Understanding Metamaterial Mechanisms

Alexandra Ion[1], David Lindlbauer[2,3], Philipp Herholz[2], Marc Alexa[2], Patrick Baudisch[1]
[1] Hasso Plattner Institute, University of Potsdam, Germany. Email: {firstname.lastname}@hpi.de
[2] TU Berlin, Germany. Email: {firstname.lastname}@tu-berlin.de
[3] ETH Zurich, Switzerland. Email: david.lindlbauer@inf.ethz.ch

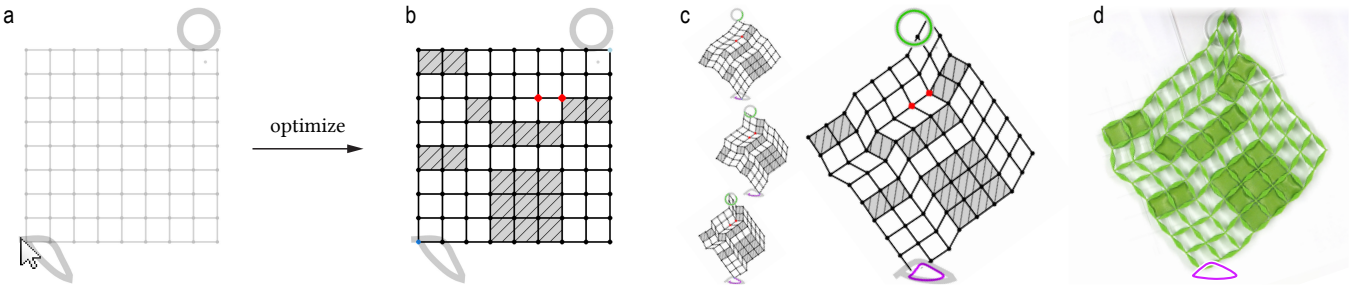

Figure 1: We define the underlying working principles of metamaterial mechanisms, which allows us to implement a computational design tool. (a) It takes user-drawn paths and (b) optimizes the cell configuration which implements the transformation. (c) In this example, we show the leg of a walker. (d) The fabricated result matches the optimized motion closely.

ABSTRACT

In this paper, we establish the underlying foundations of mechanisms that are composed of cell structures—known as metamaterial mechanisms. Such metamaterial mechanisms were previously shown to implement complete mechanisms in the cell structure of a 3D printed material, without the need for assembly. However, their design is highly challenging. A mechanism consists of many cells that are interconnected and impose constraints on each other. This leads to unobvious and non-linear behavior of the mechanism, which impedes user design. In this work, we investigate the underlying topological constraints of such cell structures and their influence on the resulting mechanism. Based on these findings, we contribute a computational design tool that automatically creates a metamaterial mechanism from user-defined motion paths. This tool is only feasible because our novel abstract representation of the global constraints highly reduces the search space of possible cell arrangements.

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300877

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; • **Hardware** → **Emerging technologies**.

KEYWORDS

Metamaterials, fabrication, computational design.

ACM Reference Format:
Alexandra Ion, David Lindlbauer, Philipp Herholz, Marc Alexa, Patrick Baudisch. 2019. Understanding Metamaterial Mechanisms. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3290605.3300877

1 INTRODUCTION

The recent rise of widely accessible fabrication machines, such as 3D printers or laser cutters, generated interest in non-experts to create and design their own devices. Their strive towards a future of personal- rather than mass-fabrication is supported by HCI researchers [4], who investigate techniques to directly interact with the machine [25, 28, 47], use real-world objects for content creation [45, 46] or embed mechanisms [49] and electronics [35]. These works were mainly concerned with creating the outside shape of 3D objects. Since 3D printing uniquely allows users to freely arrange matter in space, researchers started to generate internal structures that, e.g., optimize the strength-to-weight ratio of 3D objects [21], allow arbitrarily shaped objects to spin [3], or to float in pre-defined poses [30].

Pushing this idea further, researchers engineer microstructures that deform in a desired way. These structures are

usually arranged on a regular grid and together define the properties of the material they form [5]. This concept is known as metamaterials. Such metamaterial structures can be designed to change the materials' elasticity [37], to absorb energy [12, 40], or to change their shape [24].

Recently, we [18] pushed the concept of metamaterials further by going beyond materials and create complete mechanisms from cellular structures. Our mechanisms consist of two types of cells that are carefully arranged to move in concert to achieve the macroscopic mechanical movement. Along with this novel concept, we demonstrated relatable application examples including a door latch or a Jansen walker. However, the design of these objects was manual and difficult it still remains unclear what types of mechanisms can be implemented with such metamaterials.

In this paper, we investigate the underlying working principles of such metamaterial mechanisms. To do so, we analyze the interaction of the two types of cells, identify the underlying topological constraints of metamaterial mechanisms, and ultimately implement this domain knowledge into a computational design tool for non-expert users.

Understanding metamaterial mechanisms

In this work, we set out to understand the underlying mechanisms that inform the design of our metamaterial mechanisms [18], which implement a transformation of an input movement to an output movement. The door latch in Figure 2a-b, for example, transforms pushing the handle down (input) into a retraction of the bolt (output). To achieve a well-defined transformation, metamaterial mechanisms combine two types of cells on a regular grid. The individual cells (Figure 2c-d), are very simple—they are rigid or can shear.

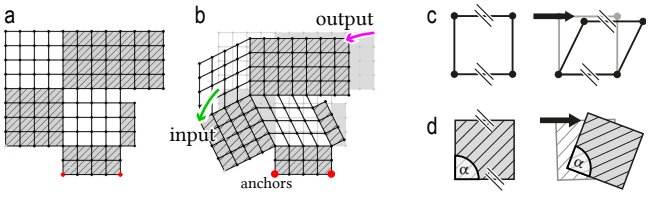

Figure 2: (a) This metamaterial door latch is defined by their microstructure, (b) which transforms the green input path (pushing the handle) into the pink output path (retracting the bolt). It consists of only (c) shearing and (d) rigid cells.

Since we already demonstrated relatable application examples in our previous paper [18], in the remainder of this paper we will discuss metamaterial mechanisms on a higher level of abstraction. We will leave out the device applying the metamaterial and instead focus on the core of the mechanism: the transformation of input movement into a desired output movement, both of which are user-defined.

Understanding cell constraints and how they interact. To achieve the movement that implements the desired mechanism, the cells need to be arranged on a grid to play together in a well-defined way. More formally, 'play together' means that each individual cell has constraints that it propagates to its neighbors. For example, opposing edges of shear cells remain parallel (parallelism constraint) and rigid cells additionally maintain their angle (angle constraint). Since the cells are connected in two dimensions, their constraints can interact. We illustrate such constraint interactions in Figure 3. This example shows how adding a single cell (marked in blue) prevents 7 other shear cells on the grid from shearing.

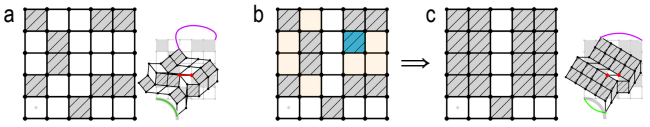

Figure 3: (a) In this example, (b) setting one cell rigid (c) prevents 7 cells from shearing, changing the output drastically.

Large search space. The example depicted in Figure 3 illustrates that interactions of constraints are unobvious and that by understanding them, we can reduce the search space drastically. We drill down on this example in Figure 4. Here, the naive approach of simply swapping cell types to find a configuration that reaches a user-defined path results in $2^7 = 128$ equivalent mechanisms. This is because any changes within the 7 cells marked in orange in Figure 3 have no effect on the resulting mechanism. While this example is only concerned with one specific scenario, the complete search space would be $2^{25} \approx 3 \cdot 10^7$. We generalize this in Section 4 and show how we reduce the search space by several orders of magnitude.

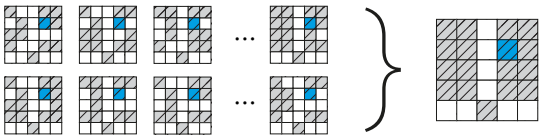

Figure 4: Here, 128 cell configurations result in the same mechanism, because the cell constraints interact.

The key insight of this work is to build an abstract representation of the cell constraints, which defines *only distinct mechanisms*. We encode constraints that edges impose on each other in a graph whose connected components define their degrees of freedom. We work directly in the reduced space of distinct mechanisms, rather than exploring the space of all possible cell configurations.

