# GeoMatries: Machine Learning-Empowered Simulation for 4D Printing and Morphing Materials Design

By

Humphrey Yang

## ABSTRACT

Materials in nature are mostly active, responsive, and transformative. Yet in conventional design practices, these features are often neglected and removed from the final products. By contrast, morphing matter design leverages the tunable, temporal properties of materials to program functions into artifacts, and requires accurate physical performance predictions to inform design decisions. 4D printing, in particular, is an additive manufacturing technique that manipulates residual stress to create stimuli-responsive, shape-changing artifacts. However, due to the lack of a fast and physically accurate simulation method to inform design decisions, the current design workflow of 4D printing requires intensive physical prototyping to iterate designs, making it a slow, inefficient, and indirect process.

This thesis takes 4D printing as an example of morphing matter design and responds to the workflow challenges mentioned above with SimuLearn, a data-driven simulation technique that combines numerical methods and machine learning, to mitigate the need of physical prototyping in design iterations. Compared to finite element analysis, our results show that this technique can simulate physical transformations with speed (1500 times faster) while having an identical accuracy (1.6% maximum relative error). Workflows adopting SimuLearn will be able to afford interactive and physically-informed digital iterations and extend the design space of 4D printing. Additionally, a prototype computer-aided morphing matter design tool is also implemented to expose the development guidelines of tools that adopt SimuLearn and is deployed in several design tasks to demonstrate its applicability and potential. Lastly, this thesis will also discuss the limitations, generalizability, and potential improvements of SimuLearn to guide future works and real-world deployments.

Thesis advisors:

Dr. Daniel Cardoso Llach

Dr. Lining Yao

“You say to a brick, ‘What do you want, brick?’ And brick says to you, ‘I like an arch.’ And you say to brick, ‘Look, I want one, too, but arches are expensive and I can use a concrete lintel.’ And then you say: ‘What do you think of that, brick?’ Brick says: ‘I like an arch.’”
- Louis Kahn. (Kahn, 1971)

# ACKNOWLEDGMENT

First and foremost, I want to thank my collaborators form the SimuLearn team - Yuxuan Yu, Haolin Liu, Kuanren Qian, Jianzhe Gu, Dr. Yongjie Jessica Zhang, and Dr. Lining Yao - for helping me with various aspects of this thesis and supporting me through numerous challenges over the past year. Needless to say, this thesis is an interdisciplinary research that I cannot accomplish alone, and I have learned much from each of my teammates and supervisors. In particular, I want to thank Haolin for always making time for me when I need technical supports and encouraging me through the challenges of writing this thesis.

My colleagues from the CodeLab are companions that I respect and love in the past two years. I enjoy the early-morning Inquiry talks about various aspects of computational design, the late-night debugging marathons when we took 112 together, or the intellectual brawls happened during pre-thesis seminars. These are the memories that I will cherish for the years to come, but among all of the times we spent together, I adore most the conversations we had about human-machine collaborations in design. Those are the discussions that inspired me to think about the future of computational design and are worth ruminating for the rest of my life. It is a great honor to be a part of the cohort, and I wish everyone good luck with the journeys to come.

Members of the morphing matter lab also played essential roles during my time at CMU, especially Guanyun Wang, Zeyu Yan, and Jianzhe Gu. They are the best of friends as well as the sharpest of minds to work with. In particular, Zeyu is the friend that helped me balance life under the pressure of CMU workloads. I also want to thank Jack Forman for mutual support during graduate school applications.

Among all people at CMU, I am most grateful to my advisors Daniel and Lining. I could not have become a good researcher without Daniel's training nor finish this thesis without his guidance. His lectures are what inspired the conceptualization of this thesis. Similarly, Lining is also a great mentor as well as friend to work with. In addition to her teaching, I also appreciate her words of encouragement along the journey.

My journey into computational design began at the National Cheng Kung University, and could not have come true without the mentorship from Professor Hsueh. He was the one who supported me through the moments of doubt of being an architect and enlightened me to rethink the role of architects beyond mere designers of buildings.

Lastly, I want to thank my family for the unconditional support throughout my life. Words fall short of expressing my gratitude toward them, especially Chia Ping, for accommodating our long-distance relationship. I could not have finished this thesis without her support and encouragements, our love is the beacon that led me through the darkest hours of life.

# TABLE OF CONTENTS

# INTRODUCTION

In this chapter, we will motivate and situate material-driven designs in contemporary practices, and use 4D printing as an example to expose the intricacies in material-driven design workflows. Next, we will hypothesize a strategy of CAD tool design to respond to the most significant barrier of realizing a material-driven design practice - simulations.

## Motivating a Material-Driven CAD Practice

Materials in nature are active, responsive, and transformative. Leaves curl in response to moisture changes in the environment; muscles retract in response to neural signals; even non-organic materials like crystals may vibrate when subjected to electric currents. These feedback loops are the most ubiquitous form of computation around us. However, in conventional design practices, these features are often neglected and removed from artifacts. We adore timber for its texture but dislike its moisture-responsive curving; we use thermoplastics for 3D printing but gets frustrated when they warp due to its residual stress. Inspecting common design practices, fabrication tools, and computer-aided design (CAD) software. Kahn's quote serves as an incisive criticism of the issues that current design practices are facing- the absence of active material properties.

Early developments of CAD were founded on the manufacturing of static materials (Cardoso Llach., 2015). Both its representations, tools, and fabrication machines were built for modeling shapes with precision and often neglects the active properties of materials. More recently, while new software systems like *Abaqus* (Dassault Systems., 2019) have emerged and afford us to accurately compute material deformations, they are often oriented toward engineering purposes, creating a division between engineering and design labors in material-driven CAD practices. This thesis uses 4D printing as an example of such design practices and takes the initial step to reconcile materials and geometries in the digital realm.

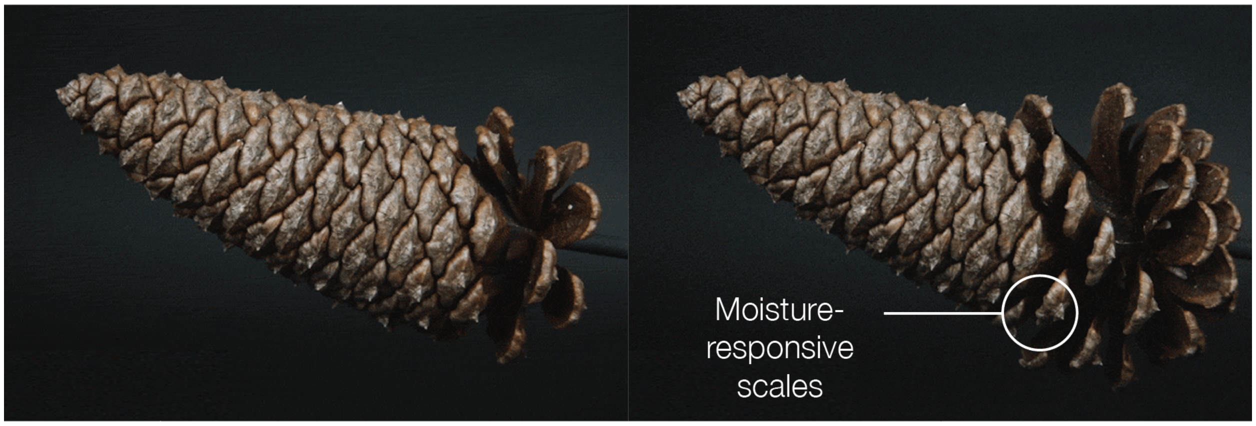


**Figure 01. Pinecone Transformation. Pinecones will transform to release its pollens or seeds when exposed to moisture. This mechanism assures the seeds to grow in ideal environments. Images courtesy of Morphing Matter Lab, CMU.**

## Challenges of Current 4D Printing Design Tools

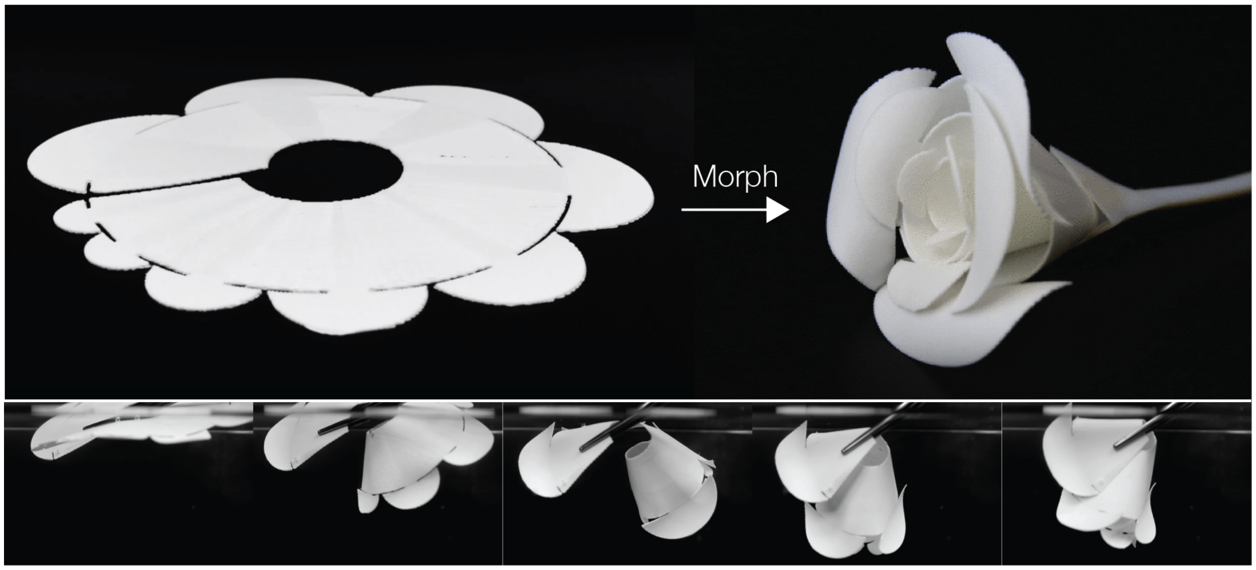


**Figure 02. 4D printed *Thermorph* rose. The artifact is printed with a desktop 3D printer as a flat sheet and triggered in hot water to transform into the targeted rose shape. Images reprinted from *Thermorph* (An et al., 2018).**

In contrast to conventional design practices, 4D printing (Tibbits., 2014) is a technique that leverages additive manufacturing and material anisotropy to produce active objects that can transform over time. This technique takes what is conventionally considered a defect in materials engineering, deformations, and leverages it to design transformative artifacts. Once exposed to the correct stimuli, these artifacts will transform to take a different shape (Figure 02). Using 4D printing as a manufacturing approach, researchers are envisioning electricity-free, gearless actuators and self-deploying structures that can improve fabrication time, reduce manufacturing complexity, and save packaging spaces.

While 4D printing has been applied to various material systems, thermoplastics have been the most popular subject due to the wide adoption of hobbyist fused deposition modeling (FDM) 3D printers (Figure 03). These personal fabrication devices allow users to combine rapid prototyping with CAD seamlessly, and 4D printing will further strengthen this convenience by making design iterations more economical in terms of time and material costs. However, several issues inhibit the wide adoption of this fabrication method.

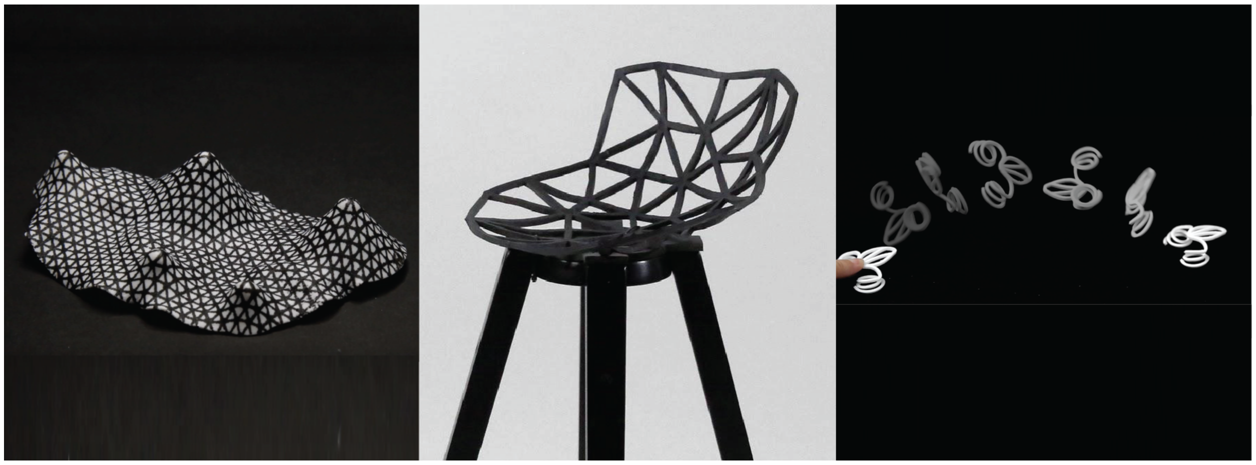


**Figure 03. Thermoplastics-based 4D Printing Design Space. Left - a landscape model from *Geodesy*; middle - a seat of chair made with *4DMesh*; right - a jumping frog toy appeared in *A-line*. Images reprinted from *Geodesy* (Gu et al., 2019), *4DMesh* (Wang et al., 2018), and *A-line* (Wang et al., 2019)**

## What You See Is Not What You Get

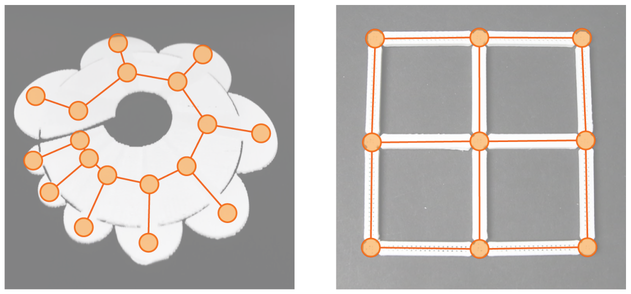


**Figure 04. Comparison of topologies. The rose (left) from *Thermorph* (An et al., 2018) has a tree-like topology and the grid design (right) from *4DMesh* (Wang et al., 2018) has a net-like topology.**

In conventional design practices of 3D printing, the model we see in CAD environments are accurate representations of the artifact. Whereas in 4D printing, the model represents either the pre-triggering or transformed shape and does not represent the artifact's temporal statuses. For simple geometries that have tree-like topologies, such as *A-line* (Wang et al., 2019) and the origami structures of *Thermorph* (An et al., 2018), we can find simplified abstractions (e.g., forward kinematics and parametric geometries) to visualize the transformation process. However, for complex geometries that have net-like topologies, such as the mesh patterns of *4DMesh* (Wang et al., 2018) and *Geodesy* (Gu et al., 2019), their transformations are much harder to predict as they involve dynamic interactions between elements (Figure 04). Moreover, in larger scales, physical factors like gravity, buoyancy, fixed-end conditions, and nonlinear material properties also become dominant in transformation processes and cannot be ignored.

## Unlimited Design Space

Standard CAD tools like *Rhinoceros* (Robert McNeel & Associates., 2019) are designed for general modeling purposes, providing us with a myriad of modeling commands - the metaphorical tools - to choose from and wield at will to create arbitrary designs across scales. However, given a 4D printing technique and material system, most of the commands may not comply with its limitations and therefore unworkable with. Furthermore, the computational geometries of CAD are designed to represent homogeneous objects, making them less compatible with the core mechanism of 4D printing - material anisotropy. Therefore, the ideal CAD tool for active materials should strategically confine the design space and streamline the design process to avoid users' frustration.

## Design Workflow

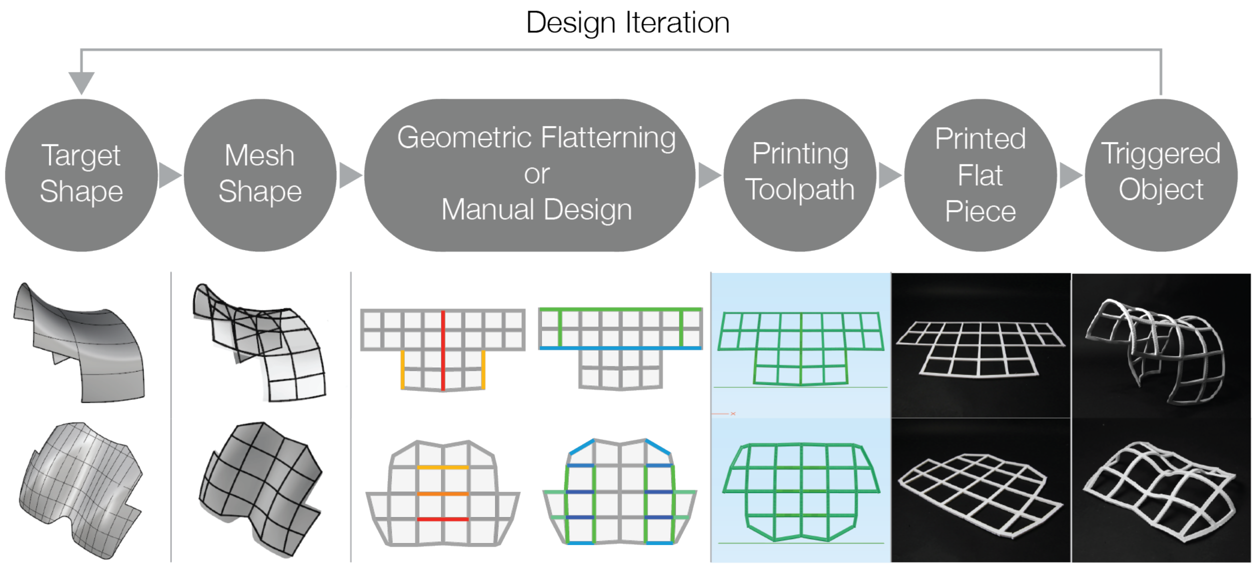
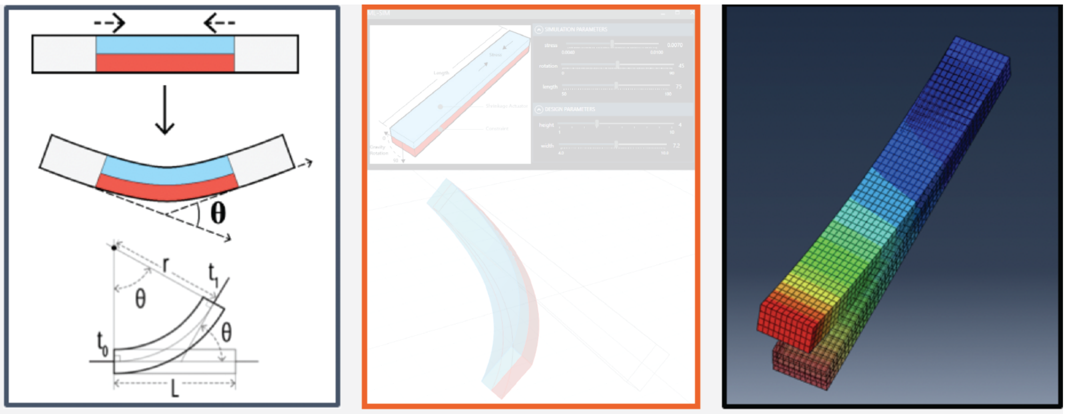


**Figure 05. Current 4D printing design workflow. This workflow demands physically prototyping and indirect pattern manipulation to iterate designs. Images reprinted from *4DMesh* (Wang et al., 2018).**

In this thesis, we categorize 4D printing design workflows into two kinds - forward- and inverse-design workflows. Forward-design workflows allow users to freely configure materials and design without an explicit goal in mind; inverse-design workflows demand users to have a transformation target prior to using the tools. Both workflows share a common issue - inefficient design iterations. Users must physically prototype the designs in order to evaluate them, which takes a long time to perform (Figure 05). The printing process itself may easily take tens of minutes or hours to perform. Instead, if the design tool affords users to iterate without having to prototype the artifacts physically, the workflow will become truly economical and efficient.

## Simulation Tools

Currently, there are two simulation methods available to us - geometrical and numerical methods. On the one hand, geometrical methods are fast to compute but are often bespoke and ignore certain physical factors in the computation, resulting in physical inaccuracy. On the other hand, numerical methods such as finite element analysis (FEA) are generalized but prohibitively slow to compute and does not provide real-time feedback. Both methods have their targeted applications - geometrical methods are commonly used in the entertainment industries, where the audiences require only fast and seemingly correct simulations; numerical methods are often used for engineering purposes, where the priority is physical accuracy and computational time is less crucial. However, the ideal simulation method for 4D printing design tools should be sufficiently accurate to inform design decisions and reasonably fast to facilitate real-time design iterations (Figure 06).



| Method | Geometrical | Ideal | Numerical |
|---|---|---|---|
| Speed | Fast (0.02 sec.) | Relatively Fast (1 sec.) | Slow (36 sec.) |
| Phyiscal Accuracy | Inaccurate | Sufficiently Accurate (< 5%) | Accurate |
| Target Domain | Graphics | Engineering | Engineering |

**Figure 06. Simulation method comparison. This comparison is based on a thermoplastic beam of dimension 75 mm * 7.2 mm * 4 mm. Images reprinted from *4DMesh* (Wang et al., 2018).**

## Hypothesis and Goal

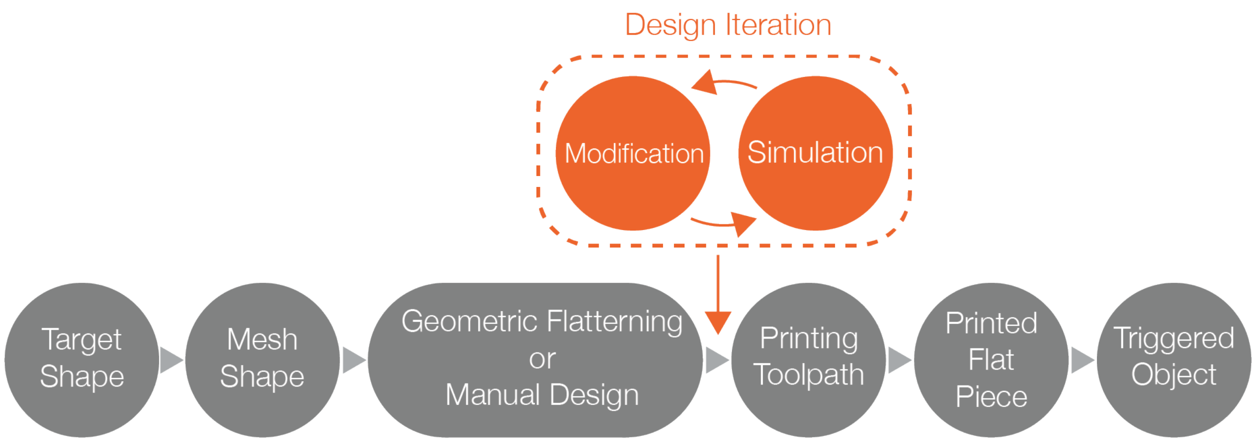


**Figure 07. Hypothesis overview. We propose a data-driven simulation technique and design tool to augment 4D printing design workflows, such that iterations no longer require physical prototyping.**

This thesis proposes a data-driven simulation technique to augment 4D printing and morphing matter design workflows. This simulation method takes FEA results as the source of data to guarantee sufficient physical accuracy and inform design decisions in CAD tools, and use machine learning (ML) regressors to generalize from the data and facilitate efficient computations. Specifically, the maximum simulation error should be smaller than the material's compliance (<5% for the material system of this work), and the speed should be fast enough to enable real-time iterations (~ 1 second). This simulation approach will breakdown a design into consistently represented building blocks - actuators - to simulate their transformations and enable geometrical reconfigurations of design. Design tools adopting this simulation engine will afford users to, in a bottom-up and forward-design manner, design patterns by composing actuators together without needing to have an explicit target shape in mind. Workflows adopting this tool also does not require physical prototyping to iterate designs (Figure 07). Compared to designing with inverse-design algorithms, this modality of design also liberates users from the geometrical limitations of inverse-design algorithms.

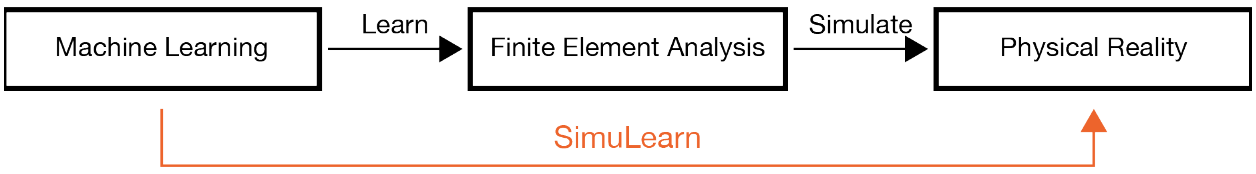


**Figure 08. Data-driven simulation.**

Based on the hypothesis above, this thesis will provide a pipeline to develop such data-driven simulators (also known as SimuLearn, Figure 08), utilize it to derive proofs-of-concept for this technique, and illustrate the design tool's capability with several design tasks.

## Outline

**Vision and Concept (Chapter 1 and 3)**

- Simulation-empowered design workflows.
- Hierarchical composition of 4D printing.
- Data-driven simulators and design tools.

