq d_m$, where d_m is a minimum interference distance we set as 0.4mm. This resembles the electrostatic potential energy following Coulomb's law, which pushes networks apart from each other. The cutoff distance reduces the computational cost considering the $O(N^2)$ complexity of the long-range force energy.

We do not update the position of the points on the end, which will be connected to the channel of beams. We used Python with the NumPy library to implement air tunnel optimization then store the result as a JSON file. We use Rhino 6 with Grasshopper to read the JSON file in real-time and generate the joint models for 3D printing (Figure 3.18 c,d).

3.7.4 GUI Implementation Details

We used JavaScript with React-Three-Fiber library (a WebGL-based graphics library) [88] to implement the design tool, simulator, durability validation, and GUI. The whole platform is implemented and tested on a quad-core Macbook Pro. The browser-based design tool has been tested on Google Chrome, Safari 14, and Edge. The code has been released open-sourced (`https://github.com/riceroll/PneuMesh`) and a demo can be accessed through (`https://riceroll.github.io/pneumesh/`).

3.7.5 Fabrication Details

For positive pressure, we used ASLONG AP-370 air compressor with $7.73psi$ maximum pressure, and $0.5 \sim 1.5L/min$; for negative pressure, we used PYP370 vacuum pump rated at $-7.15psi$ and $0.5 \sim 2.5L/min$. We used Arduino UNO and electromagnetic valves to control the pneumatic actuation. We used Z Rapid iSLA600 printer to print the joints, pistons, and adaptors and used silicone tubing with a 50 durometer and 1.5 mm inner diameter. We used polypropylene black pipe with 6 mm inner diameter and used a manual electric grinder machine to cut the pipes.

3.8 Evaluation

3.8.1 Actuation Speed

The speed of actuation is affected by both the structural complexity and power of the pumps we used. In our test, we use the same physical setup detailed in the *Fabrication Details* section. Theoretically, the time from starting actuation to finishing equals $t_a = v/k$, where v is the volume of the channels, including the beams, joints, and connection parts, and k is the flow rate of the pump. In reality, the actuation time varies due to the friction and structure deformation as shape complexity increases. To verify our assumption and quantify the performance, we measured the time of actuation with regard to the number of beams. We used five structures, each with nine active actuators without blockers, as well as a varied number of passive beams. We tested the time of a cycle of full actuation. The time varies from 1.2s to 20.8s. We posit two major reasons for the variation of the actuation speed: 1) Fabrication inconsistency, including printing and assembly qualities. 2) Small distortion in the tubing changes the friction of the linear actuator which requires higher accumulated air pressure to actuate and takes more time.

3.8.2 Simulation Accuracy

We evaluate the accuracy of our simulation by measuring the difference of corresponding joint positions between experiments and simulations after actuation. For simple shapes such as a single tetrahedron, the experiment result aligns well with our simulation. As the complexity of the shape and time of actuation increases, the discrepancy between the experiment and simulation also increases. For example, for the fox in Figure 3.15, the position of the left-front node was displaced for (34.6 ± 1.8, -4.5 ± 0.4) mm after 5 cycles of actuation, while in simulation the locomotion is (36.4, 0.3) mm. The difference is 5.3% (compared to the total locomotion distance). To improve our accuracy in the future, in addition to implementing a

high-order simulator with smaller time steps, we could add acceleration sensors to measure the location and adjust the simulator on the fly.

3.8.3 Fabrication Parameters

We reported the fabrication parameters of each application in Table 3.1. From the result, we can see that we use a very limited number of channels to actuate large numbers of actuators.

Name	T_build	T_design	# beam	# actuator	# joint	# channel	# shape	W
Lobster	2h	1.5h	137	44	46	4	8	179.1g
Turtle	3h	2h	201	107	63	4	6	296.6g
Fox	1.5h	0.5h	126	33	44	2	4	164.5g
Six-feet Robot	/	0.5h	101	24	33	2	2	/
Pill-bug	0.5h	0.5h	57	18	18	3	5	72.6g`

TABLE 3.1: The parameters of each application, including (from left) the building time, the time using the design tool, number of beams, number of linear actuators, number of joints, number of channel compartments, number of static shapes, and weights.

3.9 Limitations

Speed As reported previously, there is a limit in speed due to the flow rate of the pumps and the maximum air pressure the structure can withstand. Currently, the physical prototypes are aimed at tabletop interfaces, such as construction toolkits for design, education, or physically embodied agents. Higher speeds would require increasing the power of the pump and changing the design of the connecting parts to withstand higher air pressure.

Unit Size and Shape Resolution To increase the number of beam units and shape resolution for more complex geometries without sacrificing actuation speed, each unit must be miniaturized. Currently, the unit size is limited by both the beam and joint designs. Each requires an air channel; this channel cannot be too narrow due to both our 3D printing resolution and the required flow rate of air for a given actuation speed. Different printing methods, or different speed goals, could allow further miniaturization.

Asynchronous Actuation of Different Beam Units We have mentioned in our evaluation section how the speed of actuation could vary given the same beam number and geometrical complexity, and this is largely due to the fabrication quality and varied extent of deformation at joints (due to the different transformation design). Indeed, these two factors also cause some asynchronous actuations of different beam units. For relatively complex structures shown in this paper, some beams will take a longer time to actuate than others. For now, our simulation does not take this factor into account, and this will sometimes cause discrepancies between the simulation results and the physical performance. We believe both improving hardware design (e.g., removing the soft tubing at each joint) and adapting more sophisticated fluid mechanics simulations to better capture how air flows within our channels will help tackle this challenge.

Load Carrying Capacity This is a lightweight structure and not optimized for loading. The load-carrying capability is largely limited by the soft rubber tube connections we are using. We may improve the joint design with methods introduced in previous truss-based

robotics such as the flexible joint design in Trussformer[69], so we can have flexible joints with varying deformation angles yet being rigid and load-bearing. Instead of printing communicating tubes as joints, we 3D print a scaffold for structural stability and connecting flexible tubes for air to pass through respectively. However, a more sophisticated (or articulated) joint design may bring higher requirements for the printing resolution and limit the minimum size of our structure.

Optimization and Inverse Design

Since users might wish to design quite complex truss structures with many adjustable beam units, it could take the users a lot of trial-and-error in a forward design process until they achieve a transformation or locomotion behavior they like. In addition, what the users have designed may not be most optimized in the sense that fewer air channels may be able to reach the same locomotion behavior given more time to try other design options. For example, the lobster design took us about 1.5 hours. For the next step, to better facilitate the design process of complex truss structures, we will implement an inverse design process with tailored optimization in the future. Inverse kinematics [89] or evolutionary robotics design [90] are potential references.

Chapter 4

Muscle Synergy Inspired Evolution of Actuator Network

The functionality of robots is often related to their morphology. Adapting morphology to diverse conditions and tasks may enhance their performances in locomotion, transformation, and human-robot interaction. Truss robots with numerous beams offer extensive morphological possibilities and degrees of freedom but present challenges in terms of structural and control complexities. We introduce metatruss, a design and optimization approach inspired by biological muscle synergies, that addresses these challenges through three key contributions: (1) a novel control architecture that groups actuating beams into subnetworks controlled by single modules, (2) a multi-objective optimization pipeline incorporating custom genetic algorithms to navigate design constraints, and (3) a physically realizable implementation demonstrated through multiple robot designs. For instance, a metatruss with 150 actuating beams can be efficiently grouped into eight subnetworks, each managed by its own control module—achieving a ratio of 18.75 actuating beams per control module, a significant improvement over existing models which typically achieve only 1.5 actuating beams per control module. Our method not only simplifies control but also facilitates complex and sequential motions, enhancing the robot's ability to handle diverse tasks. Our demonstrations across multiple robot designs, from quadrupedal locomotion to shape-shifting structures, confirm that metatruss can effectively undertake various robotic and morphing functions, showcasing their potential in complex, real-world applications.

4.1 Introduction

Recent studies show that a robot's morphology is pivotal to its functionality [1]. Biological organisms demonstrate this through their ability to adjust body structure and stiffness to accommodate environmental demands - octopi squeeze through small apertures, while caterpillars use peristaltic shape changes to navigate diverse environments. This adaptability becomes indispensable for robots facing tasks such as fitting into tight spaces, conforming to objects, or inducing specific human emotions and cognitive responses. However, such dynamic plasticity that enables robust operation in natural environments remains a challenge for most artificial systems.

Robotic systems with adaptive morphology demonstrate capabilities for continuous shape changes, including volumetric transformations. Bar-joint robots use interconnected bars and joints as linear or rotational actuators, but are often limited to tree topologies [12–14] or single-bar limbs [91], restricting their shape expressiveness and weight-bearing capacity. Multi-material voxel robots [50–53], composed of regular cubic units (voxels) with different material properties, offer diverse shape changes by activating specific voxels. However, they face challenges in scalability and real-world precision due to their solid volume nature

and the non-linear interactions between connected voxels. Magnetic self-reconfigurable cubic robots [54, 55] allow for voxelated shape reassembly through magnetic connections, but lack structural integrity and continuous motion, limiting their practical applications. Other approaches include robots with variable limb lengths [9, 10], morphing wheels [11], and 2D origami or soft sheet robots [92, 93]. However, these designs often have limited degrees of freedom, control precision, or are constrained to specific morphologies, highlighting the need for more versatile and scalable solutions in adaptive robotic systems that can achieve complex, three-dimensional shape changes while maintaining structural integrity and precise control.

Among the various approaches to address these challenges, variable geometry truss (VGT) systems stand out among robotic designs that offer morphological complexity and adaptability. VGTs, composed of beams and joints that form tetrahedral or octahedral truss structures, achieve diverse transformations such as rotation, twisting, linear, and volumetric scaling through actuator beams. This flexibility enables VGTs to perform standard robotic tasks like locomotion [41, 42], manipulation [43, 44], and target-reaching, as well as specialized activities requiring morphological adaptations [45–48]. Despite their advantages in degrees-of-freedom (DOFs) and versatility, current VGTs face scalability issues due to the complexity of their control systems, which scale exponentially with the number of beams [40]. Therefore, existing VGT with physical implementations are either having a few tetrahedral units [43, 56, 57] or only a few beams are actuable [45], restricting their achievable motions.

Previously, researchers have introduced a novel approach to simplify the control of complex truss robots [63], a strategy for grouping actuator air channel into networks, termed C-networks. By grouping pneumatic actuators with interconnected joints, each subgroups of the actuators in the same C-network can be actuated simultaneous with a single air valve as the controller (Fig. 4.1 b). Despite that the total number of actuators do not change, the number of controllers decreased. With varying combination of the actuation states of the C-networks, the metatruss deforms into different morphology and the number of possible morphologies exponentially scales as the number of the C-networks increases (Fig. 4.1 b). Moreover, under a temporal sequence of actuation signals, the truss transforms into a series of morphologies and performs a sequential motion. Additionally, they enabled each actuator to have different preset contraction ratio through a blocker structure. This approach aims to simplify the system and control complexities inherent in complex truss robots. Although it introduced a design and simulation tool that allows designers to assign the beam connectivity manually, no optimization or automated design pipeline was introduced. As the truss becomes more intricate and tasks grow in complexity, manually navigating C-network assignment becomes tedious and intractable.

In nature, humans and other animals, despite having hundreds of muscles and billions of muscle cells, execute complex movements without consciously controlling each muscle's contraction. Research indicates that animals may use a control strategy known as synergy [61, 62, 100, 101]. This mechanism, also present in humans, reduces neural pathway complexity [100]. With synergy in human motor control, intricate actions like walking or jumping are executed by coordinating a muscle network periodically, eliminating the need for conscious control of every individual muscle. As such, many muscles operate concurrently when engaging in activities that require collaborative muscle actuation. While the topic remains under debate, several researchers argue that this coordinated approach achieves an optimal balance between actuator count and control complexity, significantly reducing the brain's computational burden [102–104]. Inspired by biological muscle synergy, where complex movements are achieved through efficiently coordinated muscle groups rather than individual control, we propose a similar principle for metatruss. We hypothesize that there exists an optimal number of C-networks for a given metatruss, beyond which additional networks yield diminishing increase in performance across various multiple tasks.

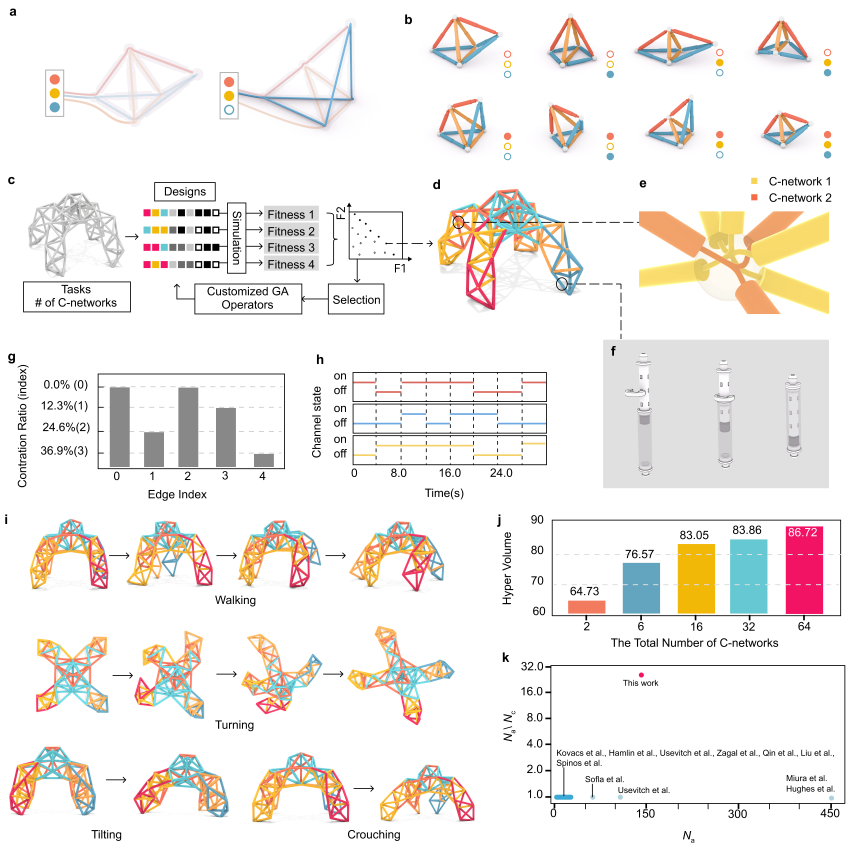


FIGURE 4.1: **Overview of the metatruss system. a,** A metatruss with double tetrahedron topology, consisting of 9 actuators and three inter-connected air channels, C-networks. The actuators belonging to the same C-network expand simultaneously, causing the shape change of metatruss. **b,** Variety of achievable morphologies through the combination of binary on/off states for three C-networks. Here, three C-networks can yield 2^3 possible truss configurations with two tetrahedrons. **c,** Illustration of the multi-objective genetic algorithm with customized operators used for the metatruss design optimization. **d,** The C-network assignment, where actuators of the same color belongs to the same C-network. **e,** The customized joint structure features inner air channels with selective connectivity, enabling unified control for actuators sharing the same air pressure. **f,** Each beam has a discrete contraction level within one of the four percentages $r \in \{0.0, 0.12, 0.24, 0.36\}$, preset manually with a blocker design. **g,** Contraction levels in a metatruss design. **h,** Open-loop binary control signals in a metatruss design. **i,** Six-channel quadruped metatruss optimized to achieve four distinct target motions through four open-loop controls: walking, turning, tilting, and crouching. **j,** Experimental results indicating diminishing performance gains with an increased number of C-network channels. **k,** Comparison with existing research featuring VGT with optimized control [40–43, 45, 46, 56, 57, 60, 94–99], highlighting how our optimizer minimizes the number of control units (N_c) while increasing the number of actuatable beams (N_a).

To validate this hypothesis, we developed a multi-objective optimization pipeline using a tailored hierarchical genetic algorithm (Fig. 4.1c, d). This approach was chosen because the discrete nature of C-network assignments and the topological constraints of the network

make traditional gradient-based optimization methods unsuitable. Our genetic algorithm incorporates custom operators that respect both symmetry and connectivity constraints while exploring the design space. The pipeline simultaneously optimizes three key aspects: the assignment of actuators to C-networks (determining which actuators work together), the preset contraction levels of individual actuators (defining actuators' motion), and the temporal actuation sequences (controlling when each C-network activates). This multi-level optimization allows us to find designs that balance the competing demands of control simplicity and task performance while maintaining physical feasibility.

Our hypothesis is substantiated through an empirical study. We demonstrate that for a complex truss robot with multiple functional objectives—specifically, a quadruped metatruss bot tasked with four distinct functions—a limited number of C-networks can yield competitive performance (Fig. 4.1d, i, j, Supplementary Video 1). This finding is significant when compared to existing literature. Our proposed design method and optimization approach notably enhance the ratio of total actuating beams to the complexity of the control system, offering a unique contribution to the field (Fig. 4.1 k). Furthermore, we showed five different truss topologies with various tasks to show the topology and task diversity of our system. We build a physical prototype of one metatruss to validate the physical feasibility and compare its trajectory with simulation to show the high accuracy of the simulator.

4.2 Results

4.2.1 Overview

Our metatruss, based on the pneumatic shape-changing truss design from PneuMesh [63], is a tetrahedron-based structure composed of pneumatic linear actuators and 3D-printed joints. Each actuator expands to a maximum length under positive pressure P_+ and contracts to one of four preset lengths under negative pressure P_-, adjustable via a reconfigurable blocker structure (Fig. 4.9 c, d). The joints have selective inner air channels that connect incident actuators, grouping them into subsets called C-networks (Fig. 4.1 a, b) Actuators within a C-network share air pressure and operate simultaneously, independent of other networks. Each C-network has a binary state: active (P_+) or inactive (P_-). The metatruss achieves various morphologies through different combinations of C-network states (Fig. 4.1 a, b). Detailed mechanism and fabrication information can be found in the Methods section Mechanism and Fabrication Details.

A metatruss can achieve specific shapes or perform sequential motions through activation signals of its C-networks, enabling tasks that require locomotion or shape changes. While previous work [63] demonstrated hand-designed C-network assignments and actuation signals for given tasks, our work automates this process for more complex truss topologies and diverse tasks. Given a metatruss topology, initial joint positions, tasks, number of C-networks, and C-network symmetry, our optimizer finds the optimal C-network assignment, contraction levels, and actuation signals to maximize the metatruss's multi-objective performance across the specified tasks. For a detailed problem definition, refer to Problem Statement of Metatruss Optimizer. The specific truss topologies and tasks explored in this paper are described in Truss Topologies and Tasks.

To optimize the metatruss design, we developed a highly-damped dynamical simulator that balances computational efficiency with physical accuracy. While existing simulators like Finite Element Methods offer high fidelity but are computationally intensive, and pure kinematic approaches are fast but oversimplified, our approach strikes a middle ground necessary for evolutionary optimization. The simulator approximates quasi-static behavior through significant damping while accurately capturing essential physical interactions including length

constraints, gravity, ground collision, and friction. This design choice enables rapid evaluation of thousands of design iterations while maintaining sufficient accuracy for real-world transfer, as validated through our physical prototypes. The simulator's performance and accuracy are thoroughly examined in Simulator, where we demonstrate comparable accuracy to established physics engines while achieving substantially faster computation times necessary for our genetic optimization pipeline."

Building on our simulator, we developed an optimization framework tailored to the unique challenges of metatruss design. At its core, our approach transforms the complex problem of C-network design into a tractable form by encoding network assignments, contraction levels, and activation signals into a simple yet expressive integer-based representation (Representation). This optimization framework addresses the essential topological constraints of C-networks while enabling efficient evolutionary optimization (Constraints). To handle multiple competing objectives while maintaining design diversity, we enhanced the NSGA-II algorithm with an elite preservation mechanism (Computation Pipeline). A key innovation of our framework lies in its custom genetic operators, carefully designed to explore the design space while respecting physical and topological constraints (Operators).

We validated our framework through three complementary experimental studies. Using a quadruped robot as our primary test case, we demonstrated that metatruss performance reaches diminishing returns beyond a certain number of C-networks, supporting our hypothesis that effective control can be achieved with relatively few networks (Performance v.s. C-network numbers). We then showcased the versatility of our approach by optimizing five distinct metatruss designs for diverse tasks ranging from locomotion to shape-morphing (Task and Topology Diversity). Finally, we bridged the simulation-reality gap by building and testing a physical prototype, confirming both the practical feasibility of our designs and the fidelity of our simulator (Physical Validation). Detailed descriptions of the truss designs and their corresponding tasks can be found in Truss Topologies and Tasks, with numerical results and implementation details in Numerical Results and Implementation Details.

4.2.2 Metatruss Simulator

Various approaches exist for simulating truss robots, each with distinct pros and cons. Finite element methods (FEM), such as Karamba3D, offer detailed analysis of load distribution and micro-deformations but are computationally intensive [105]. Rigid body simulators like Newton Game Dynamics [45], Open Dynamics Engine [96], Mujoco, and Bullet [12–14] provide a balance of speed and accuracy, making them suitable for interactive use and optimization. Some researchers focus on kinematic analysis, assuming quasi-static motion and fixed contact points, which allows for simpler inverse kinematics solutions but limits task diversity and dynamic scenarios [41, 42, 60].

To simulate metatruss, We employ a highly-damped dynamical simulation model. Although rooted in dynamic simulations, this model utilizes significant damping factors and incremental adjustments to the rest lengths of connecting beams, effectively approximating quasi-static behavior. This approach de-emphasizes the dynamic processes in favor of the final, converged states. The model integrates four types of forces: length-constraint forces, gravity, ground collision, and friction. These forces collectively shape the system's physical behavior, and their computations are performed using explicit integration methods. The incorporation of damping ensures that the system approaches a near-equilibrium state at each step, approximating quasi-static behavior while retaining computational efficiency. The model details can be found in Simulator Details.

As genetic algorithm requires lots of evaluation on generations of designs, and the result needs to be transferred to a metatruss robot in real world, the simulator needs to be both efficient and accurate. To evaluate the efficiency, we compared our simulator with Mujoco.

Using motor actuators and equality constraints in Mujoco, the simulation results between Mujoco and our simulation aligned with an average difference of 3.63%. Our simulator achieved computation speeds approximately 345 times faster than Mujoco for 10,000 simulation steps (see Simulator Details).

To validate our simulator's accuracy, we fabricated a physical prototype of a pillbug-like metatruss. We compared the tracked experimental trajectory with the simulated trajectory over eight action sequence cycles, finding a trajectory difference of 4.38% relative to the total displacement of 86.0 cm (see Physical Validation). This close alignment between experimental and simulation results demonstrates the high accuracy of the sim-to-real transfer of our simulator.

4.2.3 Optimization Framework with Tailored Genetic Algorithm

In the field of VGT, researchers have developed various approaches to optimize the control and motion of truss robots, focusing on the actuation signals for individual beams or joints [41, 42, 97]. Some studies have explored co-optimization of control and morphology [7, 78, 106, 107]. However, these methods typically assume independent control of each actuator and often require continuous contraction ratios, which is not suitable for our metatruss.

Implicit encoding methods have been used to represent element attributes and actions in a continuous latent space. Compositional pattern-producing networks (CPPNs) have been particularly effective for voxel robots [50, 108], excelling at generating complex designs with symmetry, repetition, and spatial continuity. These features align well with voxel robots' regular, Euclidean topology. However, truss structures present challenges for CPPNs due to their non-Euclidean topology where the relationship between neighboring elements is not uniform or continuous in space. A small changes in an actuator's C-network index can dramatically affect metatruss performance or invalidate the structure, and the spatial properties CPPNs excel at may not be beneficial.

Other approaches like using transformers [13, 14] or L-systems [109] for tree-topology robots also face limitations when applied to metatruss designs. These methods are well-suited for acyclic, tree-like structures but struggle with the cyclic topology of trusses. Moreover, the number of edges in our metatruss is significantly larger than in typical limbed robots, adding another layer of complexity to the encoding and optimization process.

Given the unique challenges of metatruss optimization—including C-network connectivity constraints, cyclic graph topology, and multi-objective requirements—existing implicit encoding approaches prove inadequate. Instead, we opt for discrete optimization methods, specifically genetic algorithms, which allow direct optimization on explicit encodings. The flexibility of genetic operators enables us to tailor them to our specific constraints. To address the multi-objective nature of our problem, we implement the NSGA-II algorithm, facilitating simultaneous optimization of metatruss designs across multiple performance criteria.

Design Representation

To efficiently and concisely describe a metatruss design compatible with the genetic algorithm, we use a one-dimensional integer array as its representation. We represent every parameter, including the C-network assignment, contraction level, and actuation sequences in integers, and concatenates them into an 1D integer vector. Specifically, this array is structured into three segments, each encapsulating specific design parameters of the metatruss (Fig. 5.2a):

- **C-network Assignment**: The initial segment of the array captures the affiliation of each beam to a specific C-network. Integers within this segment correspond to the indices of C-networks to which each beam is assigned (Fig. 5.2b).

- **Contraction Level**: The second segment represents the preset contraction levels for the beams. Each integer in this section signifies a preset level, which corresponds to a predefined contraction ratio (Fig. 5.2c).
- **Actuation Sequences**: The final section captures the dynamic aspects of the metatruss design – the actuation sequences. Each integer here indicates the on/off states for the air valves that govern each C-network at every time step (Fig. 5.2d). The actuation sequences are flattened into a one-dimensional array and concatenated into the representation.

This encoding represents all the information of a metatruss design as an integer vector that is suitable for the genetic algorithm to optimize. The detailed definition of the representation can be found in Representation Details.

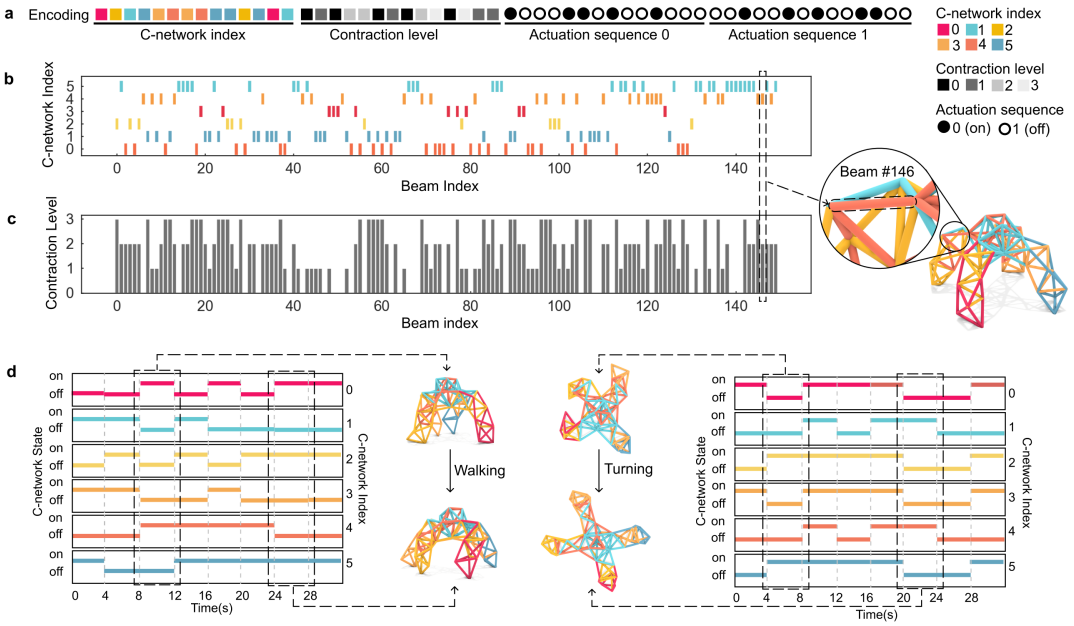


FIGURE 4.2: **Representation and Elements of a metatruss Design. a,** A 1D integer array serving as the design representation. This array encompasses C-network indices, contraction levels, and on/off control signals. **b,** In a quadruped robot example, each beam designated a unique C-network index indicated by color. **c,** Preset contraction ratios r derived from the product of contraction level and a fixed increment, $\Delta = 0.12$. **d,** Task-specific sequences of on/off control signals assigned for actions like walking and turning.

C-network Topology Constraints

As shown in the design of the actuators and joints Mechanism and Fabrication Details, actuators within the same C-network share the same air pressure through a continuous air channel network. It implies that a C-network is an undirected connected graph. Any two actuators within the same C-network needs to be physically connected, and there exists a path of actuators connecting them, where all the actuators in the path are assigned to the same C-network. We define this as the connectivity constraint to ensure the physical validity of the C-networks.

The second is symmetry constraint. Symmetry, a concept often observed in nature, has been recognized for its ability to increase the efficiency of robotic movements [110]. By integrating symmetry into the design and control of robots, the parameter space can potentially

be substantially reduced, thereby enhancing the search process's efficiency. Here, we define a symmetry constraint during the C-network assignment optimization process. This involves defining symmetry at various levels, including the joint, beam, and truss, and introducing the C-network symmetry configuration (Fig. 4.3). The details of symmetry definition can be found in Symmetry Definitions.

The formal definition for the constraints can be found in Constraint Details.

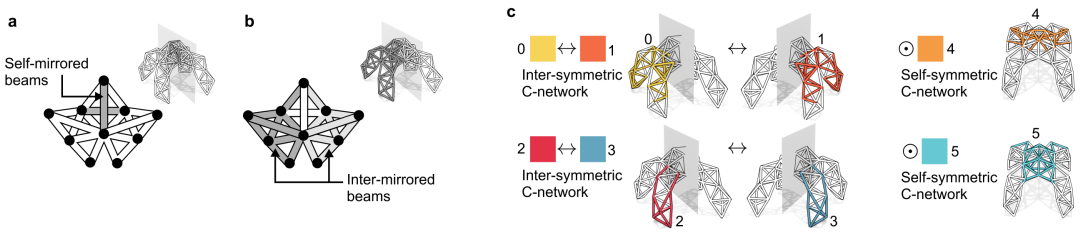


FIGURE 4.3: **Metatruss Symmetry Constraints. a-b,** A symmetric metatruss consisting of self-symmetric (**a**) and inter-symmetric beams (**b**). **c,** Preset C-network configurations designating individual C-network as self-symmetric or specify C-network pairs as inter-symmetric.

Multi-objective Computation Pipeline

To optimize towards multiple objective, a weighted sum of multiple objective function evaluation values is the straightforward way. However, when different objectives have conflicting requirements, the optimization often ends up at a middle ground which leads to solution that is not performing the best at any of the objectives.

One advantage of genetic algorithm is that instead of optimizing a single design, it optimizes a generation of designs, where each design has different advantages in some of the tasks. NSGA-II is an algorithm that encourages the diversity in the designs through a non-dominated sorting and a crowding distance sorting. Specifically, instead of sorting in one dimension with weight combination of performances, NSGA-II computes a rank (R) which shows to what degree a design is not dominated by other designs, where a design dominates another means that a design outperforms another on all objectives. Within the same rank, NSGA-II computes a crowding distance (CD) to evalute how many designs with similar performance exist, and sort them later to encourge design in a sparse performance space to enhance the diversity. The details of NSGA-II can be found in NSGA-II Explanation.

In generation-based optimization, a common challenge arises when a subset of designs consistently outperforms others, leading to their propagation through mutation, crossover, or regeneration operators. As a result, less dominant designs may be prematurely discarded, losing the opportunity to evolve and reach their potential. To address this issue, we introduced an elite pool mechanism (Fig. 4.4 b). This approach maintains a separate elite pool in addition to the traditional evolution pool. At regular intervals, the best-performing designs are moved temporarily to the elite pool, creating space for newer designs to evolve in the main pool. After a few iterations, these elite designs are reintroduced to the evolution pool for further optimization, allowing for a more balanced and diverse exploration of the design space and improved Pareto performance. Details of the elite pool mechanism can be found in Elite Pool Strategy for Optimization Across Generations.

Each optimization starts with the input of the given topology, and initial joint positions, the symmetry and C-network configurations, as well as the objectives (Fig. 4.4, a). A generation of designs are initialized, and simulated, and evaluated through multiple objective

functions. The resulting objective values are sorted through NSGA-II. The top performed designs are kept while the rest is discarded. Every a few generations, the top performed designs will be moved to an elite pool. Once the elite pool is full, all the elite designs are moved back to the active pool and continue the evolution. The details can be found in Optimization Process in One Generation with NSGA-II.

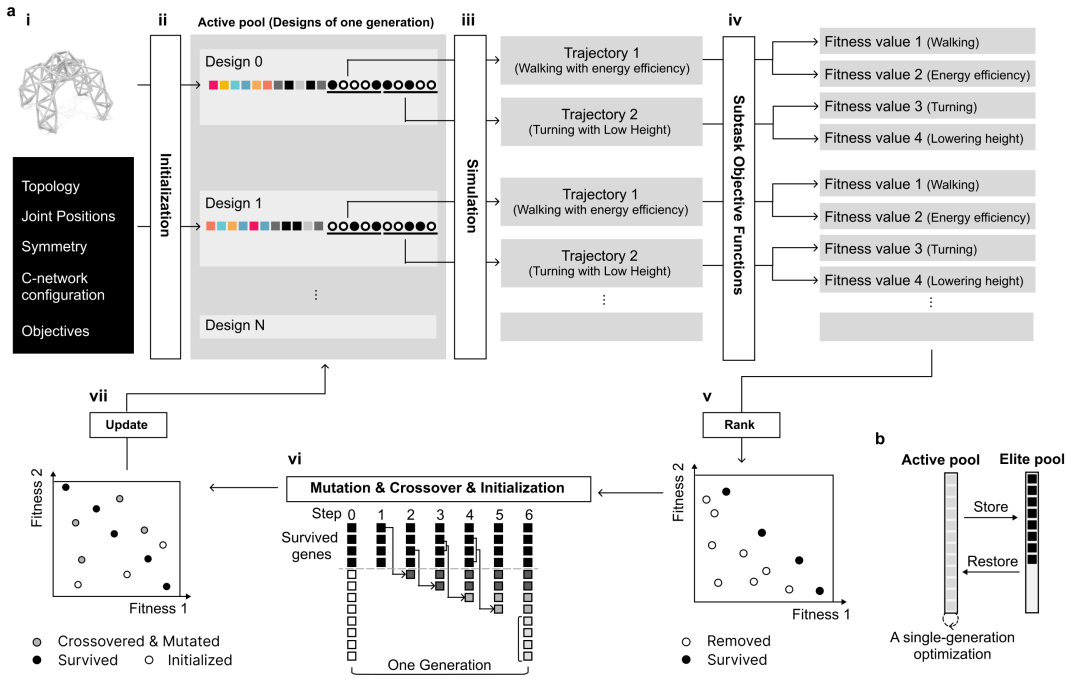


FIGURE 4.4: **The Optimization Pipeline for metatruss Structures: a,** The single-generation optimization involving each training generation to update the active gene pool via NSGA-II-based selection, mutation, crossover, and initialization. (i) The input setting includes the predefined topology, joint positions, symmetry along the $y = 0$ plane, C-network configurations, and objectives. (ii) The initial active gene pool is formulated through a tailored initialization operator. (iii) Different trajectories of the robotic behaviors resulted from simulation. (iv) Within each GA iteration, simulated design trajectories undergo evaluation using specific objective functions. (v)NSGA-II ranks and filters designs, retaining only top performers. (vi) Retained designs generate the next generation via mutation and crossover operators, complemented by additional designs from the initialization operator. (vii) Updating active gene pool. **b**, The cross-generation optimization to replenish the elite pool from the active pool. For each N_G generation, the remaining designs in the active gene pool are moved to the elite pool, indicating the end of the iteration. Once the elite pool is full, its designs are transferred back to the active gene pool.

