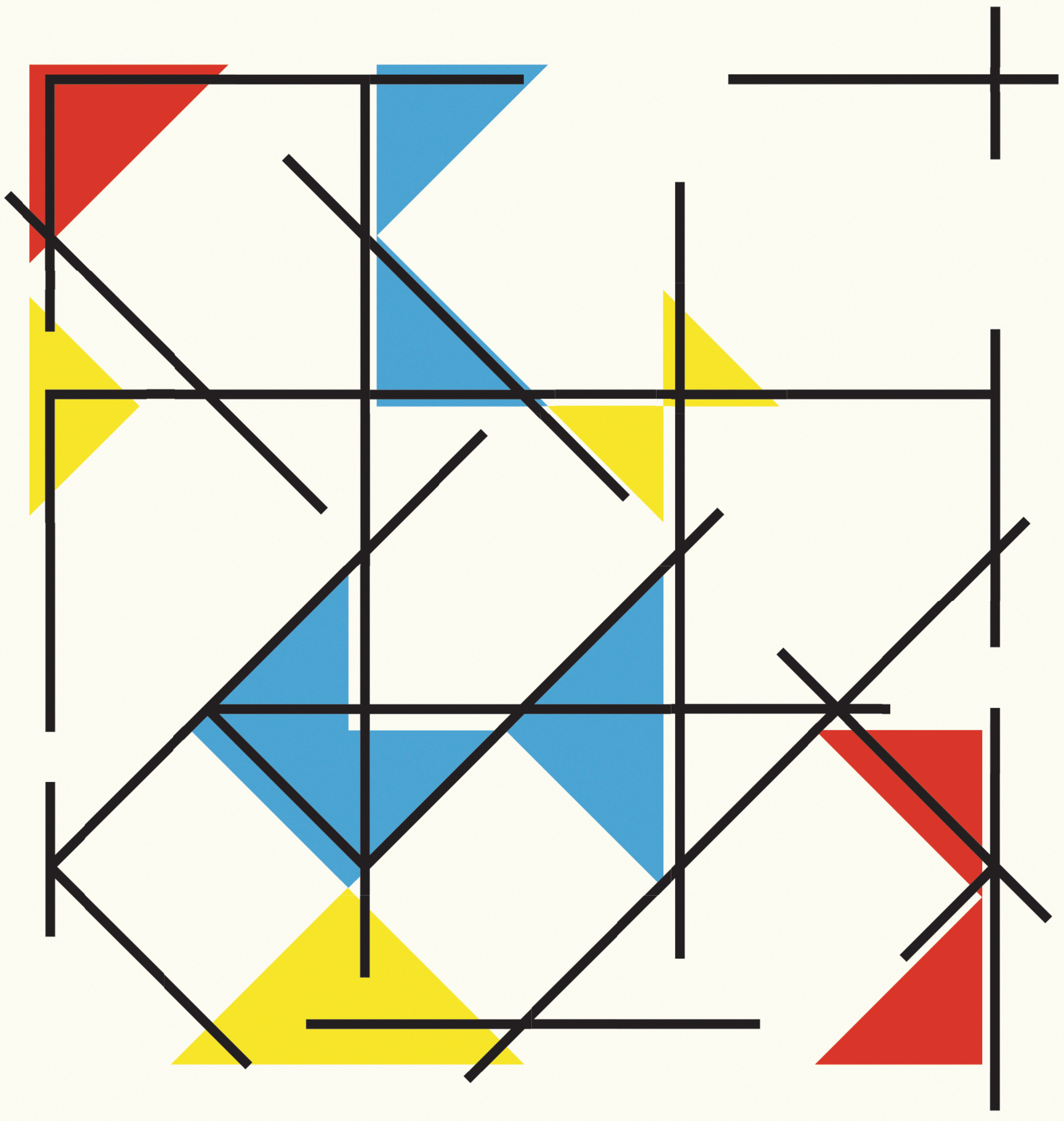
SEARCHING FOR STRUCTURE

Uncovering Combinatorial Rules to Design Metamaterials



Ryan van Mastrigt

University of Amsterdam
Faculty of Science

AMOLF physics of functional complex matter

ISBN 978-94-93391-19-2
Cover design by Ryan van Mastrigt

Searching for Structure
Uncovering combinatorial rules to design metamaterials

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op woensdag 4 september 2024, te 10.00 uur

door Ryan van Mastrigt
geboren te Utrecht

Promotiecommissie

Promotores:	prof. dr. M.L. van Hecke	Universiteit Leiden
	dr. C.J.M. Coulais	Universiteit van Amsterdam
Copromotores:	prof. dr. ir. M. Dijkstra	Universiteit Utrecht
Overige leden:	dr. Z. Zeravcic	ESPCI Paris
	prof. dr. D.M. Kochmann	ETH Zürich
	dr. ir. B. Ensing	Universiteit van Amsterdam
	prof. dr. P.G. Bolhuis	Universiteit van Amsterdam
	dr. A. Pérez de Alba Ortíz	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Contents

1 Introduction

Nature is replete with complex, emergent functionality. Think of atoms binding together to form effector molecules that regulate biological activity, chains of amino acids folding into transport proteins that move molecules across cell membranes, and cells clumping together to form muscle tissues with anisotropic elastic moduli. In all these examples, functionality emerges from the collective response of building blocks —atoms, amino acids, cells—intricately connected in some spatial structure. Abstractly, this collective response is captured in the forward structure-property relation that maps microscopic details—building block types and spatial arrangement—to a macroscopic property of interest, such as the elastic modulus. Thus, the goal of this relation is to predict for a given structure the resulting property.

Conversely, the inverse relation—property-structure—aims not to predict but to *design*: produce a structure such that the desired property emerges. In practice, this is an ill-posed problem—there may be many structures that result in the same property and a solution is not guaranteed. Crucially, there is an underlying (typically unknown) order to this inverse relation. This order is captured in design rules: conditions that a structure should satisfy to support a desired property. Without such rules, design is limited by intuition or brute force trial-and-error.

In this thesis, we encounter such forward and inverse problems in the context of mechanical metamaterials: artificially designed materials that leverage geometric effects to achieve exceptional mechanical functionalities [1, 2] such as tuneable mechanical properties [3–6], mechanical memory [7–9], steerable deformations [10] , and shape-morphing [11–13] (see Fig. 1.1 for examples). Specifically, we focus on metamaterials that feature multiple deformation pathways that cost little energy to actuate. These so-called multimodal metamaterials allow for new and exciting functionalities, such as nonlocal resonances [14], multi-shape folding [15–17], sequential buckling [18], and selectable mechanical responses [19]. However, a systematic strategy to design multimodal metamaterials is lacking.

We aim to find such systematic design strategies. To achieve this ambitious goal, we consider a family of multimodal metamaterials comprised of building blocks. The challenge is to find tilings of these building blocks that feature desired deformation pathways. However, we find that such tilings are rare exceptions in a vast sea of failed designs. This is emblematic of combinatorial problems which are ubiquitous in science, for example, in self-assembly [20–24], origami [17, 25], amorphous matter [7, 26–29],

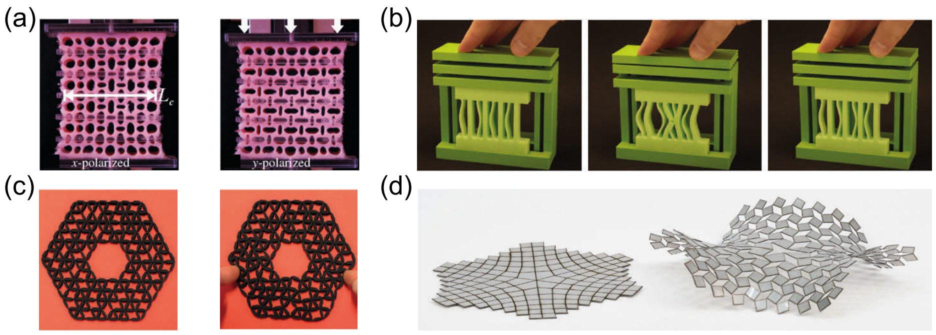


Figure 1.1: **Examples of mechanical metamaterials.** (a) The stiffness to uniaxial compression can be tuned by changing the horizontal confinement. Adapted from [3]. (b) The orientation of the slender beams flip iteratively upon compression, effectively counting the number of compression cycles. Adapted from [8]. (c) A topological defects renders a previously soft metamaterial stiff on the top side and soft on the bottom side, allowing for steerable deformation along the bottom [10]. Credit: AMOLF. (d) A kirigami-based metamaterial that can continuously morph from a flat shape (left) into another curved shape (right). Adapted from [12].

molecular design [30, 31], computer graphics [32–34], chip design [35, 36], and metamaterial design [10, 11, 19, 37].

Here, we tackle combinatorial problems using both rational design and machine learning. First, in chapter 2, we devise a theoretical framework for kinematic constraints to derive design rules. While successful, this approach can be strenuous, prompting us to explore effective alternatives. Subsequently, in chapter 3, we use neural networks to accurately delineate design space into rare compatible and abundant incompatible designs. Finally, in chapter 4 we formulate a hybrid design approach that combines computational and rational design to create metamaterials with multiple targeted spatially-textured deformations. In what follows, we provide brief background to these chapters and discuss our main findings.