**Technical Contributions (Chapter 3 - 4)**

- SimuLearn and its development pipeline.
- Graph theories for 4D printing.
- Design tool and Evaluations.

**Artifacts (Chapter 4)**

- A dataset of FEA results and a proofs-of-concept SimuLearn engine
- Design tool prototype and application examples.

With the Introduction establishing the background knowledge, the next chapter will situate this work in related literature to motivate its relevance and discuss several works that we take as inspiration. In the Methods chapter, an overview is provided to explain the experimental perimeters and high-level concepts. We will discuss our technical implementations in two-folds - considerations that apply to different materials and implementation details that are specific to our chosen material system. Next, a Results section will compare an instance of SimuLearn engines with FEA to provide performance evaluations as proofs-of-concept and discuss several design examples produced with our design tool prototype to reveal its potential. Lastly, we provide discussions on the limitations, future works, and several topics related to this work.

# RELATED WORK

This chapter is aimed to contextualize the topic, technique, and hypothesis of this thesis, and to credit several previous research that inspired our work. In particular, this chapter answers the "so what" and "how" of a data-driven fast and physically-accurate simulation method.

## Material-Driven Design Paradigms

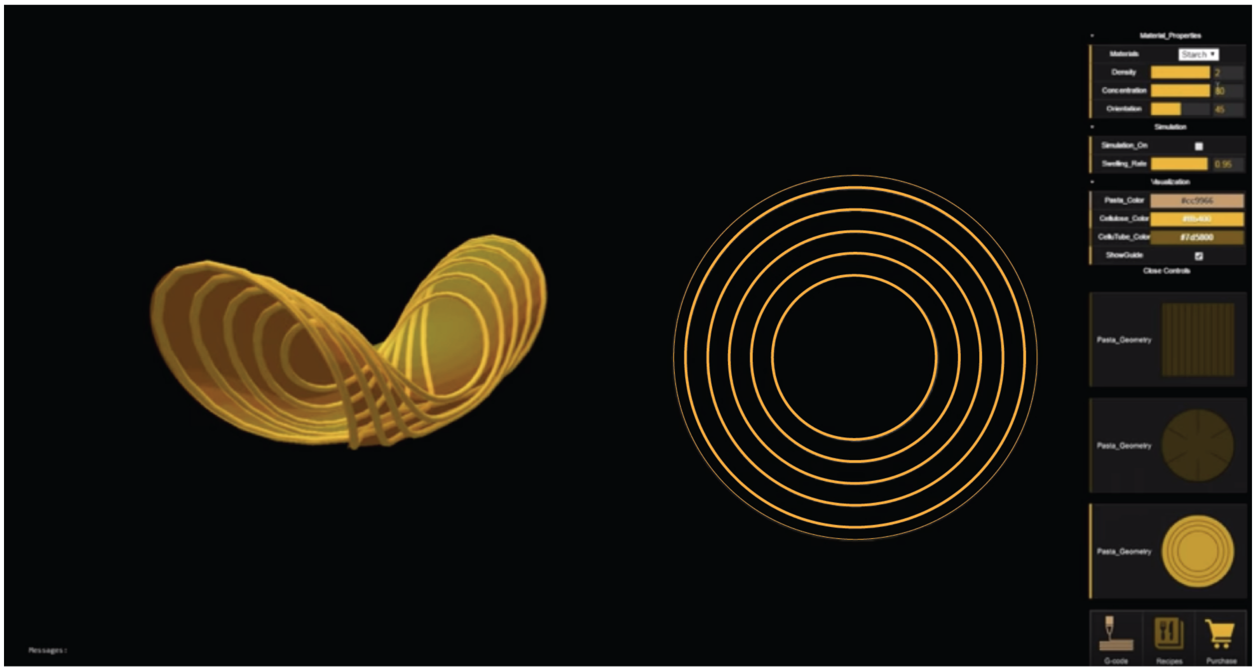

Material_Properties
Materials Starch
Density 2
Concentration 80
Orientation 45
Simulation
Simulation_On
Swelling_Rate 0.95
Visualization
Pasta_Color #cc9966
Cellulose_Color #ffb400
CelluTube_Color #7d5800
ShowGuide ✓

**Figure 09. The design tool of Transformative Appetite. The transformation is simulated by interpolating between pre-computed FEA results. Reprinted from *Transformative Appetite* (Wang et al., 2017).**

Beside thermoplastics, various materials also exhibit material properties that can be leveraged to create self-deploying objects, such as edible gels (Wang et al. 2017) or programmable hydrogels (Gladman et al. 2016). In particular, *Transformative Appetite* (Wang et al. 2017) used FEA and a data-driven method to provide deformation predictions in the design tool implementation (Figure 09), identical to the hypothesis of this thesis. However, their technique is based on geometrically interpolating between pre-calculated shapes to simulate unseen designs, and the capability of such simulator is limited to a selected few geometries. The material system and design space we are targeting at in this thesis, on the other hand, is geometrically variant and have nonlinear behaviors, and therefore cannot be simulated using the method mentioned above.

In addition to 4D printing, material properties and responses also play vital roles in a variety of design practices. For instance, compared to conventional practices, compliant mechanism (Howell et al., 2013) takes a different approach to design mechanical parts by leveraging the flexibility of materials. This method allows us to produce kinetic elements as a single piece and without using gears, allowing for applications across scales and domains - from low-cost packaging to high-cost products; from nanoscale fabrication to large-scale structures; from vehicle parts to medical devices (Figure 10). However, the nonlinear motions and force behaviors of compliant mechanism often cannot be adequately defined with simplified linear equations, and therefore requires simulation or prototyping to inform design iterations - identical to the workflow challenges of 4D printing. Thus, we speculate that this design practice can also benefit from fast and physically accurate simulations.

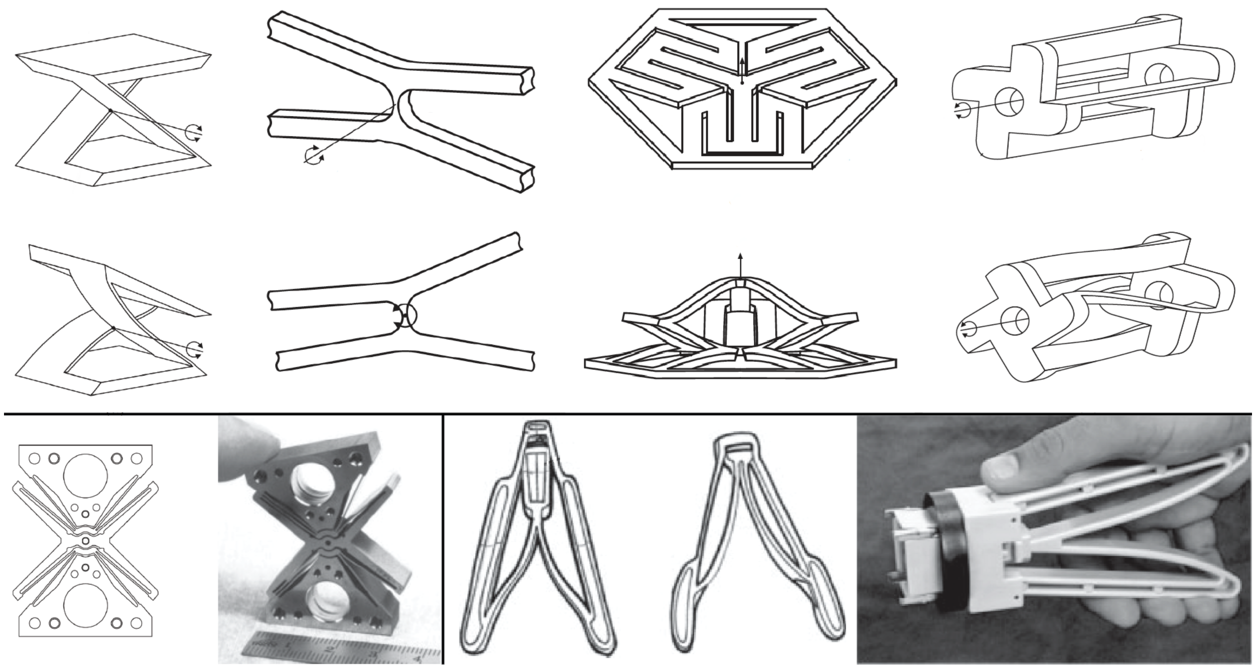


**Figure 10. Compliant Mechanisms. These mechanical structures leverage the elasticity of material to produce functional objects. Images reprinted from *Handbook of compliant mechanisms* (Howell et al., 2013).**

Beyond shape-changing materials, there also exist materials that have variable and tunable mechanical properties. *jamSheets* (Ou et al., 2014) is one of such material systems that has actively tunable stiffnesses. In their work, Ou et al. leveraged layer jamming to produce stiffness- and shape-changing interfaces and afford multi-modal functions. However, *jamSheets* designs also require certain mechanical properties and behaviors to produce functional objects, therefore necessitates physical prototyping or simulation to inform design decisions.

## Machine Learning for Simulation

This thesis takes inspiration from several works that adapt machine learning for simulation to formulate our methods. In particular, our feature vector design is inspired by related works in data-driven computational fluid dynamics (CFD); the technical pipeline is inspired by the literature that use ML to approximate FEA simulation; we take inspiration from ML-empowered Euler method simulations to design geometrically variable ML simulators and model representations.

### Computational Fluid Dynamics

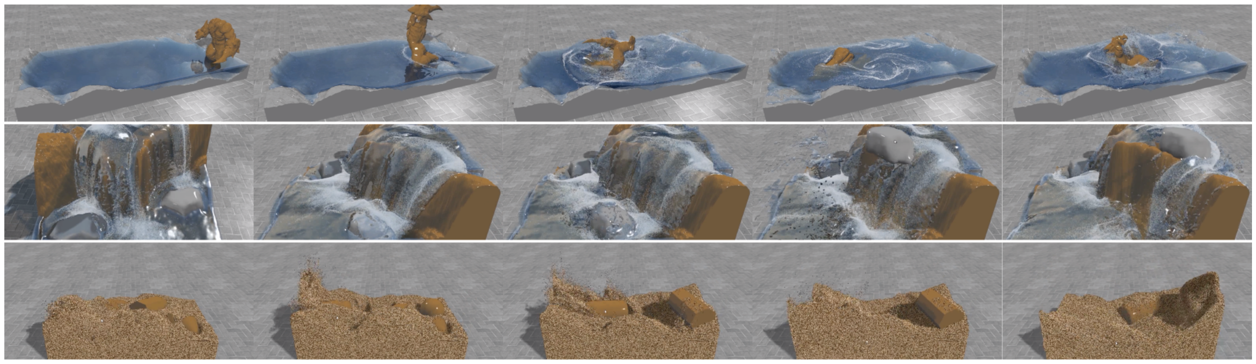


**Figure 11. PhysicsForests. The trained simulator can approximate the behaviors of various types of liquids in different environments. Images reprinted from *Physics Forests* (Apagom AG, 2019).**

In *Data-driven Fluid Simulations using Regression Forests* (Ladicky et al., 2015), the authors used machine learning techniques to speed up computational fluid dynamics. This technique takes a large dataset of simulations produced by a traditional solver to train a regressor to perform physically-based CFD simulations. The underlying abstractions and mechanisms were no different from those of conventional CFD - dividing a scene into numerous particles and compute their behaviors individually to derive their next status in time. Using a traditional CFD solver to compute the behaviors will require small time steps to guarantee the stability of simulations. However, ML regressors were able to approximate the behaviors with a large time step, boosting the computation time by one to three orders of magnitude compared to the state-of-the-art solver on a single GPU, enabling real-time interactive simulations of large scenes. Additionally, due to the variety of materials that appeared in the dataset, the trained simulator was also able to predict the movement of different types of liquid. This work is later packaged into a software system called *Physics Forests* (Apagom AG, 2019) to bring physically-based rapid CFD into animations and entertainment applications (Figure 11).

In their work, Ladicky et al. also illustrated a rule-of-thumb for feature vector design in ML simulators. In addition to the material's intrinsic properties, the feature vector should also describe the element's interactions with the environment. For instance, the feature vector should describe the ambient forces affecting the particle's movement. The algorithm used to calculate the interactions should scale linearly with the number of particles to facilitate the scalability of simulation.

## Finite Element Analysis

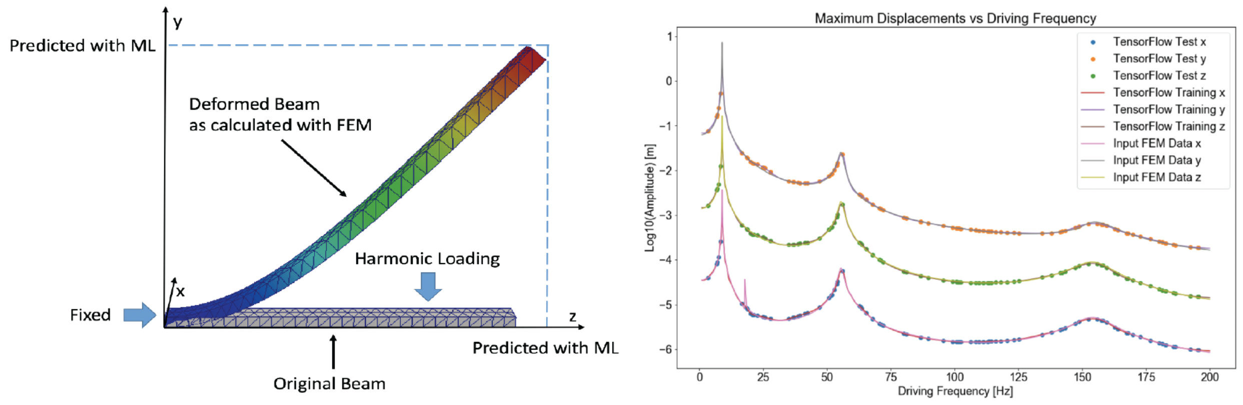


**Figure 12. Using neural networks to approximate steel beam deformation. Reprinted from *ML and FEA for Physical Systems Modeling* (Kononenko et al., 2018).**

Compared to fluid dynamics, FEA computations are even more time-consuming to compute. Whereas a CFD scene can be divided into individual particles for parallel calculation, FEA simulates a scene by solving a large linear system. On a cluster or supercomputer, a large FEA model can easily take hours to weeks to compute. With this inconvenience in mind, Kononenko et al. explored using artificial neural networks to approximate FEA solutions in *Machine Learning and Finite Element Method for Physical Systems Modeling* (Kononenko et al. 2018). In this work, the task is to predict the deformation of a steel beam under oscillating loads, and the ML model takes the boundary condition (load amplitude and frequency) of a scene as input to compute the steel beam's maximum deformation. Note that the feature vector does not describe the beam's geometry and the resulting trained ML model can only simulate a specific beam model. The validation results showed that ML regressors can successfully and accurately approximate FEA simulations (Figure 12). Furthermore, the authors also provided an implementation framework to develop FEA-based ML simulators.

While they showed that the technique is viable, Kononenko et al. also explained that developing a reliable ML-FEA simulator that takes multiple boundary conditions as input will require a large amount of data. This statement also agrees with the conclusion of an earlier work that adapts *Machine Learning for Modeling the Biomechanical Behavior of Human Soft Tissue* (Martin-Guerrero et al., 2016). In particular, the ML simulators in both works can only account for small or no model variations and are therefore geometrically limited. Nonetheless, these works both envisioned that using ML to approximate FEA will have applications in a variety of domains, including science, engineering, medicine, and entertainment.

## Euler Methods

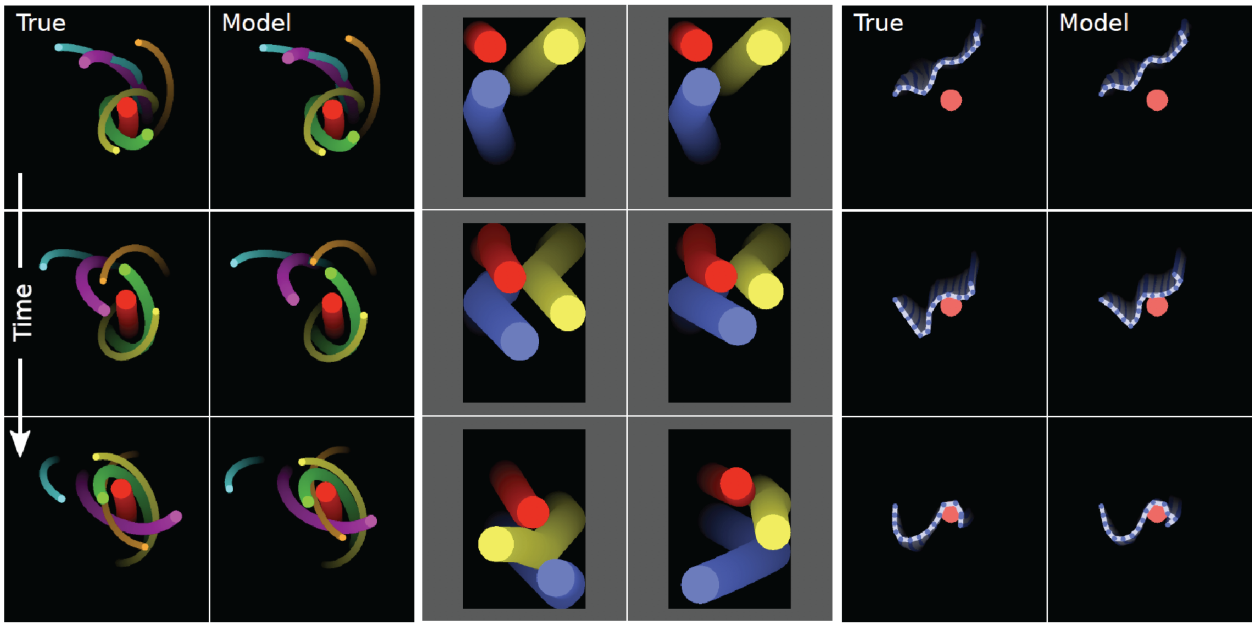


**Figure 13. Simulations with Interaction Network. Left - n-body gravity system; Middle - bouncing balls; Right - Spring-mass model. Reprinted from *Interaction Network* (Battaglia et al., 2016)**

Euler methods are simulations that are based on integrating a differentiable function over time, such as spring mass models or n-body gravitational systems. These tasks are often decomposed into smaller elements for simulation and reconfiguration, and the interactions between elements cannot be disregarded during simulations. Therefore, a critical challenge of adapting machine learning to Euler methods is to account for the interactions and reconfigurations. Among available literature in this domain, *Interaction Networks* (Battaglia et al., 2016) is most relevant to this thesis. This work leveraged graphical representations to develop a relation-centric and object-oriented ML model to perform simulations, such that the trained simulator is easily scalable and reconfigurable for different scenarios (Figure 13). Furthermore, the ML model computes the interactions of elements by exploiting the explicit structure of graphs, therefore is more efficient compared to convolutional neural networks (CNN).

In addition to its scalability and reconfigurability, *Interaction Networks* also allows sharing learning across elements in graphs, reducing the amount of data required for training. This work also adopts an iterative strategy of simulation to perform dynamic simulations[1] and enable extracting more training data from one episode of Euler method simulation. Later, this simulation technique is developed into *Graph Networks* (Sanchez-Gonzalez et al., 2018) for robotic simulation and adapted for reinforcement learning (Figure 14).

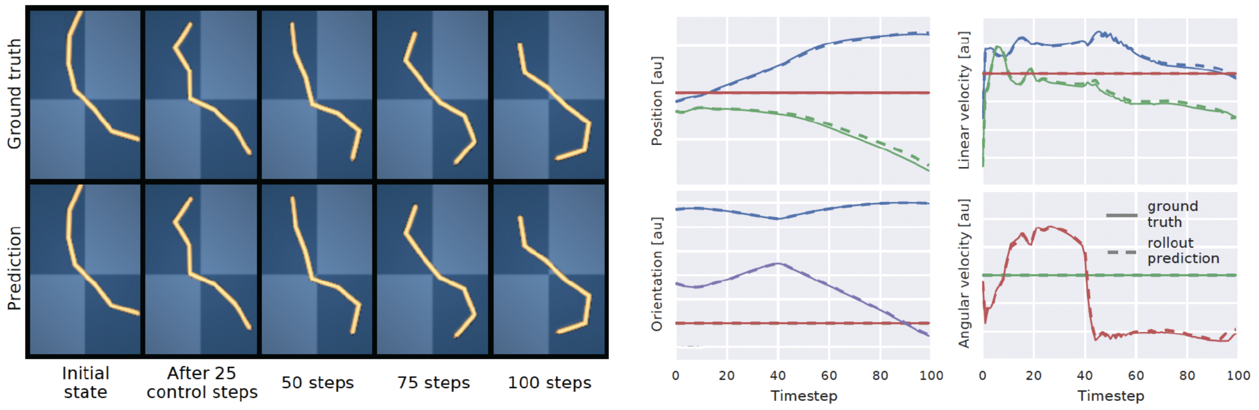


**Figure 14. Simulating an inchworm robot with Graph Networks. The simulation rollout matches the ground truth well. Reprinted from *Graph networks as learnable physics engines for inference and control* (Sanchez-Gonzalez et al., 2018).**

[1] In a dynamic simulation, the boundary condition and geometrical topology may change over time. Therefore we cannot directly compute the final result.

## Functional Simulation in Design

In addition to predicting the transformed results, simulations also help users to communicate design intentions between one another and to inform design decisions. For example, in *Printed Paper Actuator* (Wang et al., 2018), the design tool provides a library of design examples for users to play with and take inspiration from. However, the functions of these reversibly actable designs cannot be expressed using static images and models and demands an animated preview for communication (Figure 15).

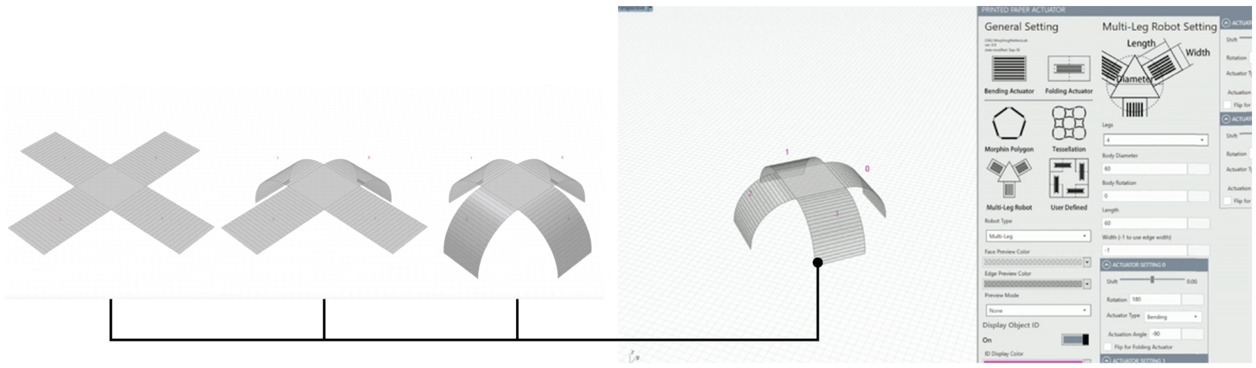


**Figure 15. Design tool interface from *Printed Paper Actuator.* Reprinted from *Printed Paper Actuator* (Wang et al., 2018).**

Identical to the example mentioned above, visualizing the transformation process also helps to inform decisions in 4D printing. In this field, although users are ultimately only concerned about the initial and transformed shape, simulating the transformation process will help us to reason over the triggering condition. For example, we can use the simulations to estimate how much space is required for the transformation to take place and plan the triggering method accordingly.

On a different note, we can also use simulations to evaluate and optimize the performance of designs. For instance, *Forté* (Chen et al., 2018) is an optimization design tools based on FEA simulations. Given a user-defined 2D structural sketch and load specification as input, the software will iteratively simulate the design's internal stress distribution with FEA to identify redundant voxels and remove them from the model, producing a structurally optimized design that takes the least amount of material to fabricate (Figure 16). Furthermore, designers can also use this software to explore design variations by altering optimization parameters.