Tailored Operators

Default genetic algorithm operators do not consider the relationship and constraints between the digits. They randomly generate, change or exchange the digits within the domain. However, a metatruss representation has symmetry and connectivity constraints, which are not explicitly expressed in the integer vector representation. Therefore, to both keep the validity of the two constraints and the randomness of the search, we developed tailored initialization,

mutation, and crossover operators (Fig. 4.5). For initialization, we need to create a random C-network from an empty metatruss. Instead of randomize the digits and check if the constraints are valid, we developed a network growth approach. It first randomly generates new C-network assigned edges on top of existing C-networks, which assures the connectivity constraint. Meanwhile, the C-network index is filtered based on the symmetry constraints. This ensures the symmetry constraint, connectivity constraints of C-networks, ans well as the randomness of the search. The details can be found in Operators

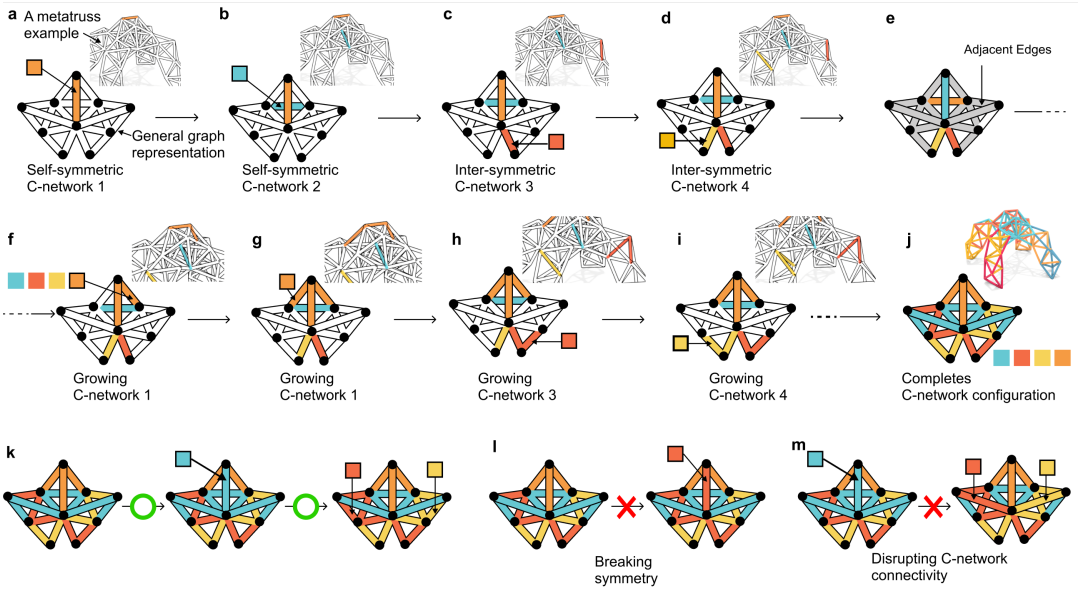


FIGURE 4.5: **Metatruss Operators including C-network Initialization, Mutation and Crossover. a-j,** A four-C-network initialization process. One beam for each of the four C-networks is randomly selected and assigned a valid C-network index, adhering to C-network and beam symmetry constraints (**a-d**). Then beams connected to those already assigned are also assigned valid C-network indices through iterative selection (**e-j**). **k,** The valid mutation steps. **l-m,** The invalid mutations that break symmetry (**l**) or disrupt C-network connectivity (**m**) .

4.2.4 Performance with Varying C-network Channel Numbers

There exists a trade-off between control complexity and task performance. In an extreme scenario, if the number of C-networks equals the number of beams - implying that each actuator can be controlled independently - the metatruss will possess the maximum DOFs for control, therefore having the potential to achieve optimal performance. However, this is likely unnecessary and generates a tremendously large parameter space (e.g., the metatruss in Fig. 4.1 would have had 150 air flow control units if each actuator is individually controlled), leading to over-complicated control setup requirements. On the other hand, a robot with too few independently controllable actuators may struggle to accomplish multiple distinct tasks.

We investigated the relationship between the number of C-networks and robot performance using a quadruped robot model (Fig. 4.1b). The robot was trained to perform four tasks: walking, turning, tilting, and crouching, with C-network numbers ranging from 2 to 64 (Fig. 4.6a).

Our results reveal a non-linear relationship between performance and C-network count. For simpler tasks like tilting and crouching, which require only a single-step action, all five

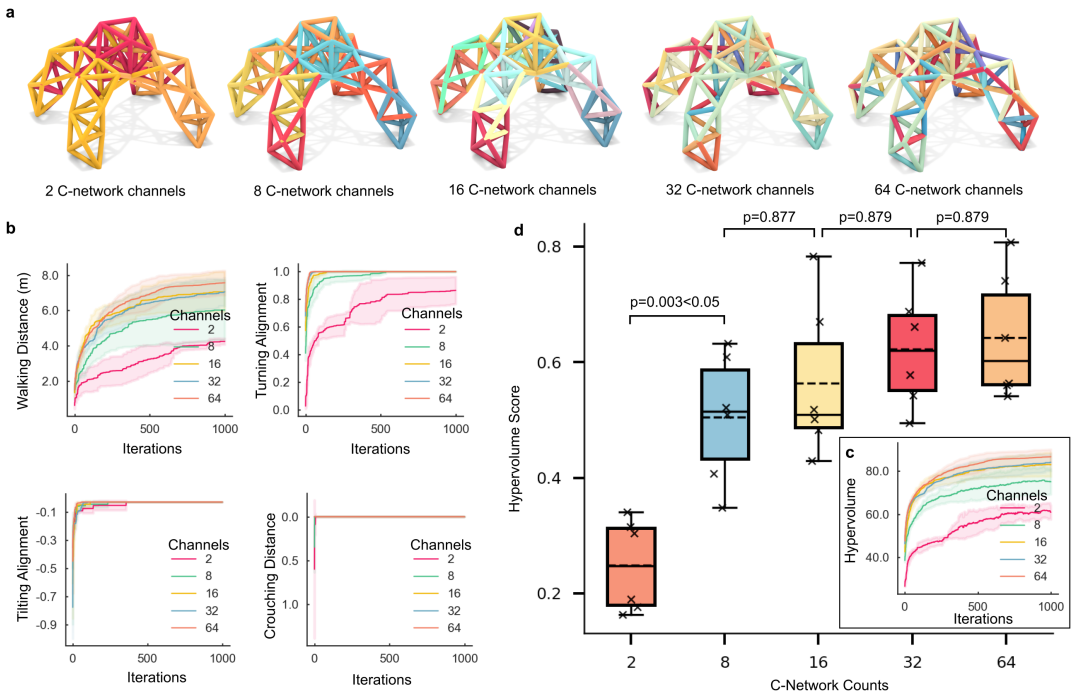


FIGURE 4.6: **Performance Trade-offs with Varying C-network Channel Numbers a,** C-network assignments for the quadruped robot with 2, 8, 16, 32, or 64 C-network channels at the 1000th iteration. **b,** Performance metrics for a quadruped robot for four target objectives including walking distance, turning alignment, tilting alignment, and crouching distance. **c,** ANOVA test results on multi-objective optimization performance for quadruped robots trained with 2, 8, 16, 32, or 64 C-network channels, each group comprising six data points. Significant differences are observed across the five groups, with a notable distinction in the first pair highlighted by Tukey's HSD. **d,** Performance of the hypervolume of the Pareto front across iterations.

robots reach the maximum value. In the task of tilting, the 2-C-network robot takes 378 iterations to converge, while the 8-C-network robot and others require 230 or fewer iterations. As tasks become more complex with longer horizons, a performance gap emerges between the 2-C-network, 8-C-network, and 16-C-network robots. Nevertheless, the performance difference diminishes as the number of C-networks increases. For robots with more than 32 C-networks, there is no significant difference in converged performance in converged performance, with only a 4.8% variation in Pareto front volume among 16, 32, and 64 C-networks (Fig. 4.6b,d).

Statistical analysis using one-way ANOVA confirmed significant differences among the five groups ($p < 0.001$). Post-hoc comparisons using Tukey's HSD showed that performance improved significantly when increasing from 2 to 8 C-networks ($p = 0.003$), but differences became statistically insignificant beyond 16 C-networks ($p > 0.877$) (Fig. 4.6c).

These findings support our hypothesis that an optimized C-network design can achieve competitive performance with a relatively small number of C-networks, balancing task performance with control system complexity. We used the hypervolume of the Pareto front as an overall performance metric to capture the multi-objective nature of the optimization problem.

The details of the implementation and analysis can be found in Numerical Results and Implementation Details.

4.2.5 Diversity in Task and Truss Topology

To demonstrate the versatility of our metatruss method, we explored a variety of truss topologies and functional objectives beyond simple locomotion tasks. Previous work on Variable Geometry Trusses (VGTs) has primarily focused on single-function designs or limited morphological changes due to control complexity [42, 45]. Similarly, other morphing robots have typically been optimized for specific tasks such as locomotion on different terrains or in water [91, 111]. Traditional limbed robots, while versatile in movement, are limited in their ability to perform significant shape changes [76].

Our method, in contrast, enables the design of multi-functional, highly adaptable structures while maintaining a simplified control system. It allows for both complex locomotion and volumetric shape morphing, similar to deformable triangle meshes in computer graphics, albeit constrained by the actuators' contraction ratios. This capability sets our approach apart from both traditional VGTs and limbed robots.

We hypothesized that our approach could optimize trusses for diverse, potentially conflicting objectives within a single design, including both locomotion and shape approximation tasks. To test this, we developed four distinct examples: a quadruped robot, a shape-shifting helmet, a lobster-inspired walking robot, and a tentacle-like actuator (Fig. 4.1b, Fig. 4.7).

The quadruped robot was optimized for four motion objectives: walking, turning, tilting, and crouching (Fig. 4.1e). The goal of this demonstration is to first show that the metatruss is capable of achieve traditional robotic tasks including locomotion and pose changes, and second to show that our computation pipeline can enable the metatruss perform multiple tasks with a single physical configuration. As expected, the robot successfully achieved all four motions, with performance improving as the number of C-networks increased up to a certain threshold (Fig. 4.6b).

The shape-shifting helmet (Fig. 4.7a-c) tested the method's capability in precise volumetric shape morphing, successfully transforming between two distinct target shapes while maintaining structural integrity. This demonstrates a capability not typically achievable with traditional limbed robots, and can potentially unlock new robotic functionalities such as changing morphology to adapt to different environmental constraints and functionality requirements, or precisely approximating different shapes for aesthetic purposs.

The lobster-inspired robot (Fig. 4.7f-i) incorporated energy efficiency alongside locomotion speed, demonstrating improved walking performance and optimization efficiency compared to single-objective optimization. This multi-objective approach goes beyond typical terrain-specific optimizations that only optimizes the speed, but also show the potential of sustainable locomotion robot with a large number of actuators.

The tentacle-like actuator (Fig. 4.7j-l) achieved high precision in reaching multiple 3D target positions, with error rates below 1e-5mm for a 135mm beam length, showcasing the method's potential for precise shape control. This shows the potential for high precision tasks such as manipulation under with truss robots.

These examples demonstrate our method's ability to optimize complex, multi-functional truss designs that can perform both locomotion and significant shape changes, while maintaining a simplified control structure. The results consistently met or exceeded our performance expectations, showcasing the potential of our approach in designing versatile, adaptive robotic systems that bridge the gap between traditional limbed robots and highly deformable structures. The details of the of topologies and tasks can be found in Truss Topologies and Tasks.

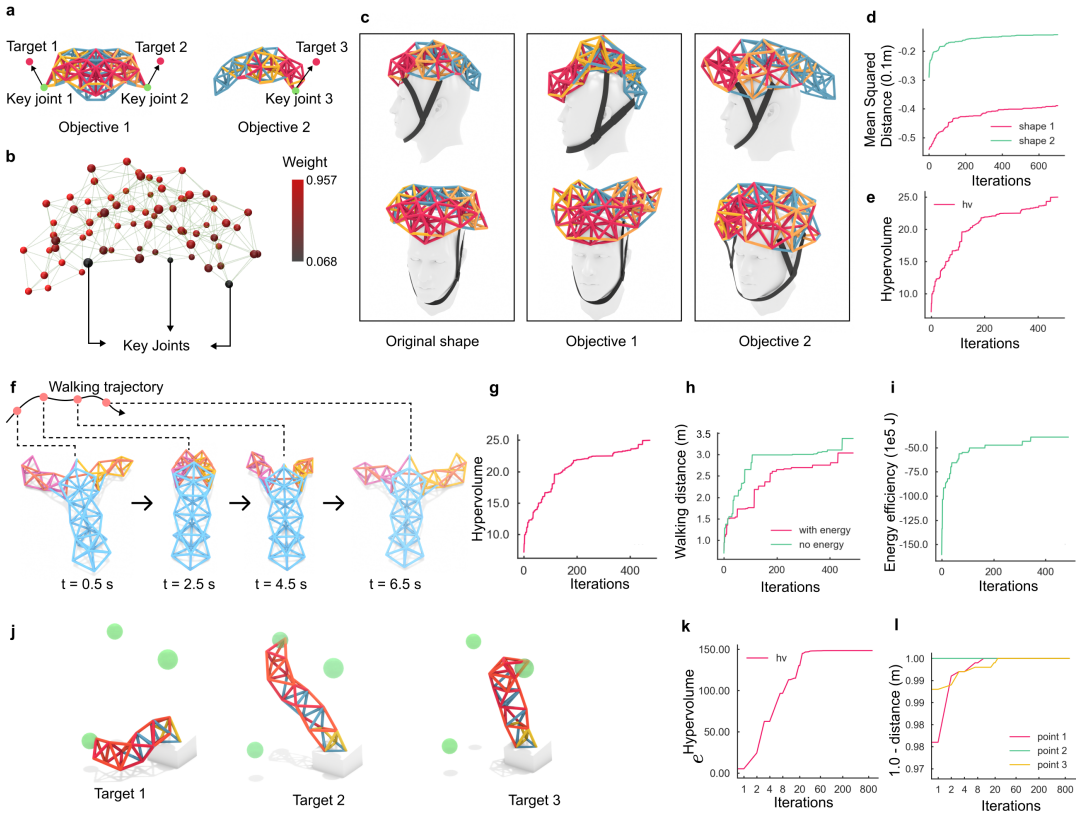


FIGURE 4.7: **Metatruss Design Variations and Performances a,** Three target positions for a morphing helmet in two objectives, with green indicating the key joints, and red being the corresponding target positions. **b,** Joint weights of the morphing helmet are indicated by color shading, with darker shades representing larger weights. **c,** The shape of the helmet before and after morphing. **d-e,** Shape-shifting helmet training metrics: hypervolume and mean square distance between the key joints and the corresponding target positions. **f,** Lobster robot's walking trajectory and morphologies at different times. **g-i,** Lobster robot metrics: hypervolume, energy efficiency, and walking distance. **j,** Tentacle robot reaching three distinct target positions. **k-l,** Tentacle robot metrics: hypervolume and mean squared distance between the tracking joint and three target positions.

4.2.6 Physical Validation

To demonstrate the feasibility of our metatruss design and assess the accuracy of our simulator, we constructed and tested a physical prototype called the "pillbug" (Fig.4.8a, Supplementary Video 5). This prototype was designed to perform a walking task with a lowered body, optimized for both locomotion speed and minimized average maximum height. The design was selected from the Pareto front after 800 iterations of training and fabricated using previously established methods[63].

We compared the experimental performance of the pillbug with our simulation predictions by tracking the trajectories of three key joints over eight action sequence cycles (Fig. 4.8b-d). Our analysis revealed a good overall agreement between the simulated and experimental results, with an average trajectory discrepancy of 3.77 cm and an average static position discrepancy of 1.26 cm. For context, the fully extended length of each beam in the prototype is 17.8 cm.

The prototype demonstrated high consistency in its motion patterns, with an average self trajectory cycle discrepancy of 0.54 cm. This indicates reliable and repeatable performance despite inherent variances in the physical system, such as friction differences between pneumatic components.

Our results validate the effectiveness of our metatruss design approach and highlight the potential for physical implementation of optimized designs. They also reveal areas for future improvement, such as accounting for friction variances in the simulation and exploring closed-loop control methods for enhanced accuracy in long-term operation.

For more detailed information, please refer to the Methods section Physical Prototype and Simulator Accuracy Validation.

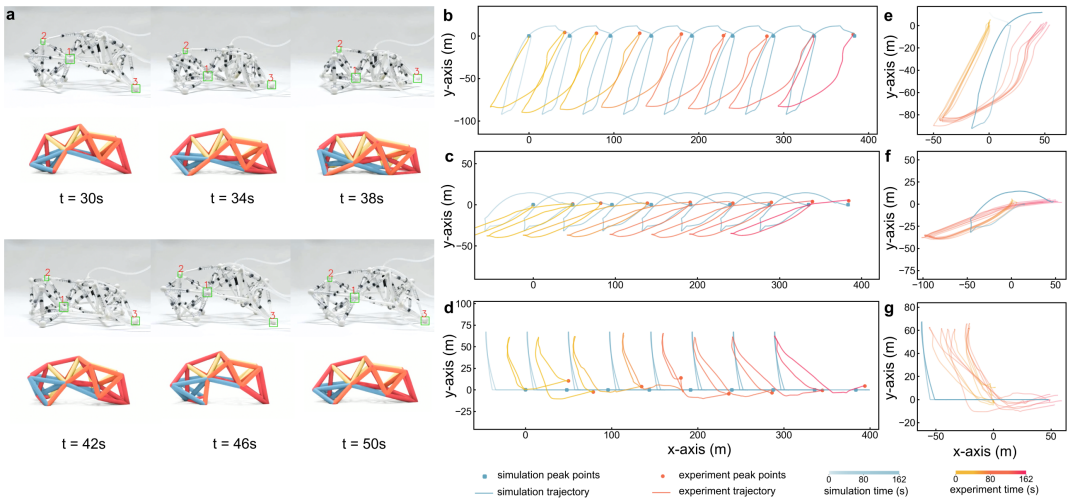


FIGURE 4.8: **Evaluation of the fabricated pillbug and simulation. a,** Comparison of the experimental and simulated pillbug performing walking with lowered body in one action sequence cycle. The three tracked joints are highlighted in green bounding boxes. **b-d,** The experiment and simulated trajectories of the three joints (Top-bottom: joint 1-3). The color gradient indicates the time of the motion. The joint positions at the beginning and ending of each action sequence cycle are highlighted with dots. **e-g,** Trajectories of each action sequence cycle from the experiment and simulated trajectories (Top-bottom: joint 1-3). All the cycles are cropped from **b-d** at the highlighted points, with starting positions aligned at the zero coordinate.

4.3 Discussion and Conclusion

In this paper, we present a metatruss design concept with a simplified control system by introducing the C-network mechanism. We implement a tailored multi-objective genetic algorithm to optimize the design of a metatruss under unique design constraints of the system that are discrete and highly relevant to the topology, such that the metatruss can achieve multiple complex motions with a limited complexity of the control system.

Our metatruss design also shows potential for fully mechanical implementation of the control system. While our current implementation relies on external control signals, the optimized open-loop control sequences could be encoded directly into mechanical logic circuits as a future work. Taking our pillbug robot as an example, its 4-bit binary control sequence with 4 time steps can be implemented using a 2-to-4 multiplexer circuit requiring only 14 logic gates including one clock unit (Fig. 4.13). Using pneumatic logic gates based on

bistable membranes [112, 113], where air pressure differences control the blocking of air tubings, these control circuits could be miniaturized and integrated directly into the metatruss structure. With mechanical logic units potentially scalable to 1cm and metatruss beams expandable to 20cm, a single metatruss robot could carry its own control circuit board. This approach would significantly simplify the control infrastructure, requiring only a constant air pressure source - either tethered through a single tube or completely untethered with an onboard compressed air tank. This demonstrates how our metatruss design could evolve from externally controlled systems to autonomous, mechanically controlled robots.

We see the potential of our metatruss design method going beyond pneumatically-driven truss robots. Other linear actuators suitable as beams in a truss-robot context could be adapted, including linear actuating beams driven by linear motors, shape memory alloy, and soft actuators such as liquid crystal elastomer [114] or muscle-based biohybrid actuators [115–118]. Another potential direction is to explore metatruss systems without a required physical subnetwork connection. Specifically, a subset of beams can be actuated under the same control signal but does not need to form an interconnected subnetwork. For example, liquid crystal elastomers of different colors can be engineered to respond to remote global lighting with specific wavelengths [119]. In such cases, the constraints from connectivity are alleviated, but the benefits of synergy and reduced control system complexity still stand. This may give us more flexibility on the algorithm side.

Our method shows potential for application in other emerging fields of robotic metamaterials and structures. Recent work has introduced strategies to design and construct classes of robotic metamaterials and 4D printed lattice structures that incorporate complex, multifunctional elements in discrete architectures [120, 121]. These approaches create materials capable of outputting multi-DoF motions, sensing capabilities, and programmable thermal and mechanical responses through the manipulation of the properties of local discrete units within 2D or 3D lattices. Our tailored multi-objective genetic algorithm, originally developed for metatruss optimization, could be adapted to optimize these lattice-based structures and potentially automate and speed up the design process, optimizing the arrangement and properties of discrete elements to achieve more complex macro-scale performances or motions while respecting manufacturing and material constraints.

Lastly, we can further explore alternative simulators and optimizers utilizing auto-differentiable simulation [107] or density method [122], which can potentially speed up the optimization efficiency through gradient-based methods and provide more capabilities on the design. For example, the density method may be used to explore an optimal initial topology of the truss structure based on a given three-dimensional mesh, as well as explore the possibility of reconfigurable topology for multi-stage robotic motions or tasks.

4.4 Methods

4.4.1 Mechanism and Fabrication Details

The metatruss design builds upon the PneuMesh framework [63], consisting of two key components: pneumatic linear actuators serving as length-changeable beams and specialized joints that connect them. Like other Variable Geometry Truss (VGT) systems, a metatruss achieves shape changes through the coordinated expansion and contraction of these beams. We present the complete mechanism and fabrication details here for comprehensiveness.

Core Components

The metatruss system is fundamentally based on two designs: a discrete-preset-contraction beam system and a selective-air-channel joint network.

Actuator Design

Each actuator is a syringe-like pneumatic device with an internal air channel and openings at both ends, allowing bidirectional airflow. Under positive pressure, all actuators expand to their maximum length. Under negative pressure, each actuator can contract to one of several preset lengths.

As shown in Figure 4.9d, the piston component contains three positioning holes where a C-shaped ring (blocker) can be installed (Fig.4.9c). When negative pressure is applied, the piston retracts until it meets the blocker. This design enables four discrete contraction ratios: 0% (no blocker), 12%, 24%, and 36%, corresponding to the three blocker positions (Fig.4.9d).

This discrete-ratio design serves two purposes. First, it simplifies the parameter space to discrete integers, making it compatible with combinatorial optimization methods. Second, it allows for preset morphological variations without increasing control complexity.

Joint Design and C-Network Implementation

The joints, which connect multiple actuators, incorporate an innovative selective-air-channel design (Fig. 4.1a). These channels enable specific groups of actuators to share the same air source, ensuring synchronized activation. We refer to these interconnected actuator groups as C-networks (Control networks). Importantly, a single joint can accommodate two independent C-networks without cross-interference, allowing for complex control patterns while maintaining system simplicity.

The joint design process involves several stages of development. First, actuators are assigned to specific C-networks. Then, internal air channels are generated for each C-network passing through the joint. The channel geometry is optimized using Kangaroo, a geometric optimization package, with two primary considerations: maintaining minimum separation between air channels and between channels and outer walls to ensure air-tightness, and minimizing channel curvature to facilitate post-processing removal of support material (Fig.4.9f). The final joint structure is created using boolean difference operations in Rhino, a parametric design tool (Fig.4.9g).

System Constraints and Challenges

While these mechanisms add flexibility to the system, they also introduce several key constraints. All actuators within a C-network must form an interconnected group to satisfy connectivity requirements. The C-networks must maintain prescribed symmetry patterns, and both contraction ratios and control signals are limited to binary/discrete states. These constraints simplify control but create significant challenges for system optimization, particularly as the metatruss scales up in size and complexity. Navigating this discrete solution space within such a structured framework becomes increasingly challenging with scale.

Fabrication Process

The fabrication process combines both 3D-printed and off-the-shelf components. We 3D print the pistons, blockers, end caps, and joints, while using commercial syringes for actuator shells and rubber tubing for pneumatic connections. The assembly process involves using super glue to construct the actuators, with barb structures 3D-printed directly on ports to secure the friction-fit rubber tubing connections.

The completed assemblies are shown in Figure 4.9, with examples of both the quadruped (h) and pillbug (i, j) configurations demonstrating the versatility of the design.

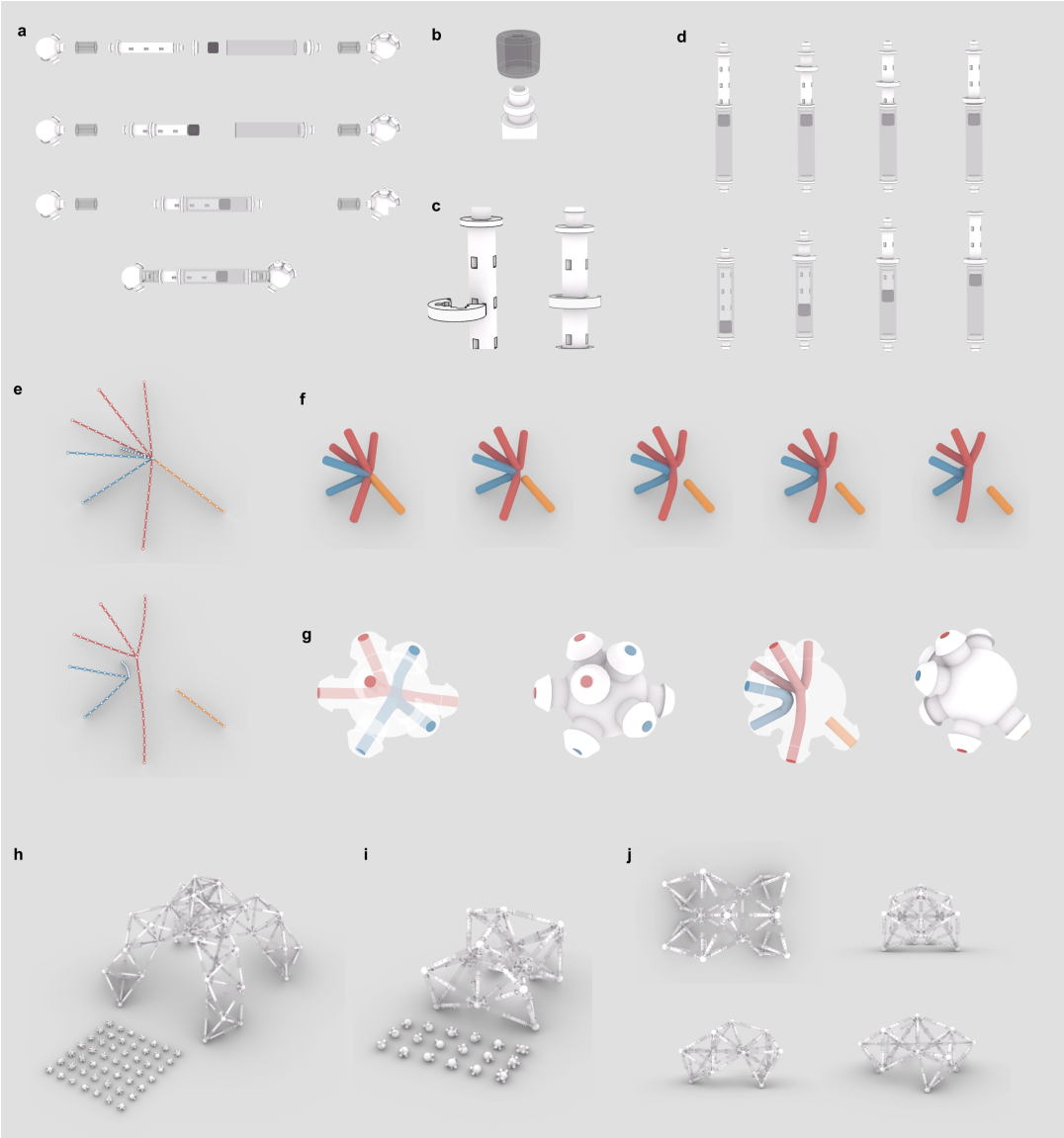


FIGURE 4.9: Metatruss Mechanism

4.4.2 Problem Statement of Metatruss Optimizer

Optimizer. Given the topology of a metatruss, the initial joint positions, and optimization configurations, which includes the N tasks, the lengths of actuation sequences, the number of C-networks and the symmetry of the C-networks, we want to find an optimal C-network assignment, the contraction level of each actuator, and N open-loop control sequences.

N metatruss motion trajectory will be generated based on N actuation sequence. Each task is a combination of one ore more objective functions. And each simulated trajectory will be evaluated by one or more objective functions. N tasks, and N control sequences correspond to M objective functions and M performance scores.

The number of C-networks and the C-network symmetry are given at the beginning of the optimization, where C-network symmetry tells whether a C-network is self-symmetric of inter-symmetric to another C-network C-network Topology Constraints.

We want find the optimal C-network assignment, contraction levels and open-loop control sequences to maximize the performances.

4.4.3 Simulator Details

FIGURE 4.10: Illustration of the simulator. **a,** Spring force vectors on beam joints, applicable when beam length l_{ij} exceeds rest length l^*_{ij}. **b,** Force vectors illustration when the metatruss contacts the ground with horizontal velocity. Besides the spring forces, each joint receives a gravity force. On ground contact, a joint encounters a vertical upward support force and a friction force in the opposite direction of the horizontal velocity component.

In our simulation, we use a dynamic mass-spring model with high damping coefficient (fig. 4.10). The simulator captures the gravity, friction and the axial forces of beams. As the friction plays a crucial role in the metatruss performance, a dynamic simulator is chosen over a semi-static simulation. Due to the asynchronized actuation, we keep the motion of the system to be as static as possible to avoid the unpredictable asynchronized dynamics from actuation. Thus, a high damping coefficient is used to capture the semi-static nature of the motion.Considering the existence of the friction and semi-static We use mass-spring model with an explicit integrator and high damping coefficient. The simulation consists of K steps within a transformation period. During each step, incremental adjustments are made to the rest lengths of beams, transitioning them from their initial length l_{ij_0} to their target lengths l_{ij_g}. Specifically, the rest length $l^*_{ij_k}$ between joints i, j at the kth step of a transformation period is calculated as

$$l^*_{ij_k} = (1 - \frac{k}{K})l_{ij_0} + \frac{k}{K}l_{ij_G}$$

Each step of simulation is further divided into multiple sub-steps, where force, velocity, and position updates are performed.

Let $\vec{p}_i$ be the position vector of point i. Let $\vec{v}_i$ be the velocity vector of point i. Let k be the spring constant, and Δl_{ij} be the difference between the actual length l_{ij} and the rest length l^*_{ij} of the spring between points i and j. Let $\vec{g}$ be the acceleration due to gravity. Let γ be the damping coefficient. Let μ be the friction coefficient.

Spring Force. Spring forces aim to maintain a certain rest length between each connected pair of points

$$\vec{F}^{\text{spring}}_{ij} = -k\Delta l_{ij}\hat{r}_{ij}$$

Where k is the spring constant, Δl_{ij} is the difference between the actual length l_{ij} and the rest length l^*_{ij}, and $\hat{r}_{ij}$ is the unit vector from i to j.

Gravitational Force. Gravitational forces act uniformly downward on all points, defined as

$$\vec{F}^{\text{gravity}}_i = m_i\vec{g}$$

, where $\vec{g}$ represents the acceleration of gravity.

Support Force. When a joint contacts the ground, support forces counteract gravity:

$$\vec{F}^{\text{support}}_i = -\min(0, \vec{F}_i \cdot \hat{z})\hat{z}$$

, where $\hat{z}$ is the unit vector perpendicular to the ground pointing upwards, and $\vec{F}_i$ is the total force on the ith joint.

Frictional Force (when in contact with the ground). Frictional forces are dependent on the horizontal velocity of the points in contact with the ground

$$\vec{F}^{\text{friction}}_i = -\mu\vec{v}_{i,\text{horizontal}}$$

, where μ is the friction coefficient.

Total Force The total force on a point i is the sum of all individual forces:

$$\vec{F}_i = \vec{F}_i^{\text{spring}} + \vec{F}_i^{\text{gravity}} + \vec{F}_i^{\text{support}} + \vec{F}_i^{\text{friction}}$$

Explicit Integration for Velocity and Position

1. Update velocity:
$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{\vec{F}_i}{m_i}\Delta t - \gamma \vec{v}_i(t)\Delta t$$
, where γ is the damping coefficient.

2. Update position:
$$\vec{p}_i(t+\Delta t) = \vec{p}_i(t) + \vec{v}_i(t+\Delta t)\Delta t$$
, where $\vec{p}_i$ is the position vector of the ith joint, $\vec{v}_i$ is the velocity vector the ith joint.

4.4.4 Simulator Comparison

To validate our simulator's accuracy and efficiency, we implemented a comparative model using MuJoCo, a widely-used physics engine known for its accuracy in robotics simulation. In the MuJoCo implementation, we modeled each linear actuator using a prismatic joint with a sliding constraint and integrated spring forces through MuJoCo's built-in force elements. The connections between actuators were modeled using ball joints with three rotational degrees of freedom. We maintained consistent parameters across both simulators, including step size, gravity constants, mass distributions, and friction coefficients. We conducted comprehensive comparisons across multiple test cases, including the four distinct motions of the quadruped robot, two motion patterns of the pillbug, and the tentacle's positioning tasks. The comparison focused on the accuracy of final joint positions, measuring the average displacement between corresponding joints in both simulators. Our analysis revealed an average positional discrepancy of 3.64% relative to the total displacement, indicating strong agreement between the two simulation approaches. Performance analysis showed significant speed advantages in our custom simulator. For a standardized test of 10,000 timesteps, our simulator completed the computation in 0.12 seconds, compared to MuJoCo's 40.80 seconds. This represents a 340-fold improvement in computational efficiency. This substantial speed increase can be attributed to two main factors: our simplified physics model that focuses on essential dynamics relevant to the metatruss system, and our optimized C++ implementation designed specifically for truss-like structures.

4.4.5 Representation Details and Symmetry Definition

Representation Details

We define a metatruss structure as a tuple (V,E), where:

$V = \{v_i\}_{i=0}^{N_V-1}$ is the set of joints (named after vertices), with each joint $v_i \in R^3$ indicating a position in three-dimensional space.

$E = \{e_i\}_{i=0}^{N_E-1}$ is the set of beams (named after edges), with each beam $e_i = \{j_{i0}, j_{i1}\}$ representing an unordered pair of distinct vertices with indices $j_{i0}, j_{i1} \in \{0,1,\ldots,N_V-1\}$.

A metatruss design includes a tuple (C,L), where:

$C = \{c_i\}_{i=0}^{N_E-1}$ is the set of C-network indices, with each C-network index c_i corresponding to the beam e_i and $c_i \in \{0,1,\ldots,N_C-1\}$, where N_C is the total number of C-networks.

$\Lambda = \{\lambda_i\}_{i=0}^{N_E-1}$ denotes the contraction levels for each beam. Each λ_i corresponds to the beam e_i and takes a value from the set $\{0,1,\ldots,N_\Lambda-1\}$, where N_Λ indicates the total number of contraction levels.

Under actuation, each beam can contract by a predefined contraction ratio r, which equals to the product of its contraction level λ and contraction increment Δ. The contraction increment is a constant value within $(0, \frac{1.0}{N_Lambda-1})$. In our work, we set $\Delta = 0.12$ and $N_\Lambda = 4$. Hence, the possible contraction ratios are in the set $\{0\%, 12\%, 24\%, 36\%\}$. This means that, depending on its contraction level, each beam can contract by one of these ratios. In practice, the value of Δ and λ are often determined by specific hardware constraints.

We take the C-network indices C, contraction levels Λ, and actuation sequences $A_0, A_1, ..A_{N_A}$, flatten and concatenate them into a 1D integer vector as the representation of a truss configuration D. This encoding represents all the information of a metatruss design as an integer vector that is suitable for the genetic algorithm to optimize.

We define $A_j = \{a_i\}_{i=0}^{N_{A\,j}-1}$ as the actuation sequence for the metatruss, where $N_{A\,j}$ is the number of actions in the jth action sequence. During simulation, metatruss will take in actions in sequence and actuate the truss. Each action $a_i \in \{0,1\}^{N_C}$ controls the on or off state of each C-network (Fig. 5.2b,c). During the actuation of each action, beams in each C-network expand or contract based on the action. τ is the time duration of each action. The total actuation time T equals to $\tau \cdot N_{A\,j}$ for the jth action sequence A_j.

A metatruss goes through a set of action sequences $\{A_0, A_1, \ldots, A_{N_A-1}$, where each action sequence A_i is expected to achieve one task S_i. A task S_i consists of a set of subtasks. For example, $S_i = \{s_l, s_t\}$ represents the metatruss is expected to perform both locomotion and turning under the same action sequence for the ith task (Fig. 5.2d).

Symmetry Definitions

We define the self-symmetry and inter-symmetry of joints and beams, as well as the symmetry definition of a metatruss (Fig. 4.10). On top of that, we define the self-symmetry and inter-symmetry of c-networks as the C-network symmetry configuration used for c-network mutation and crossover.