1.1. Mechanical metamaterial design

Throughout this thesis, we take a mechanism-based approach to metamaterial design. Below, we briefly discuss the basics of metamaterial design and the challenges that arise.

In general, mechanical metamaterials feature "soft" deformation modes: pathways of deformation that cost little energy. These soft modes impart

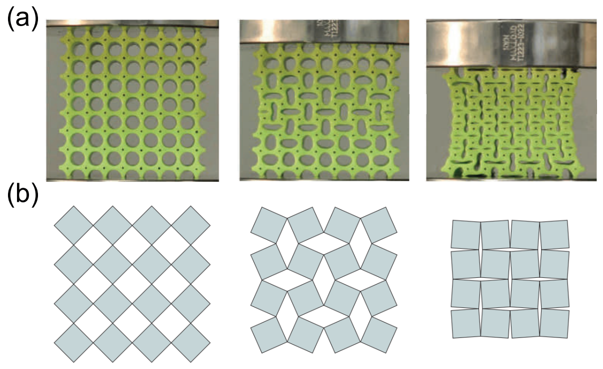


Figure 1.2: **Auxetic flexible and mechanism-based metamaterials.** (a) A rubber sheet (green) perforated with holes deforms auxetically when compressed from the top. Adapted from [2]. (b) Rigid bars (black lines) enclose rigid squares (gray) that are connected by hinges. These squares freely rotate in an auxetic deformation. Adapted from [39].

the metamaterial with exceptional mechanical properties. For example, a block of rubber perforated with multiple holes placed in a square pattern displays an auxetic response when compressed from the top—the metamaterial contracts horizontally under vertical compression [Fig. 1.2(a)]. To understand this auxetic deformation, we neglect the complex nonlinear elasticity of this highly heterogeneous material and instead consider a simpler bars-and-hinges framework. This framework consists of rigid bars connected by hinges that freely rotate. The metamaterial's deformation can then be emulated in a frame of connected rigid squares that feature a deformation that costs zero energy, i.e., does not stretch any of the bars. This so-called zero mode effectively rotates the squares in an alternating pattern [Fig. 1.2(b)], closely resembling the deformation of the metamaterial. This particular deformation is termed the counter-rotating squares (CRS) mode [2, 38, 39] and provides an effective description of the mechanics underpinning the metamaterial's auxetic response. Thus, this bars-and-hinges framework is an attractive, mechanism-based approach to describe mechanical metamaterials. Below, we discuss how we use this framework to calculate zero modes, classify the structural integrity of frames, and how we can use this to design metamaterials with desired mechanical properties.

1.1.1. Calculating zero modes

To calculate the zero modes of a frame [Fig. 1.3(a)], we determine the frame's compatibility matrix $\mathcal{C}$. This matrix maps small displacements of the hinges $\mathbf{u}$ to elongations of the bars ϵ to first order in $\mathbf{u}$. Zero modes, i.e., deformations that do not stretch any of the bars, span the null space, or kernel, of $\mathcal{C}$. In other words, $\mathbf{u}_{\text{ZM}}$ is a zero mode if $\mathcal{C}\mathbf{u}_{\text{ZM}} = \mathbf{0}$ [Fig. 1.3(b)]. Inversely, the equilibrium matrix $\mathcal{Q}$ maps stresses on the bars σ to loads on the hinges $\mathbf{l}$ and is equal to the transpose of the compatibility matrix $\mathcal{C}^T$. This equivalence follows from requiring the virtual work done by the hinges and bars to be equal [40]. Stresses that do not induce any load (net forces) are states of self-stress and span the null space of $\mathcal{Q}$, i.e., σ_S is a state of self-stress if $\mathcal{Q}\sigma_S = \mathbf{0}$ [Fig. 1.3(c)].

For a given frame with N hinges and N_B bars in d dimensions, the $N_B \times dN$ compatibility matrix $\mathcal{C}$ can be composed and a set of basis vectors that span the null space $\mathcal{N}(\mathcal{C})$ can be determined. In other words, this set of vectors span all the zero modes of the frame. The dimensionality of this set, $\dim \mathcal{N}(\mathcal{C})$, corresponds to the total number of distinct zero modes of the frame. In practice, we neglect the trivial zero modes: rigid translations and rotations of the entire frame. Thus, we find that the number of nontrivial zero modes $N_{\text{ZM}} = \dim \mathcal{N}(\mathcal{C}) - f(d)$, where $f(d) = d(d+1)/2$ is the number of rigid body motions in d dimensions. Similarly, the number of states of self-stress N_{S} is equal to the dimensionality of the null space of the $dN \times N_B$ equilibrium matrix $\mathcal{Q}$, i.e., $N_{\text{S}} = \dim \mathcal{N}(\mathcal{Q})$. For example, for the metamaterial in Fig. 1.3(a) we find that $N_{\text{ZM}} = 1$ [Fig. 1.3(b)], $f(2) = 3$ and $N_{\text{S}} = 2$ [Fig. 1.3(c)]. For large frames it is cumbersome to calculate the zero modes and states of self-stress by hand. Instead, the null space can be computed numerically by singular value decomposition (SVD) algorithms.

1.1.2. Classification of structural integrity

In practice, a simpler structural integrity classification of a frame may be used that does not require explicit calculation of $\mathcal{C}$. This classification follows from a counting rule that can be derived directly from the definitions of the compatibility and equilibrium matrices, $\mathcal{C}$ and $\mathcal{Q}$, using the

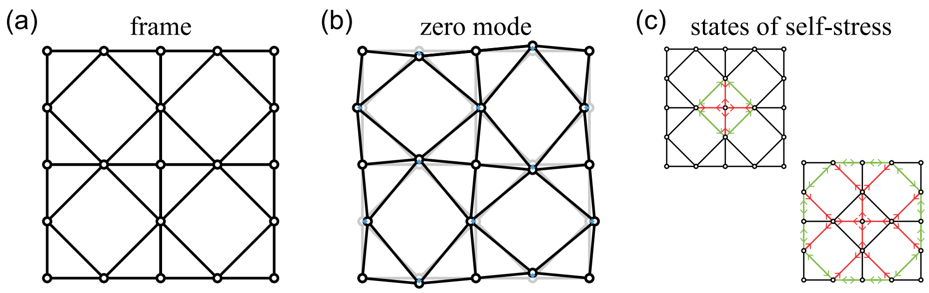


FIGURE 1.3: **Bars-and-hinges framework.** (a) Example of a frame consisting of rigid bars (black lines) and hinges (white circles). (b) The frame of (a) deformed by its single zero mode: a small deformation of the hinges (blue arrows) with respect to the rest frame (gray) that does not stretch any of the rigid bars. We note that this deformation is equivalent to the counter-rotating squares deformation of Fig. 1.2(b). (c) The two states of self-stress supported by the frame of (a). Compressive (tensile) stress on the bar is indicated by the ingoing (outgoing) arrows and green (red) color. A bar under stress induces a force on its connected hinges. In a state of self-stress the net force on each hinge is zero.

rank-nullity theorem:

$$\begin{aligned} N_{\text{ZM}} - N_{\text{S}} &= \dim \mathcal{N}(\mathcal{C}) - \dim \mathcal{N}(\mathcal{Q}) - f(d) \\ &= \dim \mathcal{N}(\mathcal{C}) - \dim \mathcal{N}(\mathcal{C}^T) - f(d) \\ &= dN - n_r - (N_B - n_r) - f(d) \\ &= dN - N_B - f(d) \equiv \mathcal{P}, \end{aligned} \tag{1.1}$$

where n_r is the rank of the compatibility matrix $\mathcal{C}$. $\mathcal{P}$ is commonly referred to as the Maxwell-Calladine count [41, 42].

This count $\mathcal{P}$ can be used to classify the structural integrity of a frame into three distinct cases:

i $\mathcal{P} < 0$: hyperstatic, the frame necessarily supports states of self-stress

ii $\mathcal{P} = 0$: isostatic, equal number of modes and states of self-stress,

iii $\mathcal{P} > 0$: hypostatic, the frame necessarily supports zero modes.

We note that this classification does not provide the number of modes N_{ZM} of a frame. Instead, it gives a lower bound: $N_{\text{ZM}} \geq \mathcal{P}$ as both N_{ZM} and N_{S} are non-negative integers. The N_{ZM} exceeds this lower bound if there are degenerate rigid bars that result in additional states of self-stress. As we will show, this structural integrity classification is useful to design for spatially extended zero modes.