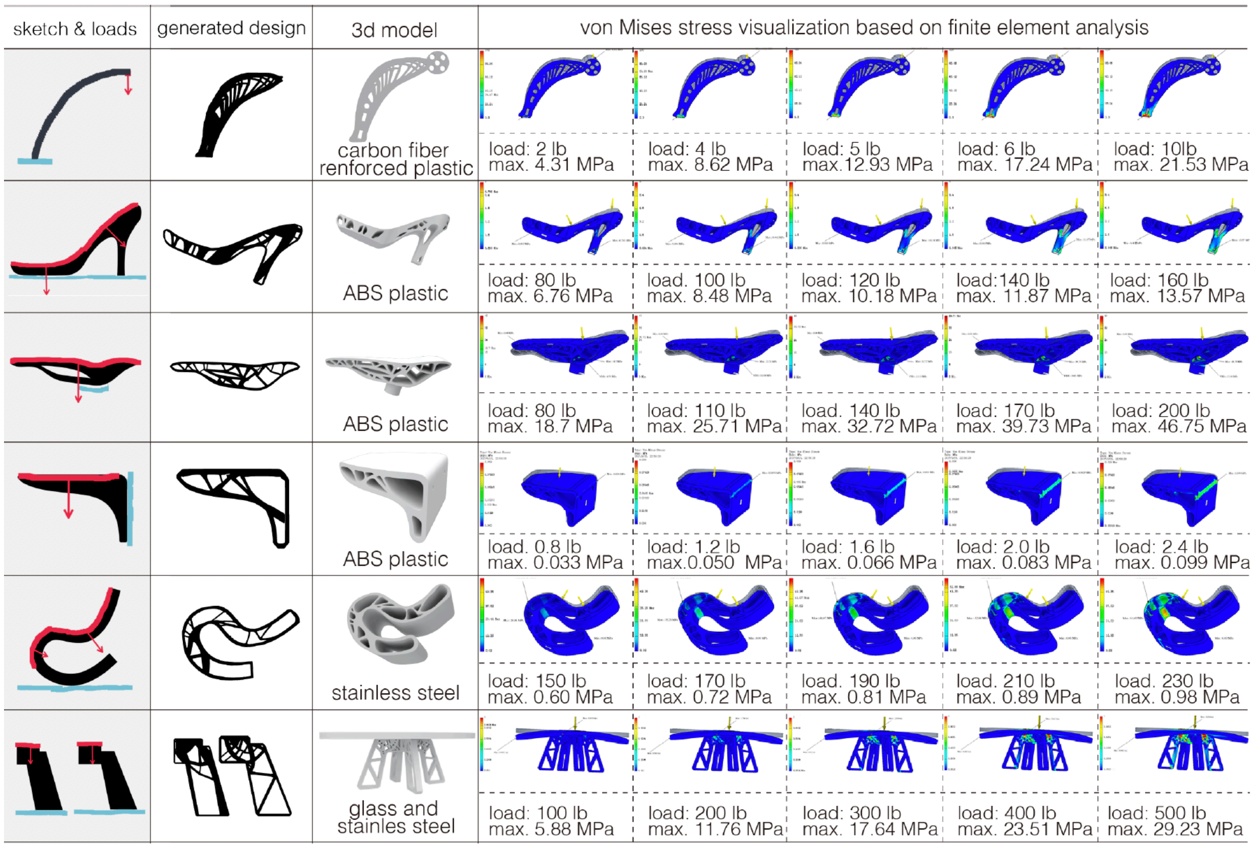

sketch & loads
generated design
3d model
von Mises stress visualization based on finite element analysis
carbon fiber renforced plastic
load: 2 lb max. 4.31 MPa
load: 4 lb max. 8.62 MPa
load: 5 lb max.12.93 MPa
load: 6 lb max. 17.24 MPa
load: 10lb max. 21.53 MPa
ABS plastic
load: 80 lb max. 6.76 MPa
load. 100 lb max. 8.48 MPa
load. 120 lb max. 10.18 MPa
load:140 lb max. 11.87 MPa
load: 160 lb max. 13.57 MPa
ABS plastic
load: 80 lb max. 18.7 MPa
load: 110 lb max. 25.71 MPa
load. 140 lb max. 32.72 MPa
load: 170 lb max. 39.73 MPa
load: 200 lb max. 46.75 MPa
ABS plastic
load. 0.8 lb max. 0.033 MPa
load: 1.2 lb max.0.050 MPa
load: 1.6 lb max. 0.066 MPa
load: 2.0 lb max. 0.083 MPa
load: 2.4 lb max. 0.099 MPa
stainless steel
load: 150 lb max. 0.60 MPa
load: 170 lb max. 0.72 MPa
load. 190 lb max. 0.81 MPa
load. 210 lb max. 0.89 MPa
load. 230 lb max. 0.98 MPa
glass and stainles steel
load: 100 lb max. 5.88 MPa
load: 200 lb max. 11.76 MPa
load: 300 lb max. 17.64 MPa
load: 400 lb max. 23.51 MPa
load: 500 lb max. 29.23 MPa

**Figure 16. Structural Optimization with *Forte*. This software system leverages FEA to produce structurally efficient designs. Reprinted from *Forte* (Chen et al., 2018).**

*Unshackling Evolution* (Cheney et al. 2013) is another research that harnesses simulation engines to compose optimization design tools. Given a target function (e.g., moving as fast as possible), this work leverages genetic algorithm and compositional pattern producing networks to derive optimized soft-robot designs by iterations. Noticeably, the fitness of each design is determined by using a dynamic soft-body simulator to predict its performance. In this particular use case, as the algorithm involves the evaluation of a considerable amount of robots, simulation speed becomes a critical factor to the viability of this technique. Likewise, *Forté* also leveraged simplified and coarsened FEA to reduce simulation time and enable real-time interactions.

# METHODS

With the backgrounds and relevance of our scope in mind, this chapter will firstly introduce the perimeters and technical concepts of our work and document the technical implementations of SimuLearn in detail. We will also discuss a design tool development framework that is adapted to data-driven simulators to complement our simulation technique.

## Overview

### Material System

This thesis adopts a 4D printing material system identical to that of 4DMesh, in which we use thermoplastics to produce mesh-like structures[2] made of bending-beam actuators that can transform when heated. The following section provides a brief introduction to the mechanism and condition of transformation.

#### Transformation Mechanism

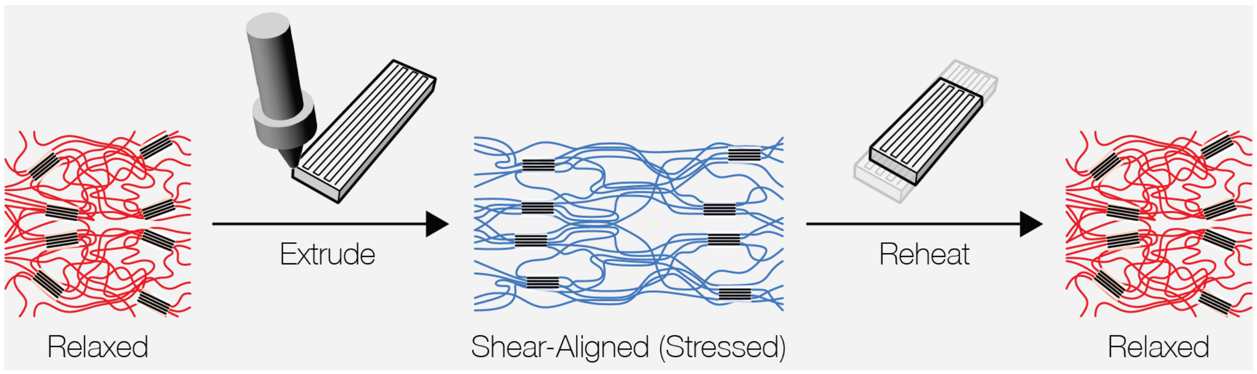


**Figure 17. Material Mechanism.**

Polylactic Acid (PLA) is a type of thermoplastic that has microscopic polymer chain structures. These polymer chains are flexible when liquid, and rigid when solid. In raw materials, these polymer chains are randomly coiled and relaxed, and will not result in material deformation. However, when extruding liquid PLA through a 3D printer nozzle, the narrow channel will cause the polymer chains to shear-align and become stressed. If the PLA is cooled down before the polymer chains can fully relax, the shear-alignments will lead to residual stresses in the material. As a result, when we heat printed materials above the glass transition temperature[3] ($T_g$, 60 ℃) of PLA, the polymer chains will release their energy and cause the extruded paths to shrink along the printing direction (Figure 17). We can characterize this transformation by calculating its shrinkage ratio with respect to the original length.

[2] In this thesis, we will also use “grids” and “patterns” as synonyms of “mesh-like structures”.

[3] At the glass transition temperature, while the PLA is still solid, the material and its polymer chain will become soft and malleable.

## Fabrication, Toolpathing, and Design

Leveraging the material property of PLA, we can deliberately arrange the tool paths to program anisotropic deformations into shapes. Take a beam shape for example, we can program them into bending actuators by printing the bottom half along the longitudinal axis and the top half perpendicularly[4], such that the bottom half will shrink along the axis of the beam when heated while the top half does not, resulting in the beam to bend downward (Figure 18). Note that the constraint will still shrink along the beam's width direction, but the deformation is small and negligible. Identically, we can program the beam to bend on the opposite direction by switching the top/bottom toolpath assignments or create a passive, non-actuator beam by printing both halves with perpendicular toolpaths.

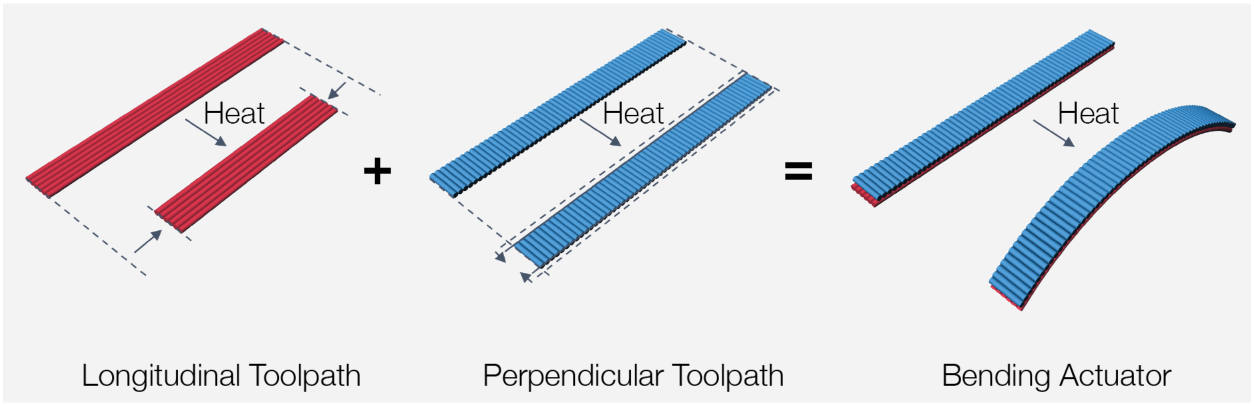


**Figure 18. Actuator Toolpathing. We can composite anisotropic toolpath blocks togethers to create bending actuators.**

With the aforementioned bending actuator designs, we can then produce larger scale transformative artifacts by composing multiple bending beams together into mesh-like structures. Joint blocks between adjacent beams are printed with alternating toolpath directions[5] to minimize transformation and avoid them from complicating the transformation process. Depends on the bending direction assignment of individual beams, the structures can transform into a variety of different shapes. In this work, the composed patterns are required to have a quad-mesh, grid-like configuration, and the width and thickness of individual beams are fixed to 7.2 mm and 4 mm respectively to confine the design space.

## Triggering

Printed mesh structures are triggered to transform inside a hot water bath (Figure 19). Compared to hot-air triggering, water provides more uniform heating over the transformation process. We glue one joint of

[4] A block that has toolpath directions perpendicular to the longitudinal axis of beams is also called constraints.

[5] Odd and even layers' toolpath directions are perpendicular to each other.

a structure to a fixed support to suspend it in the water tank during transformation and avoid from sinking to the tank bottom. The water bath is sufficiently large to avoid the grids from hitting the walls during transformations and is heated to 80 ℃, 20 ℃ higher than the $T_g$ of PLA to allow for ample transformation time before the bath cools below $T_g$, and the PLA solidifies. Furthermore, artifacts are retrieved from the tank when the water temperature drops below $T_g$ to avoid our actions form deforming the artifacts.

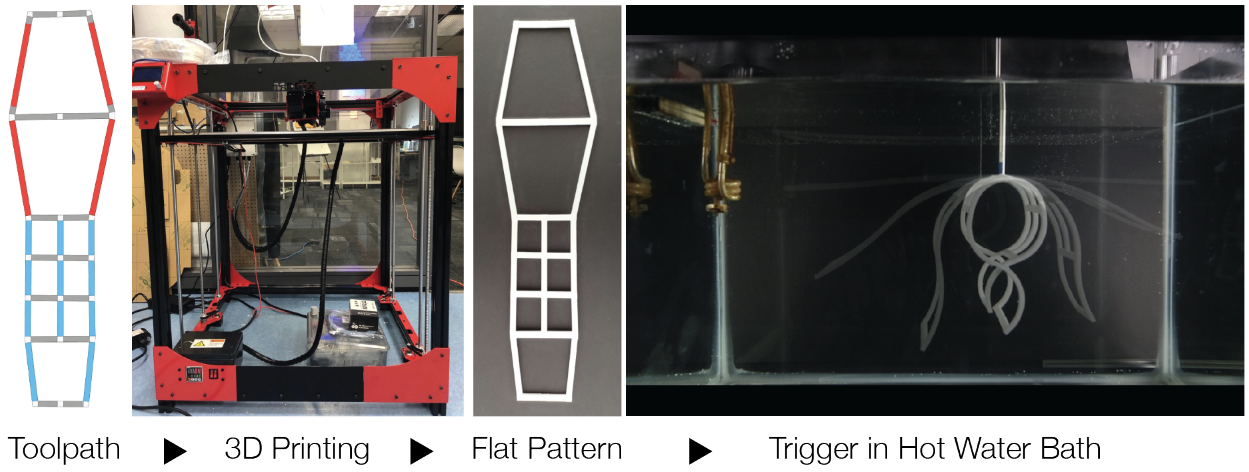


**Figure 19. Physical prototyping workflow. A toolpath (.gcode) file is printed with commercial 3D printers, and the printed artifact is triggered in a hot water bath.**

## Hierarchical Composition

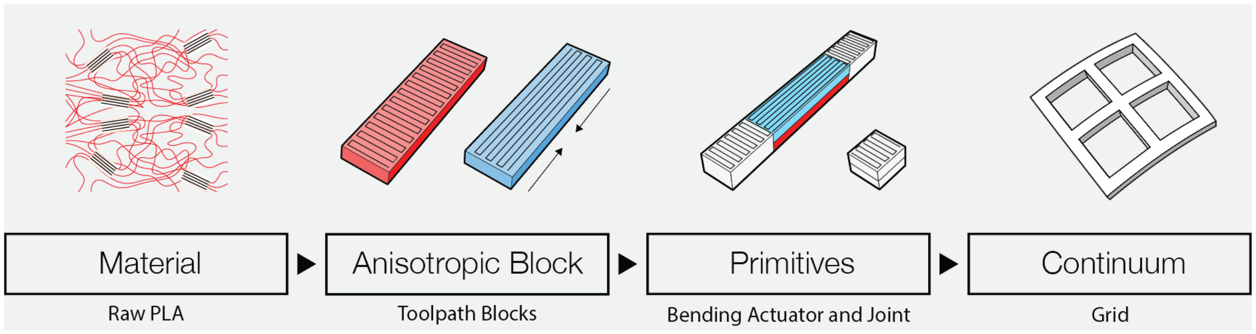


**Figure 20. Compositional hierarchy of our material system.**

This thesis takes inspiration from structured representations and inspects the design of our material system by breaking it into a chain of composition. Given an active material, we use additive manufacturing machines to structure it into anisotropic toolpath blocks. We then organize multiple toolpath blocks into actuators that can achieve certain transformation behaviors. Lastly, we composite multiple actuators into 4D printing artifacts that can transform on demand (Figure 20).

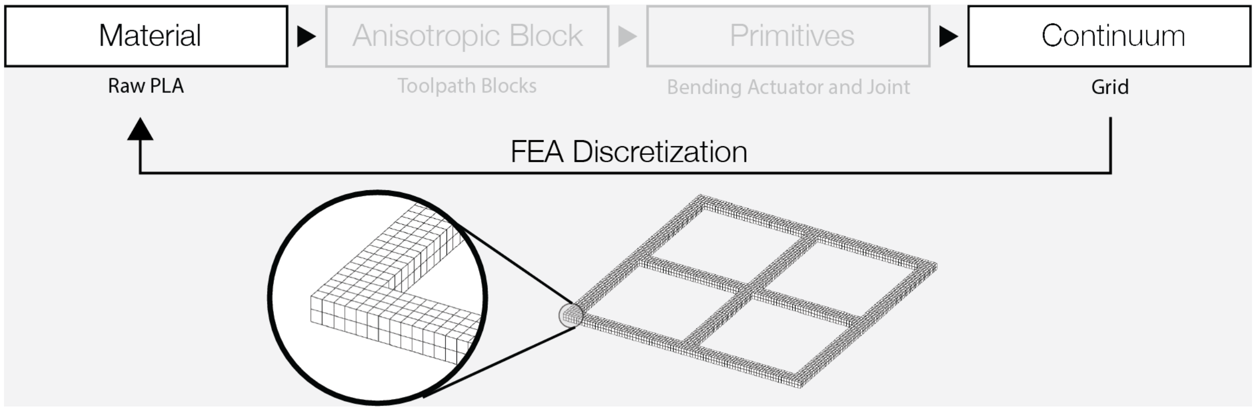


**Figure 21. FEA Perspective of Simulation. In our material system, FEA decomposes a model into a high-resolution voxel body, creating 30k to 200k elements to compute.**

The FEA perspective of simulation discretizes a continuum into voxels of materials, going from one end of the composition hierarchy to the other to create simulation units. While this method is generalized and adaptable to different geometries, it also creates lots of simulation elements to compute during simulations (Figure 21). On the other hand, general purposed CAD software systems allow us to operate across scales and levels in the compositional hierarchy, exposing users to an overwhelming amount of modeling commands and design options.

Instead of being unconstrained in the compositional hierarchy, this thesis limits the development of simulators and design tools at a single level of abstraction - actuators - to expedite design processes and simulations. At a single level of abstraction (Figure 22), the elements can be easily described with several parameters and are limited to a specific compositional topology. While this approach narrows down the design space, it also allows us to compose dedicated design tools that take fewer parameters to work with and streamline design workflows. Similarly, choosing a single hierarchical level will allow for a consistent and uniform representation of building blocks, enabling data-driven simulations to take place. Furthermore, actuators are high-level abstractions in our hierarchy of composition and will allow us to simulate transformations with fewer elements. Rather than discretizing a design into a high-resolution voxel body, we can go one step down in the hierarchy of composition and simulate the transformation of individual actuators and joints (building blocks).

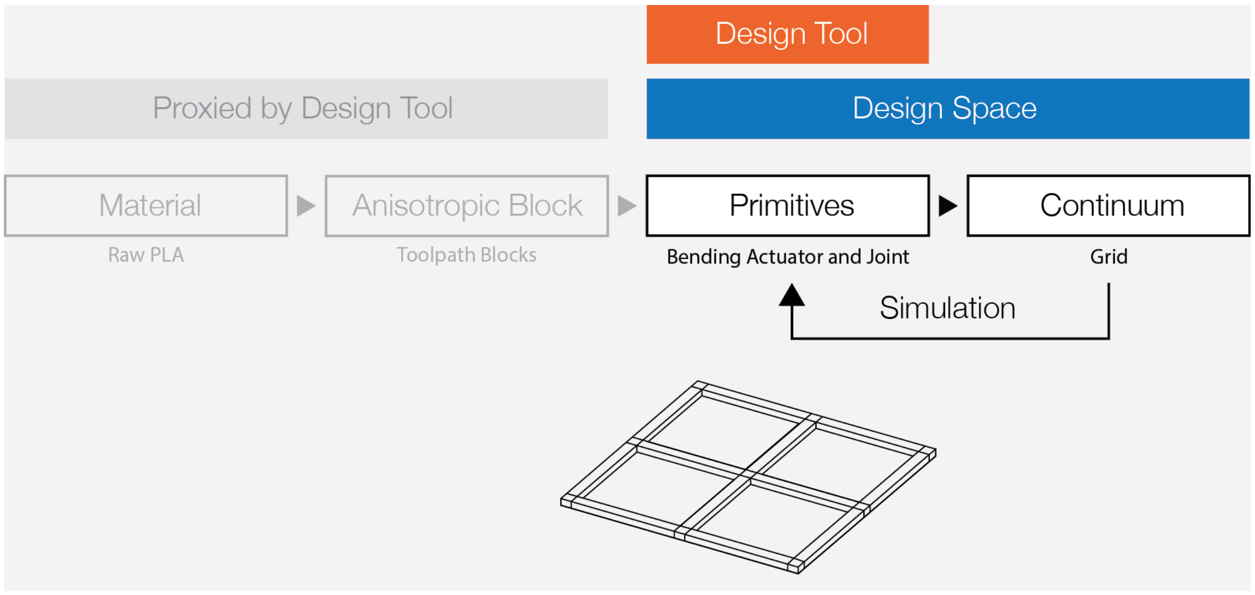


**Figure 22. Design tool at a limited level of abstraction. Limiting the level of abstraction at primitives will allow us to develop dedicated design tools and compose simulation methods that compute with fewer elements.**

## SimuLearn

### Concept

The data-driven simulation method we developed is called SimuLearn. This technique takes FEA result as training data and leverages graph convolutional networks (GCN) to produce simulation engines for compositional designs. An intuitive interpretation of its computation is, given an input grid design, the model will decompose it into its building blocks, update their statuses individually, assemble the blocks back together to derive the grid's next status in time, and iterate this process until the transformation is converged (Figure 23). Compared to directly computing the final shape of transformation, this iterative approach will allow us to collect more training data from an episode of FEA simulation and to visualize the transformation process in design tools.

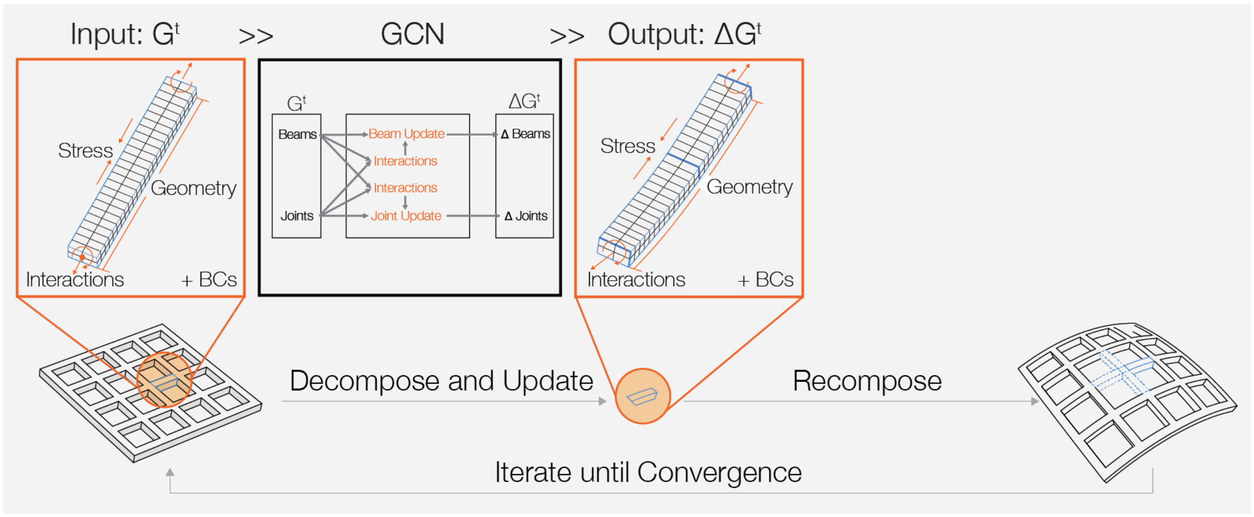


**Figure 23. Simulearn computation flow.**

Compared to FEA, SimuLearn operates at a higher level of abstraction and has fewer elements to compute. The mathematical differences between FEA and SimuLearn also result in the later to run faster than the former. FEA represents a model as a  linear system whose size (number of entries) is quadratic to the number of simulation elements, and the computation is based on solving this linear system - a sequential operation that usually takes lots of time to compute. On the other hand, SimuLearn conceives the transformation as a function of time, divides this function into multiple segments, and utilizes machine learning as regressors to approximate these segmented functions. This way, the computation is simplified into a combination of matrix multiplications and additions - parallelizable operations that computers are efficient at and have dedicated hardware for (GPGPU), therefore running faster compared to FEA. However, since SimuLearn is merely an approximation of FEA simulations, it is expected to have a lower accuracy compared to FEA.

### Technical Pipeline

With the SimuLearn technique in mind, we implement a development pipeline (Figure 24) consisting of three steps for SimuLearn engines. Each step in the pipeline has its unique challenges and goal. The FEA modeling step is aimed to establish a physically accurate model of the material system and its actuation environment, and configure the FEA solver and data structure to facilitate information retrieval at later steps. Once the FEA model is established, we can then use it to generate raw simulation results for data extraction. Note that an FEA raw result will contain the information (position, stress distribution) of every element at each time step, creating lots of information redundancy that may cause the ML model difficult to converge or prone to overfit. Therefore, we further distill these raw results at the data processing step to create an effective, less biased, and concise dataset for ML training. Lastly, we use GCN to generalize from the dataset to acquire fast and physically accurate simulation engines.

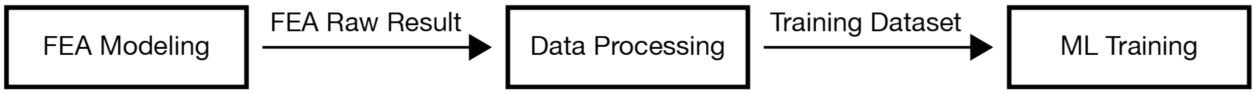


**Figure 24. Technical pipeline overview.**

## Apparatus

We implement our pipeline on a consumer-grade desktop computer (Intel i9-9900k 8C16T processor, 64 GB RAM, Nvidia P5000 Graphics Card with 16 GB Memory) to produce proofs-of-concept for the SimuLearn technique. This implementation uses the non-linear FEA software Abaqus to compute physical simulations, the ML library *Pytorch* (PyTorch, 2019) to construct ML models leveraging its CUDA backend, and the modeling software *Rhinoceros* and *Grasshopper* (Rutten et al., 2019) to process and visualize geometries, and compose design tools.