Computational design tool. Our constraint-based representation of the metamaterial mechanism and the resulting reduction of the search space makes a computational design

tool feasible. We present a computational design tool that optimizes a cell configuration for user-defined paths. Figure 1 shows how users define the size of the mechanism they are looking for and draw the input and output paths. Our heuristic optimization searches for a cell configuration that satisfies these boundary conditions.

Novel types of mechanisms. With our computational approach, we not only ease the creation process for users, but we also discover new types of mechanical transformations that were not known before. Metamaterial mechanisms were manually designed to demonstrate useful mechanisms, such as a door latch or pliers. However, we show in Figure 5a that the transformations they implemented were basic transformations, such as scaling. In this work, we demonstrate non-linear transformations, as illustrated in Figure 5b, such as self-intersections, oscillations and smoothing. We believe that the approach will foster more complex metamaterial mechanisms.

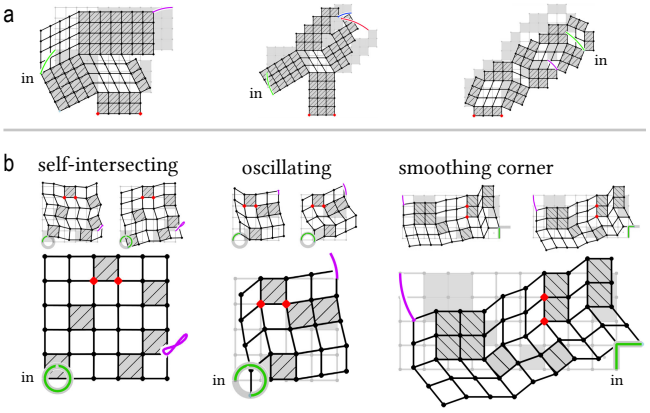

Figure 5: (a) The hand-designed mechanisms in [18] only realize simple transformations. (b) In this work, we discover more complex and even non-linear transformations.

Contributions & limitations

Our main contribution is an understanding of the underlying workings and constraints of metamaterial mechanisms. This enables a computational design tool that would otherwise not have been possible.

The naive approach of swapping cells on the grid to find a cell configuration that implements a user-defined path transformation is computationally infeasible due to an exponentially growing search space. We contribute an abstract representation of the constraints. Our constraint graph representation reduces this search space significantly.

We show that metamaterials can realize more complex and even non-linear mechanisms—a fact that was unknown before.

We focus on understanding the constraints on the most basic cells, i.e., the square shear and rigid cells. We do not explicitly implement rotated or pre-sheared cells, as we suggested in [18]. Since the topology is the same, we show that our constraint graph applied to those cells as well.

2 RELATED WORK

Fabrication of functional objects. Personal fabrication devices such as 3D printers or laser cutters allow users to fabricate personalized physical objects. Researches presented interactive systems that ease the process of designing static objects, e.g., pen holders or booklets, by interacting directly on the machine [25, 28, 47]. To match digital 3D models with existing physical objects, researchers proposed augmented reality environments to ease content creation [45, 46]. 3D printing has been shown to be an effective tool to making existing objects (e.g., pliers) more accessible by fabricating adapters [8], or for ad-hoc repairs on the go (Mobile Fabrication [34]).

Since 3D printers can arrange material freely, researchers explored how altering the material distribution on the inside of objects can alter their functionality. Examples include optimizing the strength-to-weight ratio of objects [9, 21], making objects stand or float in user-defined poses [30] or allowing arbitrary shapes to spin reliably [3].

Most 3D printers, however, are limited to creating objects from plastics. To benefit from superior mechanical properties of traditional construction materials, researchers provided software tools that allow user to insert material such as metal rods, sandpaper [7], fabrics [33], or even water bubbles [48] into printed objects. Combining different types of materials was also demonstrated to be an effective assembly approach. Using pre-stretched material and printing rigid struts on top allows for fabricating 2.5D shapes in a flat state that curve up after fabrication [16, 29]. To actively control folding of objects, researchers employed conductive layers on paper sheets that can be controlled electrically [44].

We want to push assembly-free mechanisms that can be printed from a single material [18] further and contribute a computational tool that assists users in creating them.

Fabrication of linkages and compliant mechanisms. The availability of fabrication machines not only fosters creating functional objects that are defined by their shape, such as tools or adapters, but also the implementation of mechanisms. One popular application are mechanical characters. Researchers created tools to help users integrate, e.g., walking or driving mechanisms into arbitrary 3D models [49].

To assist users in creating a new mechanism, researchers presented computational tools that optimize the position of gears [10] or links in a linkage (i.e., a mechanism that connect links with hinges) [41] to match a target locomotion path of a figure within a confined space [2].

Reducing the number of parts of mechanisms eases the fabrication process for the user, since the need for assembly disappears [11]. Such mechanisms are known as compliant mechanisms, i.e., deforming structures that allow for implementing a mechanism by rendering selected parts thin. While traditional mechanisms use very stiff (rigid) parts that are connected by hinges to transform motion or forces, compliant mechanisms consist of one part with very thin areas that allow for hinging behavior. Since the movement is performed by deformation there is virtually no friction, no need for lubrication, and thus for maintenance [17]. Since they consist of only one part, compliant mechanisms miniaturize well and are commonly used for high-precision mechanisms, such as micro-electromechanical systems (MEMS) [1] or teleoperation devices [13].

Very simple compliant mechanisms are holders, where you can snap in another object. The snap-fit, however, should be firm enough to hold the object, yet soft enough to push the object in—a ratio that needs to be optimized [42]. Recently, Megaro et al. [23] presented a system that creates complex compliant mechanisms (e.g., a hand with all joints) using fully-functional linkages as input. We, however, allow users to draw their desired mechanical transformation, without the need to define a traditional mechanism first.

We build on the foundations of these works and draw inspiration from their optimization procedures, although they are very different from ours. Since they optimize for continuous parameters, such as length of links, position of gears, etc., they make use of gradient-based methods. We, however, operate in a discrete, thus gradient-free space.

Mechanical metamaterials. Metamaterials are artificial structures with mechanical properties that are defined by their usually repetitive cell patterns, rather than the material they are made of [5, 32]. Such structures allow for unique material properties, such as materials that shrink in two dimensions upon uniaxial compression [20, 36], that damp impacts [12, 40], or localize elasticity [6, 22, 37].

On a macroscopic level, these structures are designed to implement specific material properties. On a microscopic level—if we think of the unit cells—they can be seen as many small compliant mechanisms that are interconnected on a grid. While regular tilings of such cells are well understood [14, 15, 27, 38, 39], researchers only recently started to vary the parameters across a metamaterial [24, 26], yet maintain the same topology of cells.

However, non-uniform tessellations, i.e., when cell topologies are changed freely across the material, are not well understood. To our knowledge, metamaterial mechanisms [18] are the only metamaterials that fall into this category. Therefore, we base our work on these metamaterials to set up a first understanding of such non-uniform tessellations.

3 ANALYSIS OF CELL INTERACTIONS

The mechanisms we consider can exhibit intricate behavior even though they are built from very basic building blocks: shearing and rigid cells. Figure 3 already demonstrated how drastically a metamaterial can change after performing a small local change, e.g., swapping a shear cell for a rigid cell. These properties are non-obvious but crucial to understand.

In order to understand the movements of a mechanism, we need to simulate its physical behavior numerically. For larger grids the optimization procedure can be time consuming. This can hinder interactive exploration of the space of mechanisms and is especially problematic when sampling and simulating a lot of mechanisms to find one exhibiting some pre-defined behavior.

We describe how we model the constraints of a mechanism, which reduces the number of variables in the physical optimization from all grid points to a few edge vectors. This enables a significantly more efficient implementation and gives insights into the degrees of freedom of a mechanism.

Understanding the constraints

Since our metamaterial consists of shearing and rigid cells, we observe two types of constraints in our cells: (1) parallelism constraints, i.e., opposing edges always remain parallel and (2) angle constraints, i.e., angles of rigid cells remain unchanged. Furthermore, all edges maintain their lengths. This means most edges cannot move independently, e.g., the same vector can represent edges that have to remain parallel. Edges that have to maintain a certain angle can also be represented by a reference edge that needs to be rotated in order to get the second one. To this end we build a constraint graph in which each node represents a cell edge and an arc between nodes the fact that one edge can be constructed by rotating the other (rotations also include the identity transformation).

Figure 6 illustrates the graph representation for each cell type individually. For shear cells, opposing cell edges always remain parallel. The constraint graph consequently only contains arcs between opposing cell edges. Figure 6b illustrates that rigid cells are represented by a complete graph. This means that transforming one cell edge defines the transformation of all other edges. In other words, if we know how one edge is, e.g., rotated, we know the transformation of the entire cell, because opposing edges remain parallel and adjacent edges maintain their angle.