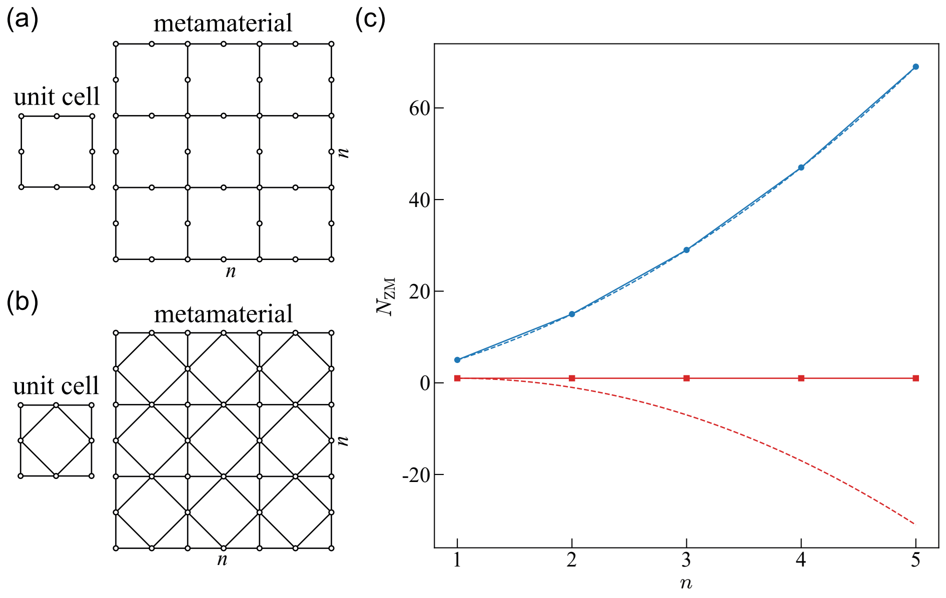


Figure 1.4: **Mode-scaling of hypostatic and hyperstatic frames.** (a) Example of a hypostatic metamaterial. The unit cell (left) is tiled on a $n \times n$ square lattice to form a larger metamaterial (right shows $n = 3$ tiling). (b) Example of a hyperstatic metamaterial. (c) The number of modes N_{ZM} of the hypostatic (blue circles) and hyperstatic (red squares) metamaterials and their lower bounds $\mathcal{P}$ [Eq. (1.1)] (dashed lines) as a function of the tiling parameter n. The hypostatic metamaterial necessarily contains many spatially localized zero modes, because N_{ZM} follows $\mathcal{P}$ and thus scales quadratically with n. The hyperstatic metamaterial features only a single intensive mode, which corresponds to the spatially extensive counter-rotating squares mode [Fig. 1.3(b)].

1.1.3. Towards mechanism-based design

To show how the Maxwell-Calladine count [Eq (1.1)] can be used for metamaterial design, we consider two different metamaterial designs: a hypostatic design [Fig. 1.4(a)] and a hyperstatic design [Fig. 1.4(b)]. Like most metamaterial designs, these metamaterials are composed of a unit cell that can be repeated to create a larger $n \times n$ tiling of unit cells. Generally, the goal of mechanism-based metamaterial design is to create a frame that supports spatially extended zero modes. Thus, at first glance, you might expect that the design should be hypostatic, as such designs are guaranteed to support zero modes.

Instead, we find that hypostatic designs are likely to support mostly spatially localized zero modes. For example, consider the hypostatic metamaterial of Fig. 1.4(a). The Maxwell-Calladine count gives $\mathcal{P} = 2n^2 + 4n - 1$. As $N_{\text{ZM}} \geq \mathcal{P}$, the number of zero modes N_{ZM} must increase quadratically as well [Fig. 1.4(c)]. We note that spatially localized modes necessarily contribute to the scaling of the number of modes N_{ZM} with the tiling parameter n, such that this hypostatic metamaterial must contain many localized modes.

Inversely, hyperstatic designs are less likely to support spatially localized modes. For example, consider the hyperstatic metamaterial of Fig. 1.4(b). Note that the frame in Fig. 1.3(a) is a 2×2 realization of this metamaterial. The Maxwell-Calladine count gives $\mathcal{P} = -2n^2 + 4n - 1$, such that the frame is hyperstatic for $n \geq 2$ and must contain states of self-stress. In fact, recall that we found two states of self-stress for the $n = 2$ tiling in Fig. 1.3(c). Moreover, an analysis of the states of self-stress for larger tilings reveals that these two states of self-stress are repeated periodically in the frame, such that the number of states of self-stress proliferates as $N_{\text{S}} = 2(n-1)^2$. From Eq. (1.1), it then follows that the number of modes $N_{\text{ZM}}(n) = 1$ [Fig. 1.4(c)]. That is, the number of modes is constant and independent of n. Therefore the frame supports a single, intensive zero mode regardless of n: the spatially extended CRS mode shown before in Fig. 1.2(b) and Fig. 1.3(b). Such intensive modes do not result in a new zero mode under spatial translation, because that would constitute a dependence on n. Consequently, intensive modes are likely to span the entire structure or localize on the edge(s) of the material. Thus, the scaling of $\mathcal{P}(n)$ with n is important to the presence of localized zero modes in the frame.

Generally, the scaling of $\mathcal{P}$ is a polynomial of, at most, degree d in n. Surprisingly, this is not true for states of self-stress. We note that states of self-stress are always localized in frames with open boundary conditions, such that either there are no states of self-stress $N_{\text{S}} = 0$ or the number must be proportional to $N_{\text{S}} \propto n^d$. In other words, $N_{\text{S}}(n)$ is either zero or a polynomial of degree d with a positive d-th degree coefficient. This localization follows from the observation that a state of self-stress is always self-contained, i.e., adding extra bars and hinges to the frame does not alter existing states of self-stress. For example, the state of self-stress in Fig. 1.5(a) does not change under an increase of the lattice size. Conversely, this localization argument does not hold for zero modes. For zero modes, any new bars and hinges that are added to the frame can alter the zero modes. For example, the CRS mode in Fig. 1.5(b) expands to newly added hinges under an increase of the lattice size. Thus, for a frame to feature

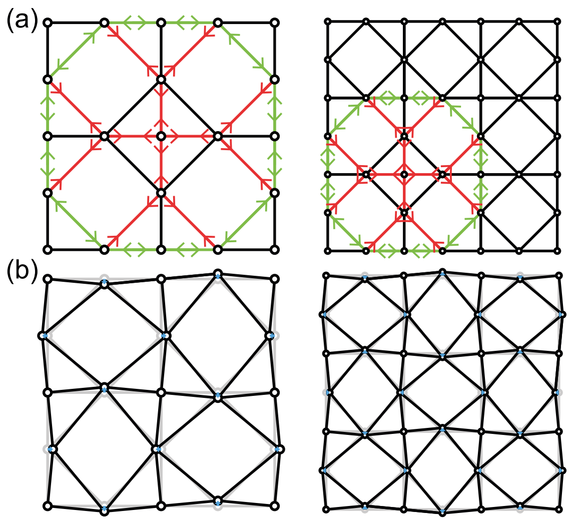


Figure 1.5: **Localization of states of self-stress and zero modes.** (a) A state of self-stress (left) is always localized, because it does not change under growth of the lattice (right). (b) A zero mode does not have to be localized, in this example the zero mode of the 2×2 lattice (left) extends under growth of the lattice (right).

no localized modes we require either a constant positive $\mathcal{P}$ and $N_\text{S} = 0$, or $\mathcal{P} \propto -n^d$ and $N_\text{S} \propto n^d$ such that all n-dependent terms cancel.

To summarize, to design metamaterials composed of a $n \times n$ tiling of unit cells with spatially extended zero modes and few localized modes, the Maxwell-Calladine counting $\mathcal{P}$ [Eq. (1.1)] should scale as $\mathcal{P} \propto -n^d$. For such frames, the number of states of self-stress N_S can cancel the higher order terms of the polynomial $\mathcal{P}$ and result in a positive, constant number of modes N_ZM. However, a central challenge remains: how do we design unit cells that feature desired, spatially extensive zero modes?

1.2. Combinatorial design

The space of possible frame geometries is limitless. How to effectively explore this space to designs frames with desired mechanical properties is an open problem. Here, we take a combinatorial approach: we limit ourselves to a discrete set of building blocks that we combine to form larger structures. This approach renders the design space finite and turns the design problem into a combinatorial tiling problem: how do we find the right tilings of building blocks that yield desired properties? We illustrate the appeal and challenges of this combinatorial approach by example below.

1.2.1. Unimodal design

Consider the building block in Fig. 1.6(a). This building block consists of two two-dimensional cells. In turn, each cell consists of four rigid triangles connected by four two-dimensional hinges, resulting in a single zero mode. Requiring the two cells to be connected and perpendicular constrains the cells' zero modes, yielding one zero mode for the entire building block [Fig. 1.6(a)].

Larger metamaterials can be constructed by tiling this building block in a cubic $n \times n \times n$ lattice. A simple count of the deformational degrees of freedom and kinematic constraints yields $\mathcal{P} = -2n^3 + 3n^2$, such that the resulting frame is hyperstatic for $n > 2$. As $\mathcal{P} \propto -n^3$, tilings of this building block likely support few extensive, spatially localized zero modes. Moreover, the building blocks have fewer symmetries than the cubic lattice they live on. This allows for orientation of the building block in three distinct ways [Fig. 1.6(b)] and 3^{n^3} possible cube configurations. However, most random configurations are kinematically frustrated, i.e., they do not support a zero mode [11].