In terms of fabrication, we use off-the-shelf materials (Polymaker PLA) and a hobbyist-grade 3D printer (Modix Big 60) for physical prototyping. The extruder nozzle diameter is strategically set at 0.6 mm to balance between fabrication time and actuation performance. While using smaller nozzle diameters will enable us to program larger deformations into actuators, they will also increase the printing time due to denser infill toolpaths. Identically, wider nozzles will reduce the fabrication time, but the resulting actuators are unable to perform sufficiently large transformations. As for the fabrication parameters (e.g., printing speed, layer thickness), we favor the configuration that allows us to prototype with speed while ensuring a good printing quality.

# Pipeline Implementation

## FEA Modeling

### Discretization

We discretize the mesh-like patterns of our material system into cubic voxel[6] bodies for FEA. In addition to the baseline mesh quality requirements of FEA - regular voxel dimensions and no pointy voxel corners, each block[7] also has an odd number of nodes[8] on the length, width, and height directions to enable sampling midpoint positions of edges and on cross-sections.

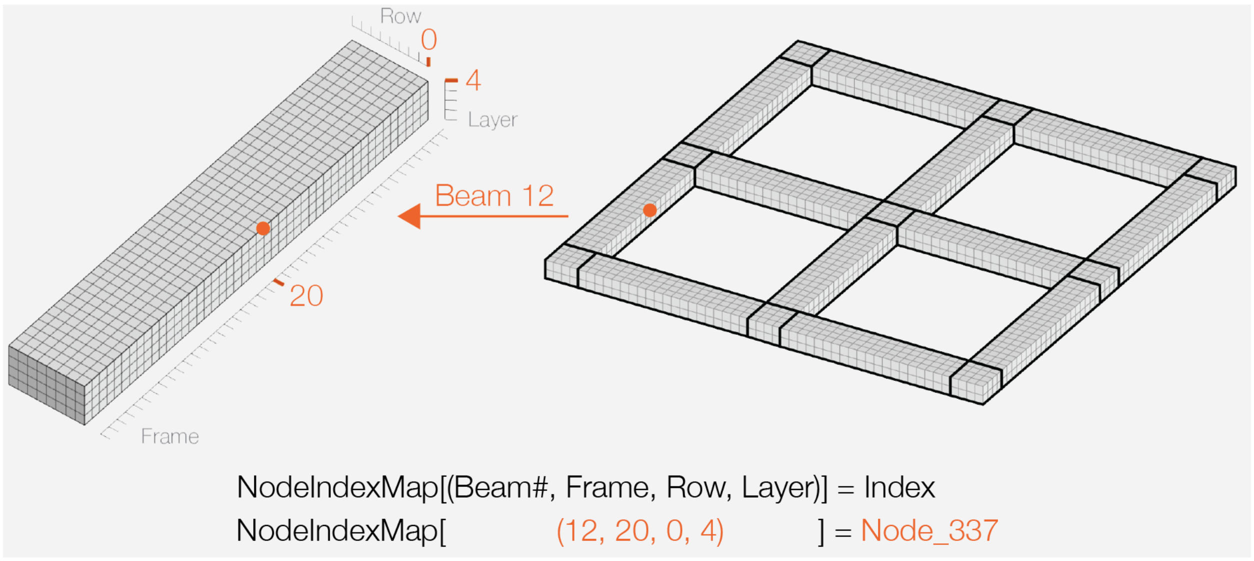

Row
0
4
Layer
Beam 12
20
Frame
NodeIndexMap[(Beam#, Frame, Row, Layer)] = Index
NodeIndexMap[ (12, 20, 0, 4) ] = Node_337

**Figure 25. Model discretization and index map. With this map, we can retrieve the index of a point by using its relative position in the model.**

In the raw simulation results produced by Abaqus, the nodal and elemental information are stored as flat lists that are difficult to index. The index of a node does not provide any information about its parent block, nor its position in the blocks. For instance, if we want to acquire the position of a beam's center point, we have no way of knowing the index of that node in the list. To address this issue, during the model discretization process, we assign each node with a unique tuple key to indicate their relative position in our model (Figure 25). The tuple keys record the index of the nodes' parent blocks and their positions along the length, width, and height directions of blocks, and are associated with the nodes'

[6] In Abaqus, this corresponds to C3D8 elements.

[7] We will refer to the collection of Joints and Beams as blocks in the following sections.

[8] In FEA conventions, points are called nodes and voxels are called elements.

indices in the information lists. Likewise, the voxels are also associated with a tuple key to enable convenient indexing.

**Boundary Conditions**

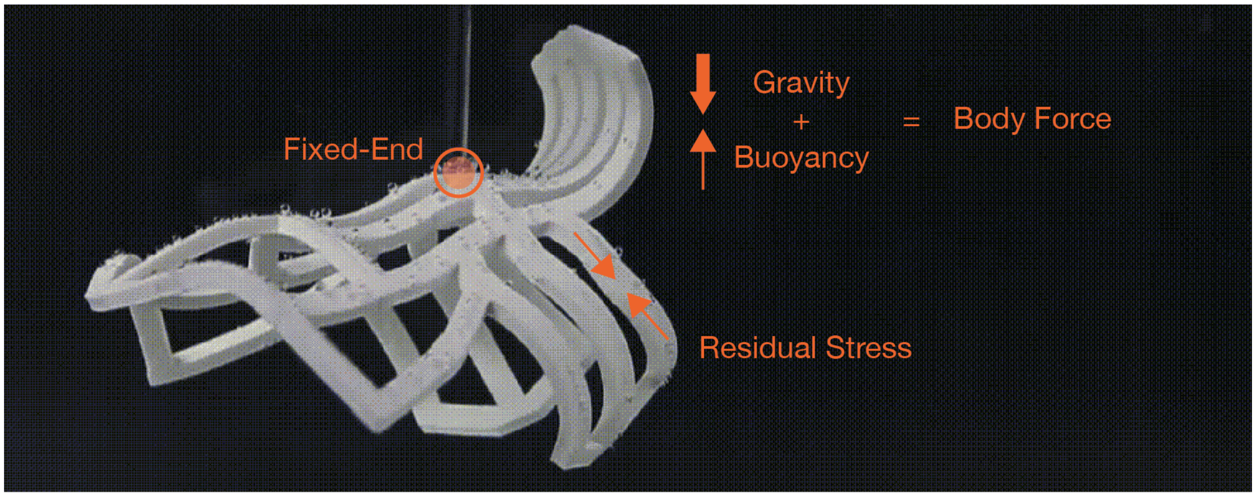


**Figure 26. FEA boundary conditions. We use three boundary and initial conditions in our model, including residual stress, body force, and fixed end conditions.**

In reality, in addition to the material's intrinsic properties, the transformation process is also affected by the physical factors in the triggering environment. We can account for these factors by modeling them as boundary conditions (BC) in our FEA model. In our case, the material system is mainly influenced by three physical factors (Figure 26). Firstly, we assign a fixed-end condition to the joint that the grid is held at during the triggering process. More accurately, since the fixed joint is glued to the support during the transformation process, only the gluing interface - the top face of joint - is fixed in the simulation. Secondly, the artifacts are simultaneously subjected to buoyancy and gravity in the water bath. We integrate these two factors into a uniform body force applied to the whole model. Lastly, depending on the local toolpath, each voxel is assigned with a stress field oriented along the printing direction to describe the residual stress induced by the printing process. We can compute the oriented stress field $= [\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{13}, \sigma_{23}]$ by using the following formulas:

$$\sigma_{11} = \sigma \cos^2 \alpha$$

$$\sigma_{22} = \sigma - \sigma_{11}$$

$$\sigma_{33} = 0$$

$$\sigma_{12} = -\sigma \sin \alpha \cos \alpha$$

$$\sigma_{13} = 0$$

$$\sigma_{23} = 0$$

Where $\sigma$ is the magnitude of the residual stress; $\alpha$ is the printing direction's polar angle; $\sigma_{11}, \sigma_{22}$, and $\sigma_{33}$ are respectively the stress components on the x-, y-, and z-axis; $\sigma_{12}, \sigma_{13}, \sigma_{23}$ are the shear stress components on the xy, xz, and yz plane.

**Material Property**

Using FEA to simulate physical transformations demands a mechanical material definition to produce accurate results. We can collect this information either from literature or by experiments. Specific to the PLA that we are using, while its density is immediately available in the literature, its mechanical properties at the triggering temperature are not yet characterized in available works and data sheets. Therefore, we conduct dynamic mechanical analysis (DMA, Figure 27) to characterize its properties for our FEA model.

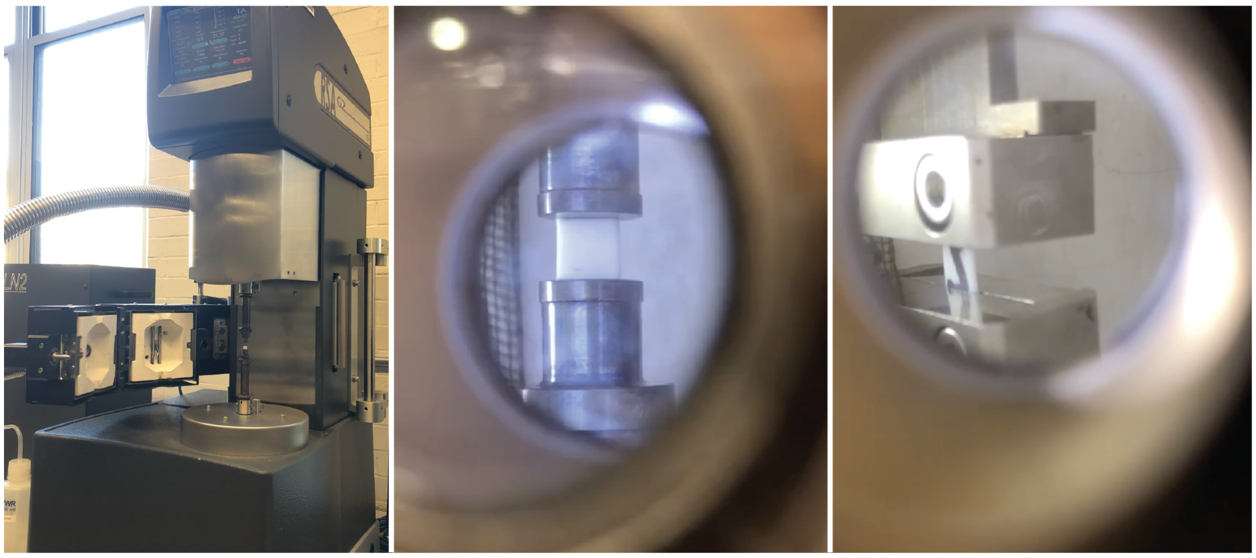


**Figure 27. DMA test. Left - DMA machine; middle - compression test; right - tension test.**

At the triggering temperature, PLA exhibits two distinctive mechanical properties - viscoelasticity and hyperelasticity - and we design our DMA test to characterize both components individually with an RSA-G2 Solids Analyzer by TA Instruments. This instrument has an enclosed, insulated, and temperature-regulated chamber to accurately recreate the triggering temperature for our experiments. We characterize the viscoelasticity component of PLA by putting 3D printed and stress-released samples under oscillating (0.01 to 100 Hz) tensile displacement loads to measure the material's storage (E') and loss (E") modulus, and we can directly import these results into our FEA software for material modeling. As for the hyperelastic part, we sample its stress-strain relationship by putting the samples under compressive and tensile loads to measure their deformations. Once we acquire the stress-strain data, we then perform curve-fitting on them with various polynomial hyperelastic models to identify the one with the lowest error (Marlow model) and import the result into the FEA software. Note that we perform both tests multiple time with different samples and average the results to mitigate experiment error, and the test samples are stress-released before experiments.

Lastly, in order to quantify the residual stress induced by the printing process, we perform multiple rounds of FEA simulation with different geometries and stress values and compare the results with physical ground truths (i.e., printed and triggered designs) to determine the residual value that leads to the lowest error. We then use this stress value for all future simulations.

**Solver Setting**

We configure our FEA solver to iteratively compute and log the transformation over time to adapt to the iterative strategy of SimuLearn. Specifically, an episode of FEA is divided into multiple equally-spaced time steps for simulation, and the solver will save the temporal results at a regular interval. For instance, our solver will divide an episode of simulation into 100 increments and save the results at a 10 step interval, resulting in 11 frames (including the initial input) in total. Consequently, a SimuLearn engine trained with this data will perform an episode of simulation with ten increments and output the temporal statuses accordingly.

# Data Processing

## Data Generation

In order to gather episodes of FEA simulations to train ML models, we compose a parametric script to batch-generate grid designs (Figure 28, 29). Given a user-specified grid size[9], the script will produce a skeletal grid by randomly sampling from predefined ranges of parameters and shift vertices around to diversify output geometries. Each edge in the grid is randomly assigned into an upward- or downward-bending actuator, or a passive unit. The skeletal grids are then converted into 3D models and processed into FEA input files with the aforementioned FEA modeling method.

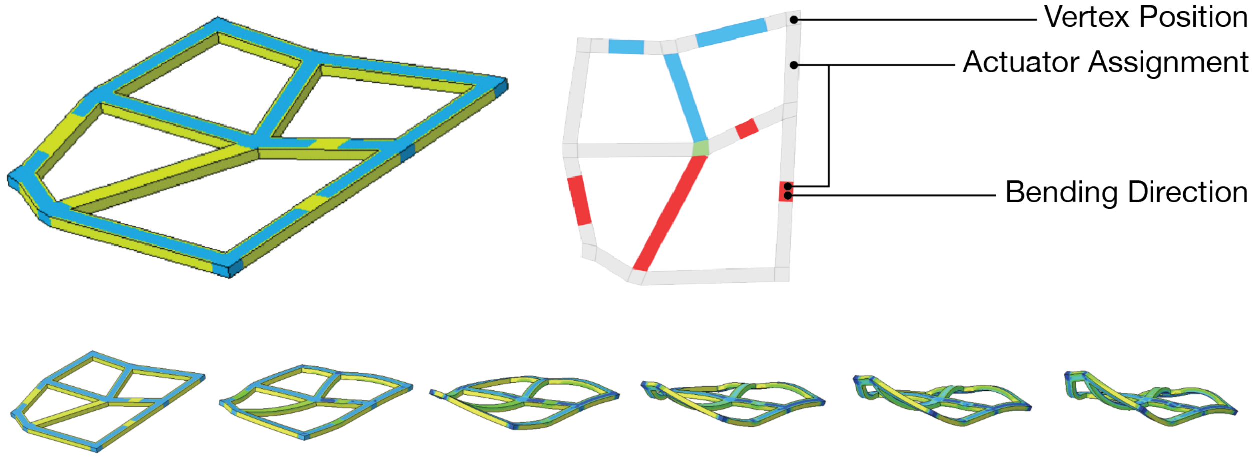


**Figure 28. Parametric grid generator.**

When using random sampling to produce grid designs, we should use importance sampling instead of uniform sampling to generate parameters to avoid implicitly biasing our dataset. For instance, we use randomly generated vectors to shift vertices around, and the vectors are defined as V(θ, r) in polar coordinate systems, where θ is the polar angle and r is the radius. While uniformly sampling θ is unbiased, doing so on r will cause the vectors to concentrate around the origin and result in a dataset that is biased toward regular grid designs.

[9] We use grid size to describe the number of vertices along the x- and y-direction of grids.

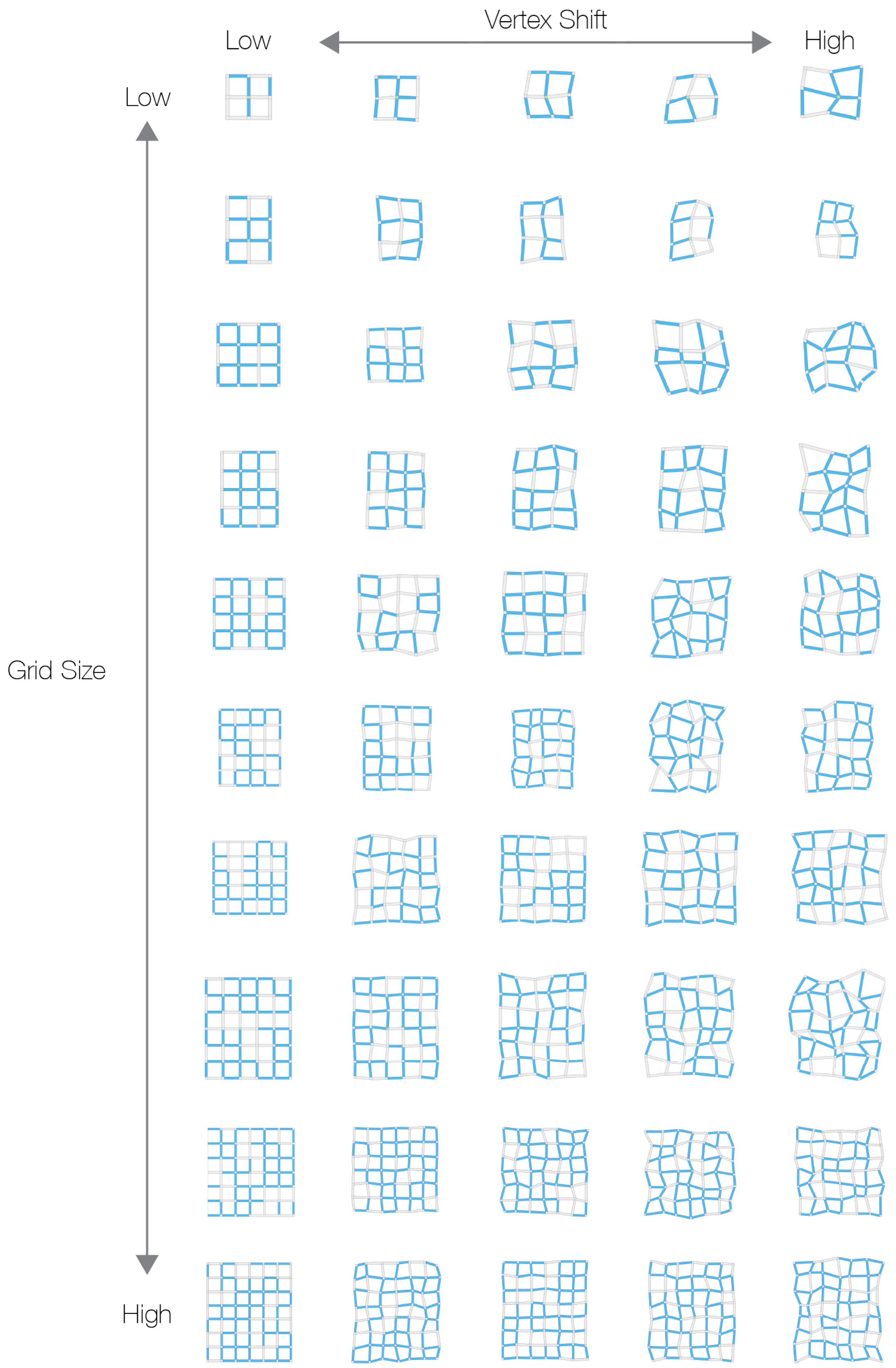


**Figure 29. Grids generated by our script.**

## Representation and Feature Extraction

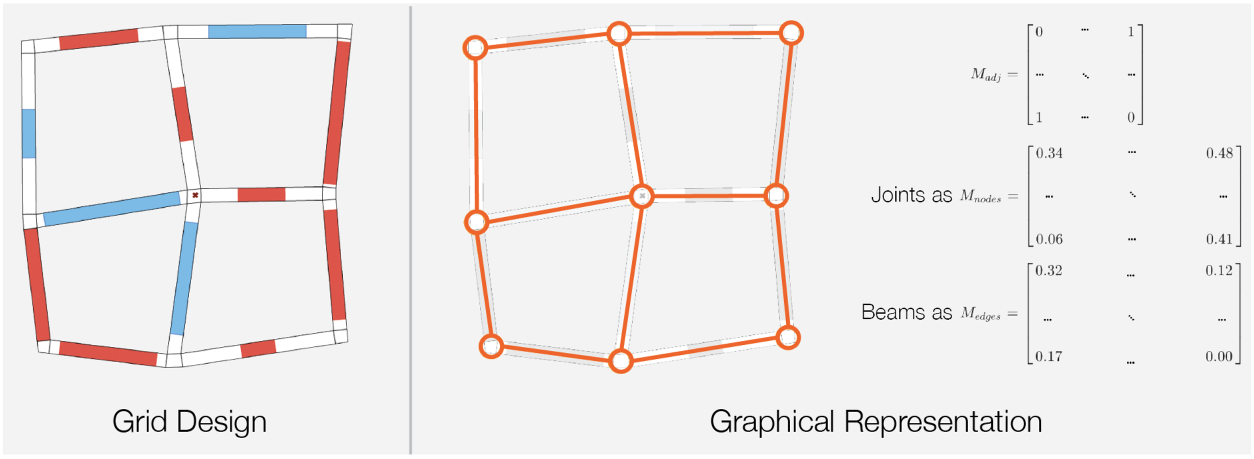


**Figure 30. Graphical representation of grid designs. We can regard the beams and joints in a grid as edges and nodes, and represent the compositional pattern as an abstract graph.**

The computed FEA raw results are converted into readable files (.csv files) using *Abaqus2Matlab* (Papazafeiropoulos et al., 2017). Nevertheless, we also have to convert the grid models into representations compatible with GCN. Given a grid model, we can regard it as an abstract graph $G = (M_{adj}, M_{edges}, M_{nodes})$, in which the joints are nodes and beams are edges, and represent it concisely using three matrices - a binary adjacency matrix $M_{adj}$ to describe the connectivity between blocks and two feature matrices $M_{edges}$ ($M_{beams}$) and $M_{nodes}$ ($M_{joints}$) to encode the information of beams and joints (Figure 30). The rows and columns of $M_{adj}$ correspond to the beams and joints in our grid respectively and the entries are set to one when two blocks are contiguous and zero otherwise. The beams and joints are individually described with a feature vector consisting of three terms - geometry, stress, and boundary conditions that the blocks are subjected to (Figure 31).

The minimal information required to reconstruct an arched beam's geometry is three cross-sections at the start, middle, and end of it, and we can rebuild the solid beam by lofting these cross-sections into a tube. Since the cross-sections are approximately rectangular and have identical dimensions, we can represent them concisely by describing the spatial plane that they are situated and centered on with a center point and two 3D vectors. While other representation methods may construct the cross-section more accurately, such as directly describing the four corners of the rectangular cross-sections, they will also lead to longer feature vectors, and the differences are visually trivial. As for the stress term, we sample the stress fields at the Gaussian quadratures around the cross-sections' corners. These positions are observed to undergo the most significant stress change during the morphing process, leading to large numerical variances that will make our dataset easier to generalize from. In the boundary condition term, we use a binary indicator to indicate whether the beam is a bending actuator or a passive beam and a 3D vector to describe the beam's relative positions with respect to the fixed joint. In total, the feature vector length of beams is 223.

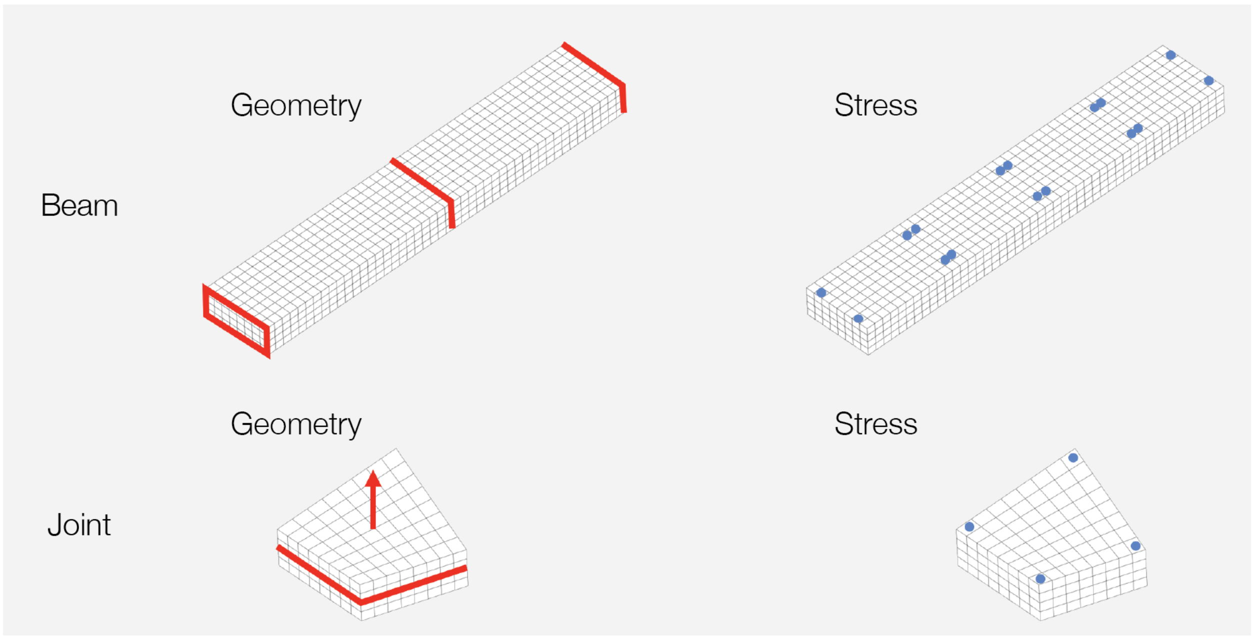


**Figure 31. Geometry and stress representation of blocks. The geometry of beams are represented as the loft of three cross sections, and joints are represented as a double-sided extrusion of a surface. We sample the stress at positions that undergo the largest stress change.**

Compared to the tube-like shapes of beams, the joint blocks have approximately box-like shapes and are nearly rigid throughout transformation processes. Therefore, a joint's geometry can be concisely represented as the double-sided extrusion of a quadrilateral surface located at the middle of the joint. More accurately, we describe the surface using one center point and four 3D vectors pointing from the center point to the surface corners and capture the extrusion with a 3D vector pointing from the surface center to the joint's top-face center point. The stress fields are sampled at the Gaussian quadratures[10] around the corners of the joint. Lastly, the joint's boundary condition term comprises a binary indicator to described the joint's fixed-end condition and a 3D vector to describe its relative positions with respect to the fixture. In total, the feature vector length of joints is 70.