Determining the degrees of freedom

We think of the degrees of freedom (DoF) as a set of edge vectors that can be transformed independently. This property is also illustrated in Figure 6, where the constraint graph of the shear cell in (a) consists of 2 connected components, which indicates that the cell has two independently moving

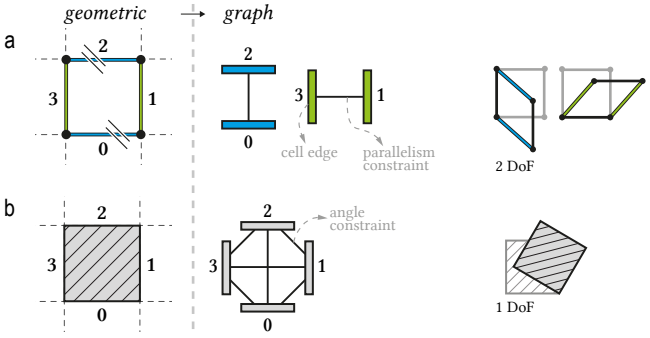

Figure 6: We model the constraints as graphs. (a) A shear cell is represented as 2 subgraphs that model the 2 independently moving adjacent edges and the parallelism constraint of opposing edges. (b) A rigid cell is a complete graph, showing that one edge defined the entire cell.

parts, i.e., the green and blue edges. The rigid cell in (b) moves as a whole, thus the graph consists of one component. In general, the DoF of our metamaterial are defined by the number of connected components in the constraint graph.

Building the constraint graph

We build the entire constraint graph by connecting the constraints of single cells shown in Figure 6 to their neighbor cells, based on coinciding edges. For example, Figure 7 shows how to proceed for a metamaterial with 4 cells. We start at the lower left cell and add its vertical edges to the constraint graph. The cell to the right shares the middle vertical edge, thus we link its other vertical edge to the graph, because they need to remain parallel. The two shear cells on the right are processed analogously. The rigid cell, which cannot change its angle, effectively couples edges of the top right and lower left cell. Due to the parallelism constraints, entire rows and columns are linked.

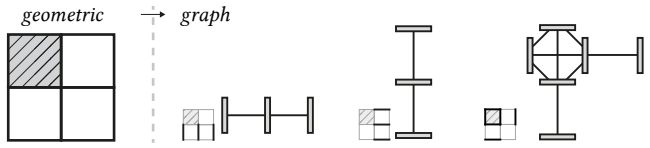

Figure 7: We build the constraint graph by connecting the single cell constraints to their neighbor cells based on coinciding edges.

The notion of DoF generalizes to entire mechanisms. The final graph in our example consists of 3 connected components, which means that the configuration has 3 DoF. Knowing one edge vector in each component uniquely determines all other edge vectors and thereby the whole mechanism. Therefore, we can formulate an optimization problem only involving 3 instead of 12 edges, as detailed in Section 6.

In a cell grid that consists only of shear cells, the edges in each row and column represent one connected component. The maximum number of DoF on an empty $n \times m$ grid is therefore $n+m$. Introducing a rigid cell joins the components of a row and a column into one, reducing the DoF by one.

The influence of anchors

So far, we only considered relative transformations of edges, i.e., edges are transformed with respect to each other. However, since the input of a mechanism is absolute, we need to fixate (i.e., *anchor*) the cell configuration in order to calculate the vertex positions in absolute space. Anchoring an edge in a cell configuration reduces its DoF by one. The same reasoning as for transforming edges, as discussed above, applies; since one edge of a connected component is defined (here, fixed to the ground), all edges in the component are defined.

4 REDUCING THE SEARCH SPACE

Since each cell on a $n \times m$ grid can have 2 states, rigid or shearing, we have 2^{nm} different possible configurations. However, not all configurations will generate a unique mechanism since shearing cells can become rigid because of the constraints. Consider the mechanism in Figure 8. The constraint graph reveals that the green and blue cells cannot shear. This is the case when *all edges* of a cell are contained in the *same connected component*. Since the rotation of both potentially independent edges of the shear cell are defined by the same connected component, their angle is constraint to the original 90°and can thus not shear.

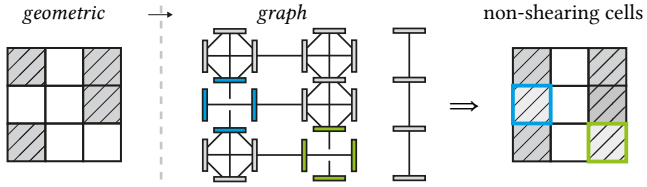

Figure 8: It might be non-obvious how many DoF this metamaterial has. Our constraint graph reveals that it contains 2 cells that cannot shear, i.e., the blue and the green one, leaving only 2 connected components, thus 2 DoF.

Since the blue and green cells cannot shear, the two cell configurations in Figure 8 are equivalent. We only want to consider *unique mechanisms* in our optimization. But how many of these unique mechanisms exist? To answer this question, we enumerate all possible connected component configurations. It is indeed possible to give an explicit formula for the number of all unique mechanisms on a $n \times m$ grid. We present the derivation of this function in Appendix A. Here, we show the empirical verification of the formula for all configurations on grids with $n < 5$. In Figure 9, we show the number of all 2^{nn} mechanisms (gray) along with

the number of unique mechanisms (blue). While this number still grows rapidly with increasing grid size, it reduces the space of configurations on a grid from growing quadratically with respect to n in the log-plot to linearly. This enables us to consider much larger grids when searching for mechanisms with certain properties.

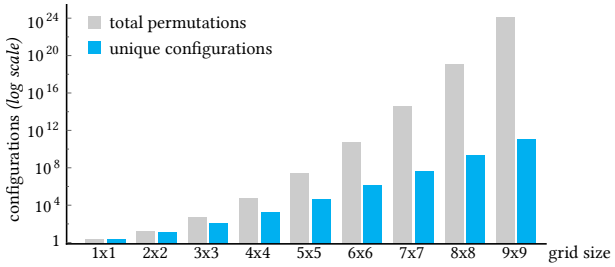

Figure 9: The number of all configurations (log scale) on a square grid compared to the number of all unique mechanisms.

5 SIMULATING THE MOTION

To simulate the deformation of the object when a vertex of a cell moves, we use a simple elastic material model. We assume rigid edges and that deformation occurs when two edges incident to a corner are rotated relative to each other.

Through the constraint graph all edges are defined as soon as one representative edge per connected component is known. To define the remaining edges, we traverse the constraint graph, e.g., using breadth-first search, starting at the representative edge. When moving across an arc the appropriate rotation is applied. When all edges are known we start to reconstruct cell vertices. To this end we traverse the grid itself starting at an anchored vertex. Each vertex is effectively reconstructed by summing all edge vectors along a path connecting the vertex to the anchor.

Direct numerical simulation would try to find a set of grid vertices minimizing this deformation energy subject to a set of non-linear constraints, edges have to maintain their length, opposing edges in a cell have to stay parallel and angles in rigid cells stay at 90°. Additionally, certain vertices are fixed, either in their original position or by an external force driving the mechanism.

Analyzing the constraint graph allows us to formulate a constrained optimization problem that has the same solution but is much more efficient than the direct approach.

The shape of the whole mechanism is determined by fixing the DoF, which requires two numbers per connected component. The current state of the deformation can therefore be modeled by a state vector $x \in \mathbb{R}^{2\,N_d}$ where N_d is the number of DoF. Each state vector uniquely determines all edge vectors and is associated with the deformation energy

$$D(x) = \sum_{i=0}^{N-1} \left(\alpha_i(x) - \frac{\pi}{2}\right)^2,$$

where $\alpha_i(x)$ stand for an interior angle of the i-th cell and N for the number of cells. This energy models the fact that each cell will, depending on the material used, resist shearing, even for non-rigid cells. Note that we only need to take one angle per cell into account because all angles in a cell produce the same energy. To simulate the deformation, we find a minimizer of D with respect to the following constraints:

C1. The edges are rigid and therefore have to be fixed in length. Constraining all DoF to unit length will ensure this property for all edges in the mechanism.
C2. Cells cannot invert, i.e., change their orientation. We thus assure that cell areas always remain positive.
C3. Anchors are fixed in position.
C4. The corner that is being dragged is fixed in position.