Crucially, design rules can be derived for this metamaterial, enabling rational design for target deformations. To derive these rules, in- and outgoing angular deformations are mapped to "spins" that live on edges in a graph-like framework where vertices correspond to building blocks [Fig. 1.6(b)]. Note that there is a parity symmetry: flipping all spins results in the same zero mode modulo the sign. A set of spins is compatible, i.e., corresponds to a zero mode, if for all edges connected to a vertex the spins satisfy one of the distinct patterns, similar to ice rules in spin ice systems [43]. These patterns correspond to building block orientations and parity of the zero mode. Thus, a simple check of these patterns gives the set of building block orientation that support the given zero mode [Fig. 1.6(c)-i]. We note that configurations of this building block cannot support more than one mode. This stems from the coupling between building block deformations: all hinges deform and thus couple to all neighboring building blocks. As each block supports only a single zero mode, there is at most only one compatible deformation of the building blocks.

Moreover, these rules can be used to design spatially-textured deformations on the faces of a cubic configuration. In this case, there are simple

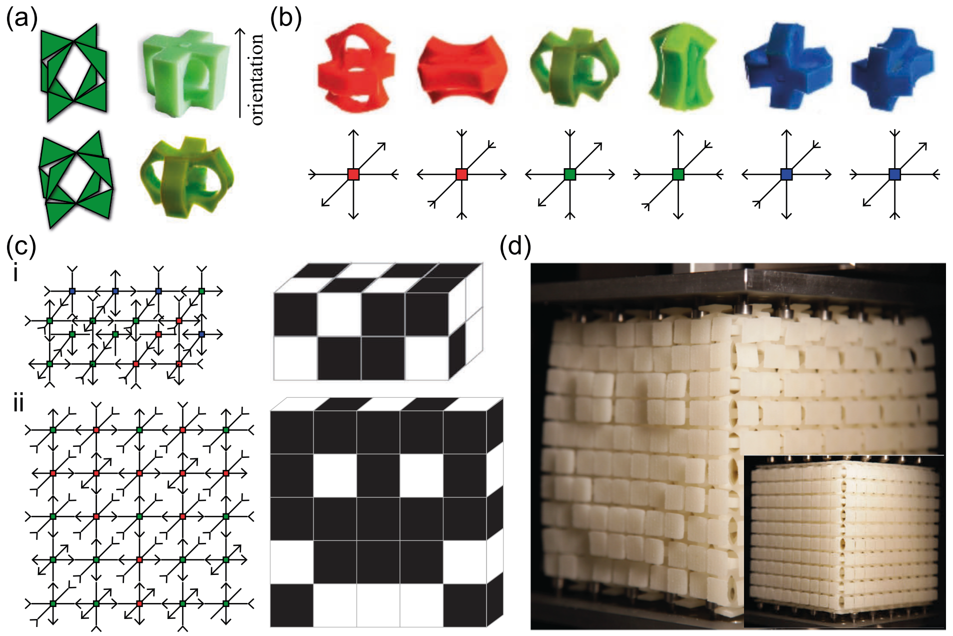


Figure 1.6: **Combinatorial metamaterial design.** (a) Schematic (left) and 3D printed (right) building block (top) supports one zero mode (bottom). (b) The zero modes for the different orientations (colors) can be represented in a graph, where ingoing (outgoing) arrows on the edges represent increasing (decreasing) angular deformations. The vertex of the edges represents the building block orientation. Note that there is a parity symmetry—flipping all arrows yields the same zero mode modulo the sign. (c) Collections of deformations (left) correspond to a zero mode if the arrows on the edges of each vertex correspond to one of the configurations in (b). On the surface of the configuration (right), a zero mode results in a pattern of ingoing (black) and outgoing (white) deformations. This pattern can be designed by inverting the checkerboard pattern of red and green building blocks (ii). (d) A $10 \times 10 \times 10$ cube that compresses to show a smiley on the cube's face. The inset shows the cube before compression. Adapted from [11].

rules to yield any desired spatial pattern on one of the faces. First, only a single slice across the cube parallel to the face of choice needs to be defined. The rest of the cube are simply copies of this slice due to the parity symmetry. In this slice, sections that all move inward or outward should follow a checkerboard pattern of blue and red/green blocks [Fig. 1.6(c)-ii]. By inverting this pattern, sections switch from all inward to all outward and vice versa. These rules were used to design a metamaterial that, when compressed, deforms to show a smiley face [Fig. 1.6(d)].

In conclusion, this combinatorial approach to metamaterial design allows for design of complex aperiodic metamaterials with desired textured shape-changes. However, several open questions remain. How do we extend this framework to building blocks beyond simple ingoing and outgoing deformations? How do we design a metamaterial that features multiple shape-changes, e.g., that can change into a smiley and a frowny?

1.2.2. Multimodal metamaterial

To answer these questions, we extend the combinatorial design approach to multimodal metamaterials that feature multiple zero-energy deformation modes. Generally, metamaterials can be categorized into one of three classes based on the scaling of the number of zero modes with the system size: [19]

(i) unimodal: a single zero mode,

(ii) plurimodal: many extensive zero modes,

(iii) oligomodal: multiple intensive modes.

On the one hand, unimodal metamaterials feature a single mechanical response which is straightforward to actuate. For example, the metamaterial of Fig. 1.2 is unimodal and the auxetic deformation is actuated by uniaxial compression. On the other hand, plurimodal metamaterials support many different mechanical responses which are hard to actuate in practice. For example, the metamaterial of Fig. 1.4(a) has a lot of deformational freedom, but actuating a specific desired deformation is difficult. Oligomodal metamaterials are a middle ground between these two, allowing for multiple mechanical responses while maintaining ease of actuation.

We focus on a family of combinatorial metamaterials composed out of bimodal building blocks [Fig. 1.7(a)] that were shown, for the right tilings, to be oligomodal [19]. Moreover, specific tilings have been realized experimentally with exceptional functionalities, such as a selectable auxetic response [19][Fig. 1.7(b)] and sequential buckling [18] [Fig. 1.7(c)-(d)]. However, for tilings larger than 3×3, the design space is too large to fully explore. We believe that this metamaterial is sufficiently rich and complex to warrant further exploration of the design space. We provide a brief introduction to this metamaterial below.

The fundamental building block is shown schematically in Fig. 1.7(a). Each black line represents a rigid bar, while vertices can be thought of as

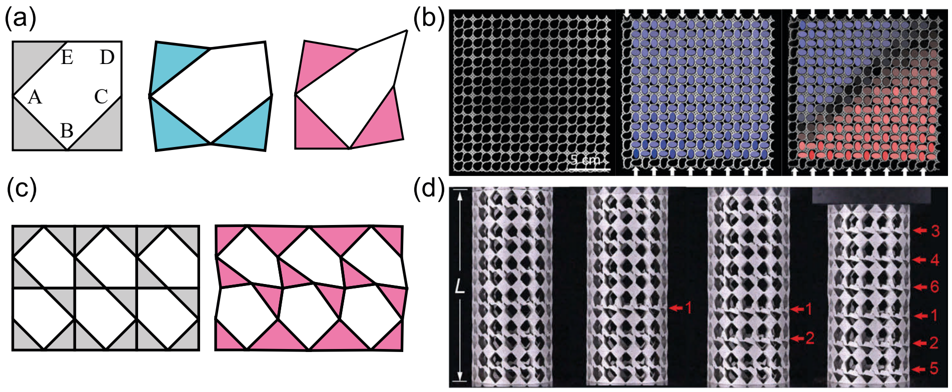


FIGURE 1.7: **Multimodal combinatorial metamaterial.** (a) The undeformed building block (gray) supports two zero modes (blue and pink). A, B, C, D, and E label the five interior angles that can change under zero-energy deformations. (b) 3D printed flexible tiling (left) of the building block in (a) that deforms in an auxetic pattern (middle) or not (right) depending on how the metamaterial is actuated (arrows indicate textured boundary conditions). The colored ellipses indicate the polarity orientation of each building block in a checkerboard pattern and serves as a guide to the eye. Adapted from [19]. (c) A 3×2 tiling of building blocks (left) deforms along a strip (right). Note that the hinges at the top and bottom edges of the tiling do not deform. (d) 3D printed metal tiling (left) of the building block in (a) in a cylindrical shape sequentially buckles upon compression, numbers indicate order of buckling. This buckling follows the deformation of shown in (c). Adapted from [18].

hinges; the 11 bars are free to rotate about the 8 hinges in 2 dimensions. The colored triangles form rigid structures, i.e., they will not deform. From the Maxwell-Calladine counting [Eq. (1.1)] we obtain $\mathcal{P} = 2 \cdot 8 - 11 - 3 = 2$. The building block contains no states of self-stress, such that $N_{\text{ZM}} = 2$. The precise deformation of these two zero modes can be derived from the geometric constraints of the building block.