[10] FEA computes the stress field of elements on the Gaussian quadratures instead of nodes.

## Feature Reduction

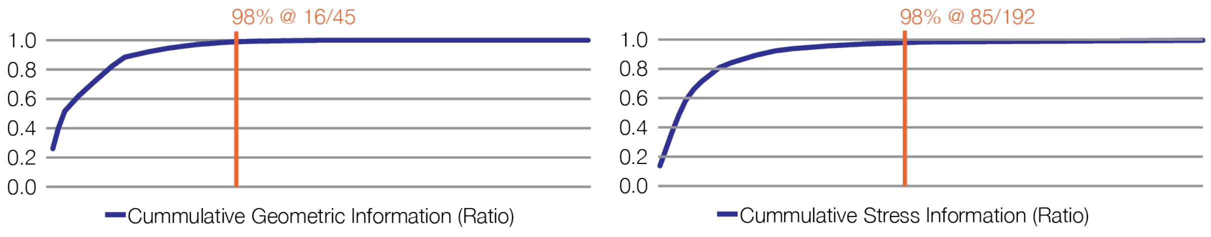


**Figure 32. Beam PCA results. At 98% information retention rate, the beam feature length is reduced by 53.4%.**

Despite that the blocks' features are already compact, a good portion of them are still redundant, especially in the stress term. In response to this problem and informed by previous work (Liang et al., 2018), we perform principal component analysis (PCA) to reduce feature redundancy. We set the information retention cutoff of PCA to 98%, which results in halving the feature dimension of beams (Figure 32) and shortening the joint feature vectors by 25%. The Eigenvectors produced by our PCA  is saved and applied to the inputs of our ML model during forward-computations.

When using PCA, an important note is that it should be applied to the geometrical, stress, and boundary condition terms individually to avoid removing critical information. For instance, only one joint in the grid is assigned with a fixed-end condition in the boundary condition term. This scarcity leads to comparatively low variance in that column across the dataset. Performing PCA on the three representation terms at once will very likely remove this information from the dataset.

## Spatial Symmetry

In reality, regardless of the position and planar rotation of an object, it should always perform the same transformation as long as it is appropriately and consistently oriented. However, since we generate our data with FEA software systems, the points of the same model may take different values depending on their location in the CAD environment, creating unnecessary variances in our dataset. We eliminate this meaningless variance by repositioning the grids such that the fixed joints are always positioned at the spatial origin. On the other hand, our data generation script produces grid models that are approximately axis-aligned, which may cause the ML model to implicitly bias. As a result, we increase rotational variances in our dataset by rotating raw FEA results to create additional episodes of simulation.

The aforementioned geometrical translations (repositions) are applied to the points in the representations' geometrical terms, and the rotations are applied to both the points and vectors. Note that when performing rotations, we should also rotate the stress fields to match the geometry.

## Machine Learning

### Graphical Network Blocks

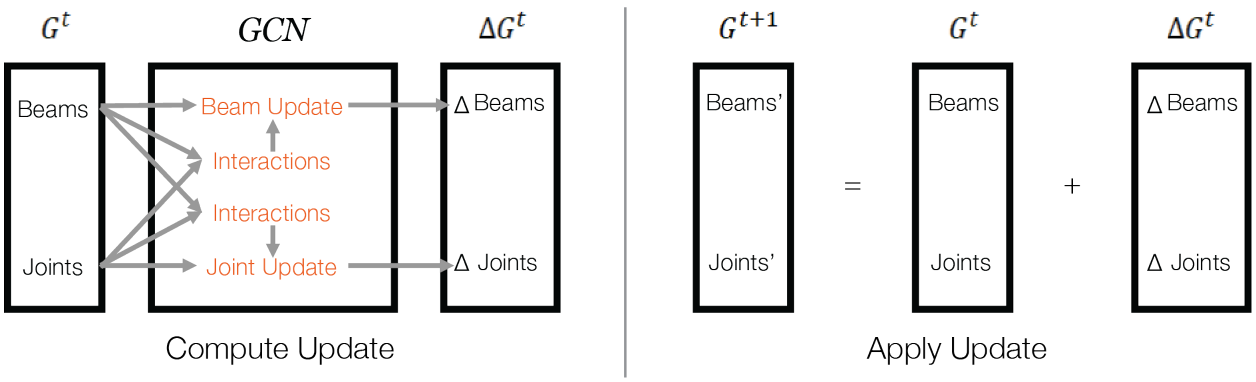


**Figure 33. GN Model and forward computation.**

Graphical network (GN) blocks are the building blocks of GCN (Figure 33). A single GN block takes the graphical representation of a grid $G = (M_{adj}, M_{beams}, M_{joints})$ as input and outputs its update $\Delta G = (\Delta M_{beams}, \Delta M_{joints})$. In particular, a GN block contains four sub-neural networks to perform different aspects of the computation. Two of the four neural networks ($f_{I-joint}$ and $f_{I-beam}$) are used to compute the interactions between neighboring blocks, and the rest ($f_{U-beam}$ and $f_{U-joint}$) is used to compute the updates of individual nodes and beams. Figure 34 describes the inputs and outputs of each sub-neural network:

| Network name | Input | Output |
|---|---|---|
| $f_{I-joint}$, joint-to-joint interaction network | $cat(X_{beam}, X_{joint}, X_{joint})$ | $I_{joint}$ |
| $f_{I-beam}$, joint-to-beam interaction network | $cat(X_{beam}, X_{joint}, X_{joint})$ | $I_{beam}$ |
| $f_{U-beam}$, beam update network | $cat(X_{beam}, conv(I_{beam}))$ | $\Delta X_{beam}$ |
| $f_{U-joint}$, joint update network | $cat(X_{joint}, conv(I_{joint}))$ | $\Delta X_{joint}$ |

**Figure 34. Inputs and outputs of sub-neural networks.**

In the interaction networks, $cat()$ denotes vector concatenation, $X_{beam}$ and $X_{joint}$ are the beam and joint feature vectors, and $I_{joint}$ and $I_{beam}$ are the latent interaction vectors with joints or beams as the receivers, respectively. Note that we regard the graphical representations of grids as directed graphs[11]. Therefore, the first joint appeared in the concatenation depicts the sender and the second the receiver. In the update networks, the $conv()$ terms in the inputs are the element-wise interaction convolutions (i.e., summing the latent interaction vectors that the element is the receiver of), and the outputs $\Delta X_{beam}$ and $\Delta X_{joint}$ are the updates of individual joints and beams.

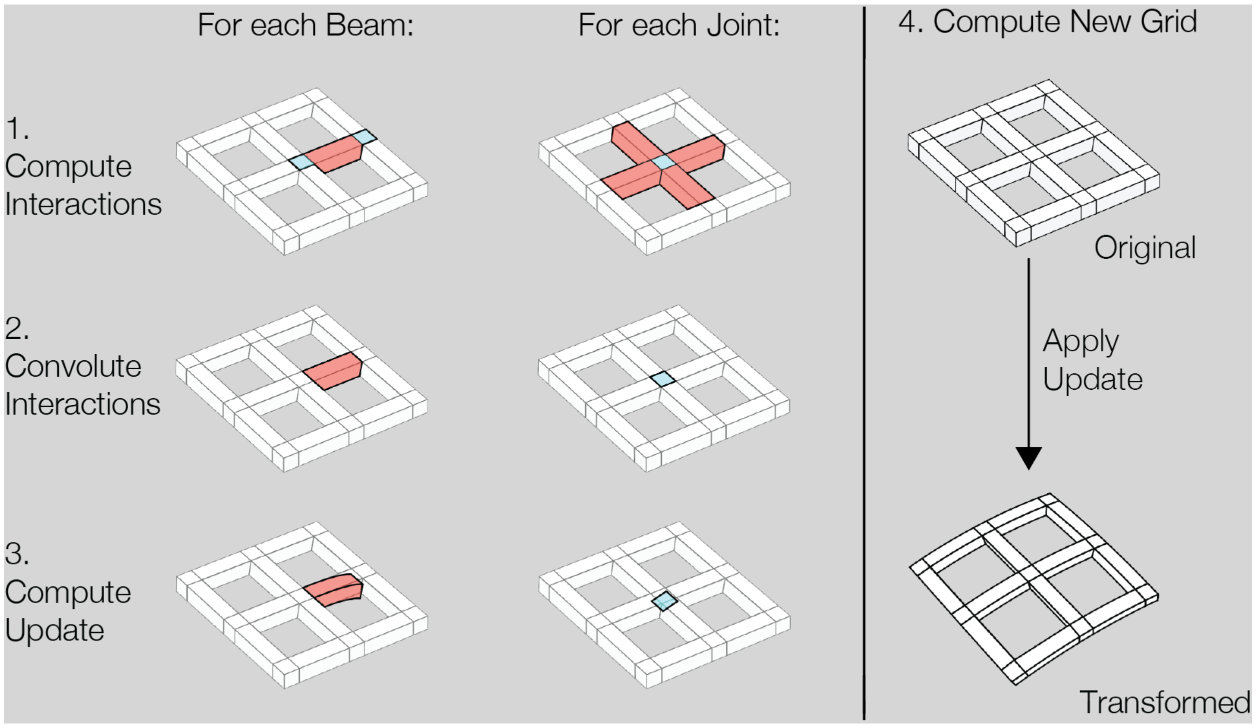


**Figure 35. GN block forward computation steps.**

During forward computations, given an input grid $G_t$, the GN block will follow these procedural steps (Figure 35) to compute $G_{t+1}$:

1. For each pair of adjacent blocks, compute $I_{joint}$ with $f_{I-joint}$ or $I_{beam}$ with $f_{I-beam}$.

2. For each block, convolute $I_{joint}$ and $I_{beam}$ that the block is the receiver of to get $conv(I_{joint})$ or $conv(I_{beam})$.

[11] In a directed graph, edges (beams) are unidirectional. For instance, given two adjacent nodes (joints) A and B in a graph, $\overline{AB}$ and $\overline{BA}$ are treated differently.

3. For each block, compute $\Delta X_{joint}$ with $f_{U-joint}$ or $\Delta X_{beam}$ with $f_{U-beam}$.

4. Compute $G_{t+1} = Gt + \Delta G_t$.

In step 1 and 2, we can use the $M_{adj}$ of $G$ to identify block adjacencies. Note that in our material system and transformation processes, $M_{adj}$ is constant and the connectivities between elements do not change over time. Therefore, we omit $M_{adj}$'s updates in our implementation. Additionally, the input interaction pairs of $f_{I-joint}$ and $f_{I-beam}$ as well as the inputs of $f_{U-joint}$ and $f_{U-beam}$ are geometrically translated (repositioned) to the coordinate origin to preserve spatial symmetry.

**Double GN**

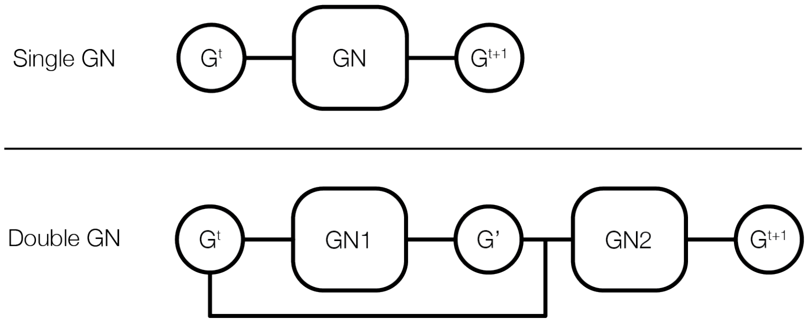


**Figure 36. Single GN and double GN model comparison. The Double GN model has a latent graph G' that allows information to propagate.**

While a single GN block can already learn and simulate transformations of the mesh-like structures, we take inspiration from Sanchez-Gonzalez et al. and construct double GN models to produce more accurate simulation engines (Figure 36). In a double GN model, the two GN blocks (GN1 and GN2) are arranged in a deep network-like arrangement and have unshared parameters. GN1 takes a graph $G_t$ as input and computes a latent graph $G'$; the feature matrices of $G_t$ and $G'$ are then concatenated together to form the input of GN2[12], and GN2 will compute the final update of the graph to get $G_{t+1}$. Compared to using a single GN block, this model is verified to have higher simulation accuracies as it allows the nodes and edges to communicate through $G'$.

[12] $GN_2$ takes the concatenated graph as input and therefore has larger neural network dimensions than $GN_1$.

## Training Method

The hyperparameters of our ML model are decided by a grid search algorithm, including the lengths of latent interaction vectors and the dimensions of the GCNs' neural networks. In particular, the neural networks are structured to have pyramid-like shapes (i.e., wide input and narrow output layers). As a GCN is essentially a combination of multiple neural networks, we directly train the models as deep networks with batch gradient descend (BGD) and an Adam optimizer. Additionally, we monitor the trends of training and validation loss, and early-terminate the training process when overfitting occurs (i.e., decreasing training loss and increasing validation loss).

# Design Tool

## Architecture

Our design tool architecture consists of four modules - composer, inspector, simulator, and slicer - each responding to different aspects of the design process (Figure 37). Given the sketch of a design and actuator assignments as input, the composer will translate it into a 3D model, program material properties accordingly, and provide visualizations to the user. The inspector will check the composed model against the design rules of the material system and visually inform users of violations. Once the model is verified, the user can then use the simulator to predict the design's transformation and modify the input to iterate. Lastly, the slicer will process the model and assigned material properties into fabrication files for physical prototyping. In this framework, the modules will proxy pattern compositions/decompositions happening at different stages of design and allow users to operate at a single level of abstraction.

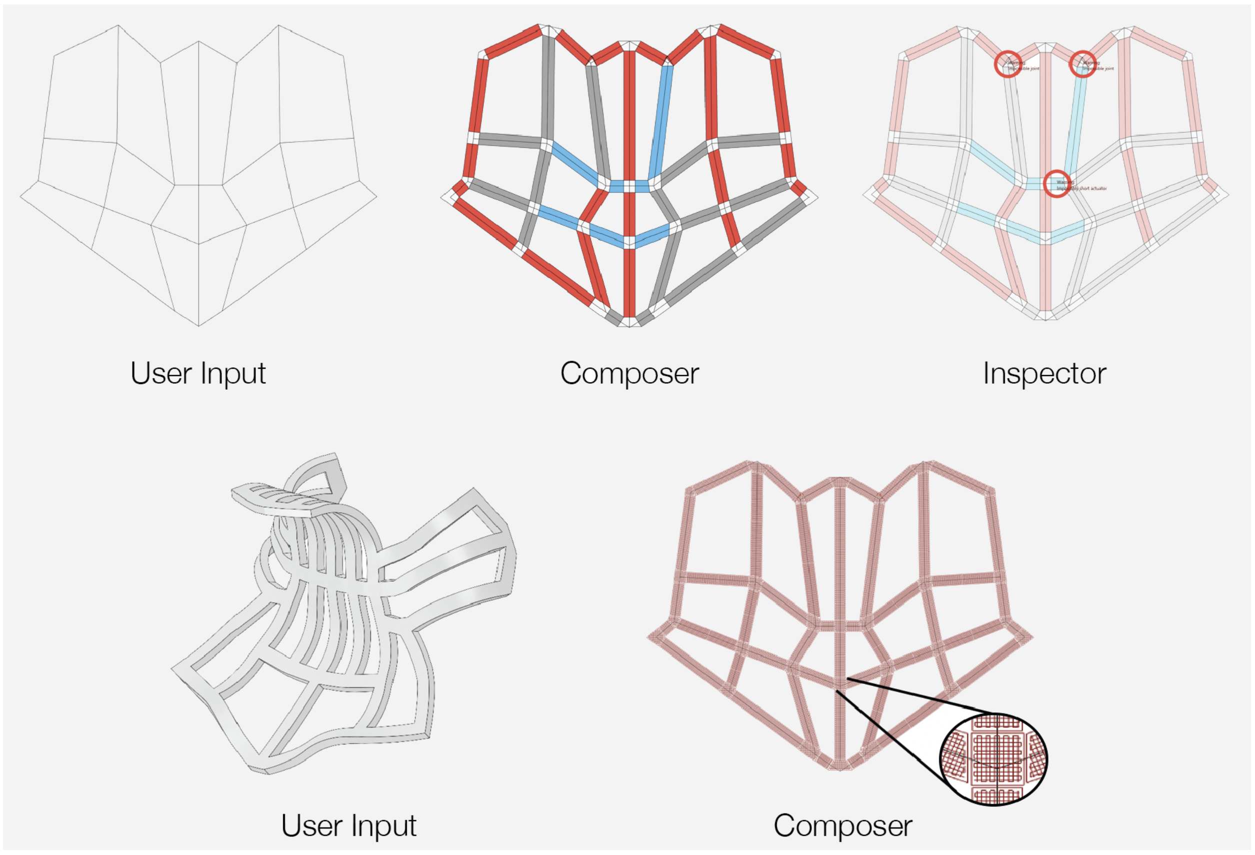


**Figure 37. Software Modules.**

## Module Details

### Composer

As a rule of thumb, the input of the design tool should be as succinct and adjustable as possible to promote design and iteration convenience. In our implementation, the design tool takes a collection of lines representing the axes of beams as input to model the geometry of grids, and designers can adjust the model by dragging the endpoints of lines or by deleting/adding edges. By default, the composer till initialize the beams as passive units, and users can assign them as actuators by typing into a text panel. Lastly, users can specify the fixed-end condition by selecting the index of joints.

### Inspector

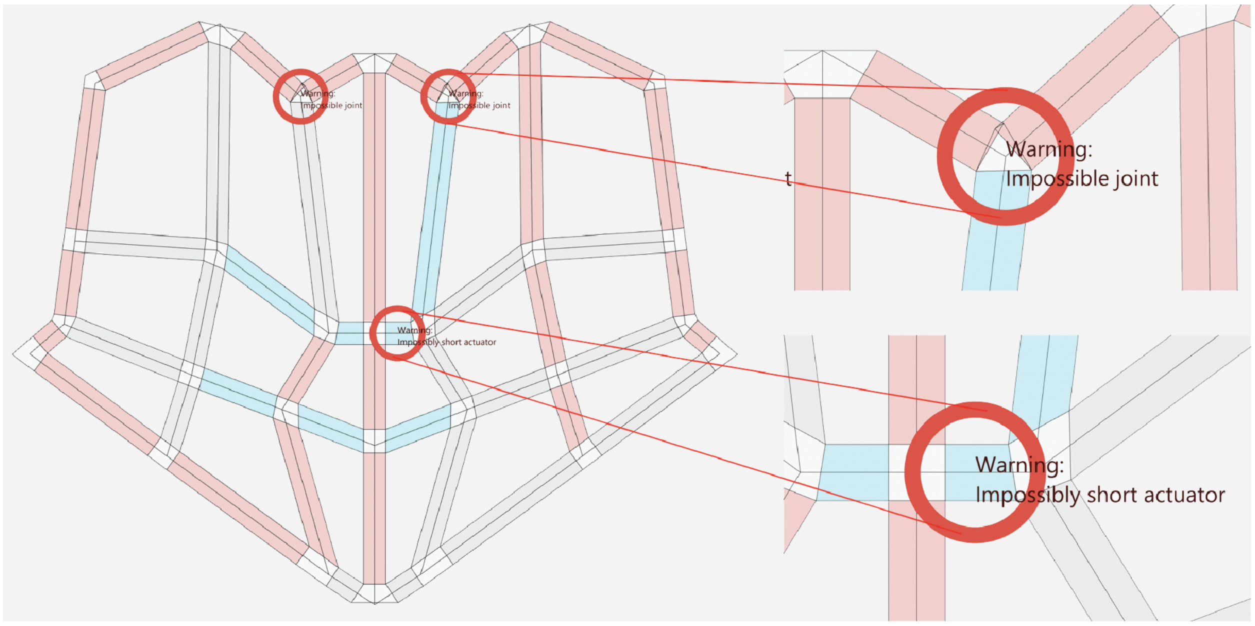


**Figure 38. Inspector Module. The inspector will check the input against the material system's design rules and inform users of violations.**

When composing dedicated CAD tools for material-driven design practices, it is essential to communicate the tools' limitations to the users to avoid frustration. These limitations originate material systems, simulators, or fabrication tools. For instance, our inspector implementation will check the 3D model and inform users when a joint violates the quad-mesh topological requirement of our material system; when the design exceeds the maximum printable area of our printer; or when the length of the beams is too long/short and therefore unseen in the simulator training dataset. In particular, since

SimuLearn is a data-driven simulation technique, the simulator cannot perform reliably on designs made of actuators that have parameters unseen in the dataset (Figure 38).

**Simulator**

The simulator takes 3D models and actuator assignments as input, converts them into graphical representations, and simulates their transformations with a pre-trained SimuLearn engine. In our implementation, the simulator will output the intermediate graphs at each simulation time step to provide animated transformations and to inform design decisions. In order to visualize the simulation results, the 3D models are reconstructed from the geometrical terms in the feature matrices of graphs.

**Slicer**

The slicer is integrated into our design tool to avoid switching between CAD and slicing software during prototyping. In our implementation, the slicer produces fabrication files by traversing down the hierarchy of composition. A design is decomposed into its building blocks (joints and beams), and building blocks into multiple toolpath blocks for rasterization. For the beams in the design, their toolpath blocks are always rasterized with a printing direction either parallel or perpendicular to the longitudinal axes of beams. The joints, on the other hand, are not required to perform transformation and are rasterized into axis-aligned toolpaths for convenience. Finally, the slicer will translate the rasterized toolpaths into G-Code files for fabrication.

# RESULTS

In this chapter, we will utilize our pipeline to produce proofs-of-concept for the techniques we developed in this thesis. The implementation is also deployed in several design tasks to evaluate the design space innovated by data-driven simulations and composition-based design tools.

## FEA Model

We use two samples of different dimensions to verify our FEA model at different scales  - 3x3 grid design that is 150 mm * 120 mm in size (Figure 39), and a 7x3 grid that is 410 mm * 100 mm in size (Figure 40). In each round of the experiment, we sample several points on the grid, measure their relative distance, and compare the physical ground truth with FEA simulation to determine the error. The maximum simulation error of the small sample is ~4.9% with respect to its dimension whereas the large sample has a maximum error of ~4.5%.