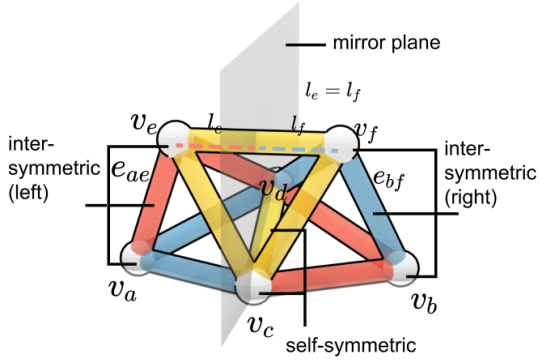


FIGURE 4.11: **Truss symmetry definition**. v_e and v_f are inter-symmetric joints because they are mirrored against the mirror plane. Similarly, v_b and v_a are inter-symmetric, while v_c and v_d are self-symmetric as they are at the mirror plane. With both joints inter-symmetric to another beam's joints, e_{ae} is inter-symmetric to e_{bf}. Blue C-network is inter-symmetric to red C-network, while yellow C-network is self-symmetric.

Joint Symmetry

Joint Inter-symmetry: Two joints are inter-symmetric if they are mirrored to each other against the mirror plane, such that the segment connecting them is perpendicular to the plane and the distance from the two joints to the mirror plane are equal. Inter-symmetry is denoted by $\leftrightarrow$. For example, $v_a \leftrightarrow v_b$ represents for inter-symmetric joints v_a and v_b.

Joint Self-symmetry: A joint is considered self-symmetric if it is located on the mirror plane. Symbolically, we use $\odot$ to represent self-symmetry. For example, $\odot v_c$ represents that joint v_c is self-symmetric.

Beam Symmetry. The symmetry extends to beams as well:

Beam Self-symmetry: If both joints of a beam are self-symmetric, the beam itself is deemed self-symmetric (Fig. 4.3a). *Beam Inter-symmetry*: A pair of beams is inter-symmetric if the corresponding pairs of joints are inter-symmetric or if one pair is inter-symmetric and the other pair is self-symmetric (Fig. 4.3b).

We categorize beams into sets E_s for self-symmetric beams and E_r for inter-symmetric beams, satisfying $E_s + E_r = E$.

Truss Symmetry. A truss is defined as symmetric if there exists a mirror plane such that every beam is either self-symmetric or inter-symmetric. While our paper exclusively utilizes symmetric trusses, our algorithm has the flexibility to be applied to asymmetric trusses as well.

C-network Symmetry. One of our goals of optimization is to have the C-networks be either self-symmetric or inter-symmetric, which allows symmetric motions. We represent the ith C-network by E_i, denoting the set of beams sharing the same C-network index i_C, where $i_C \in I_C$. *C-network Self-symmetry*: A C-network E_i is considered self-symmetric if $\forall e_m \in E_i$, $\exists e_n \in E_i$ such that $e_m \leftrightarrow e_n$ or $\odot e_m$. *C-network Inter-symmetry*: Conversely, a pair of C-networks E_i and E_j is deemed inter-symmetric if, $\forall e_m \in E_i$, $\exists e_n \in E_j$ such that $e_m \leftrightarrow e_n$. The sum of all C-networks is represented by $\sum_{i=0}^{N_C-1} E_i = E$.

For each optimization task, we predefined the C-network symmetry configuration (Fig. 4.3c), meaning that we specify the total number of C-networks as well as the count of inter-symmetric and self-symmetric C-networks. Within the C-networks identified as self-symmetric or inter-symmetric (Fig. 4.3c), we further describe relationships such as $i_{C0} \leftrightarrow i_{C1}$, which signifies the C-network with index i_{C0} is inter-symmetric with the C-network indexed i_{C1}, or $\odot i_{C2}$, meaning the C-network with index i_{C2} is self-symmetric.

4.4.6 Constraint Details

Symmetry Constraints

We define two types of symmetry among joints, beams, and the truss: self-symmetry and inter-symmetry, with respect to a given mirror plane. For joints and beams, inter-symmetry occurs when two joints or beams mirror each other against the plane, while self-symmetry occurs when they mirror themselves (Fig. 4.3a, b). At the truss level, a truss is defined as symmetric if a mirror plane exists such that every beam is either self-symmetric or inter-symmetric. While our paper exclusively utilizes symmetric trusses, our algorithm has the flexibility to be applied to asymmetric trusses as well (see **Methods** for detailed definitions).

A C-network is also assigned with symmetry in the metatruss (see **Methods**). A C-network is considered self-symmetric if it mirrors itself against the middle plane. If two C-networks mirror each other against the plane, they are considered inter-symmetric.

For each optimization task, we predefine the C-network symmetry configuration (Fig. 4.3c), meaning that we specify the total number of C-networks as well as the count of inter-symmetric and self-symmetric C-networks. Empirically, we set 33% of the C-networks as self-symmetric. For example, in a quadruped robot with six C-networks (Fig. 4.3c), four C-networks are inter-symmetric, and two C-networks are self-symmetric.

Connectivity Constraint

In our optimization task, we aim to ensure that all C-networks are connected and that they adhere to either self-symmetry or inter-symmetry in accordance with the predefined C-network symmetry configuration. The connectivity constraint ensures that the C-network forms a continuous path, enabling the efficient transmission of control signals and powers, thereby preserving the integrity of the robot's movements and actions. We have three connectivity constraints in the system:

- **Adjacent Beams**: Two beams are termed adjacent if they share a common joint. This connection represents a physical connection between the two beams at that specific joint.
- **Connected Beams**: Building on adjacency, we introduce the concept of connected beams. Two beams are classified as connected if there is a sequence of adjacent beams starting from the first beam and ending at the second. In other words, one can traverse from one beam to the other through this series of adjacent connections.
- **Connected C-network**: Extending the idea to a whole C-network, a C-network is determined to be connected if every pair of beams within that C-network is connected. This definition guarantees that there's a navigable path between any two beams in the C-network, either directly through adjacency or indirectly via a sequence of adjacent beams.

4.4.7 Optimization Process in One Generation with NSGA-II

We introduce our optimization pipeline, a tailored genetic algorithm (GA) that is illustrated in Figure 4.4 by using the previously introduced quadruped robot (Fig. 4.1e) as an example. The design of a metatruss sits in a discrete combinatorial space. The truss structure is inherently a network composed of multiple subnetworks – C-networks – that have shared nodes, but their edges are exclusive. The design of subnetworks involves C-network indices, contraction levels, and binary control sequences, which are all discrete in nature. While there are methods that treat discrete variables in a continuous manner, often used in topology optimization such as density-based approach [123], they don't fit our problem due to the connectivity constraint we impose: all beams within the same C-network need to belong to a same sub-network in a truss network, or we call them being interconnected. This constraint is not easily expressed in a continuous and differentiable form, which conventional optimization algorithms would require.

Given this constraint, we turn to the GA [124], a method adept at handling discrete and combinatorial search spaces. To ensure our specific constraints are respected, we base on the standard GA and customize its operators, enabling it to efficiently explore the design space while following our C-network connection constraints and the discrete nature of parameters.

GA is first used to initialize a generation of designs $\{D_0, D_1, \ldots, D_{N_G}\}$, where N_G is the population size of one generation (Fig. 4.4a i-ii). Each design is simulated following N_A action sequences, which are evaluated by the corresponding subtask objective functions and got N_s fitness values (Fig. 4.4a iii-v).

We use Non-dominated Sorting Genetic Algorithm II (NSGA-II) [125] for selecting designs, keeping some designs and removing others. To fill up the gene pool again, we use mutation and crossover on the kept designs, and add the newly generated ones through mutation, crossover, and initialization into the kept pool (Fig. 4.4a vi-vii). This renewal of the gene pool increases the chance of the algorithm getting higher-performing designs over time, potentially moving them to a new Pareto Front.

4.4.8 Elite Pool Strategy for Optimization Across Generations

At this stage, we introduce two distinct gene pools: an active gene pool with a capacity of N_a designs, and an elite gene pool with a capacity of N_e designs (Fig. 4.4b). Each generation, the designs in the active gene pool are assessed and sorted using the NSGA-II algorithm, based on Pareto dominance and crowding distance. A fixed percentage ρ of top-performing designs, referred to as elite designs, are preserved, while the rest are discarded. The active gene pool is then updated with new designs generated through crossover, mutation, and regeneration operators.

Every N_g generations, instead of simply preserving the elite designs within the active gene pool, these elite designs are temporarily moved to the elite gene pool. This allows the remaining non-elite designs to continue evolving, providing them with the opportunity to further optimize and potentially exceed the current elite designs.

Once the elite gene pool reaches its capacity, the designs it contains are moved back into the active gene pool. This cyclical process encourages competition between both elite and non-elite designs. The elite gene pool thus serves two key purposes: initially, it protects high-performing designs, preventing premature convergence, while promoting diversity and exploration. Later, as it reintroduces elite designs back into the active pool, it drives further exploitation of advantageous designs for high-quality solutions.

By alternating between the preservation of elite designs and their reintegration, this mechanism helps balance exploration and exploitation, ultimately leading to better Pareto performance (Fig. 4.4b).

In our paper, we use choose N_e equal to N_a, and $\rho = 20\%$. We move elites to the elite pool every $N_g = 5$ generations. Therefore, the elite pool gets full and moved back to the active pool every $N_g/\rho = 25$ generations. The detailed number of N_a for each metatruss can be found in Numerical Results and Implementation Details.

4.4.9 NSGA-II Explanation

NSGA-II introduces metrics to find the best-performing group on multi-objectives. It uses non-dominant sorting and crowding distance sorting. First, it calculates the rank (R) of a design based on whether it is dominated by other designs. The definition of dominance is as follows: Given a design 1 with multi-objective fitness $S_1 = s_{1,0}, s_{1,1}, ..., s_{1,n}$, and a design 2 with fitness $S_2 = s_{2,0}, s_{2,1}, ..., s_{2,n}$, if

$$s_{2,i} \geq s_{1,i} \forall i \in 0, 1, ..., n \text{ and } \exists j : s_{2,j} > s_{1,j}$$

we say that design 2 dominates design 1. Designs that are not dominated by any other design have rank 0. To determine rank 1, we consider only the designs not in rank 0, and find those that are not dominated within this subset. We continue this process to find subsequent ranks (Fig. 4.12a). The $R = 0$ designs form the Pareto set, and their fitness values form the Pareto front.

The crowding distance (CD) is calculated to further sort designs within the same rank and to encourage diversity. CD measures how crowded it is around a design in the fitness space. It is calculated as follows: For a design i in a particular front:

For each fitness dimension m: a. Sort the designs in the front by fitness m. b. Assign infinite distance to boundary designs. c. For all other designs, assign a distance equal to the absolute normalized difference in the fitness values of two adjacent designs. The overall CD for design i is the sum of individual distance values for each fitness dimension:

$$CD_i = \sum_{m=1}^{M} \frac{f_m(i+1) - f_m(i-1)}{f_m^{max} - f_m^{min}}$$

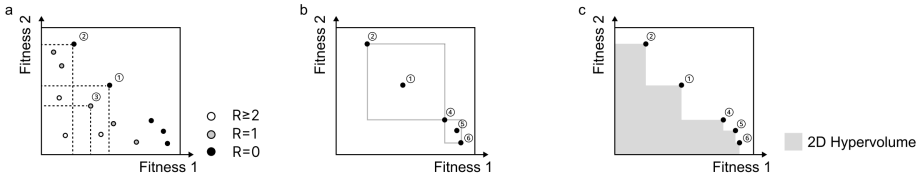


FIGURE 4.12: **NSGA-II explanation for a two-fitness optimization problem. a,** Non-dominated sorting: Designs 1 and 2 have $R = 0$ as they are not dominated by any other design. Design 3 is dominated by Design 1 but not by any other non-$R = 0$ design, so it has $R = 1$. **b,** Crowding distance calculation: For each design, the cuboid formed by its nearest neighbors in the fitness space is considered. The crowding distance is the sum of the normalized side lengths of this cuboid. For example, $CD_1 = \frac{d_{1,2}}{f_2^{max} - f_2^{min}} + \frac{d_{1,1}}{f_1^{max} - f_1^{min}}$, where $d_{1,2}$ and $d_{1,1}$ are the distances to the nearest neighbors in fitness dimensions 2 and 1, respectively. **c,** Hypervolume: The hypervolume (area in 2D) covered by the Pareto front, which serves as a measure of the quality and diversity of non-dominated designs.

where M is the number of fitness dimensions, $f_m(i)$ is the m-th fitness value of the i-th design, and f_m^{max} and f_m^{min} are the maximum and minimum values of the m-th fitness dimension. A larger CD implies that the design is more unique and potentially has distinct features, adding to the diversity of the generation. During the selection process, designs are first sorted by rank and then by crowding distance (Fig. 4.12b).

To evaluate the overall performance of a generation in multi-objective optimization, we use the hypervolume metric (Fig. 4.12c). The hypervolume is the n-dimensional space covered by the Pareto front with respect to a reference point. In the case of two fitness dimensions, it represents the area dominated by the Pareto front. A larger hypervolume indicates better overall performance and diversity of the non-dominated designs. The hypervolume provides a single scalar value that captures both the quality of the solutions and their spread across the Pareto front, making it a useful metric for comparing different generations or optimization algorithms.

4.4.10 Operators

With the constraints of symmetry and connectivity in place, traditional GA operators fall short. The standard processes, such as initialization that creates a new gene mutation that randomly selects and alters digits within a single gene, and crossover that involve swapping digits between two genes, don't align with the unique dependencies introduced by the symmetry and connectivity constraints in the metatruss. Simply applying the standard GA operators would violate symmetry and connectivity within the C-networks and beams. Therefore, we must employ customized operators that are tailored to these constraints. In the following sections, we will introduce three such operators that have been designed to function within the constraints of our optimization problem.

Initialization of C-network Indices

Standard initialization methods in GA are inadequate to navigate the unique challenges posed by our system's C-network configurations and symmetry constraints. Therefore, we've designed a specialized initialization operator that respects both symmetry and connectivity constraints.

The algorithm starts with an unassigned truss structure graph and employs the C-network configurations progressively. The first step involves earmarking a single beam for each C-network based on its symmetry property (Fig. 4.5a-d). Specifically, beams selected for self-symmetric C-networks are self-symmetric; for inter-symmetric C-networks, a beam and its inter-symmetric counterpart are chosen simultaneously.

Once this anchor layer of beams is assigned, the algorithm moves to the iterative phase (Fig. 4.5e-j). Here, it selects unassigned beams that are adjacent to already-assigned beams. The C-network assignment for these beams adheres to two key criteria: i) they must be adjacent to a beam that shares the same C-network, and ii) their symmetry properties must align with the chosen C-network. This ensures that both connectivity and symmetry constraints are satisfied.

This iterative process continues until no unassigned beams remain, thereby initializing designs that are feasible and optimized for the subsequent stages of the genetic algorithm. The detailed algorithm for the initialization and assignments of C-network indices are explained in **Methods**, "Algorithm for Initialization of C-network Indices".

Mutation

The mutation process introduces randomness into the C-network connections. It may involve changing the C-network assignment of a beam while maintaining the connectedness of beams in the same C-network. The aim here is to allow the exploration of the solution space beyond the initial population and to prevent the algorithm from getting stuck at local optima (**Methods**, Algorithm for Mutation).

Given the unique constraints of our problem, a specialized mutation operator (Fig. 4.5k) is necessary for effective optimization. Governed by a pre-defined probability p_m, the mutation process aims to explore the design space while ensuring C-network connectivity. During the mutation process (Fig. 4.5k), given a design D_m from survived designs, a random beam $e \in D_m$ has a p_m chance to be selected for mutation. The beam's C-network may be altered, subject to the following conditions: i) The new C-network index i_m must be one of the adjacent C-network indices, and ii) If e is self-symmetric, the new C-network i_m must also be self-symmetric. Otherwise, there is no inter-symmetric beam for the inter-symmetric C-network. If e has an inter-symmetric beam e', the C-network index of e' will be altered accordingly.

A fail-safe checking mechanism will be applied each time a mutation is applied. If the mutation results in the disconnection of a C-network, the operation will be reverted (Fig. 4.5l-m). The iterative process will continue until a random number r exceeds p_m. This operator ensures that the mutation is both random and constrained, facilitating the traversal of the design space without violating the system's structural or functional integrity.

Constrained Crossover Operator for Design Synthesis

The crossover operation (**Methods**, Algorithm for Constrained Crossover Operator) leverages two randomly selected survived designs as parent designs, aiming to create offspring with features from both. The process is analogous to the mutation operator but involves the exchange of beams between two designs instead of altering beams within a single design.

Briefly, a beam e_i^0 is randomly picked from the first parent design, and its corresponding beam e_i^1 in the second parent is identified. If applicable, the C-network indices of the two beams are then swapped, along with their inter-symmetric counterparts. A fail-safe checking ensures that the swapping adheres to the symmetry and connectivity constraints. The operation iterates until a successful crossover is achieved, thereby synthesizing new designs while preserving the requisite constraints.

4.4.11 Truss Topologies and Tasks

Quadruped Robot

The quadruped robot was designed with 150 actuatable beams arranged in a symmetric configuration. The robot's structure consists of four legs connected to a central body, maintaining bilateral symmetry along its longitudinal axis. This symmetry was reflected in the C-network assignments, with 30% of the C-networks designated as self-symmetric and the remainder as inter-symmetric pairs.

The quadruped was optimized for four distinct tasks: 1. Walking: Forward locomotion along the x-axis. 2. Turning: 90-degree rotation around the z-axis. 3. Tilting: Changing the orientation of the robot's top surface. 4. Crouching: Lowering the overall height of the robot.

These tasks were chosen to demonstrate the robot's ability to perform diverse movements using a single optimized configuration. The performance of the quadruped across these tasks with varying numbers of C-networks is detailed in the 4.2.4 section.

Shape-Shifting Helmet

We designed a shape-shifting helmet with two functional objectives. The two objectives represent specific target shapes the robot is trained to achieve. For each objective, the goal is to let the assigned key joints (Fig. 4.7a) approximate the corresponding target positions (Fig. 4.7a), while maintaining the rest of the non-key joints at their original locations as much as possible. When computing, we assign each joint a weighting factor. The value of the weight is based on each joint's proximity to the closest key joint along the beams. The weight of each joint diminishes as it moves further from the key joints, and the key joints themselves carry the maximum weight (Fig. 4.7b).

Figure 4.7c shows the resultant shape transformations. As demonstrated, the helmet effectively approximates each target shape (Supplementary Video 2). The training performance corresponding to each objective is depicted in Figure 4.7d,e. The plots show that the morphing helmet's ability to approximate target shapes improved steadily over the iterations.

Lobster Robot Trained for Energy Efficiency

We use a lobster-inspired walking robot to study how energy efficiency can be integrated into the functional objective (Fig. 4.7f, Supplementary Video 3). The robot that walks with energy efficiency has two subtasks. One subtask is to achieve a high locomotion speed by evaluating the displacement of the centroid after the action sequence is completed. The second subtask is to minimize the energy consumption of the robot, which is calculated by accumulating the axial force and displacement of all joints along time steps (see 4.4.12 section for details).

Figure 4.7g, h shows the lobster performance with iterations. We observed that, in this case, using multiple objectives can enhance the search efficiency and quality. For example, when searching for a solution for both the locomotion task and energy efficiency task, the search is faster and converges at a better result than searching for locomotion only (Fig. 4.7i).

Tentacles

We also demonstrated how the metatruss method could be utilized to design a tentacle actuator that approaches multiple target locations in a volumetric space with its end joint (Fig. 4.7j), with its training performance reported (Fig. 4.7k,l).

The joint at the most distal position of the tentacle is designated as the key joint. The tentacle is assigned three objectives, each including one subtask of reaching the target point

(Fig. 4.7, Supplementary Video 4). The three subtasks are to reach three different target points, each situated at the centers of the green dots in 4.7j.

This task demonstrates the metatruss's effectiveness in tasks requiring precision. By the 800th iteration, the key joint is able to achieve proximity to each target position with a distance of less than 1e-5mm, with each beam extending to a maximum length of 135mm). The tentacle shown in Fig. 4.7j is one of the designs selected from the Pareto front, exhibiting the closest distances to the three respective target positions, with deviations of 6.01mm, 2.06mm, and 3.37mm. This design is picked based on the smallest standard deviation of the three fitness values. It indicates the tentacle's ability to closely approach all three targets within a single design despite the inherent conflict among reaching three points.

Pillbug Robot

While the previous examples demonstrate the diversity of achievable morphologies and tasks, the Pillbug robot was designed specifically to showcase the physical feasibility of our metatruss design and to assess the accuracy of our simulation. The Pillbug was optimized for two tasks similar to those of the quadruped robot: walking forward and lowering its body.

The robot's structure resembles a pillbug but with four legs, featuring two smaller forelegs and two larger back legs. It consists of 50 actuatable beams, arranged to allow for both locomotion and body posture changes. We assigned an objective with two subtasks: firstly, to achieve a high locomotion speed, and secondly, to minimize the average maximum height of the metatruss. This combination of subtasks was chosen to create a low-profile, efficient walking motion.

After optimization, we selected one design from the Pareto front for physical fabrication. The fabrication process followed the approach described in our previous study [63]. The physical prototype was actuated to complete its action sequence across eight cycles, with the trajectory tracked across three of its joints for comparison with the simulated results.

Detailed information about the Pillbug robot's design, fabrication, and the comparison between experimental and simulated results can be found in the Physical Prototype and Simulator Accuracy Validation section.

4.4.12 Objective Functions

Walking. The walking subtask is defined as maximizing the directional displacement of the robot. The centroid position p_c is the position of the center of mass of all the joints. At time t, the centroid position is computed as $p_c(t) = \frac{1}{N_V}\sum_{i=0}^{N_V-1} v_i(t)$. During a given period of time expressed by $N_T \cdot I$, the displacement of the centroid along x axis is defined as the walking distance $\tilde{d_w}$. The robot is expected to achieve a walking distance as large as possible. u_x is the unit vector along the x axis.

$$\tilde{d_w}(t) = (p_c(t) - p_c(0)) \cdot u_x \tag{4.1}$$

Reorientation The reorientation subtask is to maximize the orientation alignment $\tilde{a_o}$ between the original orientation of a subset of the metatruss, represented as a unit vector u_o, with the target orientation, represented as a unit vector u_g. In this case, the metatruss starts with an initial orientation represented as a unit vector $u_o(0)$, and is expected to align with a target unit vector u_g. A subset of vertices V_o in the metatruss are assigned to determine the orientation. For the turning subtask of the quadruped robot shown in Figure 4.1e, five vertices in two incident tetrahedrons in the middle part of the robot are assigned as V_o to represent the body. $u_o(0)$ is assigned as the unit vector along the positive direction of the x-axis, while u_g is the unit vector along the positive y-axis. We define the orientation alignment fitness value as

$$\tilde{a_o} = u_o \cdot u_g \tag{4.2}$$

As the metatruss deforms, the vertices are subject to relative movement, we determine the metatruss's orientation by calculating a transformation matrix. This matrix transforms the initial vertex positions, demoted as $V_o(0)$, to approximate their final positions $V_o(T)$ at time T. We then extract the orientation information from this transformation. To achieve this, we employ the BFGS optimizer [126] from *'scipy.optimize.minimize'* [127] to optimize the 3D transformation matrix $\mathscr{F}$. This optimization minimizes the average distance between the current positions $V_o(t)$ and the transformed initial positions $V_o(0) \cdot \mathscr{F}$. Consequently, we derived the current orientation $u_o(T)$ by rotating the initial orientation $u_o(0)$ with $\mathscr{F}$.

Energy Efficiency during Walking. This function is defined as minimizing the average energy consumption during walking for a unit distance. Here the total energy consumption is calculated by summing up the work done by all the beams. The work is calculated by summing up the product of axial force f of each beam and its length change Δ_l at every time step. The energy efficiency fitness value $\tilde{\phi}$ is defined as the negative of the quotient obtained by dividing the total energy consumption by the total directional displacement as described in (4.1).

$$\tilde{\phi} = -\frac{\sum_{i=0}^{N_E-1}\sum_{t=0}^{N_T \cdot I-1} \Delta_{li}(t) \cdot f_i(t)}{\tilde{d_w}(t)} \tag{4.3}$$

Here, i indexes each beam, and t represents the time step.

Shape Approximation. The shape approximation subtask guides the metatruss to approximate a target shape, prioritizing selected key joints to reach target locations while considering the spatial relationships among all joints. As the morphing helmet example (Figure 4.7a) shows, in two objective functions, three joints are designated as key joints including one each at the front, left, and right. Their are used to guide the deformation optimization.

The subtask is formulated as minimizing the sum of the mean squared distances between the joints and their respective target positions, with a weight applied to all joints. First, a set of key joints $I_k = \{i_0, i_1, \ldots, i_{N_X-1}\}$ is selected, and their target positions $P_t = \{p_{t0}, p_{t1}, \ldots, p_{tN_X-1}\}$ are defined. For non-key joints, their target positions are set to be their initial positions and a weight is applied based on proximity to the closest key joint along the beams. Figure 4.7c demonstrates the weighting scheme, where the weight for each joint diminishes as it moves closer to the key joints, and key joints themselves carry zero weight.

Specifically, the weight for each non-key joint is determined by the distance d_i to its closest key joint along the beams, with the accumulated beam lengths defining the distance. The distance is then converted to weight using a hyperbolic tangent transformation: $w_i = \frac{1}{\pi}\tan^{-1}((d_i - \alpha)\cdot\beta) + \frac{1}{2}$, where d_i is the length of the shortest path to the closest key joint, and α and β are constants. In the context of the helmet (Fig. 4.7a-c), the $\alpha = 2.3, \beta = 2.0$.

The shape approximation fitness value X is defined as:

$$X(t) = -(\frac{1}{N_X}\sum_{i\in I_k}\|v_i(t) - p_{ti}\|_2 + \xi \cdot \sum_{i=0}^{N_V-1} w_i\|v_i(t) - v_i(0)\|_2) \tag{4.4}$$

Here, ξ indicates the importance of keeping non-key joints to their original positions. In the case of helmet, ξ is set to 1.0 to keep the overall shape, while in the case of tentacle, ξ is set to 0.0 to maximize the range of accuracy of the key joint.

Here, the metatruss maintains its overall shape while adjusting key joints to align with the target positions. This leads to a controlled deformation that respects the spatial relationships between joints.

4.4.13 Numerical Results and Implementation Details

Implementation Details

In our study, we use the notation $\leftrightarrow$ to indicate inter-symmetric channels and $\odot$ to denote self-symmetric channels. We implemented various metatruss designs with different C-network configurations. The quadruped design with 8 channels features a symmetric topology, with channels configured as $0 \leftrightarrow 1$, $2 \leftrightarrow 3$, $4 \leftrightarrow 5$, $\odot 6$, $\odot 7$, comprising 6 inter-symmetric and 2 self-symmetric channels. A simpler quadruped design with 2 channels maintains symmetry with both channels being self-symmetric ($\odot 0$, $\odot 1$).

We also explored more complex quadruped designs: a 16-channel version with 4 self-symmetric and 12 inter-symmetric channels, a 32-channel version with 8 self-symmetric and 24 inter-symmetric channels, and a 64-channel version with 16 self-symmetric and 48 inter-symmetric channels. The helmet design exhibits symmetry with three self-symmetric channels ($\odot 0$, $\odot 1$, $\odot 2$), while the lobster design is symmetric with two inter-symmetric and two self-symmetric channels ($0 \leftrightarrow 1$, $\odot 2$, $\odot 3$). Notably, the tentacle design is asymmetric, consisting of four self-symmetric channels ($\odot 0$, $\odot 1$, $\odot 2$, $\odot 3$).

In our optimization process, we set the active pool size equal to the elite pool size, with an elite percentage of 20% in every generation. As the C-network number increases, we scaled the pool size accordingly: 128 for the 8-channel quadruped, 48 for the helmet, 64 for both the lobster and tentacle, 32 for the 2-channel quadruped, 256 for the 16-channel quadruped, 512 for the 32-channel quadruped, and 1024 for the 64-channel quadruped.

We conducted our computations using Google Cloud Computing with 128 cores, achieving an average optimization time of 18 hours and 24 minutes for the 4-channel quadruped over 1000 iterations.

Performance Analysis with Varying C-network Numbers

To investigate the relationship between the number of C-networks and robot performance, we conducted a series of experiments using a quadruped robot model. The robot was trained to perform four distinct tasks: walking (maximizing forward distance), turning (90-degree rotation), tilting (changing top orientation), and crouching (lowering height). We tested configurations with 2, 8, 16, 32, and 64 C-networks, always setting 30% as self-symmetric and the rest as inter-symmetric. This 30

The genetic algorithm was run for 1000 iterations for each configuration. We used a population size of 200, with 100 designs retained after each iteration. New designs were generated through mutation (70 designs) and crossover (20 designs), with 10 new random initializations per iteration. The elite pool capacity was set to 400.

Performance was evaluated using the hypervolume of the Pareto front at the 1000th iteration. The hypervolume metric was chosen as it provides a scalar measure of the quality of a Pareto front in multi-objective optimization, capturing both the spread and the proximity to the ideal point. It was calculated using the pygmo library's hypervolume function, with a reference point set to the worst observed values for each objective plus a small offset. We conducted one-way ANOVA to compare performances across C-network numbers, with significance level set at 0.05. Tukey's Honest Significant Difference (HSD) was used for post-hoc pairwise comparisons.

The ANOVA assumptions were verified: data independence was ensured by separate testing, homogeneity of variance was confirmed by Levene's test ($F = 0.085$, $p = 0.986$), and normal distribution was verified by the Shapiro-Wilk test ($F = 0.962$, $p = 0.343$).

ANOVA results showed significant differences among the five groups ($F(4, 10) = 12.840$, $p < 0.001$, $\eta^2 = 0.138$). Tukey's HSD revealed significant performance improvement when increasing from 2 to 8 C-networks ($p = 0.003$), but no statistically significant differences

among configurations with 16 or more C-networks ($p > 0.877$). Complete pairwise comparison results are provided in Supplementary Table S1.

4.4.14 Physical Prototype and Simulator Accuracy Validation

We conducted a comprehensive evaluation of our physical prototype, the "pillbug," to validate our metatruss design and assess the accuracy of our simulator.

Experimental Setup and Data Collection

The pillbug prototype was actuated to complete its action sequence across eight cycles, with the entire process recorded on video (Fig.4.8 a, Supplementary Video 5). We tracked the trajectories of three key joints throughout the experiment, collecting 536 tracked points for each action sequence cycle (Fig.4.8 b-d).

To facilitate comparison with our simulation, we overlaid the simulated trajectory on the experimental data. We highlighted static positions at the beginning and end of each cycle to identify key points of motion and rest. The trajectory was then segmented into individual cycles at these static positions for detailed analysis.

Data Analysis and Metrics

To quantify the similarity between simulated and experimental results, we calculated several metrics:

Average Trajectory discrepancy (d_t): The mean point-wise distance between corresponding points on the experimental and simulated trajectories, averaged across the entire trajectory.

Average Trajectory Cycle discrepancy (d_c): The mean point-wise distance between each experimental trajectory cycle and the corresponding simulated cycle, averaged across all cycles and trajectories.

Average Static Position discrepancy (d_s): The mean point-wise distance between each pair of static positions across all trajectories.

Average Self Trajectory Cycle discrepancy (d_f): To assess the internal consistency of the prototype, we calculated the mean trajectory cycle for each joint and measured the distance between each experimental cycle and this mean.

Results and Conclusion

Our analysis yielded the following results:

d_t = 3.77 cm (averaged across all three trajectories), d_c = 2.68 cm, d_s = 1.26 cm, d_f = 0.54 cm. For reference, the fully extended length of each beam in the prototype is 17.8 cm. The average locomotion speed is 2.45 cm/s, which is 9.24% of its original lengths. The full displacement is 86.0 cm.

The larger values of d_t and d_c compared to d_s likely result from inherent friction differences in the pneumatic plastic syringes used as beams, causing unsynchronized actuation times under uniform air pressure. This factor was not accounted for in our simulation. The smaller d_s indicates good alignment at static positions, which may be attributed to our use of a highly-damped dynamic model and relatively low actuation pressure (+0.8 psi and -0.8 psi) in the simulation.

The fact that d_t is higher than d_c suggests accumulated errors across cycles, a common phenomenon in open-loop control systems. The low d_f value indicates high consistency in the metatruss motions despite inherent friction variances.

These results validate our metatruss design approach while highlighting areas for future improvement, such as standardizing linear actuators for uniform friction or integrating friction variance into the simulator. Additionally, the findings suggest that incorporating sensors for closed-loop control could enhance long-term accuracy, particularly given the ample interior space in the metatruss design.

4.4.15 Algorithms

Here we detailed four algorithms critical to our computational pipeline, including the algorithm for initialization of C-network indices, the algorithm for mutation, the algorithm for constrained crossover operator, and the algorithm for single training iterations with elitism.

Algorithm 1 Random Initialization of C-network Indices

$I_C = \{0, 1, \ldots, N_C - 1\}$: the set of all C-network indices
E_u: unassigned beams, E_i: beams of the ith C-network
E_s: self-symmetric beams, E_r: inter-symmetric beams
for each C-network c_i, where $i \in I_C$ **do**
    if $\exists c_{i'}$, such that $c_i \leftrightarrow c_{i'}$ **then**
        Randomly select $e, e' \in E_r$ that $e \leftrightarrow e'$
        Move e from E_u to E_i, e' from E_u to $E_{i'}$
    else if $\odot c_i$ **then**
        Randomly select $e \in E_s$
        Move e from E_u to E_i
    end if
end for
repeat
    Randomly select $e \in E_u$ adjacent to any assigned beam
    if $\odot e$ **then**
        Randomly select an adjacent C-network i that $\odot i$
        Move e from E_u to E_i
    else if $\exists e'$ that $e' \leftrightarrow e$ **then**
        Randomly select an adjacent C-network index i
        if $\exists i'$ that $i' \leftrightarrow i$ **then**
            Move e from E_u to E_i, e' from E_u to $E_{i'}$
        else
            Move e, e' from E_u to E_i
        end if
    end if
until E_u is empty

Algorithm 2 Mutation of C-network Indices

p_m: the probability of executing mutation
I_s: self-symmetric C-network indices, I_r: inter-symmetric C-network indices
$r \leftarrow$ random number in $[0,1]$
repeat
  if $r < p_m$ **then**
    Randomly pick beam $e \in E$, with C-network index i
    Find all adjacent C-network indices I_a
    if $\odot e$ **then**
      Randomly select C-network with index $i_m what is to be mutated, where \in I_s \cap I_a$
      Move e from E_i to the mutated C-network E_{i_m}
    else if $\exists e'$, that $e' \leftrightarrow e$ **then**
      Randomly select C-network $i_m \in I_a$
      if $\exists {i'}_m$, that ${i'}_m \leftrightarrow i$ **then**
        Move e to E_{i_m}, e' to $E_{{i'}_m}$
      else if $\odot i_m$ **then**
        Move e, e' to E_{i_m}
      end if
    end if
    if E_i or E_{i_m} is disconnected **then**
      Revert C-network assignment
    end if
    $r \leftarrow$ random number in $[0,1]$
  end if
until $r \geq p_m$

Algorithm 3 Constrained Crossover of C-network Indices

p_c: the probability of executing crossover
D^0, D^1: two selected survived designs for crossover
$r \leftarrow$ random number in $[0,1]$
E^0, E^1: sets of beams in D^0 and D^1
e_i^j: the ith beam of the jth selected design.
repeat
  Randomly select beam $e_i^0 \in E^0$, with C-network index j_0
  Select beam $e_i^1 \in E^1$, with C-network index j_1
  Swap the C-network indices by moving e_i^0 from $E_{j_0}^0$ to $E_{j_1}^0$, moving e_i^1 from $E_{j_1}^1$ to $E_{j_0}^1$
  if $\exists i'$, such that $e_{i'}^0 \leftrightarrow e_i^0$ and $e_{i'}^1 \leftrightarrow e_i^1$ **then**
    Swap the C-network indices of the inter-symmetric beams, by moving $e_{i'}^0$ to from $E_{j_0'}^0$ to $E_{j_1'}^0$, moving $e_{i'}^1$ from $E_{j_1'}^1$ to $E_{j_0'}^1$
  end if
  if $\exists E_k^0$ or E_k^1 is disconnected, where $k \in I_C$ **then**
    Revert the swap of C-network indices.
  end if
  $r \leftarrow$ random number in $[0,1]$
until $r \geq p_c$

Algorithm 4 Single Training Iterations with Elitism

N_p: maximum number of designs in a pool
N_s: number of designs that survives
N_m: number of designs generated by mutation
N_c: number of designs generated by crossover
N_i: number of iterations in each generation
P_e: elite pool, P_a: active pool
if Cardinality of P_e equals N_p **then**
    Move N_p designs from P_e to P_a
else
    Initialize N_p designs in P_a
end if
for $i = 0, 1, \ldots, N_i - 1$ **do**
    Evaluate the fitness of designs in P_a as $f_i, \forall i \in P_a$.
    Apply NSGA-II sorting to P_a, resulting in sorted designs $P_a = \{D_0, D_1, \ldots, D_{N_p-1}\}$ and corresponding fitness $\{\hat{f}_0, \hat{f}_1, \ldots, \hat{f_{N_p-1}}\}$.
    Retain the top N_s designs in P_a and remove the rest: $P_a \leftarrow \{D_0, D_1, \ldots, D_{N_s-1}\}$.
    for $j = 0, 1, \ldots, N_m - 1$ **do**
        Select a design $D_m \in P_a$ uniformly at random
        Add its mutation $\bar{D_m}$ into P_a
    end for
    for $j = 0, 1, \ldots, N_c - 1$ **do**
        Select two distinct designs $D_{c0}, D_{c1} \in P_a$ uniformly at random
        Add the crossover result $\bar{D_c}$ to P_a
    end for
    for $j = 0, 1, \ldots, N_i - (N_m + N_c) - 1$ **do**
        Initialize a new design $\bar{D_i}$ and insert into P_a
    end for
end for
Move the top N_s designs from P_a to P_e, clear P_a

4.4.16 Usage of Large Language Model in Writing

In the preparation of this manuscript, we utilized ChatGPT with GPT-4 specifically for grammar checking purposes. The prompts we employed were structured as follows: "[the text] Please check the grammar of this writing as a submission for a scientific journal, please don't change or distort the meaning or create new information."