To derive the zero modes to linear order, we note that they preserve the length of all bars, such that the modes can be characterized by the hinging angles of the bar. Let A, B, C, D, and E denote these angles, see Fig. 1.7(a). Going around the loop $ABCDE$, the angles add up to 3π:

$$A + B + C + D + E = 3\pi. \tag{1.2}$$

Next, we expand the angles from their rest position to linear order:

$$\begin{aligned} A &= \tfrac{\pi}{2} + \alpha, \quad B = \tfrac{\pi}{2} + \beta, \quad C = \tfrac{3\pi}{4} + \gamma, \\ D &= \tfrac{\pi}{2} + \delta, \quad E = \tfrac{3\pi}{4} + \epsilon. \end{aligned} \tag{1.3}$$

Then, from the condition that the bars cannot change length, we obtain

$$1-\cos(D)=3-2\cos(A)-2\cos(B)+2\cos(A+B), \tag{1.4}$$

and

$$\sin(B)-\frac{\sin(B+C)}{\sqrt{2}}=\sin(A)-\frac{\sin(A+E)}{\sqrt{2}}. \tag{1.5}$$

Up to first order in $\alpha, \beta, \gamma, \delta, \epsilon$, equations (1.4) and (1.5) can be rewritten as:

$$\delta=2\alpha+2\beta, \tag{1.6}$$

$$\beta+\gamma=\epsilon+\alpha. \tag{1.7}$$

Together with the loop condition (1.2), we obtain a set of three equations which express β, γ and δ in α and ϵ:

$$\begin{pmatrix}\beta\\ \gamma\\ \delta\end{pmatrix}=\begin{pmatrix}-2 & -1\\ 3 & 2\\ -2 & -2\end{pmatrix}\begin{pmatrix}\alpha\\ \epsilon\end{pmatrix}. \tag{1.8}$$

This demonstrates that we can choose the two parameters α and ϵ arbitrarily, while still satisfying Eqs. (1.2), (1.6) and (1.7), consistent with the presence of two zero modes.

By tiling together the building block in different orientations, we can create 4^{k^2} size $k\times k$ unit cells that we in turn tile to form $n\times n$ metamaterials. These unit cells—and metamaterials built from them—may have more or less zero modes than the constituent building blocks, depending on the number of states of self-stress. Previous work on 2×2 unit cells showed that each unit cell could be classified based on the number of zero modes [19]. In this thesis, we consider the previously unexplored cases of 3×3 up to 8×8 square unit cells.

To explore this design space, we first generate all 3×3 unit cell designs, a million random $k\times k$ unit cell designs for $k\in\{4,5,6\}$, and two million random designs for $k\in\{7,8\}$. For unit cells up to size $k=6$, we determine the number of modes N_{ZM} for a $n\times n$ tiling of this unit cell for $n\in\{1,2,3,4\}$ by composing the compatibility matrix and computing the dimensionality of its null space using rank-revealing QR decomposition [44]. For the larger unit cells of size $k\in\{7,8\}$, we generate $n_x\times2$ and $2\times n_y$ tilings instead. This significantly decreases the size of the compatibility matrix and thus the time it takes to calculate the number of modes.

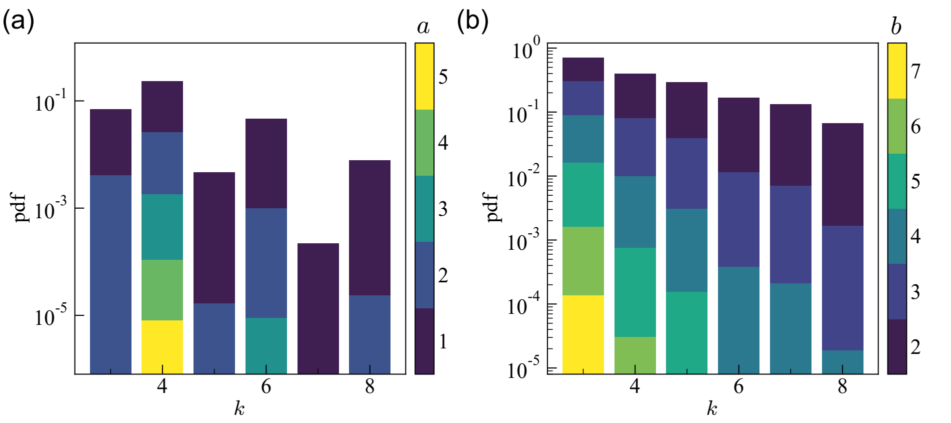


Figure 1.8: **Distribution of extensive and intensive modes.** (a) Stacked bar plot of the probability density function (pdf) for the number of extensive modes a for randomly sampled $k \times k$ unit cells. (b) Stacked bar plot of the pdf for the number of intensive modes b for randomly sampled $k \times k$ unit cells.

Surprisingly, we find that all tilings follow a linear mode scaling relation for sufficiently large n, such that:

$$N_{\mathrm{ZM}} = an + b, \tag{1.9}$$

where a and b correspond to the number of extensive and intensive modes, respectively. We note that any unit cell tiling supports the CRS mode, an intensive mode, such that $b \geq 1$. From the Maxwell-Calladine count we expect $\mathcal{P} = -k^2n^2 + 4kn - 1$, thus our metamaterial designs must contain $N_{\mathrm{S}} \propto k^2n^2$ states of self-stress. The exact scaling depends on the tiling of the building blocks.

We show the distribution of the number of extensive and intensive modes of these randomly generated unit cells in Fig. 1.8. We find that unit cell designs with a higher number of modes become exponentially more rare with increasing unit cell size k. Moreover, the size of the design space increases exponentially with k and the majority of generated designs feature only a single CRS mode. This is typical of combinatorial metamaterial design—designs with non-trivial properties become increasingly challenging to find in a design space that mushrooms with size. Without design rules, finding designs with desired properties is intractable.

In this thesis, we use this metamaterial to tackle the challenge of multimodal metamaterial design. Specifically, we derive design rules for this metamaterial in chapter 2, show that a neural network can learn such rules from examples in chapter 3, and combine computational and rational design to design metamaterials with desired spatially aperiodic deformations in chapter 4.

1.3. Machine learning for metamaterials

Machine learning (ML) forms an essential part of our strategy to design multimodal metamaterials. Below, we briefly discuss the basics of neural networks.

Machine learning is an umbrella term that encapsulates a wide range of general computational techniques to perform desired tasks. In general, such a technique consists of a *machine*: an algorithm that can be tuned to do a specific task in a process termed learning. For example, a neural network is a ML algorithm typically represented in a directed graph structure, where each node represents a (usually) real value and edges represent mathematical operations to transform values from a start node to a new value in an end node. Such networks have a graph topology, or architecture, that specifies a clear flow of information; there is an input and an output. Then, the inner machinations of this network can be tuned by altering the mathematical operations—typically affine transformations followed by a non-linear activation function—between nodes such that the network improves at its intended task. This tuning, or learning, transpires through minimization of some loss function that dictates how well the network is doing. To do so, the network requires examples: either pairs of desired input and output (supervised learning), or just inputs (unsupervised learning). This set of data that the network uses to learn is called the training set. After learning, the network is trained and can do its intended task. The trained network's performance is typically measured over another set of data that is not used to learn: the test set. A good performance over this test set is typically the goal, such a network is said to generalize well. The type of performance measure—accuracy, mean squared error, area under ROC curve—that is measured depends on the intended task. In short, the machine is a general-purpose algorithm that is tuned to perform a specific, desired task through minimization of a loss function.

Such ML techniques are broadly applicable. Any task that can be framed in terms of input-to-output can, in principle, be learned by such an algorithm [45], ranging from learning simple mathematical functions to abstract tasks such as object identification. Provided there are sufficiently many examples available, a machine can be trained and perform the desired task. The immense increase in computational power and available data over the past decade significantly fueled this approach, resulting in machines that can perform tasks that were previously thought impossible for computers.

Materials science is no exception to this trend. Over the past decade, ML has found tremendous success in classification, prediction and design of materials [46, 47], including mechanical metamaterials [48]. Most notably, to design supercompressible structures [49], soft materials [50], truss metamaterials [51], and auxetic structures [52]. However, in all these examples both the structure and property are continuously varying functions.

In contrast, we aim to classify and predict mechanical properties of combinatorial metamaterials. Such materials present unique challenges: the design space is large and intractable, compatible designs are rare, and mechanical responses are sensitive to minute changes in the design. This stands in stark contrast to smooth, continuously varying input-output relations that ML algorithms are usually trained for. Several questions thus arise: to what extend can ML algorithms learn combinatorial design rules? How does a strong imbalance of the output values influence the performance? Can a ML algorithm design metamaterials with properties outside the range of the training set? To answer these questions, we train a neural network for classification of a multimodal metamaterial in chapter 3, and combine a neural network trained for regression with a genetic algorithm to design multimodal metamaterials in chapter 4.

1.4. Thesis outlook

In this thesis, we show how, using rational design and machine learning, we can classify and design multimodal combinatorial metamaterials. Our work opens up a new route for rational and data-driven design of spatially textured soft modes in multimodal metamaterials, with potential applications in programmable materials, soft robotics, and computing *in materia*. Moreover, we foresee applications beyond the field of metamaterial design to other fields that encounter similar combinatorial problems. Below follows a detailed description of each chapter's content.