Note that we do not have to enforce parallel edges or that rigid cells maintain their interior angles. These constraints are built into the reduced representation and each state vector induces a valid cell layout. This constitutes another advantage over the naive approach where these constraints have to computed explicitly. Unfortunately, the energy and the constraints are non-linear and non-convex. However, the constraints are only quadratic and can be analytically differentiated. The energy D can also be analytically differentiated albeit yielding more complex terms due to the calculation of angles from edge vectors. We solve the problem of minimizing D subject to C1-C4 by employing the non-linear interior point solver IPOPT [43], which gives us a configuration for a specific handle position. To simulate the full deformation, we sample the desired handle path and find the correct configuration by solving the optimization problem starting from the previous configuration as an initial guess.

Invalid input. The range of the handle corner is limited by the structure of the mechanism, which is fixed by anchors. If a handle position outside this range is prescribed, the optimization problem has no solution because C4 cannot be satisfied. To yield an approximate result even for these situations we try to find a handle position, which is close to the desired one but in the range of the handle. To this end we solve another optimization problem minimizing the distance of the updated handle position $\widetilde{P_h}$ to the prescribed one P_h:

$$D'(x) = \left\| \widetilde{P_h} - P_h \right\|^2,$$

subject to the constraints C1-C3. The resulting configuration is valid and places the handle close to the desired position. Since this solution represents only one valid configuration but does not minimize D in general, we optimize D with respect to C1-C4 using the new handle position.

Evaluation

As a preliminary evaluation of the simulation, we manually created 3 mechanisms with different input paths in our software and manufactured them from laser cut acrylic and 3D printed hinges (Ninjaflex filament). We tracked the motion of the physical mechanisms using OptiTrack with up to 7 markers on each mechanism and compared it with motion paths from our simulation. We adjusted the OptiTrack data to account for scaling. The mean error of our physical cell configurations was 8.1% (SD = 8.7%), corresponding to 4.8 mm on our 60 mm cells. Figure 10 shows at one example that the recorded and simulated data matched closely.

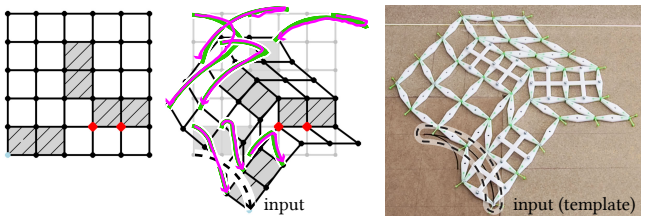

Figure 10: We tracked the physical motion using OptiTrack (blue) and compared it to our simulation (green).

6 OPTIMIZATION OF CELL PATTERNS

We aim to alleviate the problem of finding a suitable cell configuration given a set of desired input and output paths. In the following, we detail our algorithm that exploits aforementioned properties of metamaterials mechanisms such as the relation between cell transformation and anchors, and the constraint graph. Note that while most of our examples illustrate quadratic cell configurations, the approach is equally applicable for arbitrarily shaped objects. The main requirement for our approach is a simulation that correctly reproduces the deformation of a mechanism given an input path, like the one described in the previous section.

Overview

Our algorithm takes a set of user-defined input and output paths as well as a rough shape of the desired device as input. It then aims at finding a close-to-optimal fit between the motion of the mechanism and the user-defined paths. Generally, input paths are actuated (e.g., by users) and the mechanism transforms this motion to the output path. In context of the optimization, however, we do not need to differentiate between input and output paths but use them as a list of paths which a mechanism should be able to reproduce. As a first step in our algorithm, we automatically determine the positions of the anchors for the cell configuration based on the scale and direction of the paths. We then generate a mechanism that produces the desired motion. An overview of the algorithm is illustrated in Figure 11.

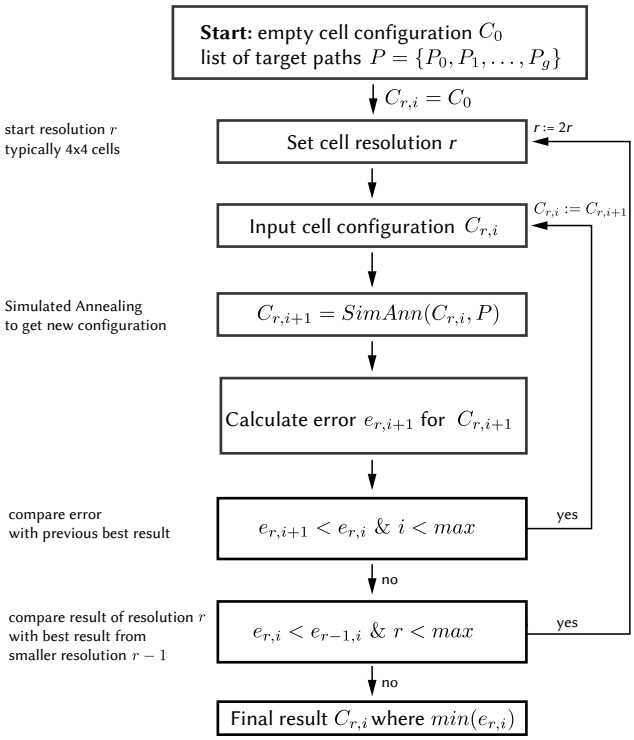

Figure 11: Overview of the process of automatically finding cell patterns given user-specified paths.

Since the behavior of a mechanism is non-linear, we create cell configurations using stochastic optimization, specifically Simulated Annealing [19, 31]. Instead of modifying the cell configuration directly, we modify the constraint graph, i.e., split and merge connected components until the algorithm converges. To speed up computation, we use a hierarchical approach where we first find an optimum configuration for scaled-down versions of the mechanism and use this configuration as seed for larger versions. The output of the algorithm is a cell configuration that produces the desired paths.

Input and representation

Users specify the shape of the cell configuration $C_0 \in 0, 1^{x,y}$, with x and y being the dimensions of the mechanism. This is a configuration of cells with undefined behavior. In the later optimization, the type of each cell is specified (e.g., they become shear or rigid cells), which ultimately governs the motion of the mechanism. Each mechanism is defined by its cells, its vertices V and edges E. The position of the anchors $A \subseteq V$ governs the motion of a mechanism and fixes the mechanism's absolute position in space. Anchors are automatically determined by our algorithm.

Besides C_0, users specify a set of desired paths P. Each path is a list of points, stored as a matrix $P_i \in \mathbb{R}^{2\times n}$. The first

point $p_{i,0}$ of a path P_i is a vertex of the cell configuration, i.e., $p_{i,0} \in V$. Note that while we use one input and one output path for clarity of exposure, the optimization generalizes to more than one input and output path.

Setting anchors

Our algorithm automatically determines candidate positions for anchors based on user-defined paths. Anchors have a big influence on the scaling and rotation of an output path. There are three main considerations that govern the positioning of the anchors (i.e., finding which edge to anchor). First, the ratio between the length of paths should be similar to the ratio between the distance between individual paths and the anchors. As shown in Figure 12a, if an input path is half the length of an output path , then the ratio for the distance between input path and anchors, and between output path and anchors should be similar. This accounts for scaling between the paths and can be calculated as

$$\min_{l} \sum_{i=0}^{m} \sum_{j=i+1}^{m} \frac{|P_i|}{|P_j|} - \frac{|e_l - P_i|}{|e_l - P_j|}$$

where m denotes the number of user-defined paths. Secondly, anchors are essentially rotation points between paths. We therefore choose the location of anchors to reflect this rotation, as illustrated by Figure 12b. Thirdly, the maximum travel of a vertex is determined by its distance to an anchor, i.e., larger distance yields larger possible travel. To allow for maximum travel, we position the anchors at maximum distance to all input or output vertices, formulated as

$$\max_{l} \sum_{i=0}^{s} |e_l - P_i| .$$

Since this operation can be performed on a scaled-down version of the configuration, we can exhaustively search the space and choose the best anchor positions. It is possible that no valid positions are found, which allows us to estimate a-priori if the user-defined mechanism is achievable. For example, if the length of a path exceeds the cell grid's diagonal, there is no solution. Users can use a larger grid or change the paths. For all other cases we optimize for the best approximation of the desired mechanism.

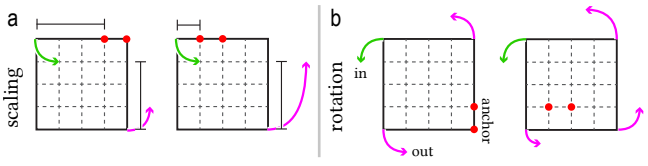

Figure 12: The anchor placement influences (a) the scaling of the output path and (b) the global rotation.