4.5 Data Availability

Correspondence and requests for code implemented, data generated and analyzed, and 3d printing models during the study should be addressed to jianzheg@andrew.cmu.edu.

4.6 Code Availability

Code implemented for genetic algorithm, tracking and data processing should be addressed to jianzheg@andrew.cmu.edu.

4.7 Supplementary Information

4.7.1 On-body Control Circuit Using Mechanical Logic Gates

Our optimization generates sequential open-loop control signals that could potentially be implemented through mechanical structures, eliminating the need for multiple electronic control inputs and requiring only a constant power supply. Taking our pillbug's forward walking task as an example, the optimized open-loop control sequence consists of a 4-bit binary signal over 4 time steps (Fig. 4.13 a). Each time step's control signal can be represented as a 2-bit input generating a 4-bit output, making it implementable through a 2-to-4 multiplexer (Fig. 4.13 b). We used logic optimization tools to analyze the truth table and derived a simplified circuit for the Pillbug's actuation sequence. The resulting circuit requires 14 logic gates, including one clock unit (Fig. 4.13 c).

For physical implementation, we identified a mechanical circuit design from previous work [112, 113] that uses pneumatic power and cylindrical units with bi-stable membranes. In this design, the air pressure differential between two chambers affects the membrane's direction (Fig. 4.13 d, e), which in turn blocks one of the air tubings. By connecting the six ports to different pressure sources, each unit can be preconfigured to function as AND, OR, or NOT gates (Fig. 4.13 f), theoretically enabling the implementation of any logic circuit. Additionally, when connected to an external tank and a constant air pressure source, these units can function as oscillators, serving as the clock component in the logic circuit (Fig. 4.13 g).

While current mechanical logic units measure approximately 4cm, they have potential for miniaturization. Similarly, our metatruss design can be scaled up through modified fabrication processes and corresponding parameter adjustments in the simulator, without requiring changes to the optimizer. Assuming mechanical logic units could be reduced to 1cm³ and metatruss beams could be expanded to 20cm length, we created renderings of a circuit board containing all required units and visualized a pillbug robot carrying this control system (Fig. 4.13 h). In this configuration, the robot would require only a single constant air power source, enabling operation either through a single tethered air tube or potentially untethered if equipped with an onboard compressed air tank.

This implementation strategy demonstrates how our metatruss system could evolve toward more self-contained, mechanically controlled robots with simplified power and control requirements.

4.7.2 This section includes:

- Supplementary Videos S1 to S5 (Available Online).
- Supplementary Table S1.

Supplementary Videos

- **Video S1:** A quadruped metatruss performs walking, turning around, lowering body and tilting the top. (Left: perspective view. Right: side or top view.)
- **Video S2:** Top: A lobster metatruss performs walking. Bottom: The same lobster metatruss performs walking with energy efficiency. Right: Training result showing the relationship between the hypervolume of the multi-objective optimization for the lobster metatruss.
- **Video S3:** A Helmet metatruss transforms into two target shapes from the same initial shape. (Left: perspective view, Right: side or top view.)

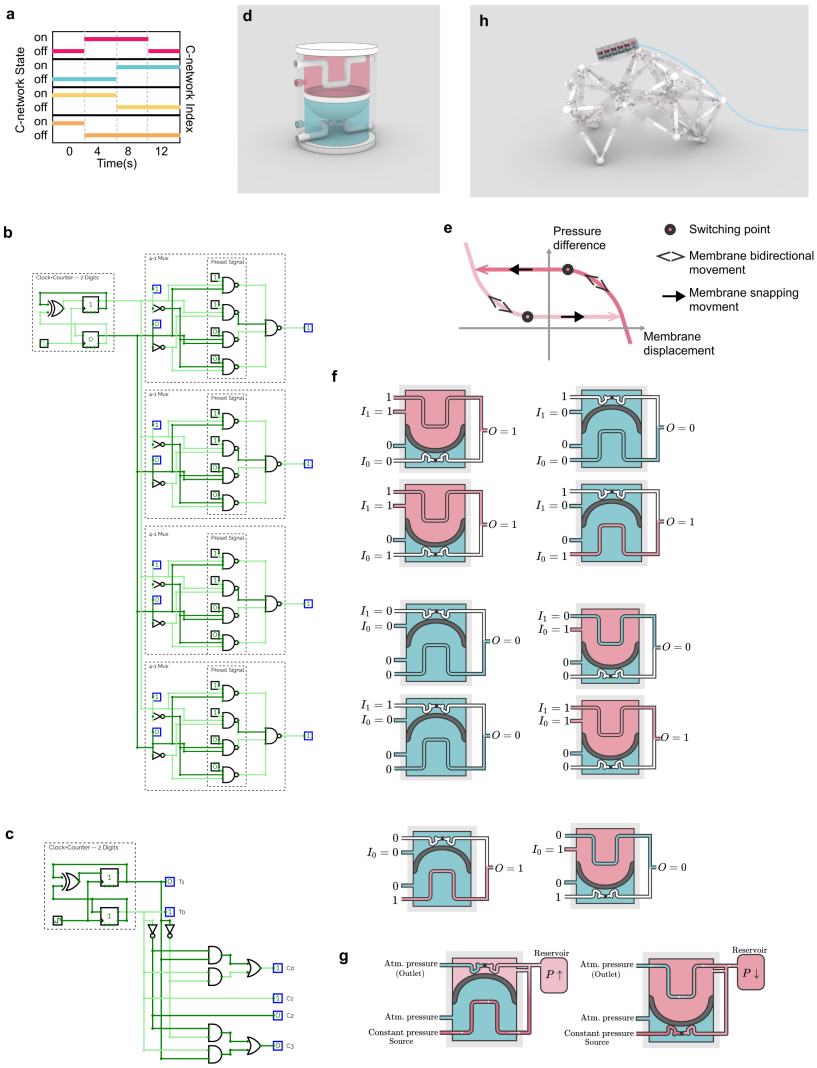


FIGURE 4.13: **On-body control circuit using mechanical logic gates. a,** Open-loop control signal optimized for Pillbug metatruss towards walking forward. **b,** 4-to-8 multiplexer circuit for the open-loop signal. **c,** Simplified circuit for the open-loop signal. **d,** The basic mechanical circuit unit. **e,** The illustration of the bi-stable mechanism of the mechanical circuit unit. **f,** The illustration of the ports connection for AND, OR, and NOT gate. **g,** The illustration of the clock unit. **h,** A rendering of the pillbug with on-body mechanical open-loop control circuit.

- **Video S4:** A tentacle metatruss reaching three different target positions. (Left: perspective view, right: side view.)
- **Video S5:** Top: tracked video of a fabricated pillbug metatruss walking forward. Bottom: The simulation of the pillbug metatruss walking forward.

Supplementary Table

TABLE S1: Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval) for the hypervolume performance of quadruped robot with 5 C-network counts, with group 0 for 2 C-networks, group 1 for 8 C-networks, 2 for 16 C-networks, 3 for 32 C-networks, 4 for 64 C-networks

Comparison	**Statistic**	**p-value**	**Lower CI**	**Upper CI**
(0 - 1)	-12.799	0.003	-22.023	-3.574
(0 - 2)	-15.764	0.000	-24.989	-6.540
(0 - 3)	-18.710	0.000	-27.935	-9.486
(0 - 4)	-19.695	0.000	-28.919	-10.470
(1 - 0)	12.799	0.003	3.574	22.023
(1 - 2)	-2.965	0.877	-12.190	6.259
(1 - 3)	-5.911	0.352	-15.136	3.313
(1 - 4)	-6.896	0.214	-16.120	2.329
(2 - 0)	15.764	0.000	6.540	24.989
(2 - 1)	2.965	0.877	-6.259	12.190
(2 - 3)	-2.946	0.879	-12.170	6.279
(2 - 4)	-3.930	0.722	-13.155	5.294
(3 - 0)	18.710	0.000	9.486	27.935
(3 - 1)	5.911	0.352	-3.313	15.136
(3 - 2)	2.946	0.879	-6.279	12.170
(3 - 4)	-0.985	0.998	-10.209	8.240
(4 - 0)	19.695	0.000	10.470	28.919
(4 - 1)	6.896	0.214	-2.329	16.120
(4 - 2)	3.930	0.722	-5.294	13.155
(4 - 3)	0.985	0.998	-8.240	10.209

Chapter 5

Closed-loop Control of Truss Robot using Reinforcement Learning

Building on the actuator grouping mechanism introduced in PneuMesh [128], we explored reinforcement learning as an approach to enable closed-loop control of truss robots. While our previous work demonstrated that grouped actuation with open-loop control can achieve complex motions, physical implementation revealed sensitivity to factors like friction variations and accumulated errors over multiple cycles. This chapter presents preliminary work investigating how reinforcement learning might address these challenges while maintaining the computational benefits of actuator grouping.

Given an initial truss structure and actuator grouping optimized through genetic algorithms, we frame the control problem as a reinforcement learning task where an agent learns to select group actuation states based on real-time position feedback. Specifically, we implement a co-design approach that couples genetic algorithms for optimizing actuator groupings with reinforcement learning for control policy development. The model incorporates customized initialization, mutation and selection functions tailored to PneuMesh system constraints, along with translation-invariant state vectors for reinforcement learning.

Using a robotic table as our test case, we demonstrate how this approach enables adaptation to varying conditions while performing multiple tasks such as moving, lowering its body, or tilting its tabletop. The results suggest potential benefits of closed-loop control, while also highlighting challenges in state representation and reward function design that warrant further investigation. The methods and findings presented here complement our open-loop control framework while pointing toward future directions in adaptive control of morphing truss robots.

This chapter details our implementation and experimental findings, beginning with an overview of the physical system and problem formulation, followed by our methodology combining genetic algorithms with reinforcement learning, and concluding with preliminary results and discussion of limitations.

5.1 Introduction

Living creatures interact with the world through a large variety of transformation behaviors. Caterpillars repetitively shrink and twist the body to climb; pufferfish expand their bellies to intimidate enemies; and macrophages deform the cytomembrane to swim and hunt other cells. In the field of robotics, one robotic system that can create complex geometries and motions is Variable Geometry Truss (VGT) [129–131]. A VGT typically consists of an arbitrary number of beams and joints, where beams are connected through joints and form truss structures composed of tetrahedrons. Each beam is a linear actuator that can expand or contract. By morphing the body with actuators, VGTs can execute various motions, including rotation, twisting, linear and volumetric scaling.

Despite its versatility and adaptability, a VGT suffers from the exponentially scaled complexity of the control system. As each beam is controlled independently, the number of control units (e.g. air tubings or linear motors) is cubically scaled with the complexity of the truss morphology. A recent VGT design, PneuMesh [128], simplifies the control system while achieving various complex motions. A PneuMesh VGT is a pneumatic driven truss where each beam is a syringe-like linear actuator with an air channel inside. Rather than individually controlling beams, PneuMesh selectively connects the air channels of beams through multi-way joints that direct air to separate beams. With this design, beams are integrated into different sub-networks, each sub-network are controlled synchronously by a single air valve. With PneuMesh, a complex VGT (up to more than 118 beams) can be controlled by a limited number of control modules (three to six control modules) but still create rich motions through the connections of beams. For example, a lobster-shaped VGT (Figure 5.1) with 67 beams can move forward and grab objects (Figure 5.2 c) with only three control modules (Figure 5.2a).

FIGURE 5.1: Left: A lobster VGT with individually controlled actuators. Right: A lobster VGT with channel grouping mechanism.

Despite the simplification of the control system, identifying an optimal channel grouping and control strategy in a forward design or manual fashion is challenging. A truss with n_e edges and n_γ channels will have a solution space with a size up to $n_\gamma^{n_e}$. Meanwhile, a physically practical solution must keep the connectivity of each channel (i.e., beams belong to the same channel group need to have ensured physical connectivity for pneumatic actuation system). Lastly, the motion performance is very sensitive to the grouping strategy, where changing the channel of a single beam might lead to entirely different motions. Other than designing the channel assignment, for every channel design and target motion, a control strategy needs to be computed. As the number of beams and tasks increases, it is not practical to manually design both the channel assignment and the control.

The aforementioned challenges lead to our proposed VGT co-design strategy. Given a predefined truss structure, we want to find a channel assignment and a set of control policies that enables the VGT to achieve multiple morphing tasks. Researchers have explore the problem of optimizing both the morphology and control of irregular-shaped robots. [132] use CPPN and genetic algorithms for a periodic control signal. [133] presented a graph heuristic search and reinforcement learning for co-designing a tree-structure robot. While these work achieved great performance for their specific systems, they have different inherent topology from PneuMesh VGT which is a 3D graph with loops and specific constraints on the channel grouping.

In this work, we introduce a hybrid model to solve the co-design problem of PneuMesh VGTs through genetic algorithms (GA) and reinforcement learning (RL). We adopted NSGA-II for the selection method of GA to improve the performance of multiple objectives without

being biased on one of the objectives. Customized initialization and mutation functions are implemented to ensure the symmetry and the connectivity of channels. To evaluate the potential of a channel design, we train a control policy with a RL controller using PPO [134] method, and simulate the motion of a VGT controlled by the trained policy. The simulation results are sent into customized objective functions to generate the ratings, or fitness.

We demonstrate the effectiveness and efficiency of our method by designing a robotic truss table and applying our method on training the table towards four objective motion tasks, including lowering the height, tilting the table top, locomotion and turning around.

5.2 Physical System

In the previous work, PneuMesh [128] introduces a VGT design that can minimize the number of control units while keeping a sufficient degree of control over the truss and achieving various motion tasks.

Traditional VGTs are over-actuated systems where the number of beams is more than what is needed for given motion tasks. Instead of having independent actuators, we selectively connect adjacent actuators through their shared joints with a customized inner air channel design (Figure 5.2a,b). Interconnected actuators are synchronized and will be actuated or de-actuated at the same time corresponding to a single control input.

In such a way,a system with n_e edges and n_c channels will have n_c binary values as control inputs at every moment. Despite the decrease in the degree of freedom(DOF), each beam can still change length, and each channel is a sub-network that can span through the entire truss. These properties endow the truss with substantial freedom to deform and achieve various motions.

A trade-off exists between the reduction of DOF and simplification of the control system. The goal of this work is to find a balance between optimizing the channel assignment and control signals.

5.3 Method

5.4 Definition and Problem Statement

We define a truss structure as a graph $T = \{E, n_\gamma\}$ consisting of n_v vertices and n_e edges, where E is the adjacency list of vertices and n_γ is the predefined number of channels. A T is associated with the channel assignment of its edges $C = [c_0, c_1, ...c_{n_e-1}]$, where c_i is the integer index of the channel assigned to the ith edge.

At every time step t, T has a corresponding state $s_t = \{V_t, U_t, V_t', U_t', k_t\}$, where V_t and V_t' are absolute and relative node positions, U_t and U_t' being absolute and relative node velocities. The actuation state of channels is a vector of n_γ Boolean values, each indicating the on/off state of one channel.

A simulator M takes in T, C and s_t, and outputs the next state s_{t+1}. T can be assigned with multiple objective functions $O = [o_0, o_1, ...o_{n_o-1}]$ that takes in the state sequence of an entire simulation $S = [s_0, s_1, s_2, ...]$ and outputs a rating vector $R = [r_0, r_1, ...r_{n_o-1}]$, where each o_i takes in S and outputs r_i. A control policy π takes in T, C, s_t, and outputs an action a_t, where a_t is assigned to each channel as the new channel states.

Given a T, its initial state s_0, and O, we want to find an optimal channel assignment C and control policies $\Pi = [\pi_0, \pi_1, ...\pi_{N_o-1}]$ that maximize the general performance on all objectives.

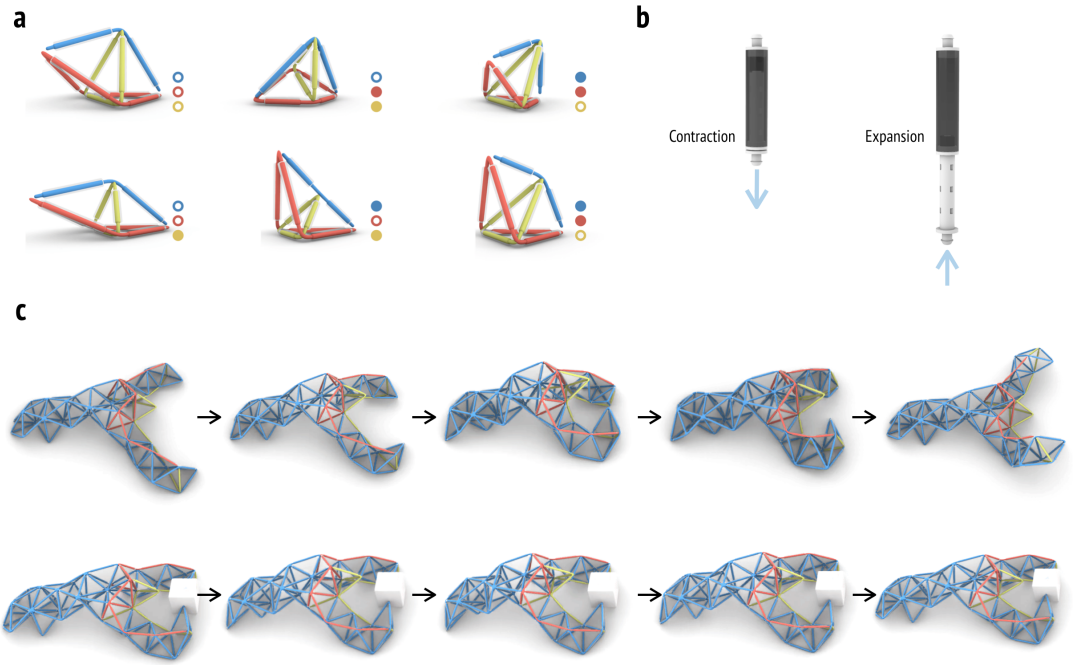


FIGURE 5.2: Physical system of channel grouped variable geometry truss.

5.5 Overview

We introduce a co-design algorithm which uses GA to optimize C and RL to optimize Π (Figure 5.3). GA is first used to initialize a generation G of channel assignments C_0,C_1, ...C_{n_g-1}, where n_g is the population size of one generation. Each channel assignment is bundled with a control sequence ξ that includes $n_s \times n_\gamma$ Boolean values, each indicating the on/off state of the i_γth channel at the i_s time step.

In each generation, every pair of C_k, ξ_k are tested by a simulator which outputs a rating vector R_k. At the end of the generation, a selection function (Appedix 5.13) following NSGA-II[[135]] keeps the population $\bar{G}$ with the best performance and discards the rest. The missing population will be filled by duplicating and mutating $\bar{G}$ through a customized mutation function (Appendix 5.12) that assures the channel connectivity and the symmetry of the truss. We also introduce an elite pool mechanism (Appendix 5.14) that temporarily keeps elite populations aside to avoid exploitation caused by the domination of elites.

However, ξ is a fixed sequence that only applied to the truss with a specific initial condition and at a specific environment. Once the initial condition changes, ξ needs to be retrained. Furthermore, errors will accumulate in a long trajectory. Instead, we train a RL model as a closed-loop control policy π_k on top an elite C_k in the last GA generation.

5.6 Channel Symmetry Requirement

If the input truss is mirrored, a channel symmetry requirement is enforced(Figure 5.5).With this requirement, each channel is either mirrored to itself or mirrored to another channel. For example, each pair of mirrored edges are either in the same channel or in two mirrored

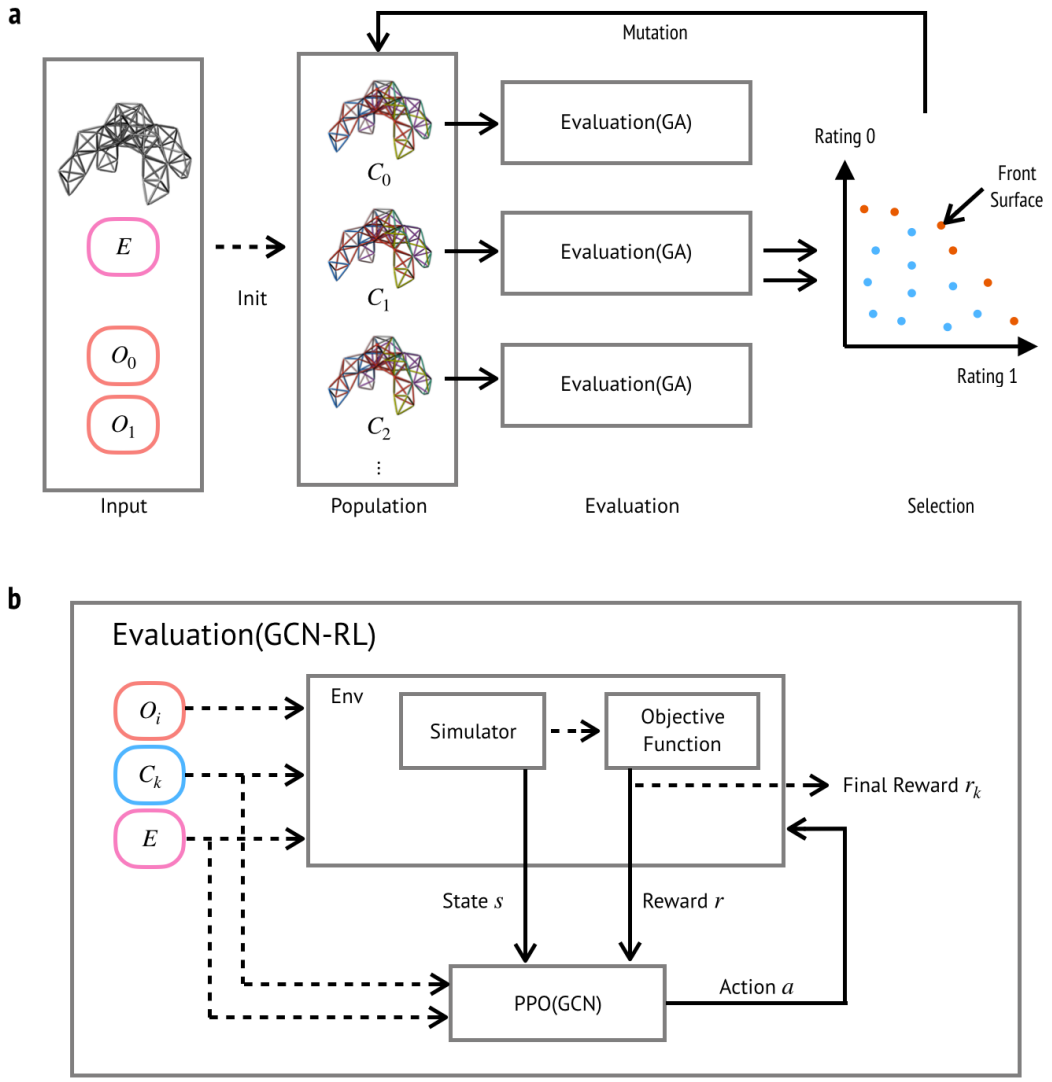


FIGURE 5.3: Computational pipeline of channel grouped VGT co-design.

channels. By default, half of the channels will be self-mirrored and the other half are inter-mirrored. In such a way, we assure the symmetry of the channels and implicitly assure the possibility of symmetric motions.

A mirroring function maps a half graph $\hat{A}$ to the whole graph $\hat{A}$. The *initialization* and *mutation* steps are performed in the half graph and mirrored to the whole graph when simulating. Other parts remains the same.

5.7 Genetic Algorithm

A GA training process consists of two nested loops. The outer loop is for exploration and the inner loop for exploitation. An evolving pool is updated at every exploitation step, and an elite pool is updated every exploration step. (C, ξ) pairs are initialized in the evolving pool and improved by iteratively applying selection and mutation (Appendix 5.12) functions. After certain generations(20 in this work), survived pairs will be put into an elite pool and a

new evolving pool will be initialized. When the elite pool is full, elite pairs will be put back into the evolving pool to further evolve. See details in (Appendix 5.11).

5.8 Reinforcement Learning

We use PPO [134] as the reinforcement learning method to train the control policy and use PytorchRL [136] for implementation. We use two multi-layer perceptrons(MLP) each with two linear layers for the actor and critic network.

Observation

The observation is calculated at every simulation time step to inform the controller the state of the truss. As mentioned above, a state vector is $s_t = \{V_t, U_t, V_t', U_t', k_t\}$. Kinematics information in the model space is location invariant and gives controller less noise than the information in the world space [137]. To calculate the relative node positions and velocities, for every truss, we define a beam in the middle as the center beam. The center beam determines the front orientation q_c as well as the center location p_c of the truss, which we use to calculate the relative positions V_t' and velocities U_t' by translating and rotating V_t, U_t based on q_c, p_c.

Action

At each time step, an action vector is generated from the controller and sent to the simulator. For a truss with n_γ channels, action is a Boolean vector with n_γ digits. The action vector is encoded as a binary value to a decimal value within $[0, 2^{n_\gamma})$. Accordingly, The output linear layer has n_γ outputs. When using deterministic policy, PPO takes the largest output as the decimal and converts it to the action vector.

Reward

Rewards are calculated by customized reward function. For each task, once the simulation reaches a maximum time steps or a target objective is achieved, the simulation finishes and gives a task specific reward. Before finishing, simulator gives 0 reward. For example, the distance of locomotion and height change are rewards for moving forward and lowering bodies. For turning left and tilting the top, a negative angle difference will be calculated based on the target and final orientations.

5.9 Preliminary Result

We demonstrate the usability and the performance of our method with a robotic table. We envision a scenario where every piece of furniture and daily objects can be robotic and have extra morphing functionalities in addition to their conventional static state. Here, a robotic table is trained to walk towards people who have limited mobility, lower itself for kids, or tilt the top in an angle for painters. in order to achieve the target design features, four objective functions were defined during the training process: moving forward, turning left by 90°, lowering the height, and tilting the tabletop. We design the truss structure of the table and fix the relative position of the top four nodes to mimic a fixed tabletop in the simulator.

We first use GA to optimize (C, ξ) for 1000 generations. We then pick a C from the last generation and train RL for 600 model updates. The experiment is run on a server with 64 cores for 2.4 hours. Figure 5.6 shows that GA effectively optimizes the design towards four objectives. Although RL improves the performance only by a small value, control policy

by RL is adaptable to a wider range of initial condition and is able to adjust the control on the fly to minimize the accumulated error. Further research needs to be conducted to test the generalizability of RL under different initial conditions, terrain conditions and external disturbance. We visualizes the design and the trajectory of the robotic table for four tasks using the training result by RL. (Figure 5.4).

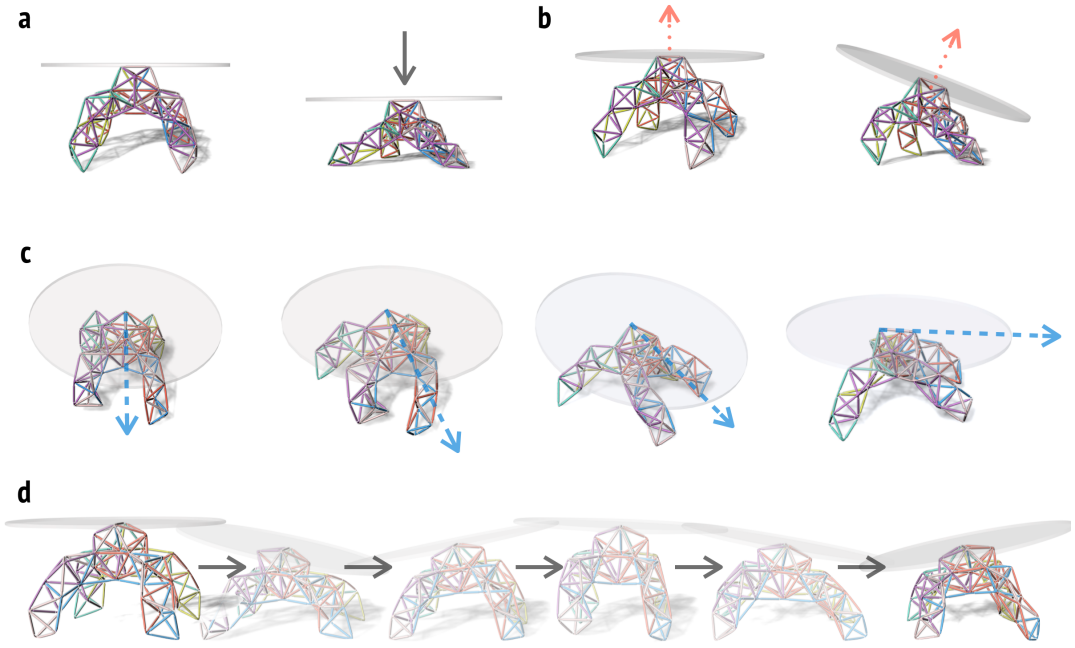


FIGURE 5.4: Four motions of a robotic table, (a) lowering the height, (b) tilting the table top, (c) rotating, and (d) locomoting forward.

5.10 Discussion

The preliminary result shows that our method is able to generate useful channel designs and controls for a complex truss with multiple diverse objectives. An immediate next step is to test the generalization ability of the method. For example, testing the method with trusses of various shapes, diverse objective functions, varying terrain conditions and external disturbance. To improve the generalization ability, we can apply graph convolutional neural network [138] to utilize the underlying graph topology information. Currently, the initial truss topology is pre-designed, but it might not be the optimal topology for the given tasks. To further improve the truss topology, a generative design such as a graph heuristic search [133] might optimize the truss topology together with the design and control.

5.11 Appendix: Genetic Algorithm Pipeline

In the first exploitation step, the evolving pool of n_g designs (C, ξ) are generated through an *initialization* function, which either randomly creates (C, ξ) pairs or duplicate from the elite pool if it is full. Ratings are generated through the *evaluation* function which simulates the truss with C, ξ. Based on the ratings, a *selection* (Appendix 5.13) function sorts and keeps a subset of the population to maximize the general population performance as well as the diversity.

In every other exploitation step, the preserved population will be duplicated and mutated to fill the evolving pool through a *mutation* (Appendix 5.12) function before the *evaluation*.

At the end of an exploitation loop, the preserved population will be added to an elite pool which has the same maximum size as the evolving pool. When the elite pool is full, elite population will be duplicated for initializing the evolving pool and the elite pool is cleared. After an exploitation step, the better elites will be selected and put back to the elite pool.

In such a way, the existing elite pool is continuously improved while the new population will have the chance to improve in each exploitation loop before being defeated by the elites.

5.12 Appendix: Initialization and Mutation with Channel Connection Constraint

Different from traditional GA where initialization and mutation are giving random value to randomly chosen digits, the digits in this problem are the channel assignment on beams, which is constrained by the channel connection rule. Beams of the same channel must be connected through joints and a random change might separate a channel and renders the solution impractical. Therefore, we use a customized channel-growing algorithm to initialize and mutate the solution.

At the initialization stage (Appendix 5), the algorithm assigns n_c edges with different channels. To assure the connectivity, a self-mirrored channel will be initialized with one self-mirrored edge. The algorithm keeps track of all the unassigned edges that are connecting to the assigned edges as E_{uc}. It randomly chooses one edge from E_{uc} and randomly assigns a channel γ that it is incident to the edge. The previous step is repeated until all the edges are assigned.

The *mutation* function (Appendix 6) is similar to *initialization*. It keeps track of all the edges that are connected with more than one channel as E_c. Every time, it selects one edge e_c from E_c and picks one different incident channel $\bar{c}$ and reassigns $\bar{c}$ to e_c. It checks the connectivity of all the subgraphs with breadth-first search and reverts the reassignment if the connectivity is broken. It repeats the previous step until finding an assignment that does not break the connectivity.

5.13 Appendix: Multi-Objective Selection Function

We adopted NSGA-II [135] as the selection function to keep the multi-objective performance.

This method first preserves solutions at the front hypersurface where no solution is worse than another solution on all objectives. Within the front hypersurface, a crowdedness distance(CD) is calculated based on how crowded the solutions are in the local performance space. If the total number of preserved solutions exceeds the threshold, the solutions with lower CD values will be preserved. Different from multi-objective functions with weighted sum that might lead to solutions that are not good at any objectives, this method improves the ratings of each objective while keeping the diversity of the genes. Meanwhile, It does not require a weight assignment to every objective.

5.14 Appendix: Elite Pool Mechanism

Every fixed number of generations, the population remaining on the front hypersurface will be added to an elite pool, and the population will be cleared and reinitialized. When the elite pool reaches the size of the population, the population in the elite pool will be put back to the population. This mechanism allows GA to explore more types of solutions by

re-initialization but at the same time exploit the best solutions by repetitively mutating the preserved population.

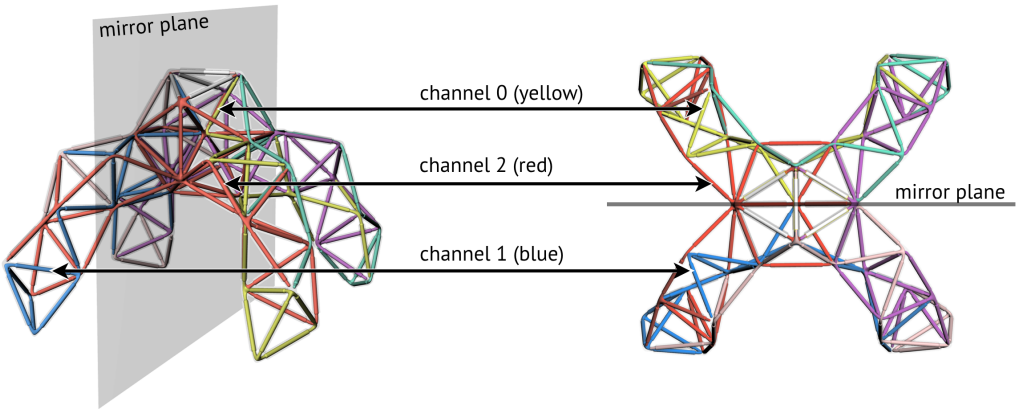


FIGURE 5.5: Channel symmetry constraint. Left: Perspective view of the truss of a robotic table. Right: Top view of the robotic table truss. Channel 0 and channel 1 are symmetric to each other with regard to the mirror plane. Channel 2 is symmetric by itself.

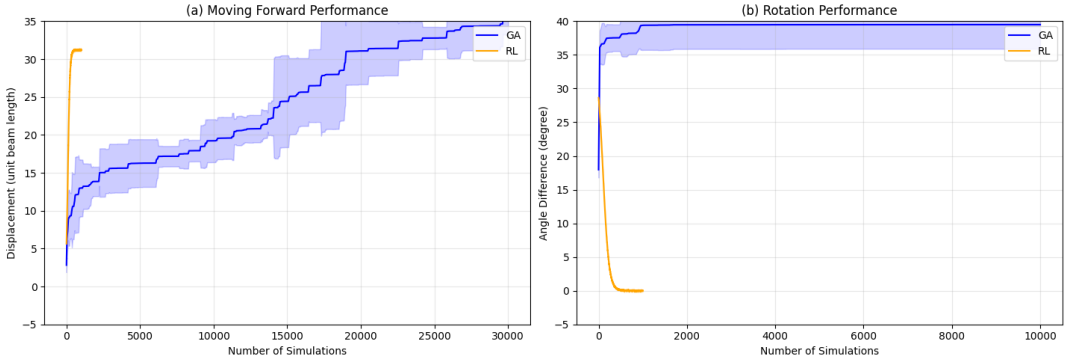


FIGURE 5.6: Performance comparison between Genetic Algorithm (GA) and Reinforcement Learning (RL) approaches for soft robot control tasks. Blue curves show the GA performance with characteristic evolutionary behavior, where the population-based optimization leads to larger performance variations and stepwise improvements. Orange curves represent the RL performance, demonstrating faster convergence and more stable learning. Shaded regions indicate the standard deviation across multiple runs ($\pm 1\sigma$ for RL, $\pm 2\sigma$ for GA). Two tasks are tested: (a) Moving forward, where displacement is measured in unit beam length. RL achieves superior performance (31.2 units) within $1,000$ simulations, while GA converges to a lower performance (27 units) after $20,000$ simulations. (b) Rotation task, where the angle difference decreases from 35° to 0°. RL completes the task in 354 simulations, significantly outperforming GA which requires $7,800$ iterations. The results demonstrate RL's superior sample efficiency and performance compared to GA for these soft robot control tasks.