In **chapter** 2, we devise a theoretical framework to keep track of kinematic constraints in a given metamaterial design. First, we provide a mathematical framework that describes the deformations of our building blocks and tilings thereof. We show that, through enforcing kinematic constraints between neighboring building blocks, we obtain a transfer matrix-like procedure that maps deformations across a design. However, there are additional local kinematic constraints that should be satisfied. Surprisingly, we show that mapping these constraints yields emergent non-local constraints on the tiling. Such constraints are unique to multimodal metamaterials. We derive these constraints explicitly for a specific type of zero mode we term strip modes. Using the results of these examples, we conjecture a set of general design rules that we verify numerically with complete agreement.

In **chapter 3**, we show that convolutional neural networks (CNNs) are able to classify with great accuracy two types of combinatorial metamaterials in rare compatible (C) and abundant incompatible (I) classes. However, due to the rarity of the C class, we cannot discern from the test set alone if the trained CNN is merely interpolating the training set or whether it has actually learned the (unknown) design rules. To answer this question, we quantify the CNN's performance over a set of random walks and compare the response of the network to the true classification. Surprisingly, we find a good agreement between the two, suggesting that the network has learned the design rules.

In **chapter 4**, we present a hybrid design strategy that combines machine learning and rational design to find designs that feature multiple desired deformations. First, we focus on designs with a high *pluripotency*, which we define to be designs that are likely to have the desired property. We use a CNN to predict the number of intensive modes—a proxy for pluripotency—and efficiently guide a genetic algorithm (GA) towards high pluripotency designs. Surprisingly, the trained CNN is able to correctly predict this number of intensive modes outside the range of the training set, which allows the GA to find ultra-rare high pluripotency designs. Subsequently, we use those designs to make a library, which we search to find a design that features the desired deformation modes. In a final step, we refine our designs to remove superfluous modes by strategically adding defects to the design. This two-step approach allows us to design 10×10 metamaterials with multiple desired spatially-textured deformations, for example, one with modes that resemble a smiley and frowny face and another with modes that resemble the letters A and U.

2 Combinatorial Design Rules

Adapted from "Emergent Nonlocal Combinatorial Design Rules for Multimodal Metamaterials" published in Physical Review E, December 2023

Combinatorial mechanical metamaterials feature spatially textured soft modes that yield exotic and useful mechanical properties. While a single soft mode often can be rationally designed by following a set of tiling rules for the building blocks of the metamaterial, it is an open question what design rules are required to realize multiple soft modes. Multimodal metamaterials would allow for advanced mechanical functionalities that can be selected on-the-fly. Here we introduce a transfer matrix-like framework to design multiple soft modes in combinatorial metamaterials composed of aperiodic tilings of building blocks. We use this framework to derive rules for multimodal designs for a specific family of building blocks. We show that such designs require a large number of degeneracies between constraints, and find precise rules on the real space configuration that allow such degeneracies. These rules are significantly more complex than the simple tiling rules that emerge for single-mode metamaterials. For the specific example studied here, they can be expressed as local rules for tiles composed of pairs of building blocks in combination with a nonlocal rule in the form of a global constraint on the type of tiles that are allowed to appear together anywhere in the configuration. This nonlocal rule is exclusive to multimodal metamaterials and exemplifies the complexity of rational design of multimode metamaterials. Our framework is a first step towards a systematic design strategy of multimodal metamaterials with spatially textured soft modes.

2.1. Introduction

The structure and proliferation of soft modes is paramount for understanding the mechanical properties of a wide variety of soft and flexible materials [2, 53–56]. Recently, computational and rational design of soft modes in designer matter has given rise to the field of mechanical metamaterials [1, 2, 39, 57–62]. Typically, such materials are structured such that a single soft mode controls the low energy deformations. Their geometric design is often based on that of a single zero-energy mode in a collection of freely hinging rigid elements [63]. Such metamaterials display a plethora of exotic properties, such as tunable energy absorption [64], programmability [3–6], self-folding [17, 65], nontrivial topology [10, 66–68] and shape-morphing [15, 69–77]. For shape-morphing in particular, a combinatorial framework was developed, where a small set of building blocks are tiled to form a metamaterial [11]. In all these examples, both the building blocks and the underlying mechanism exhibit a single zero mode, so that the metamaterial's response is dominated by a single soft mode leading to a single mechanical functionality. Often, by fixing the overall amplitude of deformation, the combinatorial design problem can be mapped to a spin-ice model [10, 11, 37] or, similarly, to Wang tilings [17, 70, 77].

In contrast, multimodal metamaterials can potentially exhibit multiple functionalities [19]. Such metamaterials host multiple complex soft modes with potentially distinct functionalities. By controlling which mode is actuated, one can tune the metamaterial's response at will. To engineer such multimodal materials, one requires precise control over the structure and enumeration of zero modes. However, as opposed to metamaterials based on building blocks with a single zero mode, the kinematics of multimodal metamaterials can no longer be captured by spin-ice or tiling problems. This is because linear combinations of zero modes are also valid zero modes such that the amplitudes of different deformation modes can take arbitrary values—such a problem can no longer be trivially mapped to a discrete tiling or spin-ice model. As a consequence, designing multimodal materials is hard. Current examples of multimodal metamaterials include those with tunable elasticity tensor and wave-function programmability [78], and tunable nonlocal elastic resonances [14]. In both works, the authors consider periodic lattices that limit the kinematic constraints between bimodal unit cells to (appropriate) boundary conditions, thereby allowing for straightforward optimization. In contrast, we aim to construct design rules for *aperiodic* multimode structures that contain a large number of simpler bimodal building blocks and that exhibit a large, but controllable number of spatially aperiodic zero modes. Such aperiodic modes allow for complex mechanical functionalities such as a strain-rate selectable auxetic response [19] and sequential energy-absorption while retaining the original stiffness [18]. For aperiodic multimode structures, the number of kinematic constraints grows with the size of the structure, so that successful designs require a large number of degeneracies between constraints. A general framework to design such zero modes is lacking.

Here, we set a first step towards such a general framework for multimodal combinatorial metamaterials. We use this framework to find emergent combinatorial tiling rules for a multimodal metamaterial based on symmetries and degenerate kinematic constraints. Strikingly, we find nonlocal rules that restrict the type of tiles that are allowed to appear together *anywhere* in the configuration. This is distinct from local tiling rules found in single-modal metamaterials which consist only of local constraints on pairs of tiles. Our work thus provides a new avenue for systematic design of spatial complexity, kinematic compatibility and multi-functionality in multimodal mechanical metamaterials.

To develop our framework, we focus on a recently introduced multimodal combinatorial metamaterial [19]. This metamaterial can host multiple complex zero modes that can be utilized to engineer functional materials. For example, a configuration of this metamaterial dressed with viscoelastic hinges allows for a strain-rate selectable auxetic response under uniaxial compression [19]. Another recent example utilizes so-called *strip modes* to efficiently absorb energy through buckling while retaining the original stiffness under sequential uniaxial compression [18]. However, the design space remains relatively unexplored and is sufficiently rich and complex that further study of this combinatorial metamaterial is warranted.

More concretely, this combinatorial metamaterial is composed of building blocks consisting of rigid bars and hinges that feature two zero modes: deformations that do not stretch any of the bars to second order of deformation [19] [Fig. 2.1(a)]. These degrees of freedom are restricted by kinematic constraints between neighboring building blocks, which in turn depend on how the blocks are tiled together. We stack these building blocks to form square $k \times k$ unit cells, and tile these periodically to form metamaterials of $n \times n$ unit cells. These metamaterials can be classified in three distinct classes based on the number of zero modes N_{ZM} as function of n: most random configurations are monomodal, due to the presence

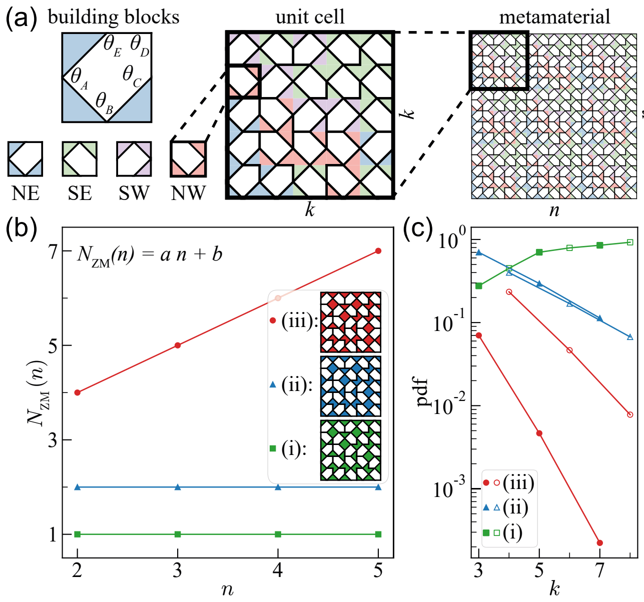


FIGURE 2.1: (a) Four differently oriented two-dimensional building blocks (left), combine into a square $k = 5$ unit cell (middle) which is tiled in a $n = 3$ grid to form a combinatorial metamaterial (right). The four orientations of the building block each have a unique color to guide the eye. The black lines represent rigid bars that hinge freely at intersections with other rigid bars. Colored regions are rigid polygons. We note that rigid pentagons with a reentrant edge are kinematically equivalent to rigid diamonds (rotated squares). (b) The number of zero modes $N_{\mathrm{ZM}}(n)$ as a function n. We distinguish between three design classes, exemplified by the three unit cells designs shown in the legend. Note that the unit cells differ only by the rotation of a single building block, yet each belongs to another class. (c) Probability density function (pdf) to find each design class through Monte Carlo sampling of the design space. Class (ii) (blue triangles) and (iii) (red circles) become exponentially more rare with increasing unit cell size k, while class (i) (green squares) becomes abundant [80]. The rate of exponential decline for class (ii) and (iii) depends on if k is odd (filled) or even (open).

of a trivial global (counter-rotating) single zero mode (Fig. 1.8) [19, 79]. However, rarer configurations can be oligomodal (constant number > 1 of zero modes) or plurimodal (number of zero modes proportional to n) [Fig. 2.1(b)].