Generation of cell configurations

Given a list of user-defined paths, any cell configuration C will transform them in a specific way. For a path P_i, this transformation is denoted as $C(P_i)$. We aim to find a cell configuration C_j with minimal difference between P_i and $C(P_i)$ while using as few DoF as possible to increase mechanical stability. We thus aim to find a solution given the objective

$$\min_{j} \sum_{i=0}^{m} \omega_i \left| P_i - C_j(P_i) \right| .$$

$\omega_i \in [0,1]$, $\sum_{i=0}^{m} \omega_i := 1$ are user-defined weights for the individual paths. We typically use equal weights for all paths (i.e., 0.5 for 1 input and 1 output path).

A trivial approach to the problem would be to randomly sample cell configurations (i.e., switching cell types randomly) and choose the best fit between the user-defined paths and the paths the mechanism produces. This, however, is not feasible, as discussed in Section 4, given the large number of possible cell configurations that yield similar movements or are rigid.

In our approach, instead of modifying the cell configuration directly, we manipulate the underlying constraint graph with respect to the DoF of a configuration. We aim for a configuration with as little DoF as possible that still can reproduce a path well. A too low number of DoF would not allow for complex motion (e.g., with changes in direction). A too high number of DoF would lead to mechanically unstable structures. Therefore, we constrain a configuration to be within DoF_{min} and DoF_{max}, which are typically chosen to be 2 and 5, respectively.

For each step in our optimization, we compute the current DoF for a configuration C_j, which govern the next step, i.e., if the next configuration shall contain more or less DoF. If the DoF should be increased, we split a chosen connected component. Conversely, we merge two connected components to decrease the DoF. If the DoF are within the limits, an operation is chosen randomly. The connected components that are affected by the operation are chosen randomly.

We use Simulated Annealing for the optimization to avoid converging to a local minimum. We calculate the current temperature T_j for each step as

$$T_j = (1 + e^{\frac{e_j}{T_0 \alpha^j}})^{-1},$$

where e_j is the error of the current iteration j. It is calculated as the sum of differences between the user-defined paths, i.e., $e_j = \sum_{i=0}^{m} \omega_i \left| P_i - C_j(P_i) \right|$. The error is normalized with respect to the length of the paths. T_0 is the starting temperature, which we typically choose as one third of the maximum number of iterations, as described below. α controls the overall falloff, typically $0.85 < \alpha < 0.99$ (in our

case $\alpha = 0.95$). To avoid that the algorithm converges in a local minimum, we restart the Simulated Annealing process several times. After each run, we compare the best results of the current and previous run. Convergence is reached if the result did not improve compared to the previous run.

Hierarchical generation. We chose a hierarchical approach for the optimization, to avoid users having to estimate the cell resolution of their configuration. We start the Simulated Annealing process for an initial configuration C_0 with a small resolution r, e.g., 4×4 cells. Once the previously described Simulated Annealing procedure converged, we double the resolution and restart the Simulated Annealing with the larger resolution $r + 1$. After convergence, we compare the error of the cell configuration C_r and C_{r+1}. In case the result did not improve, we assume C_r to be the best solution. If the result did improve, we increase the resolution again and restart the process. A typical error function for the whole process is shown in Figure 13. Since the process typically converged after 100 to 150 iterations, we set the number of maximum iterations to 200. Similarly, the algorithm typically converges after increasing the resolution 2 to 3 times. We therefore set the maximum resolution to 6.

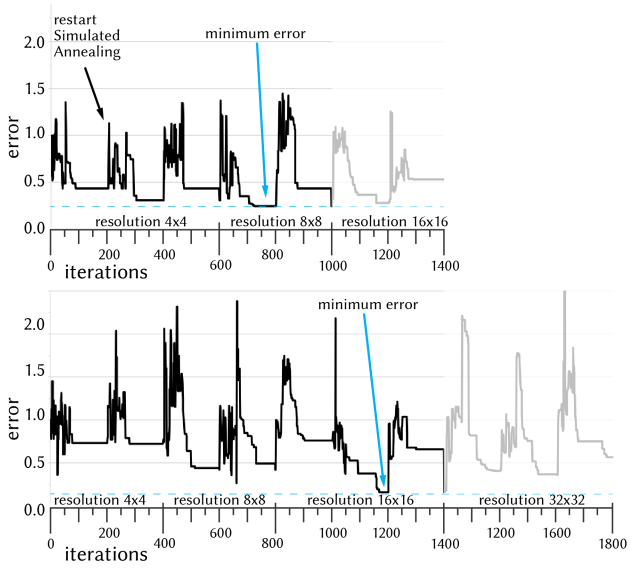

Figure 13: Typical error function for two examples. For top, the process converged after a total of 1400 iterations, with the minimum error for a resolution of 8×8 cells. Results with higher resolution (16×16) were discarded due to increased error. The bottom example converged with a resolution of 16×16 cells. For each resolution, Simulated Annealing is restarted multiple times.

Evaluation

To evaluate our optimization, we generated 10 differently-shaped cell configurations with 3 or 4 DoF as ground truth examples. We manually set the input and output vertex, an input path and the anchors. We simulated the mechanism to obtain the output path. We stored the input and output paths and the empty low-resolution cell configuration for the correct aspect ratio, which were the input to our optimization. Two examples with target and solution are shown in Figure 14. Although the resulting cell configurations are different in terms of placement of rigid cells, they could reproduce the target movement with a mean error of 2% (SD = 3%) for input paths and 7% (SD = 4%) for output paths. We computed the error as the sum of point-wise distances between the respective motions paths. Note that the cell configuration, the anchors and the target resolution were not known to the optimization but were generated.

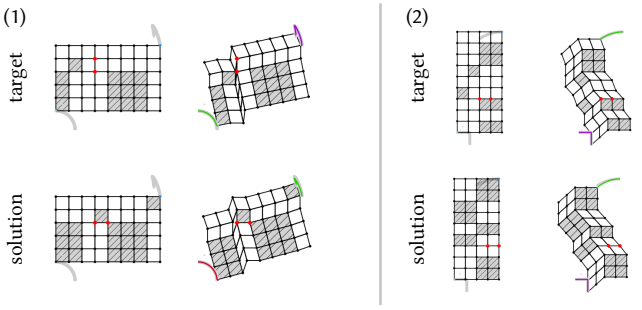

Figure 14: Two of ten ground truth examples (top) we used to test the optimization. The results in the bottom were generated without inputting the cell configuration or position of the anchors into the optimization procedure.

Implementation

While we already discussed the optimization and simulation in previous sections, we want to briefly mention the frameworks we used. We packed the metamaterial mechanism optimization, as described above, into a simple editor. The editor allows users to draw the shape of their desired mechanism and to set input and output paths, which the software will optimize for. Users can use pre-defined paths for precise motion constraints or simply draw rough paths. Time needed to generate a cell configuration depends on whether the solution requires a high-resolution configuration. If the solution is found in a low-resolution grid, the algorithm takes approximately 1 minute to converge on a commodity notebook (MacBook Pro 2015 with Bootcamp). For higher resolutions (e.g., 32×32 cells), finding the best solution takes up to 10 minutes. We are confident that this time can be decreased by parallelizing parts of the algorithm, e.g., running multiple threads of Simulated Annealing at once.

The editor is implemented in C# and uses the .NET framework 4.5. We use the Windows Presentation Foundation (WPF) as our GUI toolkit. The optimization procedure is also implemented in C#, which calls a wrapper to our C++ simulation tool. The simulation is written in C++, because it uses IPOPT [43], as we discussed in Section 5. We will provide the source code prior to the conference.

Limitations of the design tool. The editing capabilities of our editor are currently limited to 2D. Furthermore, we implemented only square cells so far. Triangular, rotated, or presheared cells, as we suggested in [18] are not integrated in the current version of the editor. However, this would be a simple extension. We also currently don't offer mesh export options, as we focused on the optimization. A conceivable option would be to implement the tool as a plugin to existing CAD tools, e.g., Autodesk Fusion 360, as they offer elaborate modeling options for the parts that embed the metamaterial.

7 EXAMPLES

The main contribution of this work is the abstract representation of metamaterial mechanisms, which ultimately allows an automatic generation of such metamaterials. While the device that embeds the metamaterial is not the focus of this work, we want to give some brief examples of how our generated metamaterials might be embedded in a device-context. The green structures in following examples are printed using rubber-like filament (Ninjaflex) on our consumer-grade 3D printer (Ultimaker 2+). More transformations are listed as examples in Appendix B.