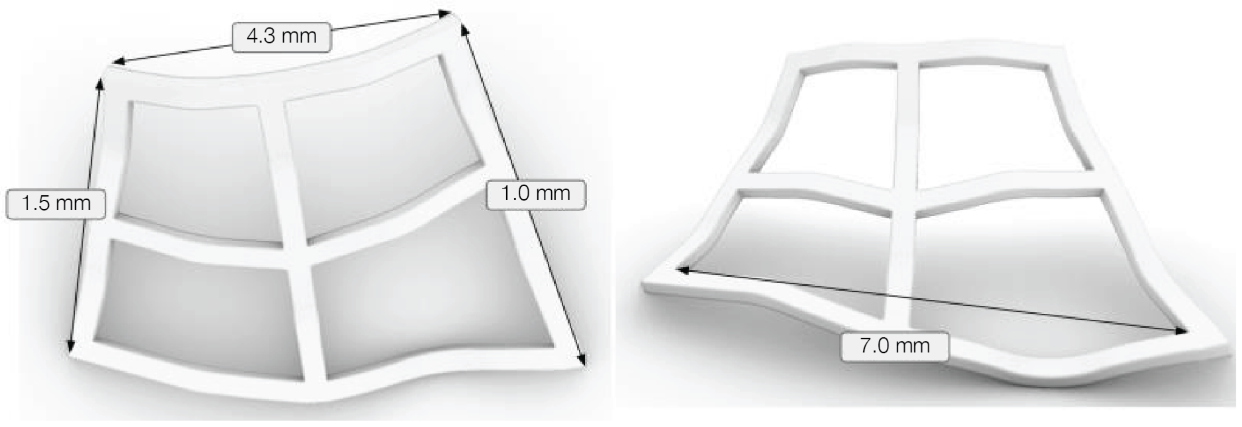


**Figure 39. 3x3 grid simulation error. The maximum error is 7 mm (4.9%).**

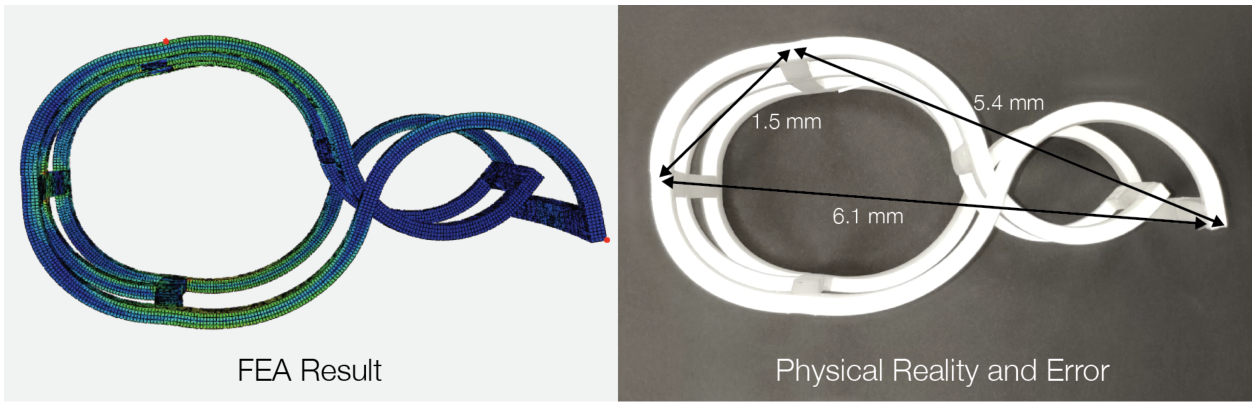


**Figure 40. 7x3 grid simulation error. The maximum error is 6.1 mm (4.5%).**

## Dataset

We use our pipeline implementation to simulate 200 3x3 grid designs to collect raw FEA results (Figure 41). The FEA solver logs ten temporal frames, and the results are rotated ten times to eliminate rotational bias, therefore allowing us to extract 100 graphs out of 1 trial of FEA. The largest grid dimension appeared in this dataset is approximately 200 mm * 200 mm, and the smallest is 100 mm * 100 mm in size. The 200 episodes of FEA are deliberately divided into three groups for different types of actuator assignment that can morph into varying shapes- all bending upward, all bending downward, or each actuator bending on a randomly assigned direction - to produce different types of transformed shapes. The first two groups of grids that have the actuators bending on the same direction will transform into bowl-like shapes, whereas the later will either become a saddle shape or an arbitrary surface.

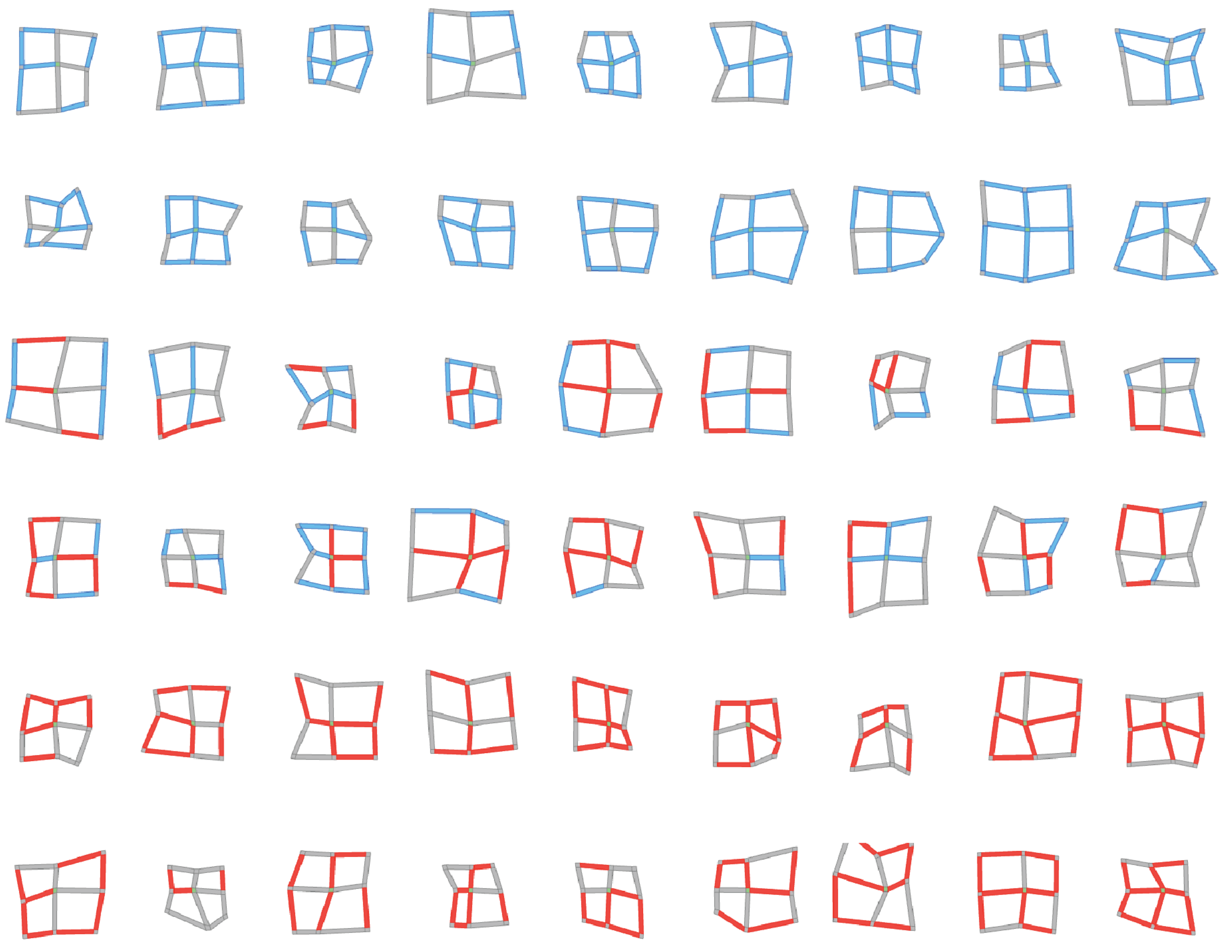


**Figure 41. A subset of the curated 3x3 grid dataset.**

## SimuLearn Performance

We use the curated dataset to train a double GN model with BGD for 1000 epochs, in which the second half has a 0.1x learning rate compared to the first half. In total, it took us 24 hours to train the model with our software implementation, and the entire pipeline produces around 204 GB of files (201 GB of FEA raw results, 1.5 GB of training data, and 0.75 GB of ML model).

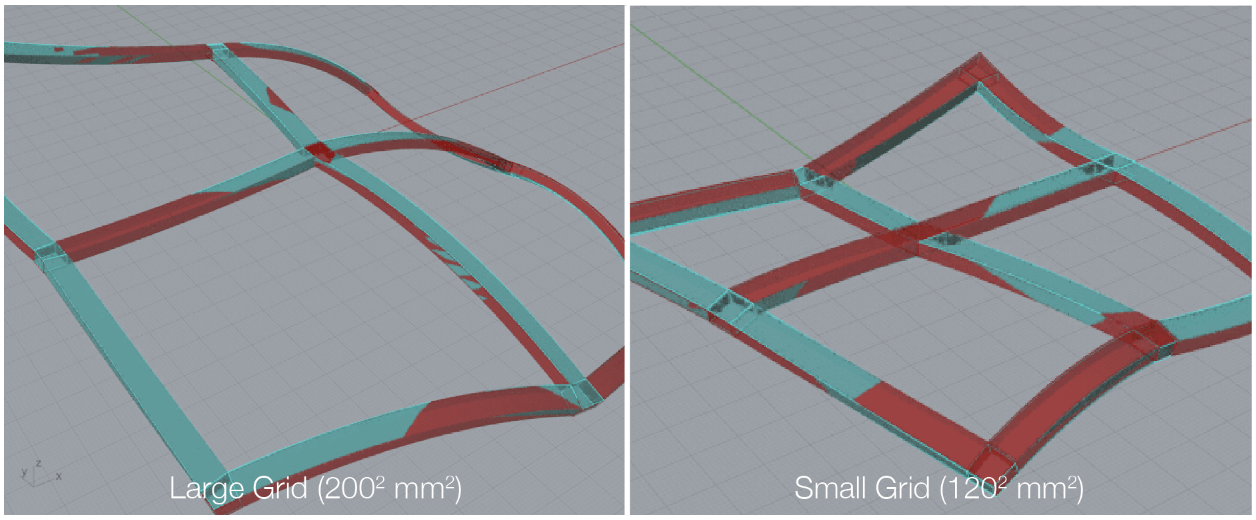


**Figure 42. Simulation Result. Blue - FEA ground truth; Red - SimuLearn result.**

The trained model is validated on 2 grids of different dimensions (Figure 43, 44). Compared to FEA ground truths (Figure 42), the maximum simulation error we measured is ~1.2 mm, which corresponds to a ~1.6% error with respect to the dimension of the grid. Compounding this error with our FEA simulation accuracy, this instance of SimuLearn engines is expected to have ~6.4% maximum error with respect to physical results. On another hand, the simulation took only 0.8 seconds to compute and runs 1500x faster than FEA (1200 seconds). In general, this result shows that SimuLearn can perform FEA-like simulations with a substantial acceleration.

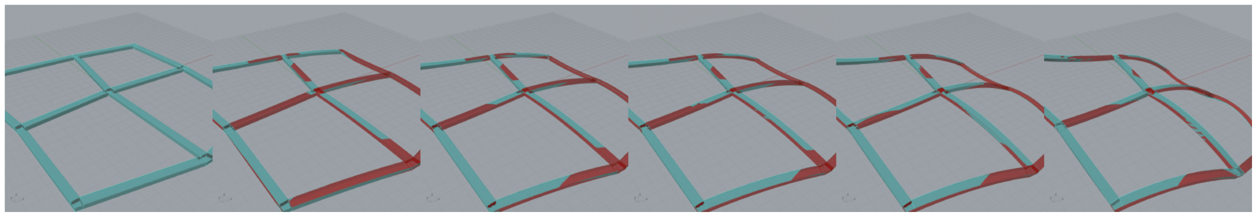


**Figure 43. Large 3x3 grid simulation rollout. Blue - FEA ground truth; Red - SimuLearn result.**

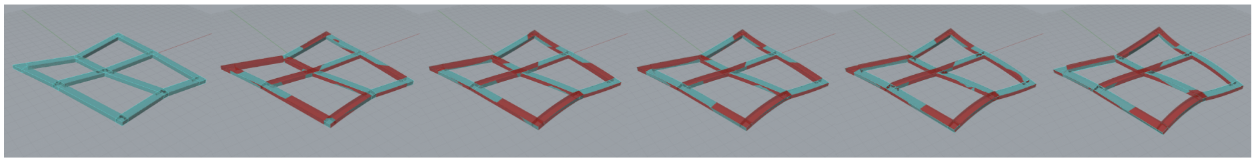


**Figure 44. Small 3x3 grid simulation rollout. Blue - FEA ground truth; Red - SimuLearn result.**

For our material system, a maximum simulation error of 5% is already sufficiently accurate due to the compliance of our material and the unaccountable factors in the physical world. Although not the dominant factors, the actuation environment has several factors that complicate the transformation process, affect the triggering result, but are challenging to model. For instance, convections within the hot water bath may slightly deform the object but are computationally difficult - if not impractical - to model. It is also impossible to consider uncontrollable external factors like deformation or shaking in our FEA simulations. In practice, we can leverage the flexibility of our material to bypass unaccountable physical factors as well as to absorb simulation errors.

While our trained SimuLearn engine approximates FEA results well, the inaccuracy of our FEA model still makes the final accuracy with respect to the physical reality insufficient for design tasks. Another important thing to note is that since our dataset generator produces grid geometries by random, regular grid designs rarely appear in our dataset and consequently causing our simulator to bias. The trained simulator performs well on grids that have irregular structures but poorly on regular patterns that are commonly seen in human-produced designs (Figure 45).

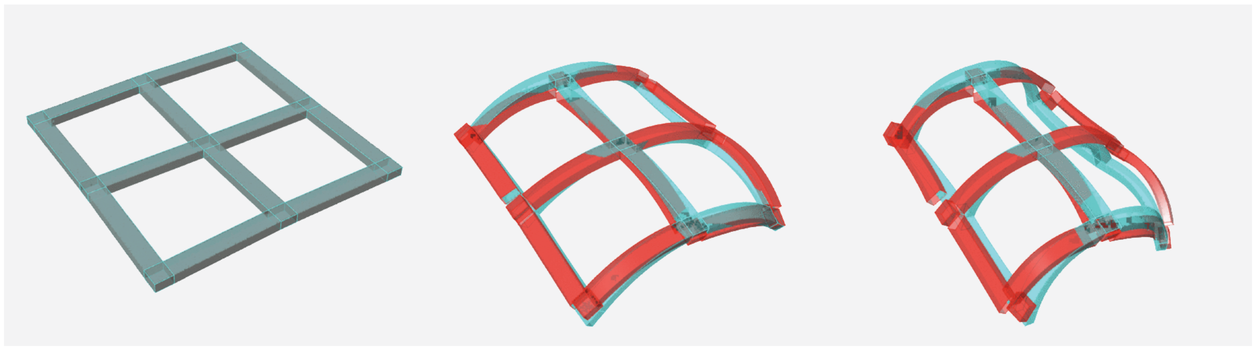


**Figure 45. Regular grid simulation failure. Blue - FEA ground truth; Red - SimuLearn result.**

## User Workflow

Our software prototype affords users to sketch, simulate, and iterate designs in a single CAD environment without having to switch between multiple software. Using our implementation, a designer can initialize a pattern by intuition, simulate its transformation, evaluate the result against certain design goal (e.g., shape-fitting), adjust the pattern, and repeat this process until the outcome is satisfactory. Compared to previous works, our design tool allows users to directly operate on patterns without having to use target shapes and geometrical algorithms as proxies. This feature promotes direct manipulation of design in 4D printing and enables intuitive design exploration and iteration.

Simulations play an essential role in the workflow mentioned above to inform effective design decisions. Ideally, the design tool will adopt SimuLearn engines to produce physically accurate and fast predictions, but our current simulator is merely a proof-of-concept that has limited versatility (i.e., only trained for 3x3 grids, not performing well on regular grids) and accuracy for design tasks. Therefore we are using FEA for simulation in our current implementation to enable an extended design space and to compromise speed for accuracy (Figure 46, 47).

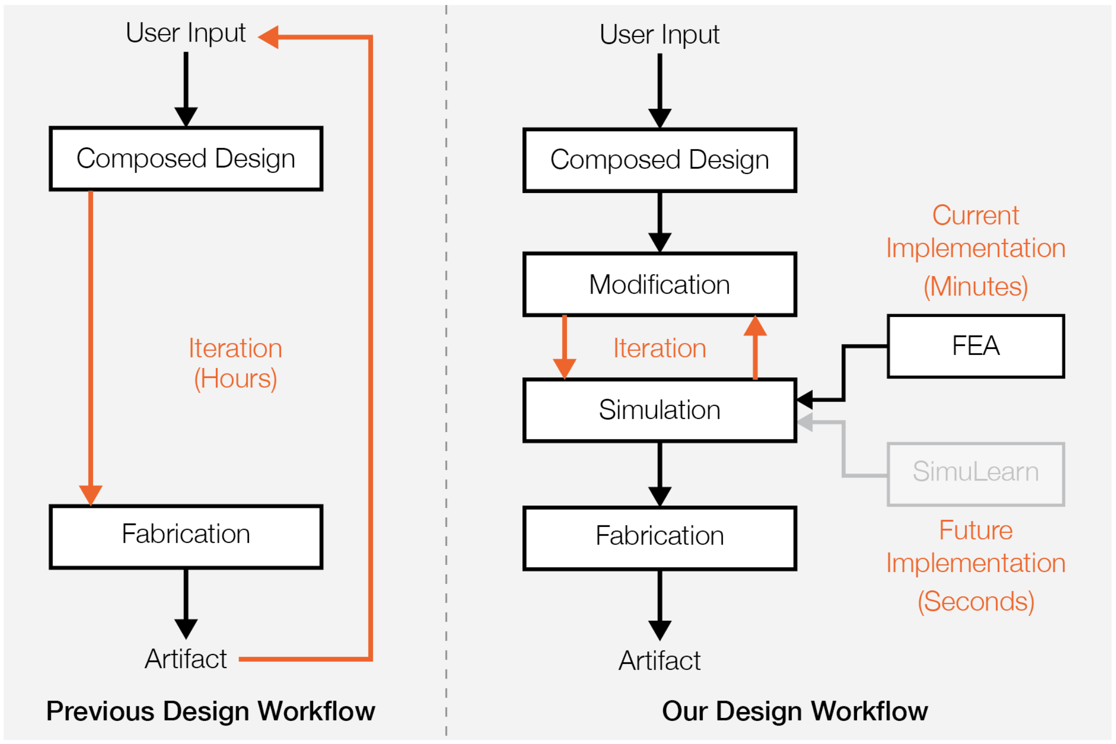


**Figure 46. Comparison of design workflows. Our implementation allows for economical iterations.**

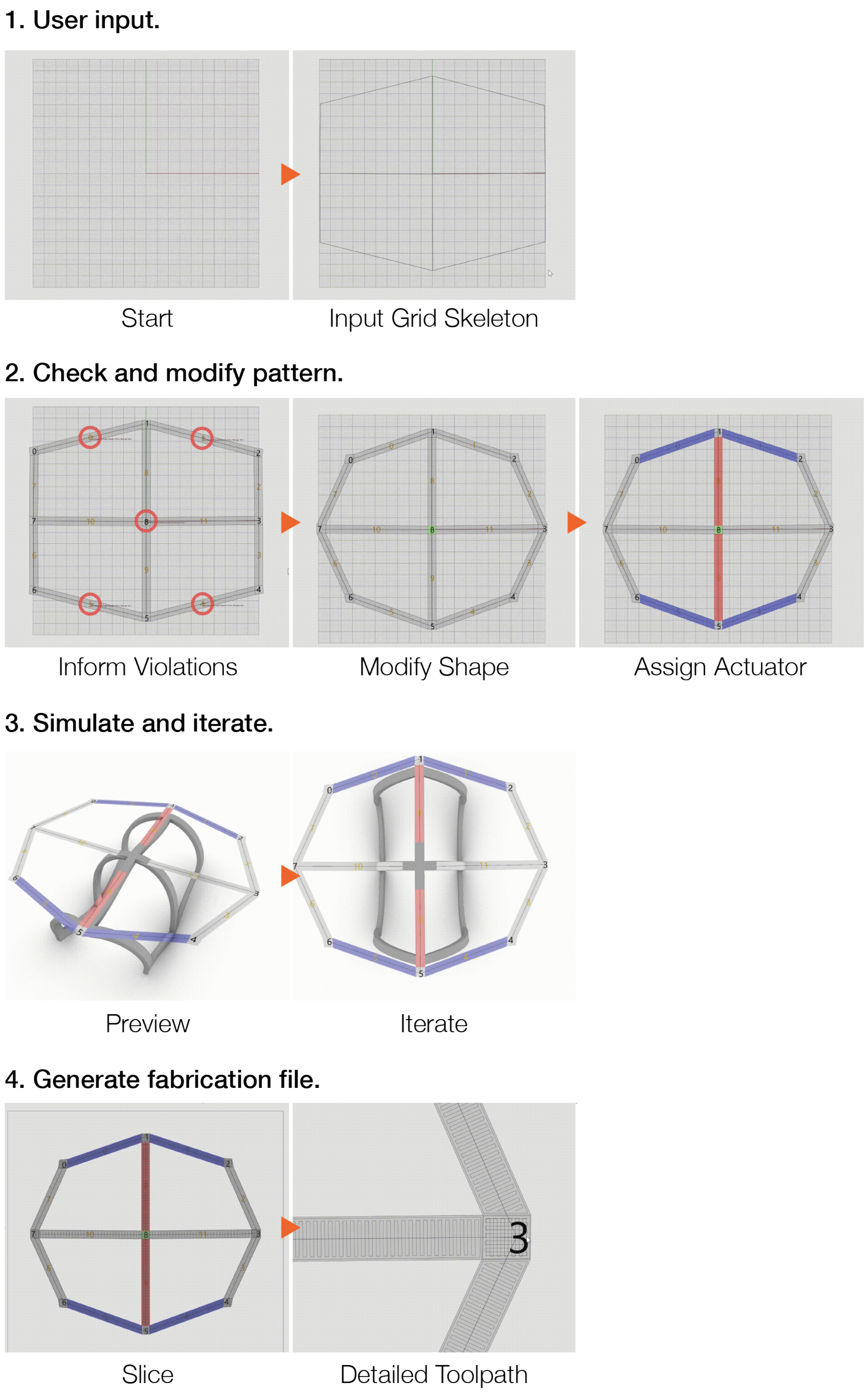


**Figure 47. Workflow with our design tool implementation.**

## Design Space

We deploy our software implementation to several design tasks to evaluate its usability and to reveal the design space enabled by it. Note that when using FEA for simulation as in our current implementation, each task took roughly three days to develop to the final product. Once we acquire a reliable SimuLearn engine, this iteration time can be reduced to less than half an hour.

### Neck Support

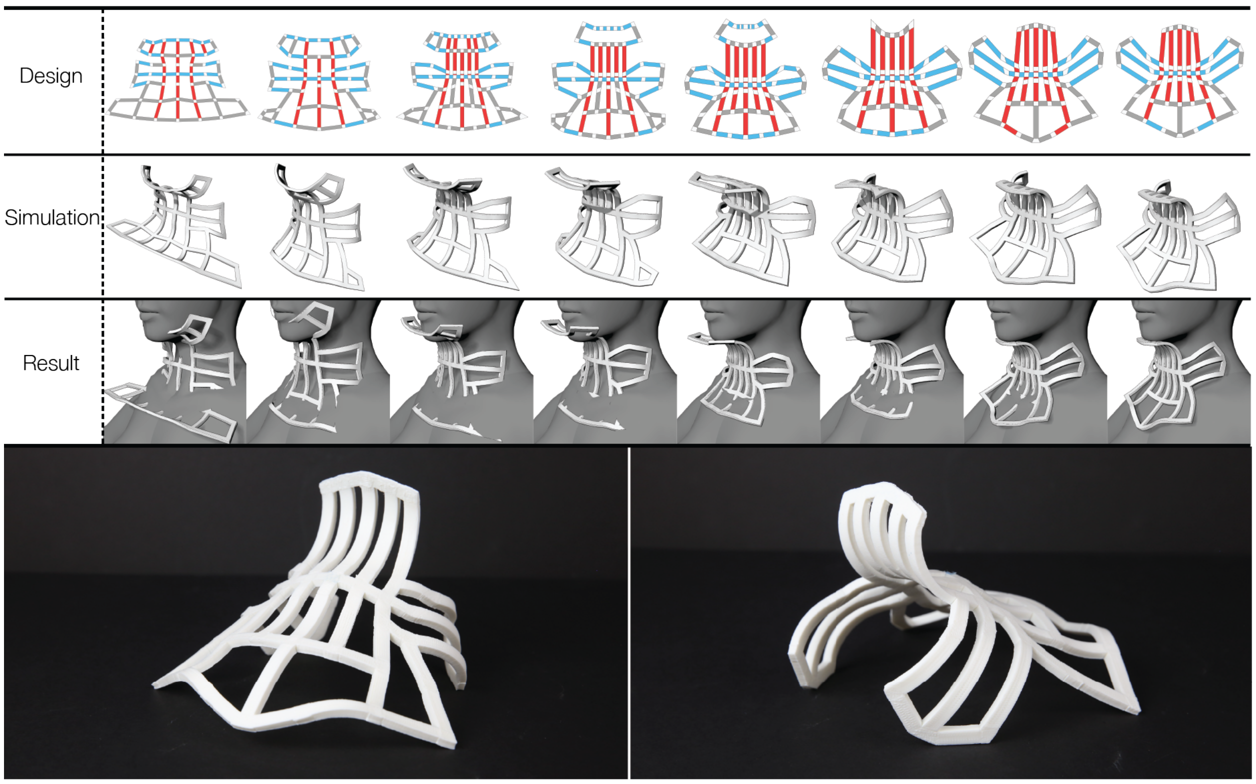


**Figure 48. Neck support design iterations. Top - design iteration; bottom - printed artifact.**

The goal of this task is to customize neck support (Figure 48) for an imaginary patient with neck injuries. The 3D model of the wearer is provided to contextualize design iterations. In the task, we use the simulation results to evaluate the fitting of the design as well as to adjust its aesthetics. In particular, this part of the human body has large curvatures and a non-developable shape, which renders it difficult to design for with currently available 4D printing tools. For instance, *Thermorph* can only produce origami-like patterns with limited structural strengths. While *4DMesh* adopts a material system identical to ours and is structurally stable, the geometrical limitations of its algorithms make it incompatible with this part of the body. Our design tool and workflow, on the other hand, is capable of handling this task.