The design space of this metamaterial was fully explored for 2×2 unit cell tilings of such building blocks [19]. For larger tilings, a brute-force calculation of the zero modes up to 8×8 reveals that this classification holds for larger unit cells (see Sec. 1.2.2). However, it is an open question how to construct design rules to determine this classification directly from the unit cell tiling without requiring costly matrix diagonalizations.

In this chapter, we focus on the specific question of obtaining tiling rules for plurimodal designs for the aforementioned building blocks. Such plurimodes drive the mechanism behind the sequential energy-absorption metamaterial [Fig. 1.7(d)] [18]. A crucial role is played by degeneracies of the kinematic constraints. These kinematic constraints follow trivially from the tiling geometry and take the form of constraints between the deformation amplitudes of adjacent building blocks. For random tilings, the kinematic constraints rapidly proliferate, leading to the single trivial mode. Checking for degeneracies between these constraints is nontrivial, as they are expressed as relations between the deformation amplitudes of different groups of building blocks. To check for degeneracies, we use a transfer matrix-like approach to map all these constraints to constraints on a small, pre-selected set, of deformation amplitudes. This allows us to establish a set of combinatorial rules. Strikingly, these combine local tiling constraints on pairs of building blocks with global constraints on the types of tiles that are allowed to appear together; hence, local information is not sufficient to identify a valid plurimodal tiling.

The structure of this paper is as follows. In Sec. 2.2 we investigate the phenomenology of this metamaterial, focusing on the number of zero modes $N_{\text{ZM}}(n)$ for unit cell sizes $3 \leq k \leq 8$. We show that random configurations are exponentially less likely to be oligomodal or plurimodal with increasing unit cell size k. Additionally, we define a mathematical representation of the building blocks' deformations that allows us to compare deformations in collections of building blocks. In Sec. 2.3 we derive a set of compatibility constraints on building block deformations that capture kinematic constraints between blocks. In Sec. 2.4 we use these constraints to formulate an exclusion rule that prohibits the structure of zero modes in collections of building blocks. Subsequently, we categorize the "allowed" mode-structures in three categories. In Sec. 2.5 we devise a mode-structure that, if supported in a unit cell, should result in a linearly growing number of zero modes, i.e., the unit cell will be plurimodal. We define a set of additional constraints on deformations localized in a strip in the unit cell that should be satisfied to support a mode with such a mode-structure. We refer to such modes as 'strip'-modes. In section 2.6 we define a transfer matrix-like formalism that maps deformation amplitudes from a column of building blocks to adjacent columns. In Sec. 2.7 we define a general framework using the transfer mappings defined in the previous section to determine if a strip of building blocks supports a strip mode of a given width W. In Sec. 2.8 we apply this framework explicitly on strips of width $1 \leq W \leq 3$ and derive a set of tiling rules for strips of each width W. Surprisingly, we find that strips of width $W = 3$ require a global constraint on the types of tiles that are allowed to appear together in the strip. Finally, we conjecture that there is a set of general design rules for strips of arbitrary width W, provide numerical proof of their validity and use them to construct a strip mode of width $W = 10$.

2.2. Phenomenology

Configuration.—We consider a family of hierarchically constructed combinatorial metamaterials [Fig. 2.1(a)] [19]. A single building block consist of three rigid triangles and two rigid bars that are flexibly linked, and its deformations can be specified by the five interior angles $\theta_A, \theta_B, \ldots, \theta_E$ that characterize the five hinges [Fig. 2.1(a)]. Each building blocks features two, linearly independent, zero energy deformations (see Sec. 1.2.2) [19]. As the undeformed building block has an outer square shape and inner pentagon shape, each building block can be oriented in four different orientations: $c = \{NE, SE, SW, NW\}$ [Fig. 2.1(a)]. We stack these building blocks to form square $k \times k$ unit cells. Identical unit cells are then periodically tiled to form metamaterials consisting of $n \times n$ unit cells; we use open boundary conditions. Each metamaterial is thus specified by the value of n and the design of the unit cell, given by the $k \times k$ set of orientations C.

Three classes.—We focus on the number of zero modes $N_{\mathrm{ZM}}(n)$ (deformations that do not cost energy up to quadratic order) for a given design. In Sec. 1.2.2, we showed that the number of zero modes is a linear function of n: $N_{\mathrm{ZM}} = an + b$, where $a \geq 0$ and $b \geq 1$ (see Fig. 2.1(b)) [79]. Based on the values of a and b, we define three design classes: Class (i): $a = 0$ and $b = 1$. For these designs, which become overwhelmingly likely for large k random unit cells [Fig. 2.1(c)], there is a single global zero mode, which we will show to be the well known counter-rotating squares (CRS) mode [2, 38, 63, 65, 69, 71, 81–84]; Class (ii): $a = 0$ and $b \geq 2$. For these rare designs, the metamaterial hosts additional zero modes that typically span the full structure, but $N_{\mathrm{ZM}}(n)$ does not grow with n; Class (iii): $a \geq 1$. For these designs the number of zero modes grows linearly with system size n, and we will show that these rare zero modes are organized along strips. Designs in class (ii) and (iii) become increasingly rare with increasing unit cell size k (see Fig. 2.1(c)). Yet, multi-functional behavior of the metamaterial requires the unit cell design to belong to class (ii) or (iii). Hence we aim to find design rules that allow to establish the class of a unit cell based on its real space configuration C and that do not require costly diagonalizations to determine $N_{\mathrm{ZM}}(n)$. Such rules will also play a role for the designs of the rare configurations in class (ii) and (iii).

As we will show, deriving such rules requires a different analytical approach than previously used to derive design rules in mechanical metamaterials [10, 11, 17, 37] The reason is for this is that each building block has two degrees of freedom yet potentially more than two nondegenerate constraints to satisfy. The problem can therefore not be mapped to a tiling problem [17, 72]. In what follows, we will define an analytic framework based on transfer-mappings and constraint-counting and use this framework to derive design rules for unit cells of class (iii).

Zero modes of building blocks.—To understand the spatial structure of zero modes, we first consider the zero energy deformations of an individual building block, irrespective of its orientation [Fig. 2.2(a)]. We can specify a zero mode m_z of a single building block in terms of the infinitesimal deformations of the angles $\theta_A, \theta_B, \ldots, \theta_E$, which we denote as $\mathrm{d}\theta_A, \mathrm{d}\theta_B, \ldots, \mathrm{d}\theta_E$, with respect to the undeformed, square configuration [Fig. 2.2(a)]. As the unit cell can be seen as a dressed five-bar linkage, it has two independent zero modes (see Sec. 1.2.2) [19, 79]. We choose a basis where one of the basis vectors correspond to the Counter-Rotating Squares (CRS) mode, where
$(\mathrm{d}\theta_A, \mathrm{d}\theta_B, \mathrm{d}\theta_C, \mathrm{d}\theta_D, \mathrm{d}\theta_E) \propto (1, -1, 1, 0, -1)$,
and the other basis vector corresponds to what we call a 'diagonal' (D) mode, where
$(\mathrm{d}\theta_A, \mathrm{d}\theta_B, \mathrm{d}\theta_C, \mathrm{d}\theta_D, \mathrm{d}\theta_E) \propto (-1, -1, 3, -4, 3)$ [Fig. 2.2(a)].
A general deformation can then be written as
$(\mathrm{d}\theta_A, \mathrm{d}\theta_B, \mathrm{d}\theta_C, \mathrm{d}\theta_D, \mathrm{d}\theta_E) = \alpha(1, -1, 1, 0, -1) + \beta(-1, -1, 3, -4, 3)$,
where α and β are the amplitudes of the CRS-mode and D-mode, respectively.