Kinetic sculptures. One example, where metamaterial mechanisms might be embedded into, are kinetic sculptures, toys, or walking automata [41]. As shown in Figure 15a, our design tool enables users to create custom walk-cycles from metamaterials.

Custom mechanisms. Users might also want to create grippers with a custom motion path. In Figure 15b, we illustrate embedding metamaterial mechanisms with different motion paths as grippers for collecting items, e.g., for picking-challenge robots. Other applications for such custom paths are e.g., fans with a path cycle, or deflectors for sprinkler which can be customized to sprinkle a specific area optimally.

Clock. Another simple example is an alarm clock, as depicted in Figure 15c . The metamaterial transforms a rotary input into an oscillating motion to strike the bells.

8 DISCUSSION & OUTLOOK

Our work aims at going beyond purely exploratory work but provide an understanding of metamaterial mechanisms. Our work opens way for researchers to explore, build and use such mechanisms. Moving away from a representation

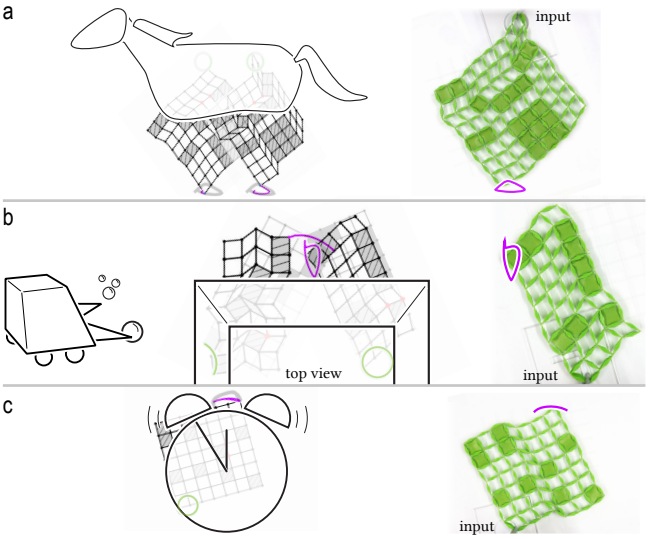

Figure 15: (a) Metamaterial mechanisms can be embedded into kinetic sculptures, or (b) implement a gripper with custom motions for, e.g., robots. (c) Another simple example is embedding an oscillating metamaterial into an alarm clock.

that is based on cells to a higher level—the constraint graph—allowed us to significantly reduce the space of cell configurations and making computational approaches feasible. Note that while the search space is highly reduced, it is still not feasible to fully enumerate the space. As discussed above, a cell configuration of 9×9 cells yields 10^{12} unique transformations (but 10^{24} cell configurations). We acknowledge that immediate commercial applications are not entirely clear at this point, since metamaterial mechanisms are still a very young research field. We believe that an accessible design tool allows not only users to design custom mechanisms, but that it is a valuable step towards exploring and further understanding the potential of cell-based materials to foster research in the long term.

Limitations and considerations

The constraint graph and optimization allow us to generate insights into the inner workings of metamaterial mechanisms and create a computational tool that enables automatic generation. There are, however, limitations to our approach.

Transformation symmetry. For a given cell configuration, moving an input vertex along the input path yields a transformation of the output vertex, i.e., an output path. If, however, the output vertex is moved along the same output path, the transformation of the input vertex is not equal to the input path. This behavior has to be taken into account when designing mechanisms that should be actuated by moving the input and the output vertex. It can, however, also be exploited to create more interesting and complex mechanisms.

Transformation complexity. In our experiments, we observed that transformations between input and output paths are limited in their complexity. Specifically, we saw that the number of inflection points between the two paths is roughly the same. An input path with one inflection points, for example, usually yields output paths with zero to two inflection points, but not of arbitrary numbers.

Extending to 3D mechanisms. Our current constraint graph models cells as constraint in length and fully defined by one angle. This does not hold true for 3D cell configurations. Therefore, there is no trivial extension of our approach into the 3rd dimension. Similar to the process of this paper, to optimize for 3D mechanisms we need to acquired an understanding of how interior angles in the volume change when a cell is subject to shearing in multiple planes. Our constraint graph must be extended to model how the new angle and parallelism constraints of an individual cell are propagated to their neighbors. Furthermore, the deformation energy D (Section 5) will be adapted to minimize these constraints. Such a basic constraint analysis is necessary for using our software for other cell structures. However, we expect our simulated annealing approach to work as presented. We already started investigating this interesting future work and replaced the 2D angle constraint with the angles of a basis in each cell as constraints, shown in our preview in Figure 16.

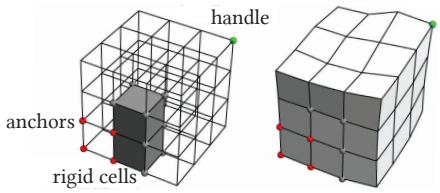

Figure 16: Preview of a 3D deformed mechanism with 2 rigid cells.

Practical extensions

There are several practical extensions to our work that concern the editor, including the simulation and generation of mechanisms.

Adapt to material properties. We used an idealized simulation that does not incorporate engineering factors such as material properties or friction. We did so to focus on the interaction between geometric constraints within a mechanism, which could be missed when taking mechanical factors such as transmission loss into account. While the mechanical properties can be optimized (e.g., through high-resolution 3D printing), the underlying constraints are inherent to the geometry. Material properties could be used to expand the repertoire of transformations. Including other types of simulation such as finite element analysis would be one beneficial extension of the editor. This would also allow us to adaptively change the stiffness, thus actuation force, of a mechanism. This can be achieved by merging cells.

Extending to different cell types. Our constraint graph, and consequently our optimization algorithm, hold true for other cells that we previously suggested in [18], such as rotated and pre-sheared cells. Our editor only needs to be extended slightly to edit such cells.

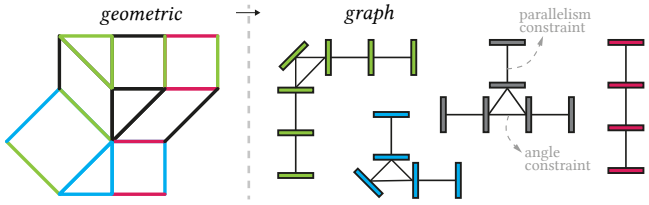

Figure 17: Our constraint graph generalizes to the cells we introduced in [18].

Considering temporal changes. We were concerned with the spatial transformation of metamaterial mechanisms. An extension of the optimization would be to also include a temporal component. Considering the speed of transformation would allow features such as keyframing and easing of motion. We plan to investigate this interesting aspect by adapting our error function and add keyframing to the editor.

9 CONCLUSIONS

We analyzed metamaterial mechanisms with respect to their topological constraints. Although the basic cells in this work (shear and rigid cells) are simple, connecting them creates complex interactions. We investigated these interactions, which we modeled as a constraint graph. This representation allows us to explore metamaterials on a more abstract level and avoid having to rely on the raw cell structure which exhibits a prohibitively large search space. Consequently, we implemented our knowledge as a computational design tool for the automatic generation of such mechanisms.

On a higher level, we think of our work as the first step towards what we like to call *heterogenous mechanical metamaterials*. We define them as metamaterials that consist of *different types* of cells. Most often metamaterials consist of cells that are topologically equivalent; all cells have the same function, but they can vary in parameter.

Metamaterial mechanisms is one example of a material that consists of *topologically different cells*. They exhibit interesting behavior yet the interactions between cells are hard to understand. In the future, we will go step-wise towards investigating more of these heterogenous metamaterials by creating generic tools that allow researchers to investigate combinations of different types of cells.