Algorithm 5 Initialization

n_γ: number of channels
Total channels $\Gamma = \{\gamma_0, \gamma_1, ...\gamma_{n_\gamma - 1}\}$, where $\gamma_i = i$ for simplicity.
Total channels of a halfgraph $\hat{\Gamma} = \{\hat{\gamma}_0, \hat{\gamma}_1, ...\gamma_{\hat{n}_\gamma - 1}\}$
$\phi : \gamma \mapsto \gamma_m$, where γ_m is the channel mirrored to γ.
$E = \{e_0, e_1, ...e_{n_e - 1}\}$, where $e_i = v_{i0}, v_{i1}$
$\hat{E} = \{\hat{e_0}, \hat{e_1}, ...\hat{e_{n_e - 1}}\}$, edges in the half graph.
$\psi : e \mapsto e_m$, where e_m is the edge mirrored to e.
$\hat{E_u}$: edges on the half graph with no channel assignment.
E_m: edges that are self-mirrored.
$C = c_0, c_1, ...c_{n_e - 1}$, $c_i \in \Gamma$, the channel assignment of each edge, initialized with -1.
for $\gamma = 0$ **to** $n_e - 1$ **do**
    $\gamma_m \leftarrow \phi(\gamma)$
    if $\gamma_m == \gamma$ **then**
        select e from $\hat{E_u} \cap E_m$
        $c_e \leftarrow e$
        select e from $\hat{E_u} \cap (\hat{E} - E_m)$
        $e_m \leftarrow \psi(e)$
        $c_e \leftarrow \gamma$
        $c_{e_m} \leftarrow \gamma_m$
        remove e from $\hat{E_u}$
        while $\hat{E_u}$ is not empty **do**
            $\hat{E_{adj}} \leftarrow$ edgesConnectingChannels(C, $\hat{E}$)
            select e from $\hat{E_{adj}}$
            $\Gamma_{inc} \leftarrow$ channelsIncidentEdge(C, e)
            select γ from Γ_{inc}
            $c_e \leftarrow \gamma$
            remove e from $\hat{E_u}$

Algorithm 6 Mutation

$\hat{E_c} \leftarrow$ edgesConnectingMultipleChannels(C, $\hat{E}$)
while True **do**
    select e from $\hat{E_c}$
    $\Gamma_{inc} \leftarrow$ channelsIncidentEdge(C, e)
    for γ in Γ_{inc} **do**
        if channelsConnected(C, $\hat{E}$, e, *gamma*) **then**
            $\gamma_e \leftarrow \gamma$
            BREAK

Chapter 6

Truss Topology and Parameter Generation with Variational Auto-encoders

While transformable truss robots offer remarkable potential for shape-morphing applications, optimizing their design remains a significant challenge due to the complex interplay between discrete topology and continuous geometric parameters. To address this challenge, we develop two complementary neural network approaches. Our first solution, a Graph Attention Network-based Variational Autoencoder (GAT-VAE), effectively optimizes C-network connectivity within fixed topologies, demonstrating exceptional accuracy in both reconstruction and connectivity prediction. To overcome the limitations of fixed topology optimization, we then introduce a Long Short-Term Memory-based VAE (LSTM-VAE) featuring a novel truss grammar that can represent loop-containing structures through a merging operation. This grammar enables simultaneous optimization of topology and continuous parameters while supporting direct shape optimization through a position prediction network. Both approaches achieve state-of-the-art performance in their respective tasks and exhibit smooth interpolation capabilities in their latent spaces. Together, they provide a comprehensive framework for truss robot optimization while suggesting promising directions for combining their complementary strengths in future work.

6.1 Introduction

While optimizing edge and node values within fixed truss topologies has proven effective using tailored genetic algorithms (GA), the fundamental challenge lies in generating optimal topologies themselves. Current truss topologies, designed by humans using conventional design tools, may not be ideally suited for specific functional objectives like locomotion or manipulation. For instance, a truss designed to resemble a quadrupedal form might not be optimal for tasks like squeezing through narrow spaces or climbing, where unconventional morphologies could prove more effective. This limitation of human design becomes particularly apparent in multi-objective scenarios, highlighting the need for algorithms that can optimize or generate topologies themselves, rather than merely optimizing parameters within a given topology.

Furthermore, while current genetic algorithms are flexible enough to handle discrete parameters like C-network indices through well-designed operators for mutation and crossover on discrete graph structures, they become inefficient as the search space scales up. Although GA can explore for global optima and is less likely to get trapped in local minima, its efficiency diminishes with increasing complexity.

There is a clear need for more efficient methods to navigate through graph topologies, which are inherently discrete and non-Euclidean. While graphs offer advantages in their

compatibility with computer data structures and their computational efficiency due to sparsity, these same characteristics present significant optimization challenges. Their discrete nature and non-Euclidean geometry mean that traditional methods of continuous optimization are not directly applicable. Similarly, approaches that convert continuous to discrete optimization, while effective for grid structures, do not translate smoothly to graph structures. This mismatch arises from the fundamental properties of graphs, necessitating novel approaches specifically tailored to graph topology optimization.

The advancement of neural networks and machine learning presents a viable solution. By employing neural networks as encoders and decoders, Variational Autoencoders (VAE) can transform discrete data and graph structures into continuous latent representations and then decode them back into discrete form. Unlike standard autoencoders that directly decode from latent vectors, VAE learns a distribution and samples latents from the latent distribution. This ensures the VAE latent space grows into a smooth and disentangled space suitable for continuous optimization.

Researchers have used grammar to represent limbed robots and have employed neural networks to encode, generate, and optimize topology and morphology [7, 8, 78]. However, these limbed robots are based on a tree topology, where each token represents a limb branching from an existing body. This type of topology fits well for most animals and limbed robots, and its hierarchical nature resembles natural language and can be well captured by language models such as transformers [79, 139, 140]. However, the volumetric shape deformation inherent in truss structures often requires loop topology, which cannot be simply represented by tree-based token grammar.

In this work, we present two complementary approaches to address these challenges, each targeting different aspects of truss optimization (6.1). 1). Our first approach employs a Graph Attention Network (GAT) based VAE for optimizing C-network assignments within fixed topologies. We initially focus on the challenging problem of C-network connectivity optimization, where GAT's natural ability to process graph structures makes it particularly suitable. Through attention mechanisms, the GAT-VAE captures both local edge relationships and global structural features of the truss design. This approach demonstrates remarkable effectiveness, achieving 99.925% reconstruction accuracy and 99.999% connectivity prediction accuracy in our experiments. The model successfully optimizes truss designs for maximum aspect ratio with a mean square error of only 0.036, while showing smooth interpolation capabilities between different C-network assignments while maintaining connectivity constraints.

However, while GAT excels at fixed-topology optimization, it cannot easily handle varying graph structures. 2). To address this limitation, we develop our second approach: a Long Short-Term Memory (LSTM) based VAE with a novel truss grammar. This grammar introduces a unique way to represent truss topologies as sequences, critically supporting loop structures through a merging operation. The LSTM-VAE simultaneously optimizes topology, C-network indices, initial lengths, and contraction ratios, while an additional position prediction network enables direct shape optimization. This comprehensive approach achieves 99.43% token reconstruction accuracy and, most importantly, demonstrates successful multi-shape optimization where a single truss can transform between multiple target shapes under different actuation states. By directly setting shape objective functions on the predicted shapes, we enable gradient-based optimization in the latent space, followed by post-optimization parameter refinement of the decoded continuous parameters to further approximate objective shapes.

These approaches complement each other while addressing different aspects of the challenge: the GAT-VAE provides deep understanding of C-network connectivity within fixed topologies, while the LSTM-VAE enables exploration of the broader design space including

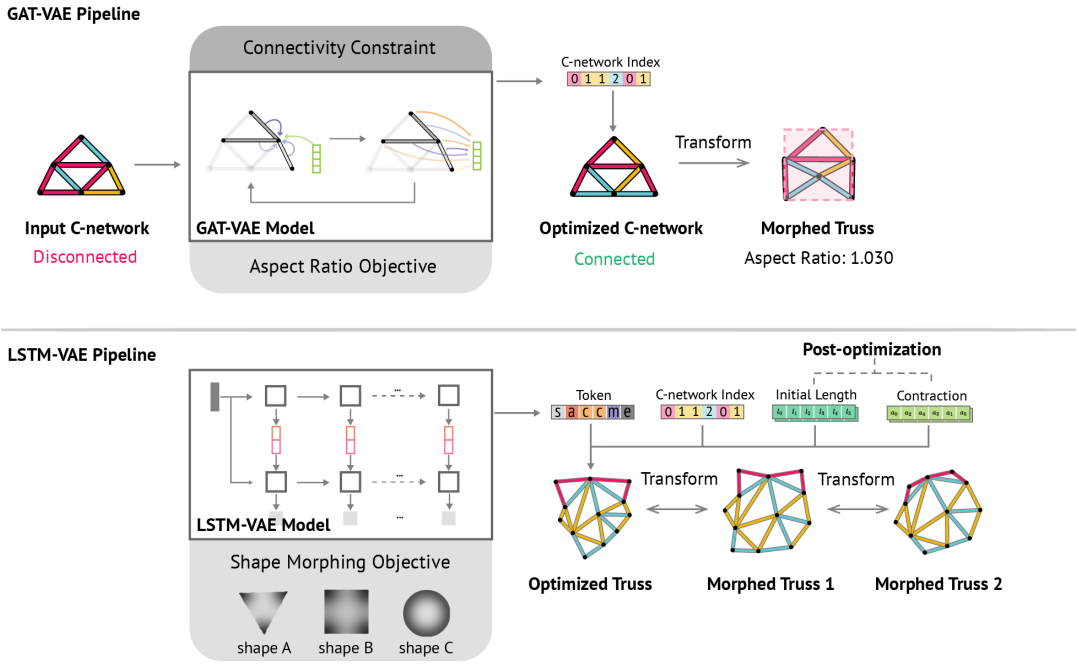


FIGURE 6.1: **Introduction. GAT-VAE Pipline** begings with an input truss topology and an initial input C-network assign. The GAT-VAE model generates the optimized C-network indices. The optimized C-network follows the connectivity constraint, and its morphed shape has a maximized aspect-ratio. **LSTM-VAE Pipeline** only takes in one or multiple objective shapes. The LSTM-VAE model generates optimized truss design with novel topology tokens, C-network indices, initial lengths and contraction ratios. A continuous post-optimization is applied on the initial lengths and contraction ratios to further refine the design. Eventually, the resulting truss robot is able to transform between multiple shapes to approximate objective shapes.

topology variation. Based on our knowledge, this is the first time an entire truss robot design can be generated from scratch with continuous optimization.

A promising future direction would be to combine these approaches, leveraging GAT's strength in understanding local and global connectivity patterns with LSTM's capability in handling variable topologies through sequential generation. While we focus on 2D trusses to explore the potential of our grammar and VAE approaches, the grammar can be straightforwardly extended to 3D truss structures, presenting an exciting direction for future work. Together, they currently provide a comprehensive framework for truss robot optimization, demonstrated through extensive experiments in shape morphing and multi-objective optimization, while pointing toward even more powerful hybrid approaches for future work.

6.2 C-network Optimization through Graph-Attentive Variational Autoencoder

Graph neural networks pass information through edges and vertices, enabling them to capture both topological and geometrical information of graph structures. Through the integration of attention mechanisms and global latent features, graph attention networks (GAT) can learn both local information and global features of a graph. This capability has led to their

widespread application in physics-informed systems with underlying graph structures, such as limbed robots and solar systems.

While genetic algorithms (GA) have proven effective for optimizing discrete parameters like C-network indices through well-designed mutation and crossover operators on graph structures, they become computationally inefficient as the search space grows [7, 8, 78]. Optimizing C-network indices while maintaining connectivity is particularly challenging, as a single misassignment can break the network's connectivity. This task requires both local understanding of edge relationships and global comprehension of network structures, making it difficult for traditional optimization methods. The discrete nature and non-Euclidean geometry of graphs further complicate the application of traditional continuous optimization methods, necessitating novel approaches specifically tailored to graph topology optimization. Our experiments demonstrate that GAT-VAE excels at this task, achieving 99.925% reconstruction accuracy and 99.999% connectivity prediction accuracy, while maintaining a mean square error of only 0.036 for aspect ratio prediction.

Truss robots exhibit both local and global information structures. Local information includes the C-network connections between incident edges and the relative positions between adjacent vertices. Global information, which can be derived from local information, encompasses the overall connectivity of C-networks and the absolute positions of vertices. By leveraging GAT's natural ability to process graph structures and capture both local edge relationships and global structural features, we can effectively optimize C-network assignments within fixed topologies.

In this section, we focus on reconstructing the C-network indices with given topology and equal initial edge length and contraction ratios. We adopt GAT as both the encoder and decoder of a Variational Autoencoder (VAE), optimizing the latent space to maximize C-network connectivity while considering the aspect ratio of the truss after transformation. Unlike standard autoencoders that directly decode from latent vectors, VAE learns a distribution and samples latents from this distribution, ensuring the latent space grows into a smooth and disentangled space suitable for continuous optimization.

6.2.1 Data and dataset

We use 2D truss topologies containing between 1 and 8 triangles as our base structures, generating 10,000 distinct topologies. To simplify the problem, we set all edges to have equal initial length and contraction ratio.

For each topology, we randomly assign C-network indices to create different C-network configurations. A C-network assignment is considered connected if all edges with the same C-network index form a connected graph component (Fig. 6.2 b). Through this random assignment process, we expand our dataset to 10,000 data points, with approximately 76% being disconnected configurations and 24% being connected configurations (Fig. 6.2 b).

Since each truss robot has three C-networks, there are eight possible actuation states (2^3) corresponding to each C-network being either actuated (contracted) or not. For each C-network assignment, we first assess its connectivity. Then, for connected configurations, we simulate all eight transformed shapes by actuating different combinations of C-networks. From these eight configurations, we extract the maximum aspect ratio, defined as the height-to-width ratio of the bounding box of the morphed shape (Figure. 6.2a). This ratio serves as a metric for the truss's shape-morphing capability.

To summarize, our dataset consists of 10,000 data points, with each point containing:

- Initial vertex positions
- C-network indices for each edge

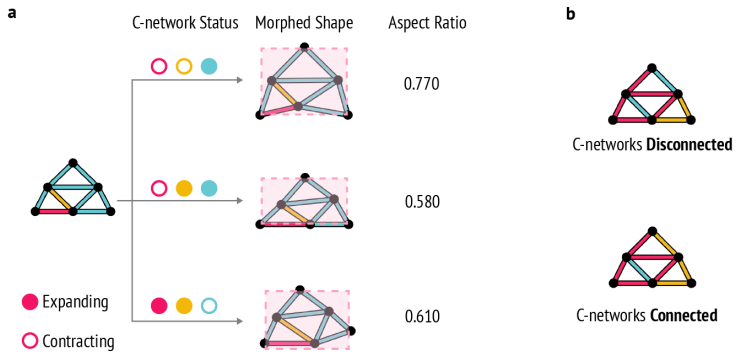


FIGURE 6.2: **Aspect ratio and connectivity. a.** Each truss robot with three C-networks has eight C-network actuation states, thus having eight transformed shapes based on the state. The height over width ratio of the bounding box of the morphed shape is called the aspect ratio. **b.** A C-network assignment is defined as disconnected, if any C-network is not a connected graph.

- A boolean value representing connectivity
- Maximum aspect ratio among all eight transformed shapes (for connected configurations)

We split this dataset into training (80%) and testing (20%) sets for model evaluation.

6.2.2 GAT-VAE Model

We develop a VAE architecture using Graph Attention Networks (GAT) as both encoder and decoder structures (Figure. 6.3).

Encoder: The encoder consists of two embedding layers: one converts each 2D vertex position $p_v \in R^2$ into initial vertex embedding e_v^0, and another transforms the one-hot vector of C-network indices $c_e \in \{0,1,2\}$ into initial C-network embedding e_c^0. A learnable embedding e_g^l initializes the global embedding. The embedded graph is then processed through the GAT encoder network for n_g iterations. The resulting global embedding is passed through two fully connected layers to produce the mean latent vector μ and the standard deviation latent vector σ. A latent vector e_g^z is sampled from the normal distribution $\mathcal{N}(\mu,\sigma)$.

Decoder: The decoder takes as input the same truss topology without C-network assignments. Vertex positions are embedded into initial vertex embeddings, while C-network embeddings are initialized with a learnable embedding e_c^l. The global embedding is set to the sampled latent vector e_g^z from the encoder. After n_g iterations of the GAT decoder network, the decoded vertex embedding e_v^{decode} and decoded C-network embedding e_c^{decode} are reconstructed into C-network indices using softmax followed by argmax operations.

GAT Update: Each GAT update iteration comprises three message-passing processes:

1. **Vertex Update:** For each vertex v, the C-network embeddings of all incident edges $\{e_{ci}^n\}_{i\in\mathcal{N}(v)}$ and the vertex embedding e_v^n are aggregated through a vertex attention layer:

$$e_v^{n+1} = \text{Attn}_v(\{e_{ci}^n\}_{i\in\mathcal{N}(v)}, e_v^n) \tag{6.1}$$

2. **C-network Update:** For each edge e, the vertex embeddings of its two incident vertices $\{e_v^{n+1}{}_0, e_v^{n+1}{}_1\}$ and the C-network embedding e_c^n are aggregated through a C-network

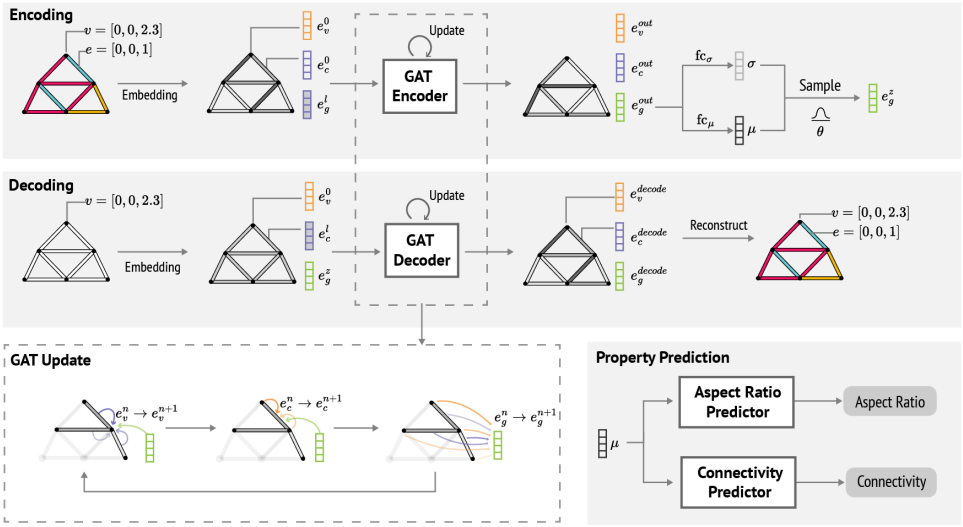


FIGURE 6.3: **GAT-VAE Model** is composed of the encoding module, decoding module and property prediction module. The **encoding module** takes in a truss design and converts it to a latent vector through GAT updates. The **decoding module** takes in a truss design without C-network information and a latent vector, and uses the same GAT update process to decode the latent vector. The decoded embeddings are reconstructed into the truss design. The **property prediction module** includes two multi-layer perceptron networks that takes in the latent vector and predicts the connectivity and maximum aspect ratio of the truss design.

attention layer:

$$e_c^{n+1} = \text{Attn}_c(e_v^{n+1}{}_0, e_v^{n+1}{}_1, e_c^n) \tag{6.2}$$

3. **Global Update:** All updated vertex embeddings $\{e_v^{n+1}{}_i\}$, C-network embeddings $\{e_c^{n+1}{}_i\}$, and the global embedding e_g^n are aggregated through a global attention layer:

$$e_g^{n+1} = \text{Attn}_g(\{e_v^{n+1}{}_i\}, \{e_c^{n+1}{}_i\}, e_g^n) \tag{6.3}$$

Two separate neural networks are trained to predict the connectivity and maximum aspect ratio. The mean latent vector μ serves as input to each network. Both networks consist of three fully connected layers with ReLU activation functions. A softmax function is applied after the connectivity predictor.

The total loss $\mathscr{L}$ consists of four components:

$$\mathscr{L} = \beta \mathscr{L}_{KL} + \mathscr{L}_{recon} + \mathscr{L}_{conn} + \mathscr{L}_{aspect} \tag{6.4}$$

where $\mathscr{L}_{KL}$ is the KL divergence between the encoded distribution and standard normal distribution weighted by β, $\mathscr{L}_{recon}$ is the reconstruction loss of C-networks, $\mathscr{L}_{conn}$ is the cross-entropy loss of connectivity prediction, and $\mathscr{L}_{aspect}$ is the mean square loss of maximum aspect ratio prediction.

6.2.3 Implementation

The GAT encoder and decoder each contain three attention layers to process local, edge-level, and global graph information. We set the number of GAT update iterations $n_g = 4$ to ensure

sufficient information propagation across the graph, noting that more complex topologies might require larger n_g values.

Training Schedule: The model is trained for a total of 2000 epochs with the following schedule:

- Batch size: Initially 128, reduced to 32 after epoch 200 to fine-tune learning
- Learning rate:
  - Initial: 1×10^{-3}
  - After epoch 50: 1×10^{-4}
  - After epoch 500: 1×10^{-5}
- KL divergence weight β:
  - Initial: 1×10^{-8}
  - Increased by factor of 10 when reconstruction accuracy reaches 99%
  - Maximum value: 1×10^{-2}

Dataset Split: We utilize 7000 data points for training (70%) and 2000 for testing (20%), maintaining a balanced distribution of connected and disconnected configurations in both sets.

Optimization: For latent space optimization, we employ the Newton-CG method from SciPy, which effectively handles the continuous optimization in the learned latent space while maintaining the constraints imposed by the VAE's learned distribution.

6.2.4 Performance

The model achieves exceptional performance across all key metrics:

- **Reconstruction Accuracy:** 99.925% on C-network assignments, demonstrating the model's ability to capture and reproduce complex graph structures
- **Connectivity Prediction:** 99.999% accuracy, indicating near-perfect understanding of global connectivity patterns
- **Aspect Ratio Prediction:** Mean Square Error (MSE) of 0.036, with a dataset mean maximum aspect ratio of 0.89

These results demonstrate that the GAT-VAE successfully learns both local and global features of truss designs. The high reconstruction accuracy indicates that the model effectively captures the discrete C-network assignment patterns, while the near-perfect connectivity prediction shows its ability to understand global structural properties. The low MSE in aspect ratio prediction (approximately 4% relative to the mean) suggests that the model accurately captures the geometric transformation properties of the designs.

The performance metrics also validate our choice of network architecture and training schedule. The gradual decrease in learning rate and increase in β allows the model to first focus on reconstruction accuracy before refining the latent space structure through the KL divergence term. This approach helps create a smooth, continuous latent space suitable for optimization while maintaining high reconstruction fidelity.

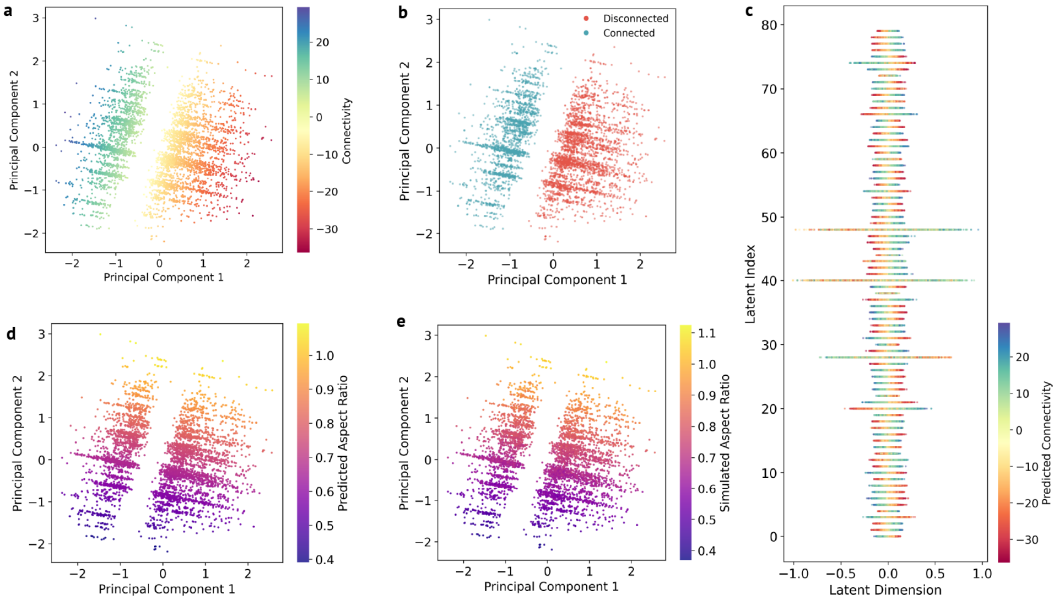


FIGURE 6.4: **GAT-VAE latent space visualization.** The latent vectors are projected to two principal dimensions using PCA. **a.** The predicted graph connectivity value shows a smooth transition mainly along the first principal component dimension. **b.** The actual label of graph connectivity shows that the model clearly and correctly separates the latent vectors into connected and disconnected categories. **c.** 80 out of 128 latent dimensions are visualized individually, showing most of the dimensions are learning meaningful information about the connectivity. **d.** The predicted maximum aspect ratio shows a smooth transition mainly along the second principal component dimension, perpendicular to the direction of connectivity prediction. **e.** The actual maximum aspect ratio is very close to the predicted maximum aspect ratio with small difference.

6.2.5 Latent Space Visualization

We analyze the learned latent space through Principal Component Analysis (PCA) to understand how the model organizes different aspects of truss designs. The visualization reveals a remarkably structured organization of the latent space (Figure 6.4).

When examining the connectivity encoding, we observe a clear separation between connected and disconnected designs primarily along the first principal component. The predicted connectivity values (Figure 6.4a) demonstrate a smooth transition across this axis, closely matching the actual connectivity labels (Figure 6.4b). This alignment validates the model's high connectivity prediction accuracy and suggests it has learned meaningful representations of structural connectivity.

To better understand the latent space's structure, we visualize 80 out of 128 individual latent dimensions (Figure 6.4c). The analysis reveals that most dimensions actively contribute to connectivity categorization, indicating an efficient use of the latent space without redundant dimensions.

The maximum aspect ratio visualization (Figure 6.4d,e) reveals another interesting pattern: the aspect ratio varies smoothly primarily along the second principal component, notably orthogonal to the connectivity direction. This orthogonality suggests the model has learned to encode geometric properties independently from connectivity features. The close

correspondence between predicted and actual maximum aspect ratios further confirms the model's ability to capture geometric transformation properties accurately.

6.2.6 Interpolation

To demonstrate the model's generative capabilities, we perform latent space interpolation between different truss configurations. Figure 6.5a showcases two distinct interpolation scenarios that highlight different aspects of the learned representation.

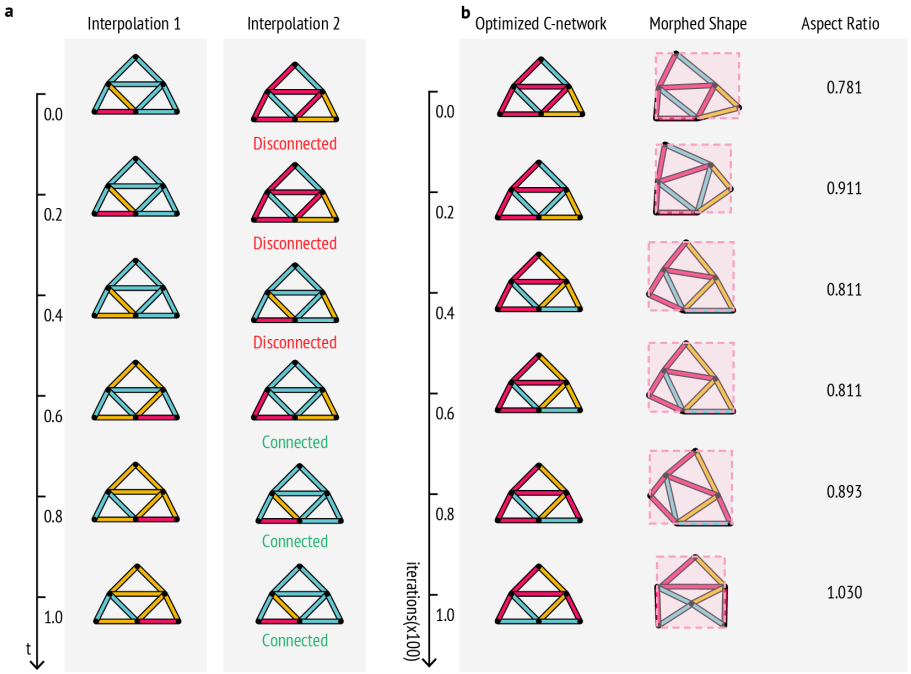


FIGURE 6.5: **GAT-VAE interpolation and optimizations. a.** The model is able to interpolate the C-network assignments between two trusses with the same connectivity (interpolation 1) or different connectivity (interpolation 2). **b.** The model optimizes the C-network assignment towards a larger maximum aspect ratio, which shows a continuous improvement on the aspect ratio of the transformed shape.

In the first scenario, we interpolate between two connected C-network configurations. We generate four intermediate designs through linear interpolation in the latent space, revealing a smooth transition that maintains connectivity throughout the path. The C-network patterns evolve gradually between the start and end configurations while preserving structural validity, demonstrating the model's ability to navigate the space of connected designs.

The second scenario presents a more challenging task: interpolation from a disconnected to a connected configuration. This transition demonstrates how the model has learned to "repair" disconnected configurations, showing a gradual improvement in connectivity as we move through the latent space. The smooth progression of designs suggests that the model has learned meaningful features about what makes a configuration connected, rather than treating connectivity as a binary property.

These interpolation results provide strong evidence that the VAE has learned a smooth and continuous latent space where neighboring points correspond to similar designs. The successful navigation between different connectivity states, while maintaining physical validity, confirms that the model captures meaningful features rather than merely memorizing training

examples. Furthermore, the smooth transition of both discrete (C-network assignments) and continuous (geometric) properties indicates that the model has developed a unified representation that handles both aspects coherently.

The clear organization shown in the latent space visualization, combined with the successful interpolation results, confirms that the GAT-VAE has learned a rich and well-structured representation of truss designs. This representation not only enables analysis of existing designs but also supports the synthesis of new ones through principled latent space exploration.

6.2.7 Shape Aspect-ratio Optimization

To validate the practical utility of our learned latent space, we investigate its effectiveness for shape optimization by focusing on the aspect ratio as a geometric objective. Our optimization process leverages the continuous nature of the latent space and the model's ability to predict geometric transformations accurately.

The optimization process begins with randomly selected C-network assignments from our test set. We first encode each design into its corresponding latent vector, which serves as the initial point for optimization. Using the Newton-CG optimizer in SciPy, we then navigate the latent space to maximize the predicted aspect ratio. The optimization takes advantage of the differentiable nature of our neural network, using gradients from the aspect ratio predictor to guide the search through the latent space.

Figure 6.5b visualizes the optimization trajectory through a series of decoded designs. The sequence demonstrates a continuous improvement in the maximum aspect ratio of the transformed shapes, with each intermediate design maintaining valid C-network connectivity. This smooth progression validates that our latent space captures meaningful geometric relationships and that the aspect ratio predictor provides useful gradients for optimization.

The success of this optimization approach highlights several key advantages over traditional methods like genetic algorithms. First, by operating in a continuous latent space rather than directly on discrete C-network assignments, we can use efficient gradient-based optimization techniques. Second, the neural network provides rapid feedback through its predictions, requiring simulation only for final validation rather than during the optimization process. This significantly reduces computational cost compared to methods that require full simulation for each candidate solution. Finally, the smooth nature of the learned latent space helps avoid the discontinuous jumps often encountered when directly modifying discrete graph structures.

These results demonstrate that our GAT-VAE not only learns to represent and reconstruct truss designs but also creates a latent space suitable for practical design optimization tasks.

6.2.8 Discussion

Our GAT-VAE approach demonstrates effectiveness in handling the challenging task of optimizing discrete categorical information on graph structures. The model shows particular strength in understanding and maintaining C-network connectivity while optimizing for geometric objectives, achieving high accuracy in both reconstruction and prediction tasks. This success highlights the potential of combining graph neural networks with variational autoencoders for problems involving both discrete and continuous optimization.

A key advantage of our approach lies in its computational efficiency. Traditional methods like genetic algorithms search through discrete C-network configurations directly, requiring full simulation for each candidate solution. In contrast, our model provides gradient-based optimization through the learned latent space, using the neural network for rapid evaluation and requiring simulation only for final validation. This efficiency becomes particularly valuable as the complexity of truss designs increases.

However, our current approach has a notable limitation: it operates only on fixed topologies. While the model excels at optimizing C-network assignments and predicting geometric transformations, it cannot generate new topological structures or modify existing ones. This constraint becomes particularly relevant when the optimal solution might require a different topological arrangement of triangles or edges.

Looking forward, one promising direction involves extending this work to handle variable topologies. A potential approach could use GAT to predict operations on edges, such as adding triangles, merging with incident triangles, or removing triangles. By encoding these operations in the edge embeddings, we could enable the model to generate distinct graph topologies in an auto-regressive fashion. This extension would significantly expand the design space accessible to our optimization approach.

6.3 Grammar-based Topology Generation through Sequential Variational Autoencoder

After successfully demonstrating that neural networks can capture topological and geometrical information in fixed graph structures through GAT-VAE, we faced a more ambitious challenge: could we teach neural networks to not just understand existing truss designs, but to create entirely new ones? This meant moving beyond simply optimizing C-network assignments to simultaneously crafting the topology itself, along with initial edge lengths and contraction ratios. However, we first needed to solve a fundamental problem: how could we represent a truss structure in a way that neural networks could both understand and generate?

Looking to the broader field of robotics and machine learning, we found that researchers had made remarkable progress in developing grammars for various discrete structures with hierarchical topologies. From limbed robots [7, 8, 78] to CAD models [79, 139] and molecules [140], these approaches shared a common thread: they all relied on having a unique "root" node from which the structure could grow. This root enabled representation as a sequence of tokens through familiar tree-traversal methods like breadth-first or depth-first search. But here is our central challenge - truss structures, with their distinctive looped graph topology, have no such natural starting point.

To address this challenge, we developed a novel truss grammar with a key innovation: a "merging" operation that allows us to represent any truss with a specified starting edge as a unique sequence of tokens. Rather than merely growing new elements atop existing ones, our grammar introduces the ability to connect the ends of neighboring branches, forming the loops that characterize truss structures. This approach successfully represents any truss with disk-like topology - that is, where the internal space of each element represents the truss geometry's internal space, and the resulting geometry is topologically equivalent to a disk. While this excludes certain exotic configurations (such as 2D trusses shaped like "O" or "∞"), it captures the vast majority of practical truss designs.

We implemented this representation using multiple long-short term memory (LSTM) networks as encoder and decoder models to reconstruct both topology and associated parameters. Additionally, we trained a separate LSTM network to directly predict eight sets of transformed vertex positions. This direct position prediction, compared to the aspect-ratio predictor in GAT-VAE, offers complete freedom in defining various shape objectives for transformation.

The trained VAE achieved high reconstruction accuracy on both training and test sets, while demonstrating meaningful and smooth interpolation within the latent space. To validate our approach, we present three compelling examples: one showing how generated truss designs can approximate a single objective shape, and two demonstrating the more complex task of transforming between two or three different objective shapes.

6.3.1 Truss Grammar

The truss grammar represents topology through a sequence of five distinct tokens (Fig. 6.6). Each sequence begins with a starting token **s**, which initializes the topology with a single edge. This edge is placed into an empty open edge queue (Fig. 6.6 a).