### Bottle Handle

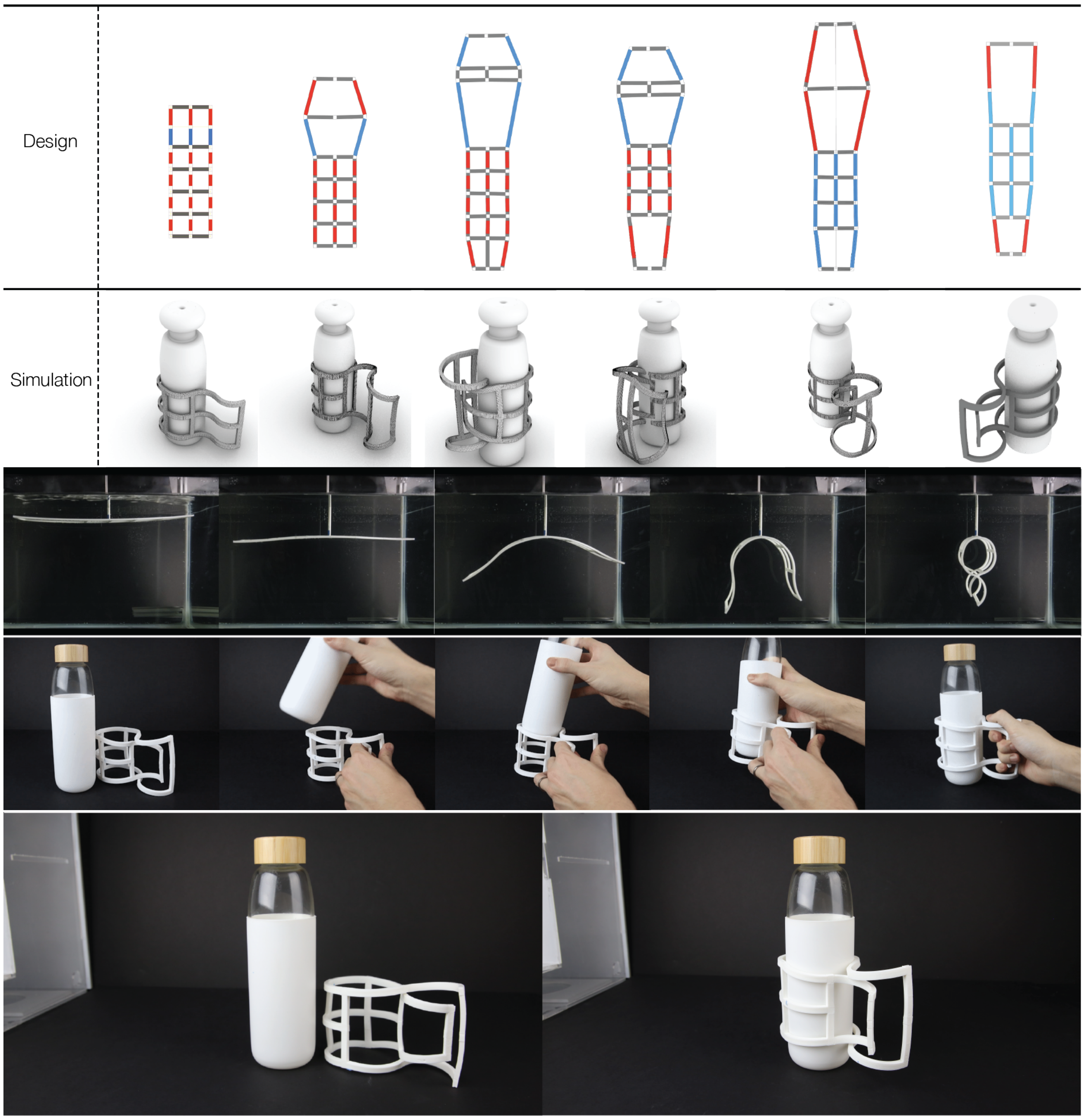


**Figure 49. Bottle handle design. From top to bottom - design iterations; triggering process of the final design; attaching the handle to the bottle; before/after assembly comparison.**

We take water bottles as an example to exemplify our tool's applicability in accessibility augmentation. A bottle handle allows users with certain disabilities to conveniently hold on and drink from it (Figure 49).

In addition to fitting the design to the shape of the bottle, we also use the simulations to explore different methods of designing the handle. Noticeably, the final design exhibits self-intersection that is previously unachievable with available 4D printing flattening algorithms as it violates the surface/mesh manifoldness assumption. Lastly, some of our designs are curling in opposite directions, and we use the simulations to identify appropriate fix-ends to avoid the artifact from hitting the support during activation.

**Modular Lampshade**

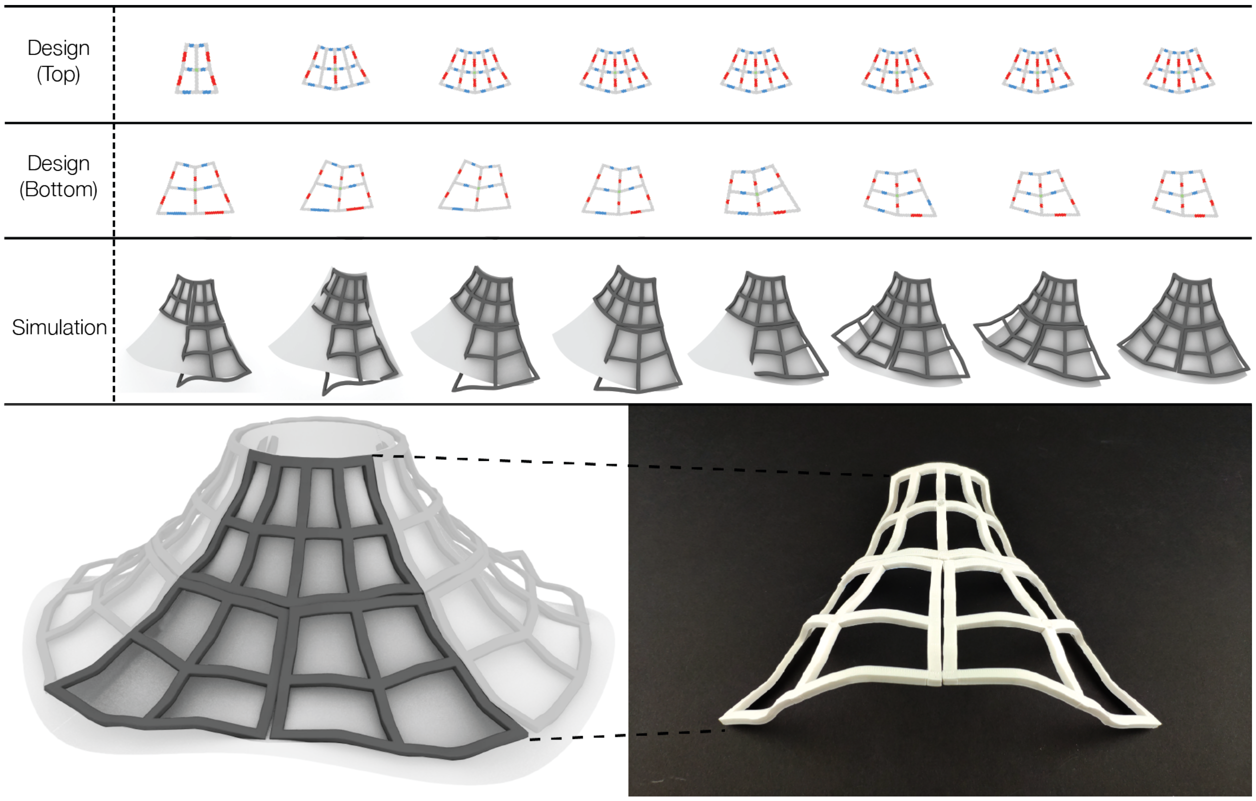


**Figure 50. Modular lampshade design. Top - design iteration; bottom left - module and assembly render; bottom right - physical prototype.**

The accuracy of simulation allows us to fabricate a large object as a combination of modular structures. In this task, we divide a lampshade into multiple segments and design each part with our tool, and use simulations to make sure that the parts do not collide or intersect with each other (Figure 50). The precision of simulation will also allow us to match the outlines of segments well and design connections between modules accurately. Note that the beams are assigned as partial actuators in this particular application to achieve small curvature.

# LIMITATION AND FUTURE WORK

The goal of this work is only to establish the concept and implementation pipeline of SimuLearn - our technical solution to bring materials and geometries one step closer in computer-aided design. Thus, the techniques we developed have their limitations that can be addressed in future works. Here we provide elaborations for each of the issues we have identified.

# Data, Data, and Data

## Parameters and Variances

Datasets are the very foundation of any data-driven technique. Identically, dataset curation also bounds the performance of SimuLearn. These limitations come in two folds - accuracy and dataset size.

When composing SimuLearn simulators, the expected error is the compound of FEA and regression error. While FEA is considered the engineering norm and ground truth of simulation, the results are often imperfect compared to the physical reality due to incorrect material or boundary condition modeling. Furthermore, machine learning regressors - especially the neural networks and gradient descend optimizer we use - cannot perfectly fit the training dataset and will always introduce error to the predictions.

The dominant factor of SimuLearn's dataset size requirements or parameter spaces is the number of design parameters. The more design parameters we take, the exponentially more data we need to train a reliable simulator. In this study, the parametric FEA input file generator is taking only a few parameters as input and setting certain parameters, such as the width and thickness of beams, as constants to confine the parameter space and data requirement. While we can technically account for these factors by curating a larger dataset, the exponential computational cost renders it impractical.

Lastly, this thesis does not investigate in depth the relationship between SimuLearn accuracy and the quality of dataset (i.e., quantity, variance). Follow-up works may wish to provide comprehensive studies of the influential factors.

## Dataset for Design Tasks

Our pipeline takes a random sampling strategy to generate data for ML training, but these randomly decided grid designs does not necessarily reflect human users' design preferences. For instance, human users tend to design regular actuators assignments (i.e., developable surfaces, clustered bending directions) and layout (i.e., approximately rectangular, trapezoidal, or parallelogrammatic), but these designs are unlikely to appear in our random dataset (Figure 51). As a consequence, the trained SimuLearn engine does not perform well on these regular patterns and often produce ostensibly wrong simulation results. The naive solution to this issue is to increase the size of our dataset, assuming that the more data we have, the more likely these regular designs will appear. Alternatively, a more effective curation method that may alleviate this issue is to invite human users to generate designs, such that the dataset reflects real-world design scenarios. However, this is a labor-intensive process, and it is difficult to generate large datasets with this curation method.

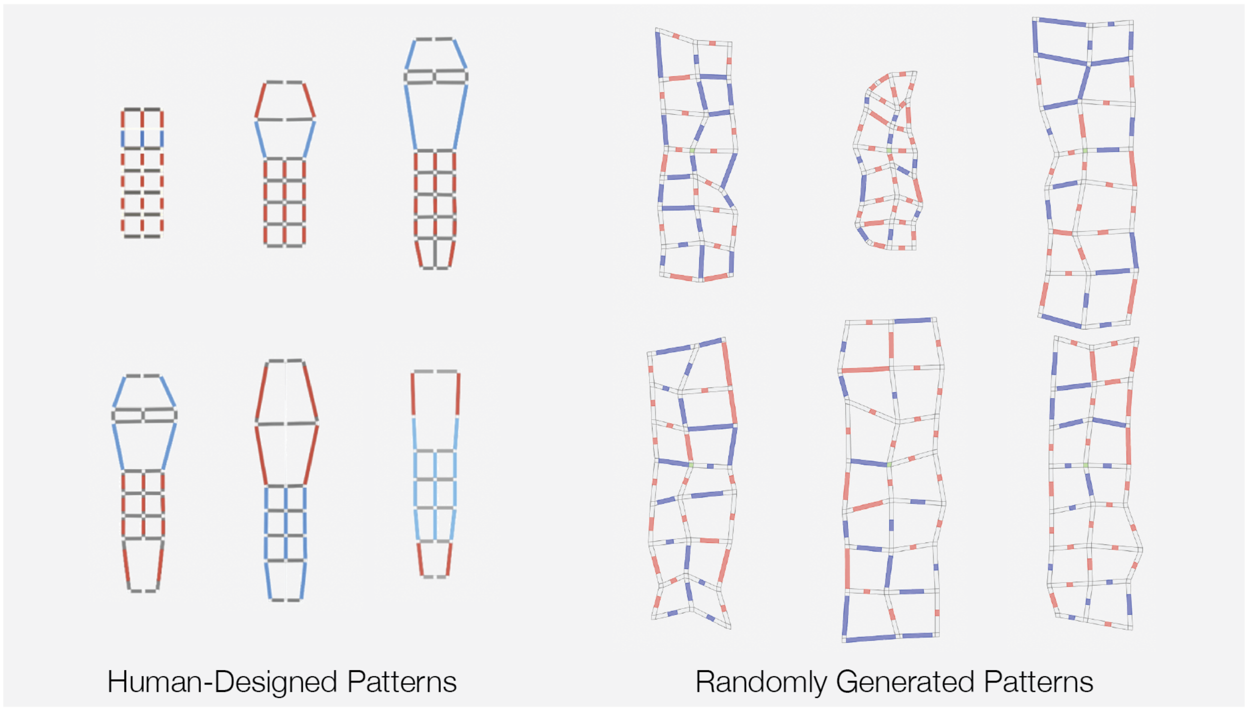


**Figure 51. Design regularity comparison. The regularity of human-designed patterns are rare to appear in the randomly generated dataset.**

## Generalizing the ML-FEA Technique

### Current FEA Model Limitations

Compared to ML regression error, FEA accuracy is currently the bottleneck to our technique. In particular, the material that we are using is susceptible to the disturbances in the triggering environment, and the transformation process always has some uncontrollable factor to it. Thus, when we measure the accuracy by maximum simulation error, it always results in poor performances. That being said, our FEA modeling may still be inaccurate, and future works may wish to characterize the material model more rigorously.

While the material definition, boundary conditions, and solver setting of our FEA model are physically-based, there are several factors that we are omitting in the current implementation, such as collisions during transformation. The bending actuators in our material system do not have high performances (i.e., large bending curvatures) and the grids appeared in this work cannot deform large enough to collide with themselves. Therefore, we ignore self-collisions to simplify our FEA model and computation. Similarly, since the artifacts are fixed to suspend in the water bath, we can ignore their collisions with surroundings. Future works that wish to consider collisions can regard them as a dynamic relation in graphical representations. For instance, if we instead trigger the 4D printed grids by letting them sink to the bottom of the water bath, we may use an extra adjacency matrix to dynamically describe contacts between beams/joints and the tank bottom, use an additional neural network to compute their interactions, and convolute the interactions into the update of blocks.

Due to the irreversible mechanism of our material system, our current simulator will iteratively update the input grid design for only a fixed amount of time to compute its transformation. In this scenario, the last output is the transformed shape of the one-time actuation. Nonetheless, our SimuLearn technique is also adaptable to reversible actuation like moisture-responsive wood curling. To simulate these physical systems, we only have to allow the simulator to iterate indefinitely. Lastly, this thesis does not exhaustively investigate the relationship between SimuLearn accuracy and the quality of dataset (i.e., size, variance). Follow-up works may wish to provide comprehensive studies of the influential factors.

**Applicable Patterns and Materials**

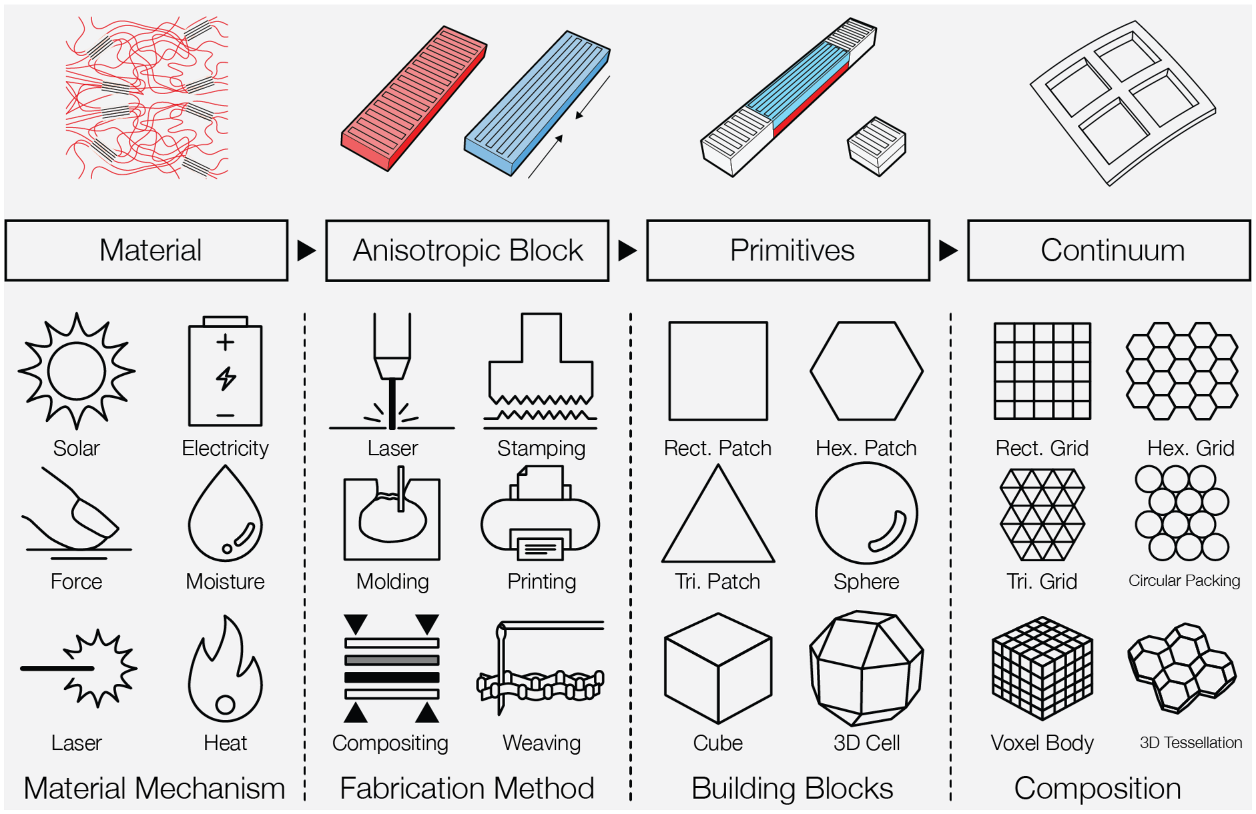


**Figure 52. Alternative material systems. SimuLearn can be adapted for other transformative mechanisms, fabrication methods, and compositional units and patterns.**

The mesh-like patterns discussed in this thesis are intuitive to abstract into graphical representations, but this approach is also adaptable to alternative compositional patterns (Figure 52). For example, we can simulate shell structures by decomposing them into small patches of surfaces, or breakdown voxel bodies into smaller, local groups of voxels for simulation - identical to mesh coarsening in finite element methods. Provided that the simulation units allow for consistent representations, we can represent their compositional designs as graphs and apply the SimuLearn technique on them. Additionally, the PLA in our material system already has a non-linear material property, indicating that this technique is also applicable to a wide range of materials. Future works may adapt the SimuLearn technique to other types of material systems that have different actuation mechanisms, fabrication methods, and compositional units and patterns.

### Graphical Representation Extension

In this work, our implementation affords compositional patterns made of bending beams and passive joints, each corresponding to the edges and nodes in our graphical representation. Nonetheless, graphical representations also allow for multiple types of nodes and edges to coexist in a pattern. For instance, we can add shrinking beams into our material system to expand its design space, and our graphs will then consist of two types of edges. To account for this, we can then supplement our graphical representation by adding a new matrix to encode the features of shrinking beams and an additional sub-networks to our GCN to compute its interactions and updates. Furthermore, this method also allows us to integrate actuators made of different materials into one design. However, increasing the classes of edges and nodes will result in a quadratic size growth of ML-FEA models, which at the same time may require more training data.

## Large Continuum Simulation

Although not being in the scope of this work, training an ML-FEA simulator to predict the transformation of large continuums reliably may be a difficult and impractical task. For instance, training an ML-FEA engine to simulate 5x5 grids will require us to curate a dataset of 5x5 grid transformation. However, using FEA to simulate 4x4 grids will take substantially more time compared to simulating 3x3 grids. Additionally, errors caused by inaccurate FEA modeling tend to propagate and amplify with increasing model size, resulting in untruthful simulations. In order to address this, a common practice is to divide large continuums into several parts, simulate them individually, and put the results back together to get the full model. This method is also compatible with our ML-FEA technique. In particular, we can extend the hierarchy of composition to account for more levels of abstraction. Take 5x5 grids for example, instead of simulating them as continuums, we can decompose them into 3x3 patches for simulation to confine error propagation. On another hand, extending the hierarchy of composition will also allow us to design with higher-level elements, further saving design efforts (Figure 53).

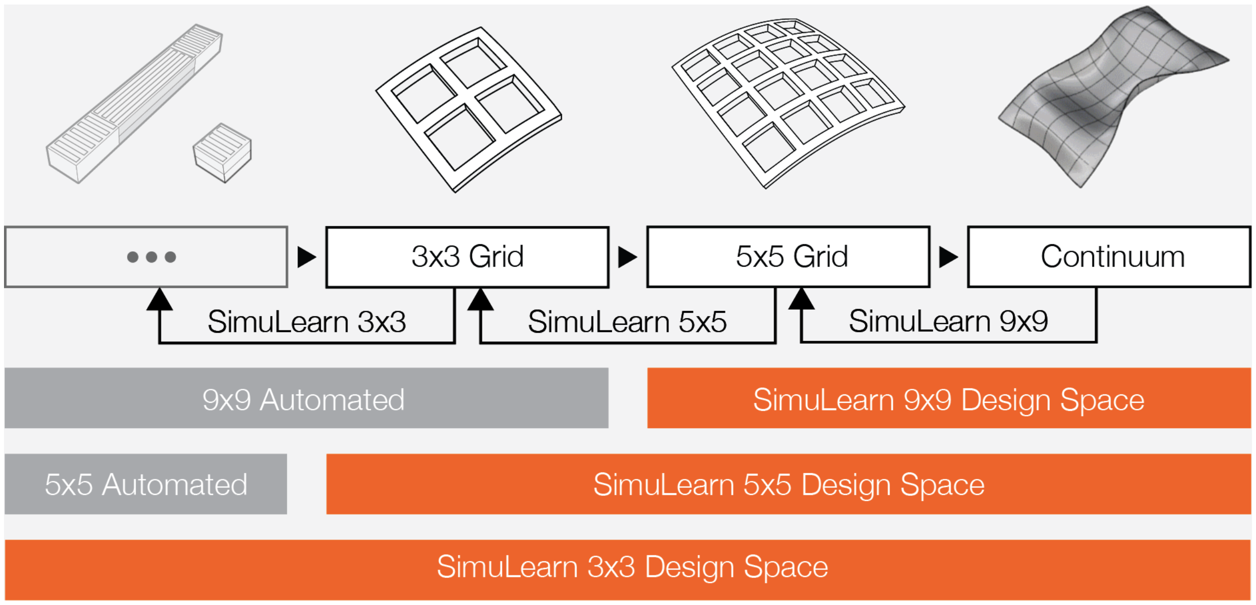


**Figure 53. Extending the hierarchy of composition. When designing larger-scale patterns, we can extend our compositional hierarchy to operate at higher-level abstractions and automate lower-level efforts.**

## Design Tool Reliability

Our current design tool implementation does not communicate well the validity of simulation results. Most commercially available FEA solvers monitor computation processes to detect divergences (i.e., unsolvable scenes) and raise errors when encountering one, but ML-based simulators guarantee results regardless of the input - even if the results are erroneous. We can resolve this issue by implementing a checker function to validate simulation results before showing them. Take our material system for example, we can check the simulation results by monitoring dislocations of supposedly adjacent faces and vertices against a threshold to identify false results.

On the other hand, this thesis mainly focuses on discussing the simulation and design tools of morphing matter, deemphasizing on the fabrication methods and tools to initiate our study. Nonetheless, providing a reliable and generalized fabrication module for 4D printing and morphing matter will also be a substantial contribution to the field that future works can address.

## Differentiable Simulation

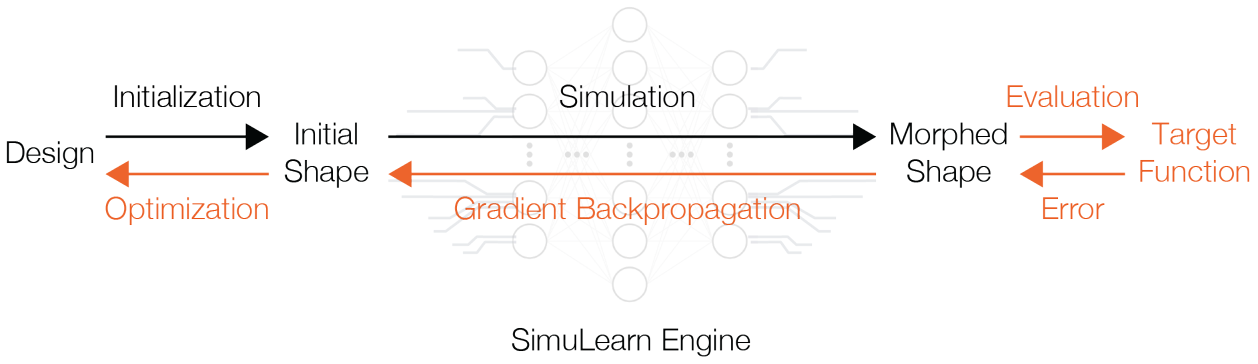


**Figure 54. Optimization leveraging SimuLearn's differentiability.**

In addition to its fast computational speed and convenience of wrapping complex physical phenomena into a singular model, using SimuLearn for simulation also enables inverse-design due to its differentiability. This approach is commonly used in robotics for policy-finding (Belbute-Peres et al., 2018), takes a gradient descent-based approach to optimize designs, and has a mathematical mechanism identical to the backpropagation algorithm of artificial neural network training. Given an initial design, we can use a SimuLearn engine to simulate its transformation, evaluate the result against an objective function to compute its loss, and back propagate the loss to the input while fixing the weights of our simulator to optimize the input design (Figure54). The objective function can take any form and represent any design goal as long as it is differentiable. For instance, if we want the design to transform into a particular explicit shape, our objective function will be the summation over the Euclidean distances between target and simulated vertices.