REFERENCES

[1] Jeffrey K. Anderson, Larry L. Howell, Jonathan W. Wittwer, and Timothy W. McLain. 2006. Piezoresistive sensing of bistable micro mechanism state. *Journal of micromechanics and microengineering* 16, 5 (2006), 943–950. https://doi.org/10.1088/0960-1317/16/5/010

[2] Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: interactive linkage editing using symbolic kinematics. *ACM Transactions on Graphics* 34, 4 (2015), 99:1–99:8. https://doi.org/10.1145/2766985

[3] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: Optimizing Moment of Inertia for Spinnable Objects. *ACM Trans. Graph.* 33, 4 (2014), 96:1–96:10. https://doi.org/10.1145/2601097.2601157

[4] Patrick Baudisch and Stefanie Mueller. 2017. Personal Fabrication. *Foundations and Trends in Human-Computer Interaction* 10, 3-4 (2017), 165–293. https://doi.org/10.1561/1100000055

[5] Katia Bertoldi, Vincenzo Vitelli, Johan Christensen, and Martin van Hecke. 2017. Flexible mechanical metamaterials. *Nature Reviews Materials* 2, 11 (oct 2017), 17066. https://doi.org/10.1038/natrevmats.2017.66

[6] Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Transactions on Graphics* 29, 4 (2010), 1. https://doi.org/10.1145/1778765.1778800 arXiv:1211.6396

[7] Xiang 'Anthony' Chen, Stelian Coros, and Scott E. Hudson. 2018. Medley: A Library of Embeddables to Explore Rich Material Properties for 3D Printed Objects. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM Press, New York, New York, USA. https://doi.org/10.1145/3173574.3173736

[8] Xiang 'Anthony' Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E. Hudson. 2016. Reprise: A Design Tool for Specifying, Generating, and Customizing 3D Printable Adaptations on Everyday Objects. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM Press, New York, New York, USA. https://doi.org/10.1145/3173574.3173736

[9] Xiang Anthony Chen, Y. Tao, G. Wang, R. Kang, Tovi Grossman, Stelian Coros, and Scott E. Hudson. 2018. Forte: User-driven generative design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vol. 2018-April. https://doi.org/10.1145/3173574.3174070

[10] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Transactions on Graphics* 32, 4 (2013), 83:1–83:12. https://doi.org/10.1145/2461912.2461953

[11] Juan Sebastian Cuellar, Gerwin Smit, Dick Plettenburg, and Amir A. Zadpoor. 2018. Additive manufacturing of non-assembly mechanisms. *Additive Manufacturing* 21 (2018), 150–158. https://doi.org/10.1016/j.addma.2018.02.004

[12] Tobias Frenzel, Claudio Findeisen, Muamer Kadic, Peter Gumbsch, and Martin Wegener. 2016. Tailored Buckling Microlattices as Reusable Light-Weight Shock Absorbers. *Advanced Materials* 28, 28 (jul 2016), 5865–5870. https://doi.org/10.1002/adma.201600610

[13] Joshua B. Gafford, Samuel B. Kesner, Alperen Degirmenci, Robert J. Wood, Robert D. Howe, and Conor J. Walsh. 2014. A monolithic approach to fabricating low-cost, millimeter-scale multi-axis force sensors for minimally-invasive surgery. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 1419–1425. https://doi.org/10.1109/ICRA.2014.6907038

[14] Aylin Gazi Gezgin and Koray Korkmaz. 2017. A New Approach to the Generation of Retractable Plate Structures Based on One-Uniform Tessellations. *Journal of Mechanisms and Robotics* 9, 4 (may 2017), 041015. https://doi.org/10.1115/1.4036570

[15] S. D. Guest and J. W. Hutchinson. 2003. On the determinacy of repetitive structures. *Journal of the Mechanics and Physics of Solids* 51 (2003), 383–391. https://doi.org/10.1016/S0022-5096(02)00107-2

[16] Ruslan Guseinov, Eder Miguel, and Bernd Bickel. 2017. CurveUps. *ACM Transactions on Graphics* 36, 4 (jul 2017). https://doi.org/10.1145/3072959.3073709

[17] Larry L. Howell, Spencer P. Magleby, and Brian M. Olsen. 2013. *Handbook of Compliant Mechanisms*. John Wiley and Sons.

[18] Alexandra Ion, Johannes Frohnhofen, Ludwig Wall, Robert Kovacs, Mirela Alistar, Jack Lindsay, Pedro Lopes, Hsiang-Ting Chen, and Patrick Baudisch. 2016. Metamaterial Mechanisms. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM Press, New York, New York, USA, 529–539. https://doi.org/10.1145/2984511.2984540

[19] Scott Kirkpatrick, C. D. Gelatt, and Mario P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680. https://doi.org/10.1126/science.220.4598.671

[20] Roderic Lakes. 1987. Foam Structures with a Negative Poisson's Ratio. *Science* 235, 4792 (feb 1987), 1038–1040. https://doi.org/10.1126/science.235.4792.1038

[21] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-Last : Strength to Weight 3D Printed Objects. *ACM Transactions on Graphics* 33, 4 (2014). https://doi.org/10.1145/2601097.2601168

[22] Jonàs Martínez, Jérémie Dumas Haichuan Song, and Sylvain Lefebvre. 2017. Orthotropic k-nearest foams for additive manufacturing. *ACM Transactions on Graphics* 36, 4 (2017). https://doi.org/10.1145/3072959.3073638

[23] Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. 2017. A computational design tool for compliant mechanisms. *ACM Transactions on Graphics* 36, 4 (2017), 1–12. https://doi.org/10.1145/3072959.3073636

[24] M. J. Mirzaali, Shahram Janbaz, M. Strano, L. Vergani, and Amir A. Zadpoor. 2018. Shape-matching soft mechanical metamaterials. *Scientific Reports* 8, 965 (2018), 1–7. https://doi.org/10.1038/s41598-018-19381-3

[25] Stefanie Mueller, Pedro Lopes, and Patrick Baudisch. 2012. Interactive Construction: Interactive Fabrication of Functional Mechanical Devices. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM Press, New York, New York, USA, 599. https://doi.org/10.1145/2380116.2380191

[26] Jifei Ou, Zhao Ma, Jannik Peters, Sen Dai, Nikolaos Vlavianos, and Hiroshi Ishii. 2018. KinetiX - designing auxetic-inspired deformable material structures. *Computers & Graphics* 75 (oct 2018), 72–81. https://doi.org/10.1016/j.cag.2018.06.003

[27] Johannes T.B. Overvelde, James C. Weaver, Chuck Hoberman, and Katia Bertoldi. 2017. Rational design of reconfigurable prismatic architected materials. *Nature* 541, 7637 (2017), 347–352. https://doi.org/10.1038/nature20824

[28] Huaishu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch, and François Guimbretière. 2018. RoMA: Interactive Fabrication with Augmented Reality and a Robotic 3D Printer. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM Press, New York, New York, USA, 1–12. https://doi.org/10.1145/3173574.3174153

[29] Jesús Pérez, Miguel A. Otaduy, and Bernhard Thomaszewski. 2017. Computational design and automated fabrication of kirchhoff-plateau surfaces. *ACM Transactions on Graphics* 36, 4 (2017), 1–12. https://doi.org/10.1145/3072959.3073695

[30] Romain Prévost, Wojciech Jarosz, Moritz Bächer, Wojciech Jarosz, and Olga Sorkine-Hornung. 2016. Balancing 3D Models with Movable Masses. In *Vision, Modeling & Visualization*. The Eurographics Association, 8. https://doi.org/10.2312/vmv.20161337

[31] D.Janaki Ram, T.H. Sreenivas, and K.Ganapathy Subramaniam. 1996. Parallel Simulated Annealing Algorithms. *J. Parallel and Distrib. Comput.* 37, 2 (1996), 207–212. https://doi.org/10.1006/jpdc.1996.0121

[32] Pedro M. Reis, Heinrich M. Jaeger, and Martin van Hecke. 2015. Designer Matter: A perspective. *Extreme Mechanics Letters* 5 (2015), 25–29. https://doi.org/10.1016/j.eml.2015.09.004

[33] Michael L. Rivera, Melissa Moukperian, Daniel Ashbrook, Jennifer Mankoff, and Scott E. Hudson. 2017. Stretching the Bounds of 3D Printing with Embedded Textiles. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1–12. https://doi.org/10.1145/3025453.3025460

[34] Thijs Roumen, Bastian Kruck, Tobias Dürschmid, Tobias Nack, and Patrick Baudisch. 2016. Mobile Fabrication. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM Press, New York, New York, USA, 3–14. https://doi.org/10.1145/2984511.2984586

[35] Valkyrie Savage, Colin Chang, and Björn Hartmann. 2013. Sauron: embedded single-camera sensing of printed physical user interfaces. In *Proceedings of the annual ACM symposium on User interface software and technology*. 447–456. https://doi.org/10.1145/2501988.2501992

[36] Krishna Kumar Saxena, Raj Das, and Emilio P. Calius. 2016. Three Decades of Auxetics Research – Materials with Negative Poisson's Ratio: A Review. *Advanced Engineering Materials* 18, 11 (nov 2016), 1847–1870. https://doi.org/10.1002/adem.201600053

[37] Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. 2015. Microstructures to control elasticity in 3D printing. *ACM Transactions on Graphics* 34, 4 (2015). https://doi.org/10.1145/2766926

[38] Christian Schumacher, Steve Marschner, Markus Gross, and Bernhard Thomaszewski. 2018. Mechanical Characterization of structured sheet materials. *ACM Transactions on Graphics* 37, 4 (2018). https://doi.org/10.1145/3197517.3201278

[39] Hamed Seifi, Anooshe Rezaee Javan, Arash Ghaedizadeh, Jianhu Shen, Shanqing Xu, and Yi Min Xie. 2017. Design of hierarchical structures for synchronized deformations. *Scientific Reports* 7, January (2017), 1–7. https://doi.org/10.1038/srep41183

[40] Sicong Shan, Sung H. Kang, Jordan R. Raney, Pai Wang, Lichen Fang, Francisco Candido, Jennifer A. Lewis, and Katia Bertoldi. 2015. Multistable Architected Materials for Trapping Elastic Strain Energy. *Advanced Materials* 27, 29 (2015), 4296–4301. https://doi.org/10.1002/adma.201501708 arXiv:1207.1956

[41] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-Based Characters. *ACM Transactions on Graphics* 33, 4 (2014), 64:1–64:9.