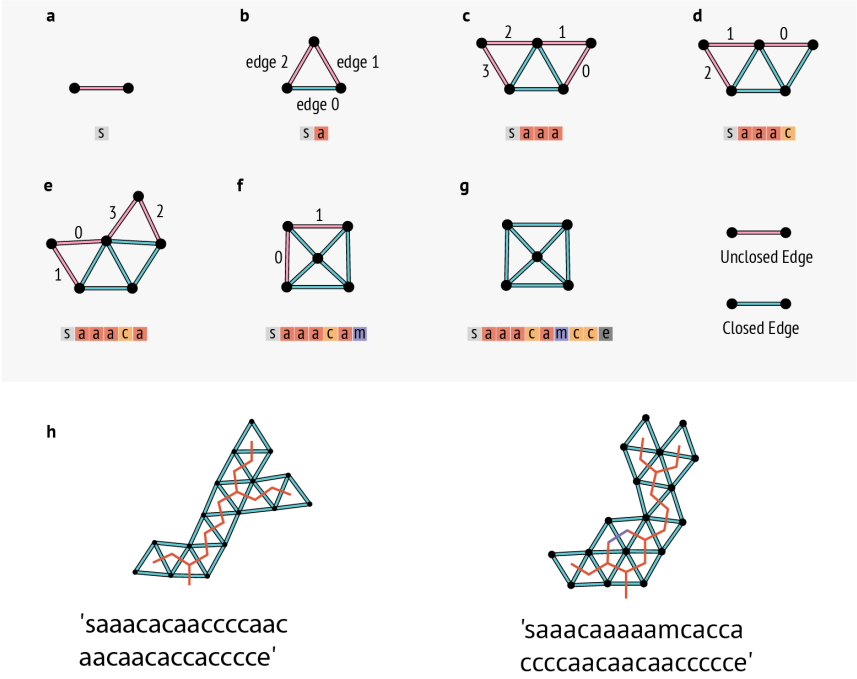


FIGURE 6.6: **2D Truss Grammar a-g.** Step-by-step demonstration of the usage of five truss topology tokens. **h.** Two examples of using a sequence of truss topology tokens to represent 2D trusses.

The generation process proceeds iteratively, with each step operating on an open edge popped from the queue. The sequence can contain four types of operational tokens:

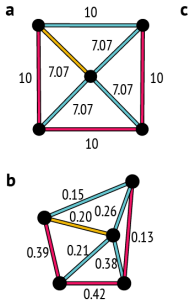
1. Adding token **a**: Creates a new triangle using the current open edge as its base. The newly formed triangle generates two additional edges, with the original open edge serving as the bottom edge. The three edges follow a specific numbering convention shown in Figure 6.6 b. Edge 1 and Edge 2 are automatically added to the end of the queue.

2. Closing token **c**: Marks the current open edge as closed, removing it from further consideration in the generation process.

3. Merging token **m**: Enables loop formation through edge merging. A merge operation is valid when the current open edge has an adjacent open edge and they share a vertex with more than one triangle. Upon merging, the two edges combine into a single edge, and the adjacent open edge is removed from the queue. Importantly, this operation does not create new triangles.

4. Ending token **e**: Terminates the sequence.

For each **a** token that creates new edges, the grammar assigns corresponding parameters: C-network indices, initial lengths, and contraction ratios. These parameters are represented as one-dimensional vectors aligned with the token sequence. For tokens that do not create new edges (**s**, **c**, **m**, **e**), the corresponding positions in these parameter vectors are filled with -1 (Fig. 6.7).



c

Tokens	s	a	a	a	c	a	m	c	c	e
C-network 1	1	1	0	1	-1	0	-1	-1	-1	-1
C-network 2	1	2	1	0	-1	1	-1	-1	-1	-1
Initial Length 1	10	7.07	10	7.07	-1	10	-1	-1	-1	-1
Initial Length 2	10	7.07	7.07	10	-1	7.07	-1	-1	-1	-1
Contraction 1	0.42	0.38	0.13	0.2	-1	0.15	-1	-1	-1	-1
Contraction 2	0.42	0.21	0.26	0.39	-1	0.2	-1	-1	-1	-1

FIGURE 6.7: **2D Truss Grammar parameters. a.** A truss before transformation initialized with C-network indices and initial edge lengths. **b.** A truss after transformation with each C-network contracting, with numbers showing the contraction ratio of each edge. **c.** Top: The topology tokens representing the truss in **a, b**. Bottom: Pairs of C-network indices, initial edge lengths, contraction ratios corresponding to each token.

The grammar naturally extends to three dimensions. In the 3D case, edges become faces, and triangles become tetrahedra. The sequence begins with an open triangle face, and the tokens operate as follows:

- **a**: Adds a tetrahedron to the current open triangle face - **c**: Closes the current open face - **m**: Merges the current open face with an adjacent open face if they share edges with more than one tetrahedron - **s** and **e**: Maintain their starting and ending functions

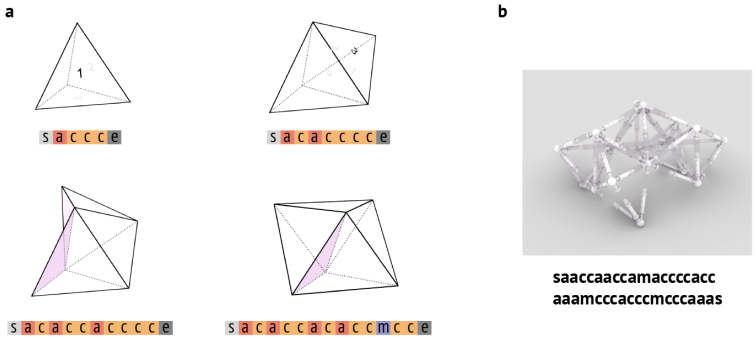


FIGURE 6.8: **3D Truss Grammar. a.** 3D truss shares the same set of topology tokens. The difference lies in that the objects of operation are changed from edges to triangles, and triangles to tetrahedrons. Adding token adds a tetrahedron instead of a triangle, and merging token merges two faces instead of two edges. **b.** A pillbug truss robot is represented by the sequence shown below the figure.

This systematic approach enables the representation of complex truss structures while maintaining consistent rules across both 2D and 3D domains (Fig. 6.8).

6.3.2 Data and Dataset

The dataset generation process consists of two phases: topology generation and parameter assignment.

For topology generation, we created 60,000 unique topologies through a controlled random sampling process of tokens **a**, **c**, **m**, **e**. For each topology, we created 10 variations, with random C-network, initial length and contraction ratios. Each topology sequence starts with an edge, and subsequent tokens are selected based on the following rules:

1. At each step, if the current edge permits a merge operation: Select randomly from **a**, **c**, **m**, **e** with equal probability

2. If merging is not valid for the current edge: Select randomly from **a**, **c**, **e** with equal probability

The sequence terminates under two conditions:

1. When an **e** token is selected and the resulting sequence is unique

2. When all edges are closed, in which case an **e** token is automatically appended

For parameter assignment, we generate 100 variations for each unique topology by randomizing three key parameters:

1. C-network indices: Selected from three possible networks

2. Initial lengths: Sampled from the range [10.0, 20.0]

3. Contraction ratios: Sampled from the range [0.8, 10]

For each parameter configuration, we simulate eight distinct actuation states and record the resulting vertex positions as ground truth data. These eight states correspond to all possible combinations of C-network actuations.

The final dataset consists of training set with 80% of the complete dataset and testing set: 20% of the complete dataset.

6.3.3 LSTM-VAE Model

The LSTM-VAE architecture leverages LSTM networks for both encoding and decoding, capitalizing on their ability to process sequential information. The model comprises three main components: the encoding module, the decoding module, and the position prediction module (Fig. 6.9).

Encoding

The encoding process begins with seven embedding layers that transform the input sequences. One layer processes topology tokens, while pairs of layers handle C-network indices, initial lengths, and contraction ratios respectively. Each embedding layer transforms its input into a 128-dimensional vector space, creating a unified representation dimension across the network. For each token in the sequence, these seven embeddings are concatenated to form a 7×128 dimensional vector. To handle variable-length sequences, all inputs are padded to a maximum length of 22 tokens, with padded positions masked during loss computation.

The Token encoder processes the token embedding sequence to generate encoded token embeddings at each step, outputting a token output vector from its final hidden state. The token embedding is concatenated with the other six embeddings to incorporate topological information into the encoding process, resulting in a combined embedding that preserves both structural and parameter information. Each specialized encoder (C-network, length, and contraction) processes this concatenated embedding to produce its corresponding output vector.

The seven output vectors are concatenated and processed through two fully connected layers to generate the mean latent vector μ and standard deviation latent vector σ. The latent vector z is then sampled from a Gaussian distribution parameterized by μ and σ using the reparameterization trick: $z = \mu + \sigma \odot \varepsilon$, where ε is sampled from $\mathcal{N}(0,1)$.

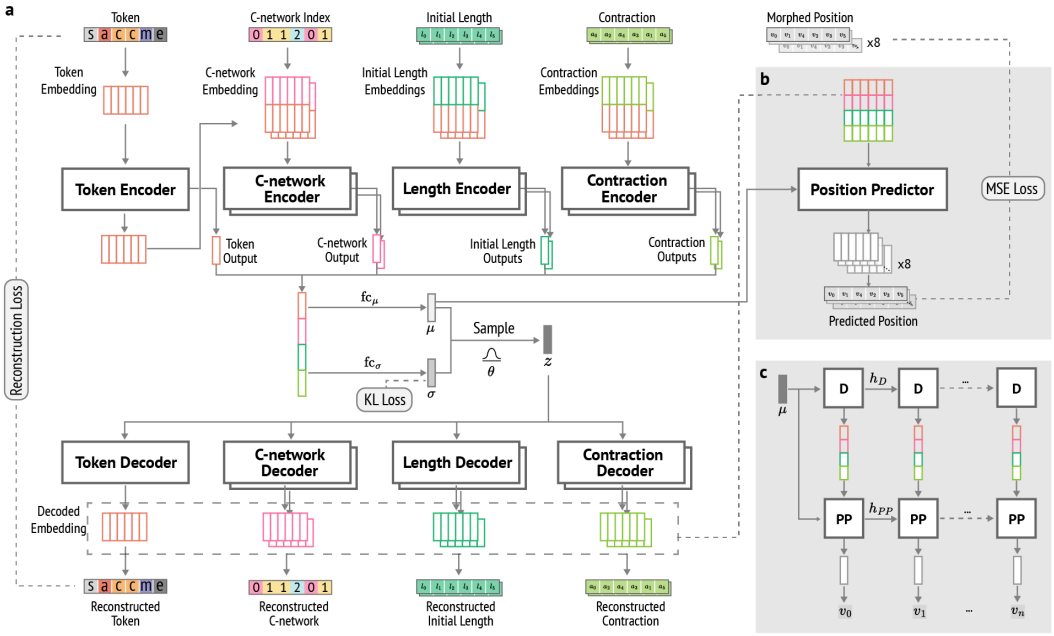


FIGURE 6.9: **LSTM-VAE model. a.** The variational auto-encoder structures has four encoder networks and decoder networks, each reconstructing the tokens, C-network indices, initial lengths, and contraction ratios of a truss design. The reconstruction loss is computed based on the input and decoded parameters from the decoders. A KL loss is calculated based on the mean and standard deviation latent vectors from the encoders. **b.** The mean latent vector is fed into the position predictor network to predict the eight sets of morphed vertex positions. An MSE loss is computed based on the predicted and simulated morphed vertex positions. **c.** Both position predictors (PP) and decoders (D) are LSTM networks. D generates the embedding sequentially, while PP takes in the embedding and predicts the vertex positions. Each time step corresponds to a generated token.

Decoding

The decoder consists of seven LSTM networks structurally identical to their encoder counterparts. Each decoder receives the sampled latent vector z as its initial hidden state, with temporal inputs initialized to zero. The decoders recursively generate their respective decoded embeddings. During training, teacher forcing is employed with a ratio of 0.5, meaning that 50% of the time the decoder uses its own previous output as input for the next step, and 50% of the time it uses the ground truth value.

The decoding process produces two types of outputs through different pathways. Token and C-network embeddings are transformed through fully connected layers into one-hot vectors matching the vocabulary size of tokens and C-network indices respectively, then converted to discrete values via argmax. Initial length and contraction embeddings are processed through fully connected layers to directly output floating-point values within their respective valid ranges.

Position Predictor

The position prediction module consists of eight parallel LSTM networks, each dedicated to predicting vertex positions for one of the eight possible actuation states. Each network

takes concatenated decoded embeddings as sequential input and uses the mean latent vector μ from the encoder as its initial hidden state. The networks process concatenated decoder outputs at each time step and transform their respective output position embeddings through fully connected layers to generate position predictions for their assigned actuation state.

Validity Predictor

To enforce token sequence validity constraints, we incorporate a validity predictor implemented as a multilayer perceptron (MLP). This component takes the latent vector as input and predicts sequence validity. The predictor trains jointly with the VAE to regulate the latent space and contributes to optimization through negative validity prediction in the loss term.

The total loss function $\mathscr{L}$ combines four components with adaptive weights:

$$\mathscr{L} = \lambda_1(t)\mathscr{L}_{KL} + \lambda_2(t)\mathscr{L}_{recon} + \lambda_3(t)\mathscr{L}_{conn} + \lambda_4(t)\mathscr{L}_{aspect} \tag{6.5}$$

where $\mathscr{L}_{KL}$ represents the KL divergence between encoded distribution and standard normal distribution, $\mathscr{L}_{recon}$ is the reconstruction loss for C-networks, $\mathscr{L}_{conn}$ denotes the cross-entropy loss for connectivity prediction, and $\mathscr{L}_{aspect}$ represents the mean square loss for maximum aspect ratio prediction. The weight coefficients $\lambda_i(t)$ follow a scheduling scheme where reconstruction weights start higher and gradually decrease while KL divergence weight increases, following common practice in VAE training. This scheduling helps establish stable reconstruction before enforcing latent space regularization.

6.3.4 Performance

The LSTM-VAE demonstrates good performance across multiple evaluation metrics on the test set. The model achieves a token reconstruction accuracy of 99.43%, indicating robust learning of the truss grammar syntax. For C-network assignments, the model attains reconstruction accuracies of 96.90% and 97.00% for the left and right channels respectively, demonstrating consistent performance across different network components.

Analysis of reconstruction accuracy across different structure sizes reveals an interesting pattern. For sequences shorter than 8 tokens, the accuracy drops to 92.02%. This lower performance on smaller structures can be attributed to their limited combinatorial space, which provides fewer examples for the model to learn the underlying patterns. As sequence length increases, the model's performance improves, suggesting better learning of structural patterns in more complex configurations.

The model also shows strong performance in reconstructing continuous parameters, achieving accuracies of 97.30% for initial lengths and 98.58% for contraction ratios. These results indicate the model's capability to simultaneously handle both discrete topological features and continuous geometric parameters.

The position prediction accuracy is evaluated by comparing the predicted vertex positions with the simulated positions from the physical engine. The model achieves an average position prediction error of 0.45 units, which is particularly noteworthy given that the average initial edge length in the dataset is 10.0 units. This represents a relative error of approximately 4.5% of the characteristic length scale, indicating high-fidelity reconstruction of the physical configuration space.

6.3.5 Interpolation

To demonstrate the model's ability to learn a meaningful latent representation, we examine the interpolation behavior between different truss designs. We first randomly select two

designs from the test set and encode them into their respective latent vectors. Linear interpolation is then performed in the latent space, with parameter $t \in [0,1]$ representing the interpolation weight between the source ($t = 0.0$) and target ($t = 1.0$) designs.

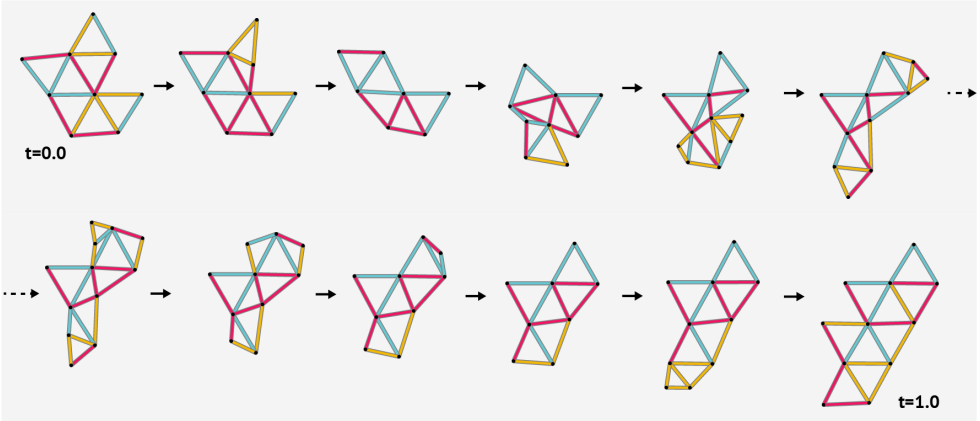


FIGURE 6.10: **LSTM-VAE latent space interpolation.** Two input truss designs (at $t = 0.0$ and $t = 1.0$) are encoded to two latent vectors. New latent vectors are interpolated evenly with t presenting the weight of the second input latent vector. The new latent vectors are decoded and shown in the order of their corresponding t values. The model is able to interpolate both continuous parameter, initial edge length, and discrete parameters, the topology and C-network indices in a visually smooth and meaningful way.

A key feature of our architecture is how different decoders interpret the same interpolated latent vector. While all decoders share the same latent input, they independently reconstruct their respective parameters - topology tokens, C-network indices, initial lengths, and contraction ratios. This separation allows each parameter type to transition smoothly in its own space while maintaining coherence in the overall design.

In our experiments, we observe an interesting pattern in how designs evolve during interpolation. Often, the topology remains stable in the early stages of interpolation while continuous parameters like initial edge lengths gradually adjust. For example, as shown in Figure 6.10, the first two interpolated designs maintain their topological structure while their initial edge lengths change to better approximate the target shape. This suggests that the model has learned to prioritize continuous adjustments before making discrete topological changes - a behavior that mirrors human design intuition where small adjustments are preferred over structural overhauls when possible.

The interpolation results reveal the VAE's capacity to capture both the discrete and continuous aspects of truss design in a unified latent space. By demonstrating smooth transitions between different designs, including changes in topology, C-network indices, and geometric parameters, the model shows its potential as a tool for exploring the design space of transformable trusses. This capability is particularly valuable for design exploration and optimization, as it allows continuous navigation through what is inherently a mixed discrete-continuous design space.

6.3.6 Multi-shape Optimization

The multi-shape optimization framework enables a single truss design to transform between multiple target shapes under different actuation states. The framework combines shape representation through signed distance fields with key point matching, allowing for precise control over both global shape and specific features (Figure. 6.11).

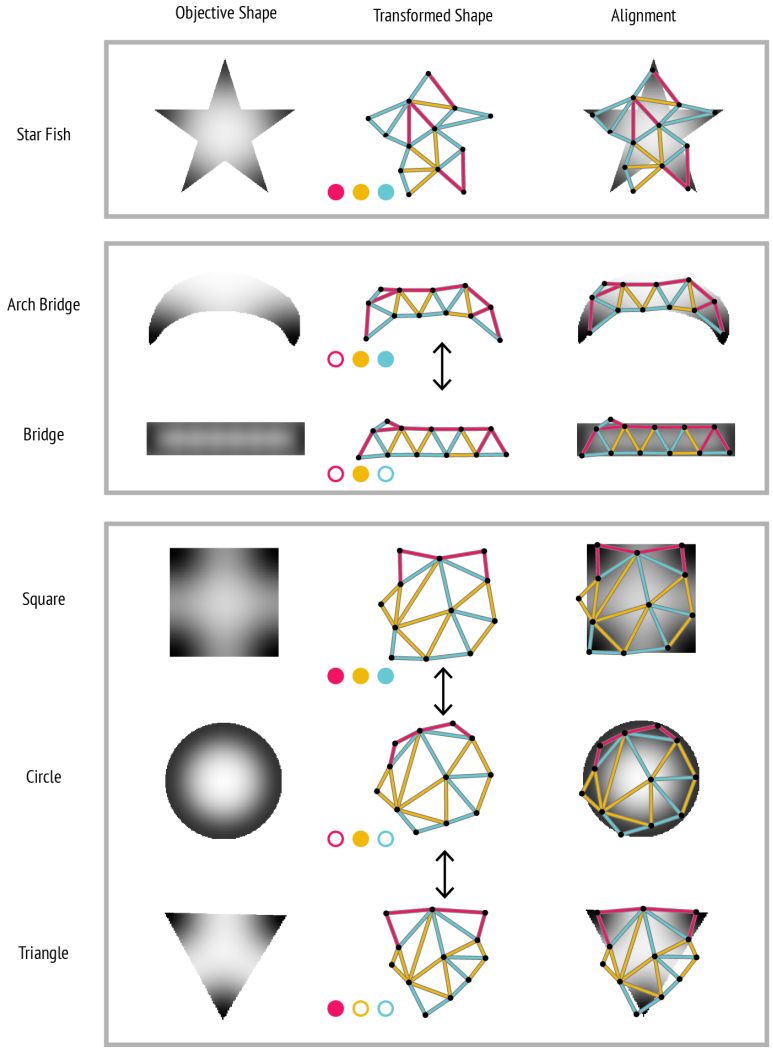


FIGURE 6.11: **LSTM-VAE shape optimization.** Left column: objective shapes, middle column: the optimized transformed shapes with the three dots showing the actuation states of the three C-networks, right column: the alignment between the transformed shapes and the objective shapes. Top-Bottom: Given one, two and three objective shape, the optimizer outputs one topology each time, which transforms into the single objective shape or transforms between two and three objective shapes.

Target Shape Representation

Our approach represents target shapes through two complementary methods. The primary representation uses a signed distance field (SDF) computed on a 256 × 256 grid, where

each pixel value represents the signed distance to the nearest boundary. Positive values indicate points outside the shape, negative values indicate points inside, and zero represents the boundary. The SDF is computed using fast marching methods from the shape's outline, providing a smooth, differentiable representation of the target geometry. Additionally, critical features of the target shape are represented through Gaussian distributions centered at key points, creating smooth gradient fields that guide precise feature matching during optimization.

Shape Loss Computation

Shape matching quality is evaluated through a dual-component loss function that considers both global shape and local features. The SDF loss component evaluates each vertex in the transformed truss against the target shape's SDF, naturally penalizing both under-coverage and over-extension. The key point loss component measures the alignment between transformed truss vertices and specified key points using the Gaussian gradient fields, ensuring accurate capture of critical features. These components are combined with appropriate weights to create a balanced objective that considers both global shape matching and local feature alignment.

Optimization Process

The optimization proceeds in two phases. The first phase optimizes in the latent space, where the primary parameter is the latent vector that the position predictor network maps to eight sets of predicted vertex positions. For each actuation state, we compute both SDF and key point losses, seeking designs that can effectively approximate one or more target shapes through different transformations.

The second phase refines the results through post-optimization. After decoding the optimized latent vector to a truss design, we maintain the discovered topology and C-network indices while fine-tuning the continuous parameters - initial lengths and contraction ratios - using gradient-based optimization to further improve shape approximation accuracy.

Experimental Results

To explore the full potential of our framework, we designed a series of increasingly challenging tests. We began with what appeared to be a straightforward task: could our system generate a truss that transforms into a single target shape such as a star. While conceptually simple, this challenge would validate the basic premise of our approach. The optimizer successfully generated a truss design that achieved the star configuration under the actuation state [0, 0, 0], demonstrating precise control over shape transformation.

We pushed the framework further: could a single truss structure encode two distinct target shapes. This represented a significantly more complex challenge, as the same physical structure needed to achieve two different configurations through combination of its actuation states. The framework produced a design that could smoothly transform between two distinct shapes while maintaining structural integrity throughout the transformation process.

The most complex test came in the form of triple-shape transformation. Here, the optimizer needed to discover a single truss design capable of achieving three different target shapes, a task that would be challenging even for human designers. The successful result demonstrated not just the framework's optimization capabilities, but its ability to uncover non-intuitive designs that might elude conventional design approaches. The truss could smoothly transition between all three target shapes through different actuation states, maintaining precise control over its geometry at each configuration. These progressive experiments revealed the framework's ability to handle complex shape-morphing objectives.

By comparing the optimized shape to the closest shapes in the dataset by finding the closest topology tokens, we found the optimized shapes are novel and visually different from any of the shapes in the dataset (Fig. 6.13 and Fig. 6.14).

6.3.7 Implementation

The LSTM-VAE architecture consists of three-layer LSTMs throughout the network. The embedding size, hidden size, latent vector size, and input size to the token encoder are all set to 128.

The training process follows a two-phase approach. In the first phase, we train on a mini-training set comprising 10% of the training data for 1,000 epochs, using a learning rate of 1×10^{-4}, batch size of 128, with σ set to 0.

The second phase extends to the full dataset for 2,000 epochs with dynamic parameter adjustments. The batch size starts at 128 and reduces to 64 after 1,000 epochs. The learning rate follows a decremental schedule: initial rate of 1×10^{-3}, decreasing to 1×10^{-4} after 100 epochs, and further reducing to 1×10^{-5} after 500 epochs.

For variational sampling, we apply a coefficient of 0.1 to the standard deviation σ. The KL divergence weight β begins at 1×10^{-8} and increases by a factor of 10 when all five reconstruction accuracy metrics exceed 95%, stabilizing at 1×10^{-2} for the remainder of training.

The optimization of the latent vector utilizes the BFGS algorithm from SciPy. The full training set includes 80% of the complete dataset.

6.4 Discussion and Conclusion

This work advances the optimization of truss robot design through two novel neural network approaches. The GAT-VAE effectively optimizes C-network connectivity and transformed shape properties within fixed topologies, achieving 99.925% reconstruction accuracy and 99.999% connectivity prediction accuracy, with a mean square error of 0.036 for aspect ratio prediction. The LSTM-VAE with our novel truss grammar enables simultaneous optimization of both discrete topological structure and continuous parameters, achieving 99.43% topology reconstruction accuracy while demonstrating successful multi-shape optimization capabilities. Both models show smooth and meaningful interpolation in their respective latent spaces, indicating they have learned meaningful representations of the design space.

Our approaches make several key technical contributions to the field. 1). We introduce a novel truss grammar that effectively represents loop-containing structures, extending beyond traditional tree-based grammars. 2). We develop a hybrid approach combining discrete topology optimization with continuous parameter tuning, creating an end-to-end differentiable framework for shape optimization in truss robotics. 3). Through our experiments, we demonstrate successful multi-shape optimization where a single truss can transform between multiple target configurations, validating the effectiveness of our approach.

While our results are promising, we identify several important limitations and corresponding future research directions. One promising direction is combining the strengths of GAT and LSTM architectures. While GAT shows excellent performance in understanding local and global connectivity, LSTM effectively processes sequential tokens. A natural next step would be to separate their functions, using LSTM solely for token reconstruction while delegating C-network indices, initial lengths, and contractions to GAT modules. This integration could potentially improve both reconstruction accuracy and optimization performance.

Furthermore, due to the limited contraction ratio and limited amount of data, there are certain target shapes are not achievable with current framework. For example, a target shape with too large contraction ratio required or curvature required is not possible (Fig. 6.15).

The extension to 3D structures presents another significant challenge. While truss grammar naturally extends to 3D by using faces and tetrahedrons as basic units rather than edges and faces, expanding the dimension significantly increases the possible design space of truss topology and transformed positions. Future work will require larger datasets and more sophisticated models specifically tailored to learning 3D truss structure representations. This dimensional expansion may necessitate architectural innovations to handle the increased complexity.

Current geometric validation also presents important limitations. The model implicitly learns truss validity from the dataset, with grammar enforcing topological validity. However, we still observe invalid geometry outputs, such as initial lengths or contractions outside their intended ranges and overlapping triangles in the resulting geometry. Future work should incorporate explicit physical constraints and validation mechanisms, possibly through differentiable physics simulation integration or explicit constraint enforcement in the loss function.

Optimization of latent space subsections presents another promising direction. Instead of optimizing the entire latent vector, we could use concatenated latent vectors where each segment contains information from specific encoders. This would enable selective optimization - for instance, if users are satisfied with the generated topology but need shape modifications, we could fix the token latent while optimizing other latent vectors, preserving topology while adjusting geometry.

Scaling to larger, more complex truss structures presents additional computational challenges that future work must address. This includes developing more efficient training strategies for larger datasets, exploring hierarchical representations for complex structures, and optimizing the inference pipeline for real-time applications. These improvements will be crucial for practical applications of our approach in real-world scenarios.

The success of our approaches in combining discrete and continuous optimization suggests broader applications beyond truss robots. These methods could extend to other domains involving graph-based structures with mixed discrete-continuous optimization challenges, such as molecular design, circuit layout, or architectural structures. Moreover, our truss grammar formulation might inspire new approaches to representing and optimizing other types of physical structures with complex topological constraints.

From a broader impact perspective, this work contributes to the growing field of automated design optimization in robotics and could potentially accelerate the development of more versatile and efficient robotic systems. However, care should be taken to ensure that automated design systems maintain physical feasibility and safety constraints in real-world applications. The integration of physical constraints and validation mechanisms will be crucial for translating these theoretical advances into practical applications.

6.5 Supplementary

The supplementary materials present further analyses of our model's capabilities and limitations in both latent space representation and shape optimization. Supplementary Figure 6.12 demonstrates the continuous interpolation between different truss designs in the LSTM-VAE latent space, showing smooth geometric transitions as the interpolation weight varies. Following this, we provide detailed comparisons between our optimized designs and the existing dataset to validate the novelty of our results. Supplementary Figures 6.13 and 6.14 showcase various optimized structures including star shapes and geometric forms, each presented alongside their closest matches from the training dataset to highlight the unique characteristics of our optimized solutions. Finally, Supplementary Figure 6.15 illustrates challenging cases where our approach encounters limitations in achieving the desired shape transformations, particularly for deformations requiring extreme scaling ratio or curvature.

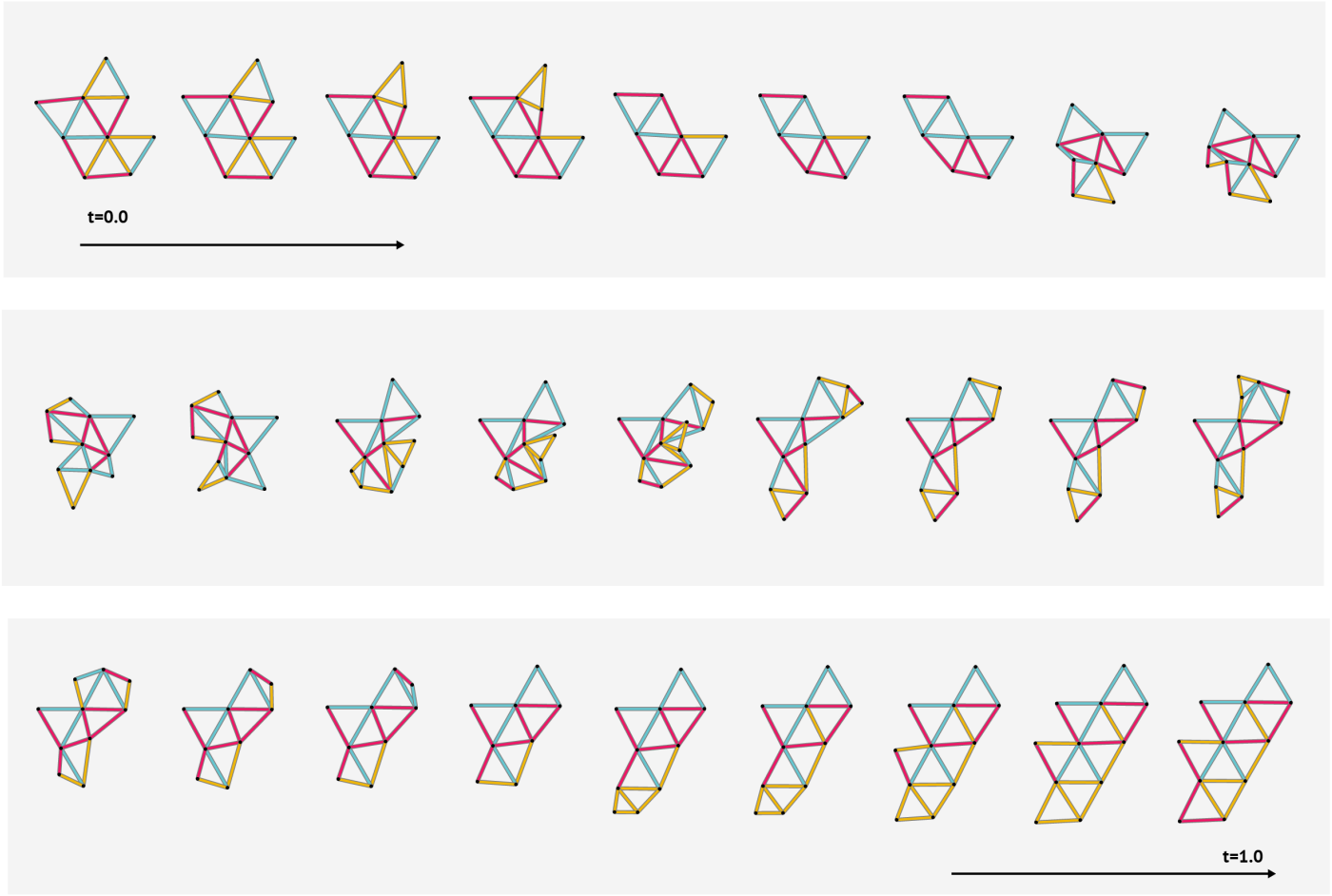


FIGURE 6.12: **LSTM-VAE latent space interpolation.** The trusses shown at the top-left and bottom-right corners represent the input designs. The t value indicates the interpolation weight applied to the latent vector of the bottom-right truss design.

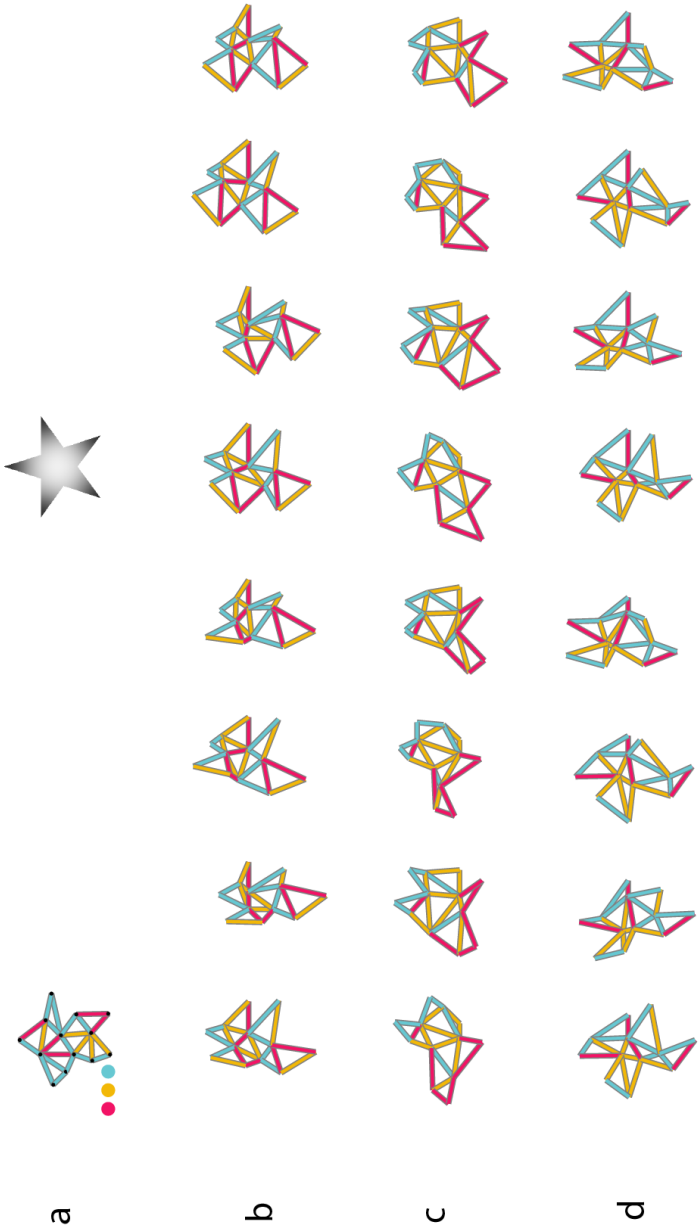


FIGURE 6.13: **Comparison of optimized star design with dataset examples. a,** The final optimized star shape and the target shape mask. **b-c,** The three closest data in the dataset with 8 transformed shapes.