Previous works of 4D printing inverse-design tools are mostly based on geometrical methods. The algorithms are adapted to specific material systems and inapplicable to others, and they often assume the target shapes to satisfy additional geometrical properties such as surface manifoldness or developability. Inverse-design via differentiable simulation engines, on the other hand, is a generalizable technique and does not require the target shape to satisfy any geometrical prerequisite other than the material system's. In a way, as long as we can simulate transformations with the ML-FEA simulator, we can use the same engine to design artifacts inversely.

# DISCUSSIONS

Beyond the techniques and implementations developed in this thesis, we are also conceptualizing a design philosophy of for material-driven practices. In this chapter, we will briefly discuss several of such thoughts and envision the CAD future enabled by data-driven simulations.

# Design Decisions

### Biases as Design Decisions

In conventional ML application domains, data biases are often induced by missing important features of the task or having poor variances of parameters and are perceived as negative properties as they lead to poorly generalized ML models. To remove these biases, a straightforward solution is to curate larger, more variant datasets, assuming that the extra data will cover more configurations of parameters. However, this approach is often impractical due to real-world limitations (e.g., storage space, labor, time).

Compared to conventional ML domains, this thesis conceives biasing as a crucial element to compose simulators successfully. Strategically biasing the dataset on important design parameters will reduce the dataset size requirement for ML training as well as help confining the design space of the material system. In our case, we prioritize the parameters that that users prefer to play with during design processes and remove less frequently adjusted ones from our implementation. In other words, dataset biases are design decisions that toolmakers will have to make when following our pipeline, and they contribute toward composing dedicated CAD tools and successful ML training.

### Rationalizing Simulator Developments

Composing data-driven simulation engines is virtually a tradeoff between implementation time and design convenience (Figure 55). In our pipeline, dataset curations are computationally heavy tasks that can easily take days - if not weeks - to perform, but the resulting fast and physically accurate simulators can accelerate design iterations substantially. Designs that would conventionally require days to physically prototype and iterate will only take minutes with our method. That being said, developing a SimuLearn engine requires thoughtful rationalization to justify its cost. Our simulation technique is economical only if the trained simulator is frequently used, such as deploying material systems to the industries or for mass-customization. For small-scale applications like academic research on novel materials, our technique is less practical, and FEA will instead be the logical method for simulation. Lastly, SimuLearn is expected to be less accurate than numerical methods like FEA and may not serve engineering domains as well as design tasks.

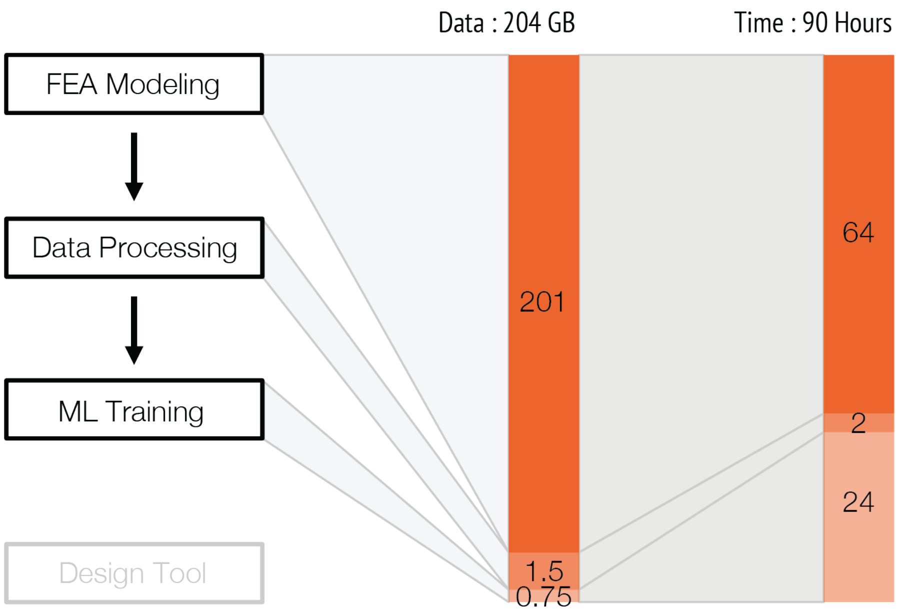


**Figure 55. 3x3 SimuLearn development costs. This development process is a tradeoff between development time and deployment convenience.**

## Data-Driven Simulation

### Concurrent Learning

While the ML-FEA simulator development workflow may appear to be linear, its implementation is, in fact, an iterative learning process. The FEA modeling step demands an accurate understanding of the factors affecting simulation results; the data processing step necessitates proper representations for the patterns and careful confinement of design spaces; the ML training validates the result of previous steps. In our implementation, we had to iterate each step several times in order to acquire a properly functional pipeline, each time expanding our knowledge of the material system. The pipeline serves not only as a tool to develop SimuLearn engines but also as an educational tool that helps implementors to gain insights.

### Beyond Mechanical Simulations

The simulation scope of this thesis - residual stress-induced deformations - is a well-studied domain in mechanical engineering. We have established accurate numerical models for this phenomenon and know well its affecting factors, therefore allowing us to utilize FEA to collect the dataset for ML training. However, there exist domains that we currently do not have an accurate numerical model for, such as the growth of *Muscular Thin Film Actuators* (Feinberg et al., 2007), and we can modify our implementation workflow to take real-world observations as the source of data. For instance, we can record videos of cellular activities, use computer vision techniques to tag and extract information from these observations, and use this data to train ML models to predict their actuation performances and behaviors (Figure 56). Perceiving the data-driven simulation technique as an engineering tool, it will enable us to design with physical systems that we do not even have deep understandings.

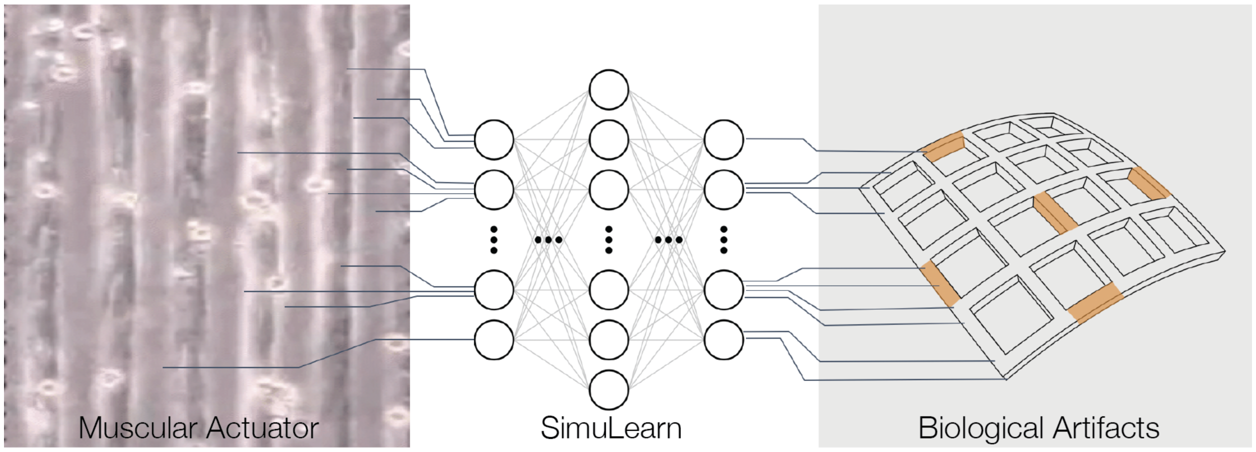


**Figure 56. Adapting SimuLearn for biological materials. We can use SimuLearn to approximate and predict material behaviors that are not yet well-characterized, such as muscular actuators. Image reprinted from *Muscular Thin Film Actuator* (Feinberg et al., 2007).**

## Ecosystems

### Manufacturing Ecosystems

We have the vision to democratize and create industrial ecosystems for 4D printing and morphing materials and look forward to their wide adoption in a variety of domains - interactive devices, self-assembling furniture, reactive architectural facades, and more. However, these fabrication techniques and practices are still in their infancy, and we have not yet seen them realized in the market. After having conversations with several industrial collaborators, we speculate that this is due to the lack of effective tools that assist us to design and fabricate with convenience and accuracy. This work is our initial step to advancing toward our vision by addressing the technical challenges of composing CAD tools in this ecosystem.

### SimuLearn Ecosystems

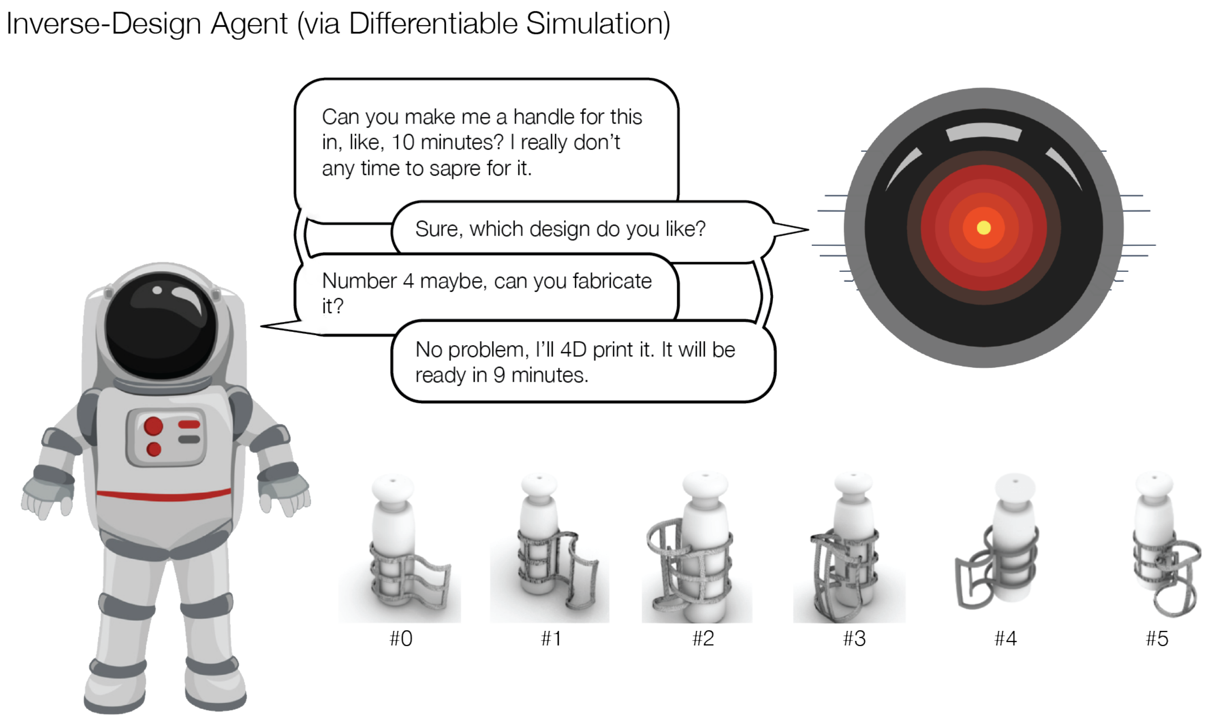
As explained in earlier chapters, curating a dataset that reflects real-world design scenarios is a difficult task. However, we can create a crowd-sourcing ecosystem that facilitates the curation of such datasets. For instance, given a material system, organizations that desire to develop SimuLearn engines can offer FEA computing services to users who wish to simulate in return of their design to collect data. Once the organizations have acquired a sufficient amount of data, they can then use it to develop SimuLearn engines and offer them as services.

## Augmentative Design Agents

The ultimate vision of this thesis - GeoMatries - is to bring geometries and material together in the CAD tools of morphing matter design, and SimuLearn is our technical solution to make such vision come true. Furthermore, this simulation technique will also enable us to create augmentative design agents that scaffold or improve design experiences (Figure 57).

In addition to its ability to perform fast and physically accurate simulations, the developed SimuLearn engines can also serve as the foundations of composing augmentative or exploratory design agents. As discussed, the differentiable computation of SimuLearn can be used for inverse-design and optimization, and we can further leverage this feature to create design agents that help us to explore design options. For instance, given a design goal, we can randomly initialize several patterns and optimize them to have good performances and offer these patterns as design options for the users to choose from.

SimuLearn also offers the basis for computers to reason over physical phenomena and design decisions in real-time, and take further initiatives in morphing matter design processes. In our current software implementation, users have to inspect the transformations to decide what to modify manually, and the computers are playing passive roles in these processes. Alternatively, we can implement additional functions to actively inform users about the consequences of modifications, or to provide guidance given design goals. For example, knowing the user's design goal, the computer can suggest modifications that will result in better performances, or inform undesired modifications (e.g., resulting in self-collisions).


Inverse-Design Agent (via Differentiable Simulation)
Can you make me a handle for this in, like, 10 minutes? I really don't any time to sapre for it.
Sure, which design do you like?
Number 4 maybe, can you fabricate it?
No problem, I'll 4D print it. It will be ready in 9 minutes.
#0
#1
#2
#3
#4
#5

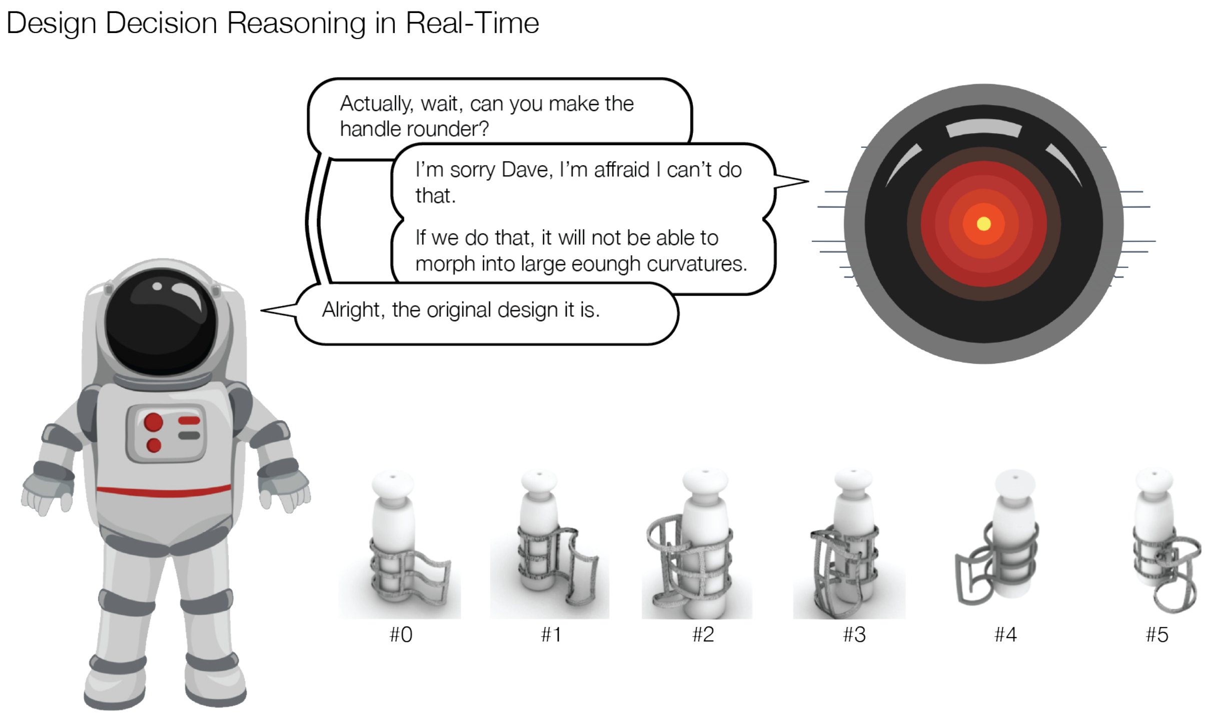

Design Decision Reasoning in Real-Time
Actually, wait, can you make the handle rounder?
I'm sorry Dave, I'm afraid I can't do that.
If we do that, it will not be able to morph into large eoungh curvatures.
Alright, the original design it is.
#0
#1
#2
#3
#4
#5

**Figure 57. SimuLearn-empowered design agents.**

# CONCLUSIONS

This chapter will discuss the contributions of our work, and recap the content and vision discussed in previous chapters.

In this thesis, we take 4D printing as an example of material-driven design paradigms to propose, prototype, and provide proofs-of-concept for a data-driven simulation technique to respond to one of the challenges we face in such a design practice - rendering material responses in digital tools. Compared to currently available simulation methods that are oriented to engineering or animation purposes, our technique is adapted to design tasks and exhibits potential to afford efficient and economical 4D printing design workflows. Our design tool prototype also shows that CAD tools adopting physically-accurate and fast simulators can enable us to explore new design frontiers of our selected material system. Beyond 4D printing, we are also envisioning our technique to be applied to various material-driven design domains.

We as designers are familiar with using materials as a factor in design processes. There are powerful tools and machines that help us to shape matters into a variety of different shapes and functions. However, under the industrial mode of making, we often favor materials that have high fabricability or even engineer matters to have one such property, averting the active, transient, and temporal behaviors of materials - even more so as we advanced into a computer-aided design and modeling era. Active material behaviors become maintenance and fabrication issues while they can be leveraged to functionalize artifacts and instill ubiquitous interactions into everyday objects. We hope this thesis can serve as the initial step to reconcile materials with geometries in the CAD tools we wield and promote a GeoMatrical - concurrently geometry- material-driven - design practice.

Lastly, we want to clarify that this simulation technique is not meant to compete with other simulation methods nor to discriminate the significance of physical prototyping in design processes. In fact, this work is only made possible by combining engineering technologies with design knowledge and is not realizable without physical artifacts informing digital implementations. As an architect, I acknowledge the importance of combining design with technology and mediating between bits with atoms. To me, the role of architects is never mere designers of buildings, but the coordinators of disciplines and the tools we make are the very theater of intellectual convergence.

# Contribution

**Vision and Concept (Chapter 1 and 3)**

- **Simulation-empowered design workflows.** We envision that fast and accurate simulation tools will augment current inverse-design workflows and enable forward-design tools of 4D printing and morphing matter to take place.
- **Hierarchical composition of 4D printing.** We provide an intellectual framework of 4D printing that helps us select the level of abstraction, limit design space, and strategize the development of CAD and simulation engines.
- **Data-driven simulators and dedicated design tools.** We propose to adopt data-driven methods to compose efficient simulation engines and make bespoke design tools to streamline design processes.

**Technical Contributions (Chapter 3 - 4)**

- **SimuLearn and its development pipeline.** This thesis establishes a pipeline consisting of three steps - FEA modeling, data-processing, and model training - to cultivate machine learning-empowered, finite element analysis-based simulators.
- **Graph theories for 4D printing.** To our knowledge, we are the first to adapt graph theories to represent the compositional design of 4D printing.
- **Design tool and Evaluations.** We conceptualize a design tool architecture adapted to our ML-FEA simulator and investigate its implication in design tasks.

**Artifacts (Chapter 4)**

- **A dataset of FEA results.** This dataset consists of 600 randomly generated samples.
- **A trained SimuLearn engine.** A simulator is developed as a proof-of-concept for our technique.
- **A design tool prototype.** This CAD tool is adapted to designing 4D printed mesh-like structures made of thermoplastic bending beams. However, due to technical limitations, this implementation is using FEA rather than SimuLearn to predict transformations.
- **Design examples.** We leverage our design tool implementation to explore the design space enabled by our technique and workflow, including wearable customizations, modular structures, and geometries that are previously difficult to achieve.

# REFERENCES

An, B., Tao, Y., Gu, J., Cheng, T., Chen, X.'., Zhang, X., Zhao, W., Do, Y., Takahashi, S., Wu, H., Zhang, T., & Yao, L. (2018). Thermorph: Democratizing 4D Printing of Self-Folding Materials and Interfaces. CHI.

Apagom AG. (2019). Physics Forests [Computer Software]. Retrieved from http://apagom.com/physicsforests/

Battaglia, P.W., Pascanu, R., Lai, M., Rezende, D.J., & Kavukcuoglu, K. (2016). Interaction Networks for Learning about Objects, Relations and Physics. NIPS.

Belbute-Peres, F.D., Smith, K.A., Allen, K., Tenenbaum, J.B., & Kolter, J.Z. (2018). End-to-End Differentiable Physics for Learning and Control. NeurIPS.

Chen, X.'., Tao, Y., Wang, G., Kang, R., Grossman, T., Coros, S., & Hudson, S.E. (2018). Forte: User-Driven Generative Design. CHI.

Cheney, N., MacCurdy, R.B., Clune, J., & Lipson, H. (2013). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. SEVO.

Dassault Systems. (2019). Abaqus [Computer Software]. Retrieved from https://www.3ds.com/products-services/simulia/products/abaqus/

David Rutten., Robert McNeel & Associates. (2019). Grasshopper [Computer Software]. Retrieved from https://www.grasshopper3d.com/

Feinberg, A. W., Feigel, A., Shevkoplyas, S. S., Sheehy, S., Whitesides, G. M., & Parker, K. K. (2007). Muscular thin films for building actuators and powering devices. Science, 317(5843), 1366–1370.

Gladman, A. S., Matsumoto, E. A., Nuzzo, R. G., Mahadevan, L., & Lewis, J. A. (2016). Biomimetic 4D printing. Nature Materials, 15(4), 413–418.

Gu, J., Breen, D.E., Hu, J., Zhu, L., Tao, Y., Zande, T.V., Wang, G., Zhang, Y., & Yao, L. (2019). Geodesy: Self-rising 2.5D Tiles by Printing along 2D Geodesic Closed Path. *CHI*.

Howell, L. L., Magleby, S. P., & Olsen, B. M. (2013). Handbook of compliant mechanisms. Chichester, West Sussex, United Kingdom: John Wiley & Sons.

Kononenko, O., & Kononenko, I. (2018). Machine Learning and Finite Element Method for Physical Systems Modeling. CoRR, abs/1801.07337.

Ladický, L. ʼubor, Jeong, S., Solenthaler, B., Pollefeys, M., & Gross, M. (2015). Data-driven fluid simulations using regression forests. ACM Transactions on Graphics. https://doi.org/10.1145/2816795.2818129

Liang, L., Liu, M., Martin, C., & Sun, W. (2018). A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. Journal of the Royal Society, Interface / the Royal Society, 15(138). https://doi.org/10.1098/rsif.2017.0844

Cardoso Llach, D. (2015). Builders of the Vision. https://doi.org/10.4324/9781315798240

Kahn, L. (1971). Transcribed from the 2003 documentary 'My Architect: A Son's Journey by Nathaniel Kahn.

Martin-Guerrero, J. D., Ruperez-Moreno, M. J., Martinez-Martinez, F., Lorente-Garrido, D., Serrano-Lopez, A. J., Monserrat, C., … Martinez-Sober, M. (2016). Machine Learning for Modeling the Biomechanical Behavior of Human Soft Tissue. 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). https://doi.org/10.1109/icdmw.2016.0042

Ou, J., Yao, L., Tauber, D., Steimle, J., Niiyama, R., & Ishii, H. (2013). jamSheets. Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction - TEI ʼ14. https://doi.org/10.1145/2540930.2540971

Papazafeiropoulos, G., Muñiz-Calvente, M., & Martínez-Pañeda, E. (2017). Abaqus2Matlab: A suitable tool for finite element post-processing. *Advances in Engineering Software, 105*, 9-16.

Pytorch. (2019). Pytorch [Computer Software]. Retrieved from https://pytorch.org

Robert McNeel & Associates. (2019). Rhinoceros [Computer Software]. Retrieved from https://www.rhino3d.com/

Sanchez-Gonzalez, A., Heess, N., Springenberg, J.T., Merel, J., Riedmiller, M.A., Hadsell, R., & Battaglia, P.W. (2018). Graph networks as learnable physics engines for inference and control. ICML.

Tibbits, S. (2014). 4D Printing: Multi-Material Shape Change. Architectural Design. https://doi.org/10.1002/ad.1710

Wang, G., Cheng, T., Do, Y., Yang, H., Tao, Y., Gu, J., An, B., & Yao, L. (2018). Printed Paper Actuator: A Low-cost Reversible Actuation and Sensing Method for Shape Changing Interfaces. CHI.

Wang, G., Tao, Y., Capunaman, O.B., Yang, H., & Yao, L. (2019). A-line: 4D Printing Morphing Linear Composite Structures. *CHI*.

Wang, G., Yang, H., Yan, Z., Ulu, N.G., Tao, Y., Gu, J., Kara, L.B., & Yao, L. (2018). 4DMesh: 4D Printing Morphing Non-Developable Mesh Surfaces. *UIST*.

Wang, W., Yao, L., Zhang, T., Cheng, C., Levine, D., & Ishii, H. (2017). Transformative Appetite. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI 17. doi:10.1145/3025453.3026019

# LIST OF FIGURES