[42] Nurcan Gecer Ulu, Stelian Coros, and Levent Burak Kara. 2018. Designing coupling behaviors using compliant shape optimization. *Computer-Aided Design* 101 (2018), 57–71. https://doi.org/10.1016/j.cad.2018.03.008

[43] Andreas Wächter and Lorenz T. Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106, 1 (01 Mar 2006), 25–57. https://doi.org/10.1007/s10107-004-0559-y

[44] Guanyun Wang, Tingyu Cheng, Youngwook Do, Humphrey Yang, Ye Tao, Jianzhe Gu, Byoungkwon An, and Lining Yao. 2018. Printed Paper Actuator: A Low-cost Reversible Actuation and Sensing Method for Shape Changing Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–12. https://doi.org/10.1145/3173574.3174143

[45] Christian Weichel, John Hardy, Jason Alexander, and Hans Gellersen. 2015. ReForm: Integrating Physical and Digital Design through Bidirectional Fabrication. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM Press, New York, New York, USA, 93–102. https://doi.org/10.1145/2807442.2807451

[46] Christian Weichel, Manfred Lau, David Kim, Nicolas Villar, and Hans W. Gellersen. 2014. MixFab: A Mixed-reality Environment for Personal Fabrication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3855–3864. https://doi.org/10.1145/2556288.2557090

[47] Karl D. D. Willis, C. Xu, K.-J. Wu, G. Levin, and Mark D. Gross. 2011. Interactive Fabrication: New Interfaces for Digital Fabrication. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*. 69–72. https://doi.org/10.1145/1935701.1935716

[48] Jonas Zehnder, Espen Knoop, Moritz Bächer, and Bernhard Thomaszewski. 2017. MetaSilicone: Design and Fabrication of Composite Silicone with Desired Mechanical Properties. *ACM Transactions on Graphics* 36, 6 (nov 2017), 1–13. https://doi.org/10.1145/3130800.3130881

[49] Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-aware retargeting of mechanisms to 3D shapes. *ACM Transactions on Graphics* 36, 4 (jul 2017), 1–13. https://doi.org/10.1145/3072959.3073710

A ENUMERATING ALL UNIQUE MECHANISMS

As noted in Section 4, all edges in a row or column necessarily belong to a single connected component. Starting with an empty $n \times m$ grid we have $n + m$ connected components. Introducing a rigid cell joins two connected components and removes one degree of freedom. Suppose we want to enumerate all unique mechanisms with k degrees of freedom. This amounts to k connected components where we differentiate between x empty columns, y empty rows and z components formed by merging rows and columns using rigid cells. There are $\binom{n}{x}$ ways to choose the columns and $\binom{m}{y}$ ways to choose the rows. The remaining $m - y$ rows and $n - x$ columns can be arbitrarily partitioned into z sets. The Stirling number of the second kind counts the number of these partitions as $S(n - x,\ z)$ and $S(m - y, z)$ where we use the convention $S(a, b) = 0$ for $a < b$. The z sets of rows and columns can be connected in $z!$ ways. This gives

$$G^{n,m}(x, y, z) = \binom{n}{x}\binom{m}{y} S(n - x,\ z)\ S(m - y, z)\ z!$$

different possibilities. Summing over all values for x, y and z we obtain the number of all possible unique mechanisms on a $n \times m$ grid. We empirically verified this number by analyzing all possible configurations on grids with $n < 5$.

B EXAMPLE TRANSFORMATIONS

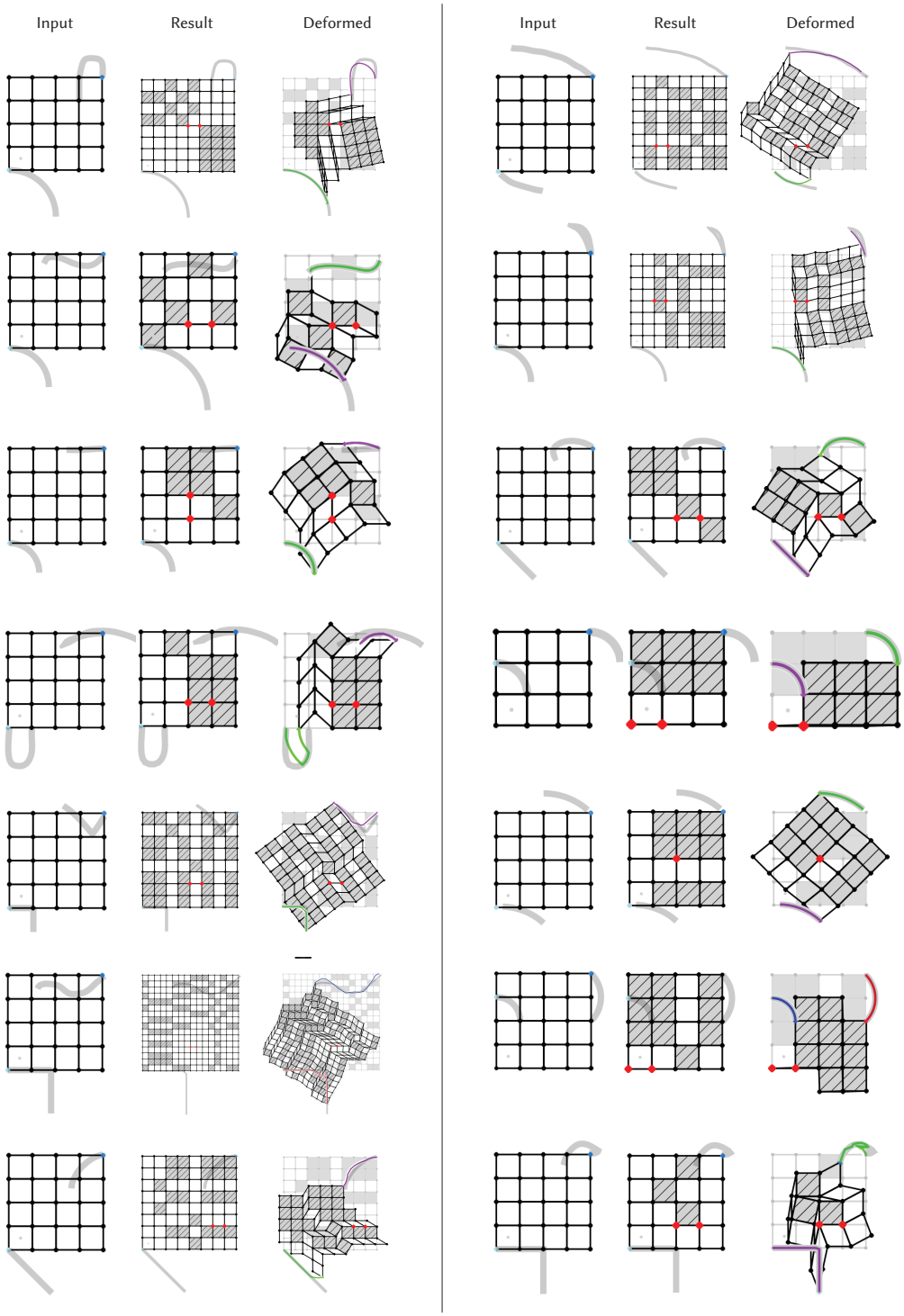

Figure 18: Examples of mechanisms generated with our algorithms. Note that for every example, two paths and the empty cell configuration was given. Anchor positions and rigid cell assignments were automatically determined.