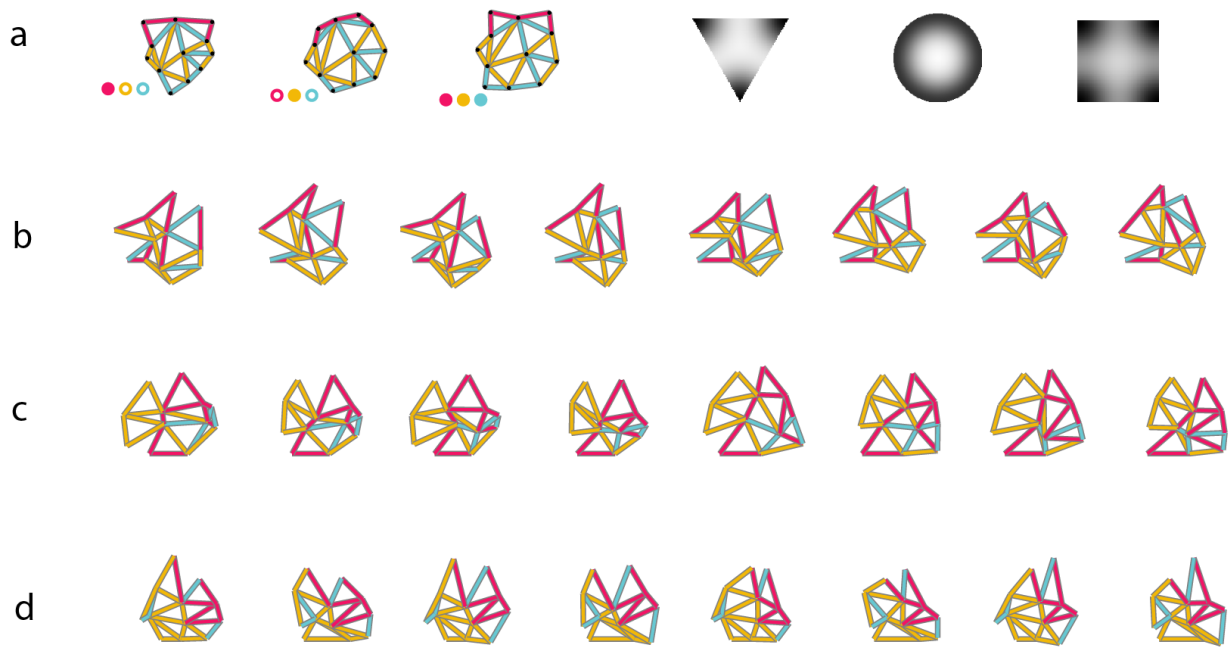


FIGURE 6.14: **Comparison of optimized geometric shape design with dataset examples. a,** The final optimized geometrical shapes and the target shape masks. **b-c,** The three closest data in the dataset with 8 transformed shapes.

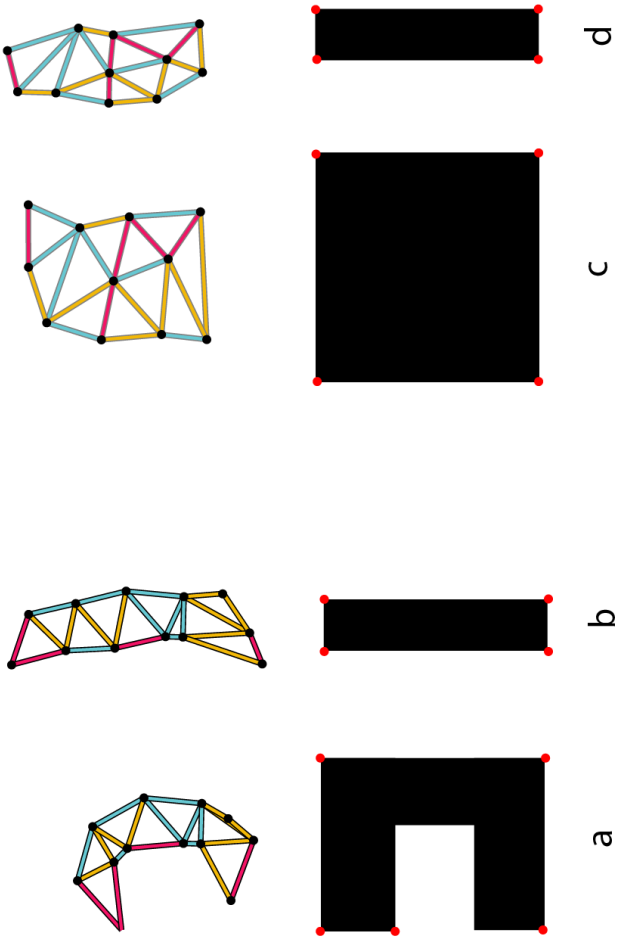


FIGURE 6.15: **Failure cases of shape optimization of strip transformations. a-b,** The optimized shapes (top) and the target shape masks of a strip bent into a C shape. **c-d,** The optimized shapes (top) and the target shape masks of a strip transformed into a square.

Chapter 7

Discussion

7.1 The Challenge of Complex Morphing Robots

Nature offers remarkable examples of creatures that can dramatically transform their morphology to adapt to their environment and tasks. From octopi that can squeeze through openings barely larger than their eye [4], to the magnificent frigatebird that can inflate its gular pouch to nearly its body size [3], to lizards that can dramatically modify their throat structure for display [2], biological systems demonstrate sophisticated capabilities in morphological adaptation that extend far beyond simple compressions or expansions.

This biological inspiration has driven developments not only in robotics but also in human-computer interaction (HCI), where shape-changing interfaces enable dynamic physical affordances and adaptive functionality. From social robots that use dynamic skin textures to convey emotional states [32], to furniture-scale shape transformation systems [141], to entirely soft, programmable shape-changing modules [26], researchers have demonstrated the broad potential of morphological adaptation. However, creating machines that can achieve such dramatic volumetric shape changes while maintaining structural integrity and control precision remains a significant challenge.

Current approaches to morphological adaptation in robotics each face distinct limitations. Limbed robots, which mirror the anatomical structure of animals [5, 6], utilize a central body with branching limbs [7, 8]. While effective for many tasks, their tree-like topology inherently constrains their shape-changing capabilities - they can move limbs but cannot fundamentally alter their volume or overall structure. Continuum robots, inspired by spineless organisms [15, 16], provide virtually infinite degrees of freedom but often lack mechanical stability and load-bearing capacity. Cubic robots based on voxel units [50, 51] achieve shape changes through selective activation but face challenges in scalability and precision due to their solid volume nature.

Among these approaches, looped graph structures (LGSs) - encompassing both truss and tensegrity robots - emerge as a promising alternative that combines structural stability with shape-changing capabilities. Their graph topology with loops, where edges can circle back to their starting nodes, provides both structural strength and extensive degrees of freedom. However, these structures face two fundamental challenges that have limited their practical implementation:

First, the curse of dimensionality (CoD) manifests as the structure scales up. Given the same size of elements and connection patterns, scaling up an LGS's volume leads to a cubic increase in the number of elements. This exponential growth affects both hardware complexity and the search space for design optimization. As a result, current physical implementations are limited to either simple structures with few tetrahedral units [43, 56, 57] or larger structures where only a small subset of beams are actuable [45]. More complex designs have been demonstrated in simulation [60], but their physical realization remains challenging due to these scaling constraints.

Second, the discrete nature of graph structures poses unique optimization challenges. Unlike continuum robots that are inherently suitable for continuous optimization [15, 16], or voxel-based robots that can be optimized in a continuous space and then discretized to a regular grid structure [53, 123], the non-regular and non-Euclidean graph structure of LGSs creates significant barriers to efficient optimization. While topology optimization methods have proven effective for regular grid structures [122], the connectivity constraints and irregular topology of truss robots make such approaches unsuitable. This discrete nature proves particularly challenging when combined with actuator grouping strategies, limiting optimization approaches to heuristics-based combinatorial methods [7] that often prove less efficient than continuous optimization techniques.

In this thesis, we addressed these fundamental challenges through a series of contributions: (1) a bio-inspired actuator grouping mechanism that reduced control complexity while maintaining functionality, (2) a specialized genetic algorithm framework that effectively optimized discrete parameters towards multiple objective tasks within physical constraints, and (3) a novel neural network approach that transformed discrete topology optimization into continuous latent space optimization, enabling topology generation. Together, these advances enabled the creation of complex morphing truss robots that were previously impractical to design and implement.

7.2 Biological Inspiration: From Muscle Synergy to Actuator Groups

Our approach to addressing these challenges draws inspiration from biological muscle synergy, where complex movements emerge from coordinated muscle groups rather than individual control. Research indicates that humans and other animals execute sophisticated movements without consciously controlling each muscle, despite having hundreds of muscles and billions of muscle cells. This mechanism, known as muscle synergy, significantly reduces neural pathway complexity [61, 62, 100]. Through synergy in human motor control, intricate actions like walking or jumping are executed by coordinating muscle networks periodically, eliminating the need for conscious control of every individual muscle. While the topic remains under debate, several researchers argue that this coordinated approach achieves an optimal balance between actuator count and control complexity, significantly reducing the brain's computational burden [102–104].

This biological principle suggested a novel solution to the curse of dimensionality in truss robots: rather than controlling each actuator independently, we could group actuators into networks that operate in coordination. As presented in PneuMesh: Actuator Grouping in Truss Robot, we introduced a novel actuator grouping mechanism through customizable connection joints, termed C-networks. Each subset of actuators within a C-network can be actuated simultaneously by a single control module, significantly reducing the redundancy in control mechanisms while maintaining the system's expressiveness.

The success of this approach was demonstrated through the construction of complex truss robots with over 100 actuable beams controlled by just a few modules. For instance, our system enabled a truss robot with 150 actuating beams to be efficiently grouped into eight subnetworks, each managed by its own control module—achieving a ratio of 18.75 actuating beams per control module. This represents a significant improvement over existing models which typically achieve only one actuating beams per control module [40, 45].

Our implementation, detailed in PneuMesh: Actuator Grouping in Truss Robot, involves carefully designed joint structures that can selectively channel actuation signals across multiple beams. Each joint incorporates an innovative selective-air-channel design that enables specific groups of actuators to share the same air source, ensuring synchronized activation.

This design not only simplifies the control architecture but also reduces the physical complexity of wiring and power distribution. The effectiveness of this approach was validated through multiple robot designs, from a six-leg walker achieving forward motion with just two air ports to complex shape-morphing structures capable of dramatic volumetric transformations.

7.3 From Manual Design to Automated Optimization

While the actuator grouping mechanism presented in PneuMesh: Actuator Grouping in Truss Robot demonstrated the feasibility of complex truss robots with simplified control, it introduced new challenges in design optimization. The assignment of actuators to C-networks needed to satisfy both connectivity constraints (actuators in the same C-network must form a connected subgraph) and symmetry requirements. As detailed in Muscle Synergy Inspired Evolution of Actuator Network, a truss with n_e edges and n_γ channels has a solution space with a size up to $n_\gamma^{n_e}$. Moreover, a physically practical solution must maintain the connectivity of each channel, meaning beams belonging to the same channel group need to have ensured physical connectivity for the pneumatic actuation system.

Existing approaches to truss robot optimization proved inadequate for our unique challenges. Previous work typically assumed independent control of each actuator [41, 42, 97], or focused on co-optimization of control and morphology for tree-like structures [7, 78, 106]. Implicit encoding methods like Compositional Pattern-Producing Networks (CPPNs), while effective for voxel robots [50, 108], struggle with the non-Euclidean topology of truss structures where the relationship between neighboring elements is not uniform or continuous in space.

As presented in Muscle Synergy Inspired Evolution of Actuator Network, we addressed these challenges through a tailored genetic algorithm incorporating custom operators that respect both symmetry and connectivity constraints while exploring the design space. Our approach transformed the complex problem of C-network design into a tractable form by encoding network assignments, contraction levels, and activation signals into a simple yet expressive integer-based representation. To handle multiple competing objectives while maintaining design diversity, we enhanced the NSGA-II algorithm with an elite preservation mechanism.

The effectiveness of this optimization framework was demonstrated through several compelling examples. Most notably, our quadrupedal robot with over 150 actuators could achieve walking, turning, crouching, and tilting behaviors using just eight control groups. Through systematic experimentation, we found that performance improvements diminish beyond a certain number of C-networks, supporting our hypothesis inspired by muscle synergy. Statistical analysis using one-way ANOVA confirmed significant performance improvements when increasing from 2 to 8 C-networks ($p = 0.003$), but differences became statistically insignificant beyond 16 C-networks ($p > 0.877$). The success of this optimization approach enabled the design of significantly more complex truss robots than previously possible.

The effectiveness of this optimization framework was demonstrated through several compelling examples. Most notably, our quadrupedal robot with over 150 actuators could achieve walking, turning, crouching, and tilting behaviors using just eight control groups. Through systematic experimentation, we found that performance improvements diminish beyond a certain number of C-networks, supporting our hypothesis inspired by muscle synergy. Statistical analysis using one-way ANOVA confirmed measurable performance improvements when increasing from 2 to 8 C-networks ($p = 0.003$), but differences became statistically insignificant beyond 16 C-networks ($p > 0.877$). This result validates our core hypothesis: strategic actuator grouping can achieve complex robotic capabilities with reduced control

complexity. While previous VGT implementations were limited to either structures with few tetrahedral units [43, 56, 57] or larger structures where only a small subset of beams were actuable [45], our approach enabled multi-behavior control in a structure with 150 actuating beams managed by eight modules, achieving a ratio of 18.75 actuating beams per control module compared to the 1:1 ratio in previous work [40, 45]. This reduction in control complexity while maintaining motion capabilities suggesting an alternative to the conventional approach of independent control for each actuator.

7.4 Bridging Discrete and Continuous Optimization

Despite the effectiveness of our genetic algorithm approach presented in Muscle Synergy Inspired Evolution of Actuator Network, scaling to larger structures and more complex tasks revealed fundamental limitations of discrete optimization. The expansive search space created by the curse of dimensionality, combined with the discreteness and connectivity constraints of actuator grouping, complicated the use of traditional optimizers. This challenge became particularly acute when dealing with multi-objective scenarios, where unconventional morphologies might outperform human-designed structures.

As detailed in Truss Topology and Parameter Generation with Variational Auto-encoders, we developed two approaches using variational auto-encoders (VAE) to transform the discrete design space into continuous latent representations suitable for optimization. Our first solution leverages graph attention networks (GAT) to optimize actuator grouping connectivity within fixed topologies. The GAT architecture's natural ability to process graph structures proved effective in capturing both local edge relationships and global structural patterns, achieving high accuracy in both reconstruction (99.925%) and connectivity prediction (99.980%).

However, while the GAT-VAE excelled at fixed-topology optimization, it could not easily handle varying graph structures. To overcome this limitation, we needed to solve a fundamental representation challenge: how to encode truss structures with loops in a way that neural networks could both understand and generate. Previous work had demonstrated success in using grammar to represent limbed robots, molecular structures [7, 78], and CAD models [140], but these approaches relied on tree-based grammars where each token represents a branching element from an existing body. Such representations cannot capture the looped topology essential to truss robots.

Our solution, presented in Truss Topology and Parameter Generation with Variational Auto-encoders, introduces a novel truss grammar that represents loop-containing structures through a merging operation. Our grammar enables the representation of any truss with disk-like topology as a unique sequence of tokens, which we combine with an LSTM-VAE to simultaneously optimize topology, C-network indices, initial lengths, and contraction ratios.

The model achieved 99.43% token reconstruction accuracy and demonstrated smooth interpolation capabilities in its latent space. As shown in Figure. 6.12, the model can smoothly transition between different designs while simultaneously modifying both discrete parameters (number of triangles) and continuous parameters (edge lengths). This continuous latent space representation enables efficient gradient-based optimization of traditionally discrete parameters and topologies.

We demonstrated our approach through three increasingly complex shape-morphing tasks (Fig. 6.11). Given one, two, or three target shapes, our model generates complete truss designs including topology, actuator grouping, and contraction ratios. The resulting trusses can transform to approximate a star shape in the single-shape case, morph between rectangular and triangular forms in the two-shape case, and achieve three distinct configurations in the three-shape case.

To verify that our model generates novel designs rather than memorizing training examples, we compared the optimized designs with their closest counterparts in the training dataset. As shown in Figure. 6.13 and Figure. 6.14, the optimized designs exhibit distinct differences in both topology and transformation strategies. For the star shape optimization, our model discovered an asymmetric structure with fewer triangles than similar training examples, achieving more efficient shape approximation. In the geometric shape transformations, the model generated novel actuation patterns that enable smoother transitions between target configurations.

However, the current approach has limitations, particularly for shapes requiring large contraction ratios or high curvature (Fig. 6.15). These failure cases suggest the need for future work in pre-evaluating target shape feasibility, either through analytical methods calculating minimum distortion ratios or through a learned neural network that predicts if the objective shapes are achievable.

To our knowledge, this work presents the first application of continuous latent space optimization to discrete looped topology structures in robotics. While previous approaches either relied on discrete optimization [7] or were limited to tree-like structures [79], our method enables simultaneous optimization of discrete topological features and continuous geometric parameters within a unified framework.

7.5 Conclusion

This thesis presents a progression of solutions to fundamental challenges in looped-graph structures (LGSs), particularly focusing on truss robots. Through a series of theoretical and practical contributions, we have advanced the field's understanding of how to design, optimize, and control complex morphing robots while managing the inherent challenges of dimensionality and discreteness.

Our first major contribution, presented in PneuMesh: Actuator Grouping in Truss Robot, introduces a synergy-inspired actuator grouping mechanism that fundamentally reimagines how complex truss robots can be controlled. Drawing parallels from biological muscle coordination, this approach enables the control of structures with over 100 actuators using just a few control modules - a dramatic reduction in complexity that makes previously intractable designs physically realizable. This innovation demonstrates that strategic reduction in control granularity can actually enable more complex overall capabilities, challenging traditional assumptions about the relationship between control complexity and functional sophistication.

The development of our optimization framework, detailed in Muscle Synergy Inspired Evolution of Actuator Network, addresses the discrete nature of truss optimization while respecting physical constraints. Our tailored genetic algorithm's success in optimizing actuator groupings within fixed topologies, while maintaining connectivity constraints, demonstrates the viability of discrete optimization for moderate-sized structures. However, its diminishing efficiency with increasing complexity motivated our transition to continuous optimization approaches.

The introduction of our dual VAE framework in Truss Topology and Parameter Generation with Variational Auto-encoders - combining Graph Attention Networks for connectivity optimization and LSTM networks for topology generation - represents a significant advancement in morphing robot design. The GAT-VAE's remarkable accuracy in reconstructing actuator groupings (99.925%) and connectivity patterns (99.999%) demonstrates the potential of continuous latent space optimization for discrete problems. Meanwhile, the LSTM-VAE's novel truss grammar enables the first approach to generating complete truss designs, including topology, actuator groupings, and geometric parameters.

These technical contributions collectively enable several key capabilities:

1. The ability to scale up truss robot complexity without proportional increases in control complexity, making previously intractable designs physically realizable
2. Efficient optimization of both discrete topological features and continuous parameters within a unified framework
3. Generation of novel truss designs optimized for specific tasks or shape transformations, moving beyond human-designed topologies
4. A practical pathway to implementing complex morphing robots through a combination of simplified control architecture and tailored optimization tools

The approaches developed in this thesis could extend beyond truss robots to other domains involving graph-based structures and mixed discrete-continuous optimization challenges. For tensegrity robots, which achieve stability through a balance of tension-only cables and compression-only rods [142], our grouped actuation strategy could be adapted to cable-based systems while our VAE framework could optimize both topology and tension distribution. In robotic metamaterials, where mechanical properties emerge from cellular architectures [143], our topology generation approach could enable optimization of both spatial allocation of elements and material properties of each element, particularly relevant for multi-material fabrication. These extensions are made possible by the fundamental similarities in their underlying discrete graph structures and optimization challenges, such as balancing geometric constraints with functional objectives while managing high degrees of freedom. The computational frameworks developed here, including grammar-based representation and hybrid discrete-continuous optimization, provide tools that could be adapted for these related domains. Additionally, advances in hardware implementation through alternative actuation mechanisms like light-responsive materials [144] and improved sensing systems could enable more precise control over morphological transformations while reducing physical constraints (See Future Work).

Chapter 8

Future Work

The solutions developed in this thesis, from synergy-inspired actuator grouping to neural network-based topology optimization, establish foundational approaches for designing and controlling complex morphing robots. While our framework has demonstrated success in truss robots, its underlying principles - the combination of grouped actuation, grammar-based representation, and hybrid discrete-continuous optimization - have potential applications beyond our initial scope. This chapter examines three key directions for extending our work: application to other physical systems like tensegrity robots and metamaterials, advancement of hardware implementation through novel actuation and sensing mechanisms, and enhancement of algorithmic capabilities for more complex topologies and control strategies. These directions not only address current limitations but also open new possibilities for adaptive robotics and morphing structures.

8.1 Extension to Other Physical Systems

Our framework's success in truss robots suggests broader applications to other morphing structures that share similar topological characteristics but present unique challenges. Of particular interest are tensegrity robots and robotic metamaterials, both of which involve complex graph structures and face similar challenges in balancing structural stability with shape-changing capabilities. These extensions not only validate the generality of our approach but also open new possibilities for adaptive structures across different scales and applications.

8.1.1 Tensegrity Robots

A natural and promising extension of our framework lies in tensegrity robots, which combine rigid rods with tensioned cables in a looped graph structure. Unlike truss robots where each edge can bear both tension and compression, tensegrity structures achieve stability through a precise balance of tension-only cables and compression-only rods. This fundamental characteristic makes them particularly appealing for applications requiring high strength-to-weight ratios and natural shock absorption capabilities [142]. However, tensegrity robots face even more stringent constraints than truss robots - beyond managing degrees of freedom, they must maintain structural stability through continuous tension balancing during any shape transformation.

Current approaches to tensegrity robot design typically rely on either modifying existing stable configurations [145] or assembling known stable units [146]. While these methods have produced functional robots, they significantly limit the design space and may miss more optimal configurations for specific tasks. Traditional optimization methods like force density [147] or position-based approaches [148] work well for simple topologies but become

computationally intractable for complex structures. Furthermore, these methods typically focus on static stability rather than dynamic performance, making them insufficient for robotic applications.

Physical System Design

To address the control complexity challenge in tensegrity robots, we propose adapting our actuator grouping mechanism to a cable-based actuation system, as illustrated in Figure 8.1. This design replaces individually controlled cables with a continuous cable that threads through multiple edges and is controlled by a single external actuator such as a winch. The approach maintains the benefits of reduced control complexity demonstrated in truss robots while accommodating the specific requirements of tensegrity structures.

The cable routing strategy requires careful design of the pathways through which cables traverse multiple edges, enabling coordinated movement patterns across the structure. This presents three primary technical challenges. First, the joints must enable smooth cable routing without interference while managing friction effects in multi-edge cable paths. Second, the system needs to ensure continuous and controllable length changes across grouped edges for precise shape control. Third, the design must maintain proper tension distribution throughout the structure to preserve structural integrity during transformation.

Computational Framework Adaptation

Extending our VAE-based optimization framework to tensegrity robots requires several key modifications to accommodate their unique constraints. The primary challenge lies in maintaining tensional integrity - tensegrity structures must maintain a precise balance of tensions, with rigid components connected only through cables under tension. Incorporating this constraint into the optimization pipeline necessitates the integration of form-finding algorithms, which have proven effective for static tensegrity structures. The modified framework distinguishes between rigid rods and tension cables in the graph representation while incorporating tension balance constraints in the optimization process. The truss grammar requires adaptation to handle the specific requirements of tensegrity structures, particularly in representing the distinct properties of rods and cables. This modified grammar must ensure that generated designs maintain stability across their range of motion, a constraint not present in pure truss structures.

Preliminary Simulator Development

Initial simulations using Mujoco demonstrate the feasibility of this approach for tensegrity robots. The simulator captures the essential dynamics of cable-actuated tensegrity structures, including tension distribution and shape deformation. These preliminary results indicate that the framework can handle the additional complexity introduced by tensegrity-specific constraints.

Several technical challenges remain for physical validation. The development of robust joint designs that maintain reliable cable routing under dynamic conditions presents a primary challenge. Implementing effective tension monitoring and control systems is equally critical, as tension distribution directly affects structural stability. The integration of actuator grouping with tensional integrity requirements necessitates careful balance between grouped actuation and tension distribution. Validating the transfer from simulation to physical systems requires systematic testing and refinement of the models to capture real-world dynamics.

The extension to tensegrity robots enables new possibilities in morphing robot design through the combination of truss and tensegrity principles. Cable-based actuation systems

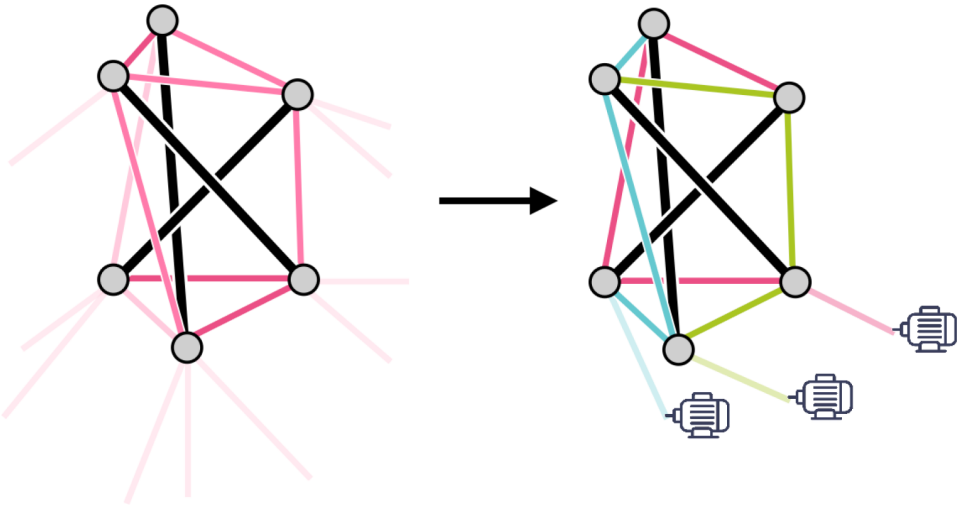


FIGURE 8.1: Illustration of the proposed tensegrity structure with cable actuator grouping and external winch motors. Same color of cables are continuous and pulled by a single external motor.

with optimized tension distribution can achieve higher strength-to-weight ratios than traditional truss designs [142]. The distributed tension network provides natural impact absorption capabilities, while the combination of rigid elements with tensioned cables enables extensive shape changes while maintaining structural stability. These characteristics make tensegrity robots particularly suitable for applications in space exploration, where lightweight, deployable structures must adapt to irregular terrain, and in architectural applications, where structures must respond to environmental conditions.

8.1.2 Robotic Metamaterials

Metamaterials represent engineered structures that derive their properties primarily from their cellular architecture rather than their constituent materials. These structures exhibit behaviors not found in natural materials, including tunable chiral, auxetic, and compliant properties [143, 149]. The behavior emerges from their repeating spatial arrangement of solid regions and voids, with truss-based cellular architectures representing one effective approach to space-filling patterns [150].

Current approaches to metamaterial design range from manual design to topology optimization methods. Some researchers have demonstrated success in achieving specific mechanical properties through discrete optimization of unit cell geometry [151, 152]. Neural networks have been employed to predict properties of given designs [153], but the generation of novel topologies optimized for specific properties remains challenging. Additionally, while existing work primarily focuses on single-material structures, modern multi-material fabrication capabilities suggest opportunities for more complex designs with varying material properties across the structure.

Our VAE framework offers potential advantages for metamaterial design through its ability to encode and optimize both topology and continuous parameters. The framework could be extended to optimize both the spatial arrangement of unit cells and the material properties of individual elements. This capability becomes particularly relevant with the advancement of multi-material 3D printing technologies, which enable the fabrication of structures with varying material properties across different components. The position prediction network

could be modified to predict mechanical properties of generated designs, enabling direct optimization for specific behavioral targets.

8.2 Hardware Implementation

The practical implementation of complex morphing robots presents several technical challenges that warrant exploration. Current hardware limitations in joint design, actuation mechanisms, and sensing capabilities directly impact the achievable complexity and precision of morphing structures. This section examines potential advances in these areas to bridge the gap between theoretical capabilities and physical realization.

8.2.1 Joint Design Evolution

Static truss structures employ straightforward joint designs that prioritize load distribution and structural stability. However, truss robots require joints capable of accommodating varying angles during transformation while maintaining structural integrity. Previous work introduced several approaches to this challenge. The spherical joint design in [154], based on [155], enables smooth rotation through a 3D-printed mechanism. Alternative approaches include the spiral zipper spherical joint [156], which employs a linkage structure allowing both rotation and joint reconfiguration.

Our pneumatic system introduces additional complexity by requiring air channels through the joints. Current designs rely on soft, compliant mechanisms [157, 158], but these limit load-bearing capacity. A promising direction combines rigid spherical joints with integrated pneumatic routing. However, this approach faces a fundamental trade-off: joint size typically scales with degrees of freedom to prevent collision, potentially limiting the structure's overall contraction ratio. Future work could address this through miniaturization of joint components or by incorporating joint constraints directly into the optimization framework.

8.2.2 Alternative Actuation Mechanisms

The current pneumatic actuation system imposes connectivity constraints that complicate both physical implementation and algorithmic optimization. Light-triggered actuation mechanisms offer a potential solution by enabling remote control without physical connections between actuators. Materials such as hydrogels respond to light through hydration changes [159, 160], while liquid crystal elastomers (LCE) contract along their nematic alignment direction under UV exposure [161].

These materials enable more sophisticated control through wavelength-specific responses. Different regions of an LCE actuator can respond to distinct wavelengths [144], similar to our C-network grouping but without physical connectivity requirements. By varying color concentrations within individual actuators, partial responses to different light sources become possible [162], enabling reuse of actuators across different transformations. This approach has demonstrated success in micro-scale applications [163], suggesting potential for miniaturized truss robots at scales of 0.5-5mm.

8.2.3 Sensing and Feedback Attachments

The preliminary work on reinforcement learning for closed-loop control, detailed in Closed-loop Control of Truss Robot using Reinforcement Learning, demonstrates the potential for

enhancing the current open-loop control system. The reinforcement learning approach processes environmental inputs to determine optimal C-network states at each time step, enabling adaptation to environmental disturbances and precise control over morphological transformations. However, implementing closed-loop control on physical systems requires accurate real-time position sensing of the truss robot's joints.

Motion capture technology offers promising solutions for real-time tracking. The modular nature of truss structures, combined with their high volume-to-mass ratio, facilitates the integration of various sensing devices. Potential approaches include distributed IMU sensors or multiple cameras tracking visual markers [164]. These sensing systems must balance precision with the added weight and complexity they introduce to the structure.

8.3 Algorithm Advancement

8.3.1 Topology Generation and Dataset Extension

Current transformer architectures demonstrate capability in processing sequences of comparable length to their training data [165], with models like Skexgen successfully handling over 200 tokens for CAD modeling. Large language models further extend this capacity [166]. Our current implementation, described in Truss Topology and Parameter Generation with Variational Auto-encoders, handles sequences up to 22 tokens, suggesting potential for significant expansion.

Neural networks have shown promising results in processing 3D data, as demonstrated by recent work in triangle mesh surface generation [167]. While these approaches focus on surface meshes without connectivity constraints or tetrahedral structures, our truss grammar naturally extends to 3D configurations as outlined in Truss Topology and Parameter Generation with Variational Auto-encoders. Expanding the dataset to incorporate longer sequences and 3D truss topologies represents a logical next step in advancing the framework's capabilities.

8.3.2 Control Integration

The reinforcement learning approach detailed in Closed-loop Control of Truss Robot using Reinforcement Learning successfully develops closed-loop controllers for truss designs optimized through genetic algorithms (Muscle Synergy Inspired Evolution of Actuator Network). The current TrussVAE implementation employs separate position predictors for each possible control signal combination, limiting extensibility to more complex control scenarios. A more scalable approach would develop a neural network that processes control signal combinations and actuator groupings into high-level features for position prediction, enabling reuse of the network structure across different grouping configurations.

Future developments could explore universal controller generation [14, 168], where controllers are conditioned on topology. This approach would enable simultaneous optimization of both structural design and control strategies, potentially discovering more efficient solutions for complex morphing tasks.

Bibliography

1. Shah, D. *et al.* Shape Changing Robots: Bioinspiration, Simulation, and Physical Realization. en. *Advanced Materials* **33,** 2002882. ISSN: 1521-4095. https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202002882 (2023) (2021).
2. Ord, T. J., Klomp, D. A., Garcia-Porta, J. & Hagman, M. Repeated evolution of exaggerated dewlaps and other throat morphology in lizards. *Journal of Evolutionary Biology* **28,** 1948–1964. ISSN: 1010-061X. https://doi.org/10.1111/jeb.12709 (2024) (Nov. 2015).
3. Edmonds, P. How the frigatebird's courtship display lures the ladies. en. *National Graphics*. Section: Magazine. https://www.nationalgeographic.com/magazine/article/how-frigatebird-courtship-display-attracts-females (2024) (Jan. 2019).
4. Levy, G. & Hochner, B. Embodied Organization of Octopus vulgaris Morphology, Vision, and Locomotion. English. *Frontiers in Physiology* **8.** Publisher: Frontiers. ISSN: 1664-042X. https://www.frontiersin.org/journals/physiology/articles/10.3389/fphys.2017.00164/full (2024) (Mar. 2017).
5. Raibert, M., Blankespoor, K., Nelson, G. & Playter, R. BigDog, the Rough-Terrain Quadruped Robot. *IFAC Proceedings Volumes. 17th IFAC World Congress* **41,** 10822–10825. ISSN: 1474-6670. https://www.sciencedirect.com/science/article/pii/S1474667016407020 (2024) (Jan. 2008).
6. Hirai, K., Hirose, M., Haikawa, Y. & Takenaka, T. *The development of Honda humanoid robot* in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)* **2.** ISSN: 1050-4729 (May 1998), 1321–1326 vol.2. https://ieeexplore.ieee.org/abstract/document/677288?casa_token=Zq_U6xbjju0AAAAA:NLV6dgRNBra7Bu09P63tm921ttrkUFdTEso04z4EmuEh6l4EK4uB_5fJ6dytP9BxC7lDoN2y (2024).
7. Zhao, A. *et al.* RoboGrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics* **39,** 188:1–188:16. ISSN: 0730-0301. https://dl.acm.org/doi/10.1145/3414685.3417831 (2024) (Nov. 2020).
8. Xu, J., Spielberg, A., Zhao, A., Rus, D. & Matusik, W. *Multi-Objective Graph Heuristic Search for Terrestrial Robot Design* en. in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, Xi'an, China, May 2021), 9863–9869. ISBN: 978-1-72819-077-8. https://ieeexplore.ieee.org/document/9561818/ (2023).
9. Sun, J., Lerner, E., Tighe, B., Middlemist, C. & Zhao, J. Embedded shape morphing for morphologically adaptive robots. en. *Nature Communications* **14.** Publisher: Nature Publishing Group, 6023. ISSN: 2041-1723. https://www.nature.com/articles/s41467-023-41708-6 (2024) (Sept. 2023).
10. Nygaard, T. F., Martin, C. P., Torresen, J., Glette, K. & Howard, D. Real-world embodied AI through a morphologically adaptive quadruped robot. en. *Nature Machine Intelligence* **3.** Publisher: Nature Publishing Group, 410–419. ISSN: 2522-5839. https://www.nature.com/articles/s42256-021-00320-3 (2024) (May 2021).

11. Sihite, E., Kalantari, A., Nemovi, R., Ramezani, A. & Gharib, M. Multi-Modal Mobility Morphobot (M4) with appendage repurposing for locomotion plasticity enhancement. en. *Nature Communications* **14.** Publisher: Nature Publishing Group, 3323. ISSN: 2041-1723. https://www.nature.com/articles/s41467-023-39018-y (2024) (June 2023).
12. Zhao, A. *et al.* RoboGrammar: graph grammar for terrain-optimized robot design. en. *ACM Transactions on Graphics* **39,** 1–16. ISSN: 0730-0301, 1557-7368. https://dl.acm.org/doi/10.1145/3414685.3417831 (2023) (Dec. 2020).
13. Gupta, A., Fan, L., Ganguli, S. & Fei-Fei, L. *MetaMorph: Learning Universal Controllers with Transformers* arXiv:2203.11931 [cs]. Mar. 2022. http://arxiv.org/abs/2203.11931 (2023).
14. Gupta, A., Savarese, S., Ganguli, S. & Fei-Fei, L. Embodied intelligence via learning and evolution. en. *Nature Communications* **12.** Number: 1 Publisher: Nature Publishing Group, 5721. ISSN: 2041-1723. https://www.nature.com/articles/s41467-021-25874-z (2023) (Oct. 2021).
15. Wehner, M. *et al.* An integrated design and fabrication strategy for entirely soft, autonomous robots. en. *Nature* **536.** Number: 7617 Publisher: Nature Publishing Group, 451–455. ISSN: 1476-4687. https://www.nature.com/articles/nature19100 (2024) (Aug. 2016).
16. Buchner, T. J. K. *et al.* Vision-controlled jetting for composite systems and robots. en. *Nature* **623.** Number: 7987 Publisher: Nature Publishing Group, 522–530. ISSN: 1476-4687. https://www.nature.com/articles/s41586-023-06684-3 (2024) (Nov. 2023).
17. Gomi, R., Suzuki, R., Takashima, K., Fujita, K. & Kitamura, Y. *InflatableBots: Inflatable Shape-Changing Mobile Robots for Large-Scale Encountered-Type Haptics in VR* en. in *Proceedings of the CHI Conference on Human Factors in Computing Systems* (ACM, Honolulu HI USA, May 2024), 1–14. ISBN: 9798400703300. https://dl.acm.org/doi/10.1145/3613904.3642069 (2024).
18. Suzuki, R. *et al. RoomShift: Room-scale Dynamic Haptics for VR with Furniture-moving Swarm Robots* en. in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (ACM, Honolulu HI USA, Apr. 2020), 1–11. ISBN: 978-1-4503-6708-0. https://dl.acm.org/doi/10.1145/3313831.3376523 (2024).
19. Suzuki, R. *et al. LiftTiles: Constructive Building Blocks for Prototyping Room-scale Shape-changing Interfaces* en. in *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction* (ACM, Sydney NSW Australia, Feb. 2020), 143–151. ISBN: 978-1-4503-6107-1. https://dl.acm.org/doi/10.1145/3374920.3374941 (2024).
20. Gonzalez, J. T. *et al. Constraint-Driven Robotic Surfaces, At Human-Scale* en. in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (ACM, San Francisco CA USA, Oct. 2023), 1–12. ISBN: 9798400701320. https://dl.acm.org/doi/10.1145/3586183.3606740 (2024).
21. Ishii, H. *et al. TRANSFORM: Embodiment of "Radical Atoms" at Milano Design Week* en. in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (ACM, Seoul Republic of Korea, Apr. 2015), 687–694. ISBN: 978-1-4503-3146-3. https://dl.acm.org/doi/10.1145/2702613.2702969 (2024).

22. Follmer, S., Leithinger, D., Olwal, A., Hogge, A. & Ishii, H. *inFORM: dynamic physical affordances and constraints through shape and object actuation* en. in *Proceedings of the 26th annual ACM symposium on User interface software and technology* (ACM, St. Andrews Scotland, United Kingdom, Oct. 2013), 417–426. ISBN: 978-1-4503-2268-3. https://dl.acm.org/doi/10.1145/2501988.2502032 (2024).

23. Nakagaki, K., Dementyev, A., Follmer, S., Paradiso, J. A. & Ishii, H. *ChainFORM: A Linear Integrated Modular Hardware System for Shape Changing Interfaces* en. in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (ACM, Tokyo Japan, Oct. 2016), 87–96. ISBN: 978-1-4503-4189-9. https://dl.acm.org/doi/10.1145/2984511.2984587 (2024).

24. Nakagaki, K., Follmer, S. & Ishii, H. *LineFORM: Actuated Curve Interfaces for Display, Interaction, and Constraint* en. in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (ACM, Charlotte NC USA, Nov. 2015), 333–339. ISBN: 978-1-4503-3779-3. https://dl.acm.org/doi/10.1145/2807442.2807452 (2024).

25. Yao, L. *et al. PneUI: pneumatically actuated soft composite materials for shape changing interfaces* en. in *Proceedings of the 26th annual ACM symposium on User interface software and technology* (ACM, St. Andrews Scotland, United Kingdom, Oct. 2013), 13–22. ISBN: 978-1-4503-2268-3. https://dl.acm.org/doi/10.1145/2501988.2502037 (2024).

26. Nakayama, R. *et al. MorphIO: Entirely Soft Sensing and Actuation Modules for Programming Shape Changes through Tangible Interaction* en. in *Proceedings of the 2019 on Designing Interactive Systems Conference* (ACM, San Diego CA USA, June 2019), 975–986. ISBN: 978-1-4503-5850-7. https://dl.acm.org/doi/10.1145/3322276.3322337 (2024).

27. Suzuki, R. *et al. ShapeBots: Shape-changing Swarm Robots* arXiv:1909.03372. Sept. 2019. http://arxiv.org/abs/1909.03372 (2024).

28. Suzuki, R. *et al. Dynablock: Dynamic 3D Printing for Instant and Reconstructable Shape Formation* in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Association for Computing Machinery, New York, NY, USA, Oct. 2018), 99–111. ISBN: 978-1-4503-5948-1. https://dl.acm.org/doi/10.1145/3242587.3242659 (2024).

29. Lu, Q., Xu, H., Guo, Y., Wang, J. Y. & Yao, L. *Fluidic Computation Kit: Towards Electronic-free Shape-changing Interfaces* en. in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (ACM, Hamburg Germany, Apr. 2023), 1–21. ISBN: 978-1-4503-9421-5. https://dl.acm.org/doi/10.1145/3544548.3580783 (2024).

30. Dunstan, B. J. & Hoffman, G. en. in *Cultural Robotics: Social Robots and Their Emergent Cultural Ecologies* (eds Dunstan, B. J., Koh, J. T. K. V., Turnbull Tillman, D. & Brown, S. A.) 13–34 (Springer International Publishing, Cham, 2023). ISBN: 978-3-031-28138-9. https://doi.org/10.1007/978-3-031-28138-9_2 (2024).

31. Barco, A., De Jong, C., Peter, J., Kühne, R. & Van Straten, C. L. *Robot Morphology and Children's Perception of Social Robots: An Exploratory Study* en. in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction* (ACM, Cambridge United Kingdom, Mar. 2020), 125–127. ISBN: 978-1-4503-7057-8. https://dl.acm.org/doi/10.1145/3371382.3378348 (2024).

32. Hu, Y., Zhao, Z., Vimal, A. & Hoffman, G. *Soft skin texture modulation for social robotics* in *2018 IEEE International Conference on Soft Robotics (RoboSoft)* (Apr. 2018), 182–187. https://ieeexplore.ieee.org/document/8404917 (2024).

33. Hu, Y. & Hoffman, G. *Using Skin Texture Change to Design Emotion Expression in Social Robots* in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* ISSN: 2167-2148 (Mar. 2019), 2–10. https://ieeexplore.ieee.org/document/8673012 (2024).

34. Hu, Y. & Hoffman, G. What Can a Robot's Skin Be? Designing Texture-changing Skin for Human–Robot Social Interaction. *J. Hum.-Robot Interact.* **12,** 26:1–26:19. https://dl.acm.org/doi/10.1145/3532772 (2024) (Apr. 2023).

35. Bradwell, H. L., Winnington, R., Thill, S. & Jones, R. B. *Morphology of socially assistive robots for health and social care: A reflection on 24 months of research with anthropomorphic, zoomorphic and mechanomorphic devices* en. in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)* (IEEE, Vancouver, BC, Canada, Aug. 2021), 376–383. ISBN: 978-1-66540-492-1. https://ieeexplore.ieee.org/document/9515446/ (2024).

36. Siu, A. F., Kim, S., Miele, J. A. & Follmer, S. *shapeCAD: An accessible 3D modelling workflow for the blind and visually-impaired via 2.5 D shape displays* in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility* (2019), 342–354.

37. Yao, L. *et al. PneUI: pneumatically actuated soft composite materials for shape changing interfaces* in *Proceedings of the 26th annual ACM symposium on User interface software and Technology* (2013), 13–22.

38. Ou, J. *et al. aeroMorph-heat-sealing inflatable shape-change materials for interaction design* in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), 121–132.

39. Roudaut, A., Karnik, A., Löchtefeld, M. & Subramanian, S. *Morphees: toward high" shape resolution" in self-actuated flexible mobile devices* in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), 593–602.

40. Hamlin, G. & Sanderson, A. TETROBOT: a modular approach to parallel robotics. *IEEE Robotics & Automation Magazine* **4.** Conference Name: IEEE Robotics & Automation Magazine, 42–50. ISSN: 1558-223X (Mar. 1997).

41. Usevitch, N. S., Hammond, Z. M. & Schwager, M. Locomotion of Linear Actuator Robots Through Kinematic Planning and Nonlinear Optimization. *IEEE Transactions on Robotics* **36.** Conference Name: IEEE Transactions on Robotics, 1404–1421. ISSN: 1941-0468 (Oct. 2020).

42. Liu, C., Yu, S. & Yim, M. *Motion Planning for Variable Topology Truss Modular Robot* en. in *Robotics: Science and Systems XVI* (Robotics: Science and Systems Foundation, July 2020). ISBN: 978-0-9923747-6-1. (2023).

43. Hughes, P., Sincarsin, W. & Carroll, K. Trussarm—A Variable-Geometry-Truss Manipulator. *Journal of Intelligent Material Systems and Structures* **2.** Publisher: SAGE Publications Ltd STM, 148–160. ISSN: 1045-389X. https://doi.org/10.1177/1045389X9100200202 (2023) (Apr. 1991).

44. Miura, K. *Design and operation of a deployable truss structure* in. NTRS Author Affiliations: Tokyo Univ. NTRS Document ID: 19840017014 NTRS Research Center: Legacy CDMS (CDMS) (May 1984). https://ntrs.nasa.gov/citations/19840017014 (2024).

45. Kovacs, R. *et al. TrussFormer: 3D Printing Large Kinetic Structures* en. in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (ACM, Berlin Germany, Oct. 2018), 113–125. ISBN: 978-1-4503-5948-1. https://dl.acm.org/doi/10.1145/3242587.3242607 (2023).

46. Spinos, A. & Yim, M. *Towards a variable topology truss for shoring* in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)* (June 2017), 244–249.

47. Curtis, S. *et al.* Tetrahedral Robotics for Space Exploration. *IEEE Aerospace and Electronic Systems Magazine* **22.** Conference Name: IEEE Aerospace and Electronic Systems Magazine, 22–30. ISSN: 1557-959X (June 2007).

48. Pieber, M., Neurauter, R. & Gerstmayr, J. *Adaptive Triangular Cell(ATC): An Adaptive Robot for Building In-Plane Programmable Structures* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* ISSN: 2153-0866 (Oct. 2018), 1–9.

49. Shah, D. S. *et al.* Tensegrity Robotics. en. *Soft Robotics* **9,** 639–656. ISSN: 2169-5172, 2169-5180. https://www.liebertpub.com/doi/10.1089/soro.2020.0170 (2023) (Aug. 2022).

50. Object, o. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. https://core.ac.uk/reader/24060140 (2024).

51. Cheney, N., Clune, J. & Lipson, H. Evolved Electrophysiological Soft Robots. en.

52. Cheney, N., Bongard, J. & Lipson, H. *Evolving Soft Robots in Tight Spaces* en. in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (ACM, Madrid Spain, July 2015), 935–942. ISBN: 978-1-4503-3472-3. https://dl.acm.org/doi/10.1145/2739480.2754662 (2024).

53. Cheney, N., Bongard, J., SunSpiral, V. & Lipson, H. *Scalable Co-Optimization of Morphology and Control in Embodied Machines* arXiv:1706.06133 [cs]. Dec. 2017. http://arxiv.org/abs/1706.06133 (2024).

54. Zykov, V., Mytilinaios, E., Adams, B. & Lipson, H. Robotics: self-reproducing machines. eng. *Nature* **435,** 163–164. ISSN: 1476-4687 (May 2005).

55. Nisser, M., Izzo, D. & Borggraefe, A. An Electromagnetically Actuated, Self-Reconfigurable Space Structure. en. **14** (2017).

56. Qin, Y. *et al. TrussBot: Modeling, Design, and Control of a Compliant, Helical Truss of Tetrahedral Modules* en. in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, Philadelphia, PA, USA, May 2022), 4218–4224. ISBN: 978-1-72819-681-7. https://ieeexplore.ieee.org/document/9812295/ (2023).

57. Spinos, A., Carroll, D., Kientz, T. & Yim, M. *Variable topology truss: Design and analysis* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* ISSN: 2153-0866 (Sept. 2017), 2717–2722. https://ieeexplore.ieee.org/document/8206098 (2023).

58. Romanishin, J. W., Gilpin, K. & Rus, D. *M-blocks: Momentum-driven, magnetic modular robots* en. in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, Tokyo, Nov. 2013), 4288–4295. ISBN: 978-1-4673-6358-7 978-1-4673-6357-0. http://ieeexplore.ieee.org/document/6696971/ (2024).

59. Kriegman, S., Blackiston, D., Levin, M. & Bongard, J. A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences* **117,** 1853–1859 (2020).

60. Usevitch, N., Hammond, Z., Follmer, S. & Schwager, M. *Linear actuator robots: Differential kinematics, controllability, and algorithms for locomotion and shape morphing* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* ISSN: 2153-0866 (Sept. 2017), 5361–5367.

61. Kerkman, J. N., Bekius, A., Boonstra, T. W., Daffertshofer, A. & Dominici, N. Muscle Synergies and Coherence Networks Reflect Different Modes of Coordination During Walking. *Frontiers in Physiology* **11.** ISSN: 1664-042X. https://www.frontiersin.org/articles/10.3389/fphys.2020.00751 (2023) (2020).

62. Giszter, S. F., Mussa-Ivaldi, F. A. & Bizzi, E. Convergent force fields organized in the frog's spinal cord. eng. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* **13,** 467–491. ISSN: 0270-6474 (Feb. 1993).

63. Gu, J. *et al. PneuMesh: Pneumatic-driven Truss-based Shape Changing System* en. in *CHI Conference on Human Factors in Computing Systems* (ACM, New Orleans LA USA, Apr. 2022), 1–12. ISBN: 978-1-4503-9157-3. https://dl.acm.org/doi/10.1145/3491102.3502099 (2023).

64. Sareen, H. *et al. Printflatables: printing human-scale, functional and dynamic inflatable objects* in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017), 3669–3680.

65. Hamlin, G. J. & Sanderson, A. C. *Tetrobot: A modular approach to reconfigurable parallel robotics* (Springer Science & Business Media, 2013).

66. Curtis, S. *et al. Tetrahedral robotics for space exploration* in *2007 IEEE Aerospace Conference* (2007), 1–9.

67. Zagal, J. C., Armstrong, C. & Li, S. *Deformable octahedron burrowing robot* in *Artificial Life Conference Proceedings 12* (2012), 431–438.

68. Usevitch, N. S. *et al.* An untethered isoperimetric soft robot. *Science Robotics* **5** (2020).

69. Kovacs, R. *et al. TrussFormer: 3D printing large kinetic structures* in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), 113–125.

70. Caluwaerts, K. *et al.* Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of The Royal Society Interface* **11.** Publisher: Royal Society, 20140520. https://royalsocietypublishing.org/doi/10.1098/rsif.2014.0520 (2024) (Sept. 2014).

71. Lessard, S. *et al. A bio-inspired tensegrity manipulator with multi-DOF, structurally compliant joints* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* ISSN: 2153-0866 (Oct. 2016), 5515–5520. (2024).

72. Sabelhaus, A. P. *et al. Mechanism Design and Simulation of the ULTRA Spine: A Tensegrity Robot* en. in (American Society of Mechanical Engineers Digital Collection, Jan. 2016). https://dx.doi.org/10.1115/DETC2015-47583 (2024).

73. Bruce, J. *et al.* SUPERball: Exploring Tensegrities for Planetary Probes. en.

74. Liu, Y. *et al.* A review on tensegrity structures-based robots. *Mechanism and Machine Theory* **168,** 104571. ISSN: 0094-114X. https://www.sciencedirect.com/science/article/pii/S0094114X21003153 (2024) (Feb. 2022).

75. Hu, J., Whitman, J. & Choset, H. GLSO: Grammar-guided Latent Space Optimization for Sample-efficient Robot Design Automation. en.

76. Kim, J. T., Park, J., Choi, S. & Ha, S. *Learning Robot Structure and Motion Embeddings using Graph Neural Networks* en. arXiv:2109.07543 [cs]. Sept. 2021. http://arxiv.org/abs/2109.07543 (2024).

77. Hu, J., Whitman, J., Travers, M. & Choset, H. *Modular Robot Design Optimization with Generative Adversarial Networks* en. in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, Philadelphia, PA, USA, May 2022), 4282–4288. ISBN: 978-1-72819-681-7. https://ieeexplore.ieee.org/document/9812091/ (2024).

78. Gupta, A., Fan, L., Ganguli, S. & Fei-Fei, L. *MetaMorph: Learning Universal Controllers with Transformers* arXiv:2203.11931 [cs]. Mar. 2022. http://arxiv.org/abs/2203.11931 (2024).

79. Xu, X., Jayaraman, P. K., Lambourne, J. G., Willis, K. D. D. & Furukawa, Y. *Hierarchical Neural Coding for Controllable CAD Model Generation* en. June 2023. https://arxiv.org/abs/2307.00149v1 (2024).

80. LeClerc, V., Parkes, A. & Ishii, H. *Senspectra: A computationally augmented physical modeling toolkit for sensing and visualization of structural strain* in *Proceedings of the SIGCHI conference on Human factors in computing systems* (2007), 801–804.

81. Parkes, A. & Ishii, H. *Bosu: a physical programmable design tool for transformability with soft mechanics* in *Proceedings of the 8th ACM Conference on Designing Interactive Systems* (2010), 189–198.

82. Tahouni, Y., Qamar, I. P. & Mueller, S. *NURBSforms: A Modular Shape-Changing Interface for Prototyping Curved Surfaces* in *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction* (2020), 403–409.

83. Wang, G. *et al. 4DMesh: 4D printing morphing non-developable mesh surfaces* in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), 623–635.

84. Hjelle, D. & Lipson, H. *A robotically reconfigurable truss* in *2009 ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots* (2009), 73–78.

85. Galloway, K. C., Jois, R. & Yim, M. *Factory floor: A robotically reconfigurable construction platform* in *2010 IEEE International Conference on Robotics and Automation* (2010), 2467–2472.

86. Leen, D., Ramakers, R. & Luyten, K. *StrutModeling: A low-fidelity construction kit to iteratively model, test, and adapt 3D objects* in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (2017), 471–479.

87. Sofla, A., Elzey, D. & Wadley, H. Shape morphing hinged truss structures. *Smart Materials and Structures* **18,** 065012 (2009).

88. drcmda. *React-Three-Fiber, github repository* https://github.com/pmndrs/react-three-fiber. 2021.

89. Jain, S & Kramer, S. Forward and inverse kinematic solution of the variable geometry truss robot based on an N-celled tetrahedron-tetrahedron truss (1990).

90. Doncieux, S., Bredeche, N., Mouret, J.-B. & Eiben, A. E. G. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI* **2,** 4 (2015).

91. Lipson, H. & Pollack, J. B. Automatic design and manufacture of robotic lifeforms. en. **406** (2000).

92. Shah, D. S. *et al.* A soft robot that adapts to environments through shape change. en. *Nature Machine Intelligence* **3.** Publisher: Nature Publishing Group, 51–59. ISSN: 2522-5839. https://www.nature.com/articles/s42256-020-00263-1 (2024) (Jan. 2021).

93. Belke, C. H., Holdcroft, K., Sigrist, A. & Paik, J. Morphological flexibility in robotic systems through physical polygon meshing. en. *Nature Machine Intelligence* **5.** Publisher: Nature Publishing Group, 669–675. ISSN: 2522-5839. https://www.nature.com/articles/s42256-023-00676-8 (2024) (June 2023).

94. Usevitch, N. S., Halsted, T., Hammond, Z. M., Okamura, A. M. & Schwager, M. *Distributed Control of Truss Robots Using Consensus Alternating Direction Method of Multipliers* arXiv:2108.06577 [cs]. Aug. 2021. http://arxiv.org/abs/2108.06577 (2023).

95. Usevitch, N. S. *et al.* An untethered isoperimetric soft robot. *Science Robotics* **5.** Publisher: American Association for the Advancement of Science, eaaz0492. https://www.science.org/doi/10.1126/scirobotics.aaz0492 (2023) (Mar. 2020).

96. Zagal, J. C., Armstrong, C. & Li, S. *Deformable Octahedron Burrowing Robot* en. in *Artificial Life 13* (MIT Press, July 2012), 431–438. ISBN: 978-0-262-31050-5. https://www.mitpressjournals.org/doi/abs/10.1162/978-0-262-31050-5-ch057 (2023).

97. Spinos, A. & Yim, M. A Linking Invariant for Truss Robot Motion Planning. *IEEE Robotics and Automation Letters* **7.** Conference Name: IEEE Robotics and Automation Letters, 1424–1430. ISSN: 2377-3766. https://ieeexplore.ieee.org/abstract/document/9669032 (2023) (Apr. 2022).

98. Liu, C. & Yim, M. *Reconfiguration Motion Planning for Variable Topology Truss* in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* ISSN: 2153-0866 (Nov. 2019), 1941–1948.

99. Sofla, A. Y. N., Elzey, D. M. & Wadley, H. N. G. Shape morphing hinged truss structures. en. *Smart Materials and Structures* **18,** 065012. ISSN: 0964-1726. https://dx.doi.org/10.1088/0964-1726/18/6/065012 (2023) (May 2009).

100. Torres-Oviedo, G., Macpherson, J. M. & Ting, L. H. Muscle synergy organization is robust across a variety of postural perturbations. eng. *Journal of Neurophysiology* **96,** 1530–1546. ISSN: 0022-3077 (Sept. 2006).

101. Ting, L. H. & Macpherson, J. M. A limited set of muscle synergies for force control during a postural task. eng. *Journal of Neurophysiology* **93,** 609–613. ISSN: 0022-3077 (Jan. 2005).

102. Macpherson, J. M. Strategies that simplify the control of quadrupedal stance. II. Electromyographic activity. eng. *Journal of Neurophysiology* **60,** 218–231. ISSN: 0022-3077 (July 1988).

103. d'Avella, A., Saltiel, P. & Bizzi, E. Combinations of muscle synergies in the construction of a natural motor behavior. eng. *Nature Neuroscience* **6,** 300–308. ISSN: 1097-6256 (Mar. 2003).

104. Rana, M., Yani, M. S., Asavasopon, S., Fisher, B. E. & Kutch, J. J. Brain Connectivity Associated with Muscle Synergies in Humans. *The Journal of Neuroscience* **35,** 14708–14716. ISSN: 0270-6474. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4635125/ (2024) (Nov. 2015).

105. Kovacs, R. *et al. TrussFab: Fabricating Sturdy Large-Scale Structures on Desktop 3D Printers* en. in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (ACM, Denver Colorado USA, May 2017), 2606–2616. ISBN: 978-1-4503-4655-9. https://dl.acm.org/doi/10.1145/3025453.3026016 (2023).

106. Rosendo, A., Von Atzigen, M. & Iida, F. The trade-off between morphology and control in the co-optimized design of robots. en. *PLOS ONE* **12** (ed Bongard, J.) e0186107. ISSN: 1932-6203. https://dx.plos.org/10.1371/journal.pone.0186107 (2023) (Oct. 2017).

107. Fang, Y. *et al. Complex Locomotion Skill Learning via Differentiable Physics* June 2022. http://arxiv.org/abs/2206.02341 (2023).

108. Howard, D. *et al.* Evolving embodied intelligence from materials to machines. en. *Nature Machine Intelligence* **1.** Publisher: Nature Publishing Group, 12–19. ISSN: 2522-5839. https://www.nature.com/articles/s42256-018-0009-9 (2024) (Jan. 2019).

109. *L-systems driven self-reconfiguration of modular robots - Dongyang Bie, Yanhe Zhu, Xiaolu Wang, Yu Zhang, Jie Zhao, 2016* https://journals.sagepub.com/doi/10.1177/1729881416669349 (2024).

110. Raibert, M. H. Symmetry in Running. *Science* **231.** Publisher: American Association for the Advancement of Science, 1292–1294. https://www.science.org/doi/abs/10.1126/science.3945823 (2023) (Mar. 1986).

111. Bongard, J., Zykov, V. & Lipson, H. Resilient machines through continuous self-modeling. eng. *Science (New York, N.Y.)* **314,** 1118–1121. ISSN: 1095-9203 (Nov. 2006).

112. Rothemund, P. *et al.* A soft, bistable valve for autonomous control of soft actuators. *Science Robotics* **3.** Publisher: American Association for the Advancement of Science, eaar7986. https://www.science.org/doi/full/10.1126/scirobotics.aar7986 (2024) (Mar. 2018).

113. Preston, D. J. *et al.* Digital logic for soft devices. *Proceedings of the National Academy of Sciences* **116.** Publisher: Proceedings of the National Academy of Sciences, 7750–7759. https://www.pnas.org/doi/full/10.1073/pnas.1820672116 (2024) (Apr. 2019).

114. Guo, H., Saed, M. O. & Terentjev, E. M. Heliotracking Device using Liquid Crystalline Elastomer Actuators. en. *Advanced Materials Technologies* **6,** 2100681. ISSN: 2365-709X. (2023) (2021).

115. Webster-Wood, V. A., Akkus, O., Gurkan, U. A., Chiel, H. J. & Quinn, R. D. Organismal engineering: Toward a robotic taxonomic key for devices using organic materials. en. *Science Robotics* **2,** eaap9281. ISSN: 2470-9476. https://www.science.org/doi/10.1126/scirobotics.aap9281 (2024) (Nov. 2017).

116. Ricotti, L. *et al.* Biohybrid actuators for robotics: A review of devices actuated by living cells. eng. *Science Robotics* **2,** eaaq0495. ISSN: 2470-9476 (Nov. 2017).

117. Won, P., Ko, S. H., Majidi, C., W. Feinberg, A. & A. Webster-Wood, V. Biohybrid Actuators for Soft Robotics: Challenges in Scaling Up. en. *Actuators* **9.** Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, 96. ISSN: 2076-0825. https://www.mdpi.com/2076-0825/9/4/96 (2024) (Dec. 2020).

118. Sun, W. *et al.* 3D Printing Hydrogel-Based Soft and Biohybrid Actuators: A Mini-Review on Fabrication Techniques, Applications, and Challenges. English. *Frontiers in Robotics and AI* **8.** Publisher: Frontiers. ISSN: 2296-9144. https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.673533/full (2024) (Apr. 2021).

119. Grabowski, P., Haberko, J. & Wasylczyk, P. Photo-Mechanical Response Dynamics of Liquid Crystal Elastomer Linear Actuators. *Materials* **13,** 2933. ISSN: 1996-1944. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7372473/ (2023) (June 2020).

120. *Design and printing of proprioceptive three-dimensional architected robotic metamaterials | Science* https://www.science.org/doi/10.1126/science.abn0090 (2024).

121. Morales Ferrer, J. M., Sánchez Cruz, R. E., Caplan, S., van Rees, W. M. & Boley, J. W. Multiscale Heterogeneous Polymer Composites for High Stiffness 4D Printed Electrically Controllable Multifunctional Structures. en. *Advanced Materials* **36,** 2307858. ISSN: 1521-4095. https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202307858 (2024) (2024).

122. Sigmund, O. & Maute, K. Topology optimization approaches. en. *Structural and Multidisciplinary Optimization* **48,** 1031–1055. ISSN: 1615-1488. https://doi.org/10.1007/s00158-013-0978-6 (2023) (Dec. 2013).

123. Tcherniak, D. Topology optimization of resonating structures using SIMP method. en. *International Journal for Numerical Methods in Engineering* **54,** 1605–1622. ISSN: 1097-0207. (2023) (2002).

124. Katoch, S., Chauhan, S. S. & Kumar, V. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications* **80,** 8091–8126 (2021).

125. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. en. *IEEE Transactions on Evolutionary Computation* **6,** 182–197. ISSN: 1089778X. http://ieeexplore.ieee.org/document/996017/ (2023) (Apr. 2002).

126. Liu, D. C. & Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical programming* **45,** 503–528 (1989).

127. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* **17,** 261–272 (2020).

128. Gu, J. *et al. PneuMesh: Pneumatic-driven Truss-based Shape Changing System* in *CHI Conference on Human Factors in Computing Systems* (2022), 1–12.

129. Rhodes, M. D. & Mikulas Jr, M. M. *Deployable controllable geometry truss beam* tech. rep. (1985).

130. Usevitch, N. S. *et al.* An untethered isoperimetric soft robot. *Science Robotics* **5,** eaaz0492 (2020).

131. Kovacs, R. *et al. TrussFormer: 3D printing large kinetic structures* in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), 113–125.

132. Cheney, N., MacCurdy, R., Clune, J. & Lipson, H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution* **7,** 11–23 (2014).

133. Zhao, A. *et al.* Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)* **39,** 1–16 (2020).

134. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

135. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* **6,** 182–197 (2002).

136. Kostrikov, I. *PyTorch Implementations of Reinforcement Learning Algorithms* https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail. 2018.

137. Sanchez-Gonzalez, A. *et al. Learning to simulate complex physics with graph networks* in *International Conference on Machine Learning* (2020), 8459–8468.

138. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

139. Xu, X. *et al. SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled Codebooks* en. arXiv:2207.04632 [cs]. July 2022. http://arxiv.org/abs/2207.04632 (2024).

140. Guo, M. *et al. Hierarchical Grammar-Induced Geometry for Data-Efficient Molecular Property Prediction* en. in *Proceedings of the 40th International Conference on Machine Learning* ISSN: 2640-3498 (PMLR, July 2023), 12055–12076. https://proceedings.mlr.press/v202/guo23h.html (2024).

141. Wang, G. *et al. 4DMesh: 4D Printing Morphing Non-Developable Mesh Surfaces* in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Association for Computing Machinery, New York, NY, USA, Oct. 2018), 623–635. ISBN: 978-1-4503-5948-1. https://dl.acm.org/doi/10.1145/3242587.3242625 (2024).

142. Shah, D. S. *et al.* Tensegrity Robotics. en. *Soft Robotics* **9,** 639–656. ISSN: 2169-5172, 2169-5180. https://www.liebertpub.com/doi/10.1089/soro.2020.0170 (2024) (Aug. 2022).

143. Panetta, J. *et al.* Elastic textures for additive fabrication. *ACM Trans. Graph.* **34,** 135:1–135:12. ISSN: 0730-0301. https://dl.acm.org/doi/10.1145/2766937 (2024) (July 2015).

144. Zhao, D. & Liu, Y. A prototype for light-electric harvester based on light sensitive liquid crystal elastomer cantilever. *Energy* **198,** 117351. ISSN: 0360-5442. https://www.sciencedirect.com/science/article/pii/S0360544220304588 (2024) (May 2020).

145. Vespignani, M., Friesen, J. M., SunSpiral, V. & Bruce, J. *Design of SUPERball v2, a Compliant Tensegrity Robot for Absorbing Large Impacts* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* ISSN: 2153-0866 (Oct. 2018), 2865–2871. (2024).

146. Wang, Z., Li, K., He, Q. & Cai, S. A Light-Powered Ultralight Tensegrity Robot with High Deformability and Load Capacity. eng. *Advanced Materials (Deerfield Beach, Fla.)* **31,** e1806849. ISSN: 1521-4095 (Feb. 2019).

147. Zhang, J. & Makoto, O. *Tensegrity structures* (2015).

148. Pietroni, N., Tarini, M., Vaxman, A., Panozzo, D. & Cignoni, P. Position-based tensegrity design. *ACM Trans. Graph.* **36,** 172:1–172:14. ISSN: 0730-0301. https://dl.acm.org/doi/10.1145/3130800.3130809 (2024) (Nov. 2017).

149. Makatura, L. *et al.* Procedural Metamaterials: A Unified Procedural Graph for Metamaterial Design. en. *ACM Transactions on Graphics* **42,** 1–19. ISSN: 0730-0301, 1557-7368. https://dl.acm.org/doi/10.1145/3605389 (2024) (Oct. 2023).

150. Zok, F. W., Latture, R. M. & Begley, M. R. Periodic truss structures. *Journal of the Mechanics and Physics of Solids* **96,** 184–203. ISSN: 0022-5096. https://www.sciencedirect.com/science/article/pii/S0022509615300983 (2024) (Nov. 2016).

151. Jenett, B. *et al.* Discretely assembled mechanical metamaterials. *Science Advances* **6.** Publisher: American Association for the Advancement of Science, eabc9943. https://www.science.org/doi/10.1126/sciadv.abc9943 (2024) (Nov. 2020).

152. Cui, Z. *et al. Robotic Metamaterials: A Modular System for Hands-On Configuration of Ad-Hoc Dynamic Applications* in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Association for Computing Machinery, New York, NY, USA, May 2024), 1–15. ISBN: 9798400703300. https://dl.acm.org/doi/10.1145/3613904.3642891 (2024).

153. Bastek, J.-H., Kumar, S., Telgen, B., Glaesener, R. N. & Kochmann, D. M. Inverting the structure–property map of truss metamaterials by deep learning. *Proceedings of the National Academy of Sciences* **119.** Publisher: Proceedings of the National Academy of Sciences, e2111505119. https://www.pnas.org/doi/10.1073/pnas.2111505119 (2024) (Jan. 2022).

154. Kovacs, R. *et al. TrussFormer: 3D Printing Large Kinetic Structures* in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Association for Computing Machinery, New York, NY, USA, Oct. 2018), 113–125. ISBN: 978-1-4503-5948-1. https://doi.org/10.1145/3242587.3242607 (2024).

155. Bosscher, P. & Ebert-Uphoff, I. *A novel mechanism for implementing multiple collocated spherical joints* in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)* **1.** ISSN: 1050-4729 (Sept. 2003), 336–341 vol.1. https://ieeexplore.ieee.org/abstract/document/1241618 (2024).

156. Collins, F. & Yim, M. *Design of a spherical robot arm with the Spiral Zipper prismatic joint* in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE Press, Stockholm, Sweden, May 2016), 2137–2143. https://doi.org/10.1109/ICRA.2016.7487363 (2024).

157. Usevitch, N. S. *et al.* An untethered isoperimetric soft robot. *Science Robotics* **5.** Publisher: American Association for the Advancement of Science, eaaz0492. https://www.science.org/doi/10.1126/scirobotics.aaz0492 (2024) (Mar. 2020).

158. Zagal, J. C., Armstrong, C. & Li, S. *Deformable Octahedron Burrowing Robot* Pages: 438. ISBN: 978-0-262-31050-5 (July 2012).

159. Zhao, Y.-L. & Stoddart, J. F. Azobenzene-Based Light-Responsive Hydrogel System. *Langmuir* **25.** Publisher: American Chemical Society, 8442–8446. ISSN: 0743-7463. https://doi.org/10.1021/la804316u (2024) (Aug. 2009).

160. Haines, L. A. *et al.* Light-Activated Hydrogel Formation via the Triggered Folding and Self-Assembly of a Designed Peptide. *Journal of the American Chemical Society* **127.** Publisher: American Chemical Society, 17025–17029. ISSN: 0002-7863. https://doi.org/10.1021/ja054719o (2024) (Dec. 2005).

161. Zeng, H., Wani, O. M., Wasylczyk, P., Kaczmarek, R. & Priimagi, A. Self-Regulating Iris Based on Light-Actuated Liquid Crystal Elastomer. en. *Advanced Materials* **29,** 1701814. ISSN: 1521-4095. https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201701814 (2024) (2017).

162. Brannum, M. T. *et al.* Light Control with Liquid Crystalline Elastomers. en. *Advanced Optical Materials* **7,** 1801683. ISSN: 2195-1071. (2024) (2019).

163. Hongo, K. *et al.* en. in *Medical Technologies in Neurosurgery* (eds Nimsky, C. & Fahlbusch, R.) 63–66 (Springer, Vienna, 2006). ISBN: 978-3-211-33303-7. https://doi.org/10.1007/978-3-211-33303-7_9 (2024).

164. Menolotto, M., Komaris, D.-S., Tedesco, S., O'Flynn, B. & Walsh, M. Motion Capture Technology in Industrial Applications: A Systematic Review. en. *Sensors* **20,** 5687. ISSN: 1424-8220. https://www.mdpi.com/1424-8220/20/19/5687 (2024) (Jan. 2020).

165. Xu, X. *et al. SkexGen: Autoregressive Generation of CAD Construction Sequences with Disentangled Codebooks* en. arXiv:2207.04632 [cs]. July 2022. http://arxiv.org/abs/2207.04632 (2024).

166. Vaswani, A. *et al.* Attention is All you Need. en.

167. Siddiqui, Y. *et al. MeshGPT: Generating Triangle Meshes with Decoder-Only Transformers* arXiv:2311.15475. Nov. 2023. http://arxiv.org/abs/2311.15475 (2024).

168. Gupta, A., Fan, L., Ganguli, S. & Fei-Fei, L. *MetaMorph: Learning Universal Controllers with Transformers* arXiv:2203.11931 [cs]. Mar. 2022. http://arxiv.org/abs/2203.11931 (2024).