Zero modes of unit cells.—We now consider the deformations of a single building block in a fixed orientation. Hence, we can express a zero mode of an individual building block m_z as $m_z(\alpha_z, \beta_z, c_z) = \alpha_z m_{CRS} + \beta_z m_D(c_z)$. The deformation of each building block is completely determined by three degrees of freedom: the orientation c_z and the amplitudes α_z and β_z of the CRS and D mode. To compare these deformations for groups of building blocks, we now define additional notation. We use a vertex representation [19] where we map the changes in angles of the faces of the building

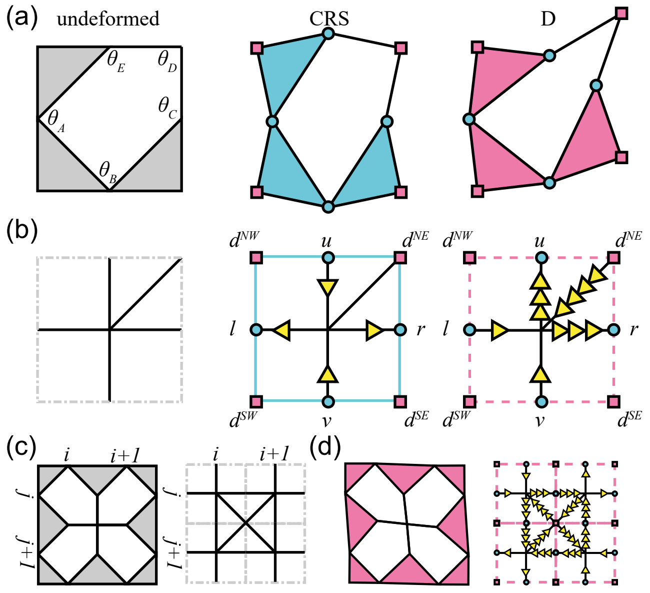


FIGURE 2.2: (a) Zero modes of the building block in orientation NE are infinitesimal deformations of the undeformed building block (left) expressed in the two basis zero modes CRS (middle) and D (right). These deformations are characterized by changes in the four angles on the faces of the block (cyan circles) and the four angles on the corners of the block (pink squares). (b) The five interior angles of the building block in orientation NE are represented by edges in the bond representation (left). We express deformations of the building block as values on these edges, which we represented as arrows. The number of arrows corresponds to the magnitude of deformation, and the direction of the arrows (incoming, outgoing) to the sign. Note that the CRS mode (middle) deforms only the angles on the faces of the building block and thus does not depend on the orientation of the building block. However, the D mode (right) does deform a diagonal edge and the mode thus depends on the orientation of the building block. (c) Building blocks are tiled together on a grid to form unit cells (left, for a 2×2 example), where the row index j increases from top to bottom and the column index i from left to right. The bond representation (right) forms the static background. (d) The static background is dressed with arrows on its bonds that represent deformations of the unit cell (left) in the vertex representation (right).

block, $\mathrm{d}\theta_A, \mathrm{d}\theta_B, \mathrm{d}\theta_C$ and $\mathrm{d}\theta_E$ to values on horizontal (l, r) and vertical (u, v) edges, and the change in angle of the corner of the building block, $\mathrm{d}\theta_D$, to the value d^c on a diagonal edge—note that the location of the diagonal edge represents the orientation, c, of each building block [Fig. 2.2(b)]. Irrespective of the orientation, we then find that a CRS mode corresponds to

[Fig. 2.2(b)]
$(u, v, l, r, d^{\mathrm{NE}}, d^{\mathrm{SE}}, d^{\mathrm{SW}}, d^{\mathrm{NW}}) \propto (-1, -1, 1, 1, 0, 0, 0, 0) = m_{CRS}$.
For a D mode, the deformation depends on the orientation; for a NE block we have [Fig. 2.2(b)]
$(u, v, l, r, d^{\mathrm{NE}}, d^{\mathrm{SE}}, d^{\mathrm{SW}}, d^{\mathrm{NW}}) \propto (3, -1, -1, 3, -4, 0, 0, 0) = m_D(\mathrm{NE})$.
We note that for a D mode in a building block with orientation c, only a single diagonal edge is nonzero. For ease of notation, we express the deformation of a building block with orientation c in shorthand (u, v, l, r, d^c), where the excluded diagonals are implied to be zero. In this notation, the D mode for a SE block is
$(u, v, l, r, d^{\mathrm{SE}}) \propto (-1, 3, -1, 3, -4) = m_D(\mathrm{SE})$,
for a SW block it is
$(u, v, l, r, d^{\mathrm{SW}}) \propto (-1, 3, 3, -1, -4) = m_D(\mathrm{SW})$,
and for a NW block it is
$(u, v, l, r, d^{\mathrm{NW}}) \propto (3, -1, 3, -1, -4) = m_D(\mathrm{NW})$.
In addition, throughout this paper we will occasionally switch to a more convenient mode basis for calculation, where the degrees of freedom of m_z are the orientation c_z and the deformations u_z and v_z.

To describe the spatial structure of zero mode deformations in a $k \times k$ unit cell, we place the building blocks on a grid and label their location as (i, j), where the column index i increases from left to right and the row index j increases from top to bottom [Fig. 2.2(c)]. We label collections of the building block zero modes $m_{i,j}(\alpha_{i,j}, \beta_{i,j}, c_{i,j})$ as $M(A, B, C)$, where A, B, and C are the collections of $\alpha_{i,j}$, $\beta_{i,j}$ and $c_{i,j}$. Such a collection $M(A, B, C)$ describes a *valid* zero mode of the collection of building blocks C if M's elements, building block zero modes $m_{i,j}$, deform compatibly with its neighbors.

2.3. Compatibility constraints

Here, we aim to derive compatibility constraints on the deformations of individual building blocks in a collection of building block C to yield a valid zero mode M (Sec. 2.2). We find three local constraints that restrict the spatial structure of such valid zero modes. First, we require compatible deformations along the faces between adjacent building blocks, and thus consider horizontal pairs (e.g., a building block at site (i, j) with neighboring building block to its right at site $(i + 1, j)$) and vertical pairs (e.g., a building block at site (i, j) with neighboring building block below at site $(i, j + 1)$)[Fig. 2.2(c)]. To be geometrically compatible, the deformations of the joint face needs to be equal, yielding

$$r_{i,j} = -l_{i+1,j}, \quad \text{and} \quad v_{i,j} = -u_{i,j+1} \tag{2.1}$$

for the 'horizontal' and 'vertical' compatibility constraints respectively. Due to the periodic tiling of the unit cells, we need to take appropriate periodic boundary conditions into account; the deformations at faces located on the open boundary of the metamaterial are unconstrained.

Second, we require the deformations at the shared corners of four building blocks to be compatible. This yields the diagonal compatibility constraint [Fig. 2.2(c)]:

$$d^{\mathrm{SE}}_{i,j} + d^{\mathrm{NE}}_{i,j+1} + d^{\mathrm{SW}}_{i+1,j} + d^{\mathrm{NW}}_{i+1,j+1} = 0. \tag{2.2}$$

We note that we again need to take appropriate periodic boundary conditions into account, and note that the deformations at corners located on the open boundary of the metamaterial are unconstrained (see App. A2.1). For compatible collective deformations in a configuration of building blocks, we require these constraints to be satisfied for all sites, with appropriate boundary conditions: either periodic or open.

2.4. Mode structure

In this section we determine an important constraint on the spatial structure of the zero modes that follows from the compatibility constraints [Eqs. (2.1) and (2.2)]. We use the compatibility constraints to derive a constraint on the mode-structure of 2×2 configurations, which in turn restricts the "allowed" spatial structures of valid zero modes M in any configuration C. To derive this constraint, we label the deformations of each building block as either CRS or D, depending on the magnitude of the D mode, $\beta_{i,j}$. We refer to building blocks with $\beta_{i,j} = 0$ as CRS blocks that deform as $m_{i,j} \propto m_{CRS}$, and to building blocks with $\beta_{i,j} \neq 0$ as D blocks. We will find that the compatibility constraints restrict the location of D and CRS blocks in zero modes.

Regardless of the unit cell configuration C, there is always a global CRS mode where all building blocks are of type CRS [19, 79]. To see this from our constraints, note that CRS blocks trivially satisfy the diagonal compatibility constraint [Eq. (2.2)], and when we take $\alpha_{i,j} = (-1)^{i+j}\alpha$, also the horizontal and vertical compatibility constraints [Eq. (2.1)]. We refer to a deformation of CRS blocks that satisfies these constraints as an area of CRS with amplitude α. Any configuration of building blocks with open boundaries supports a global area of CRS with arbitrary amplitude. Another way to see this is to note that locally, the CRS mode m_{CRS} does not depend on the building block's orientation c.

To find additional modes in a given configuration, at least one of the building blocks has to deform as type D. We now show that any valid zero mode in a 2×2 plaquette cannot contain a single D block. Consider a 2×2 configuration of building blocks with an open boundary and assume that three of the building blocks deform as CRS blocks ($\beta_{1,2} = \beta_{2,1} = \beta_{2,2} = 0$) [Fig. 2.3(a)]. These three blocks deform such that

$$u_{2,1} = -l_{1,2}. \tag{2.3}$$

However, this is incompatible with a D block at site $(1, 1)$—irrespective of its orientation, for a D block $v_{1,1} \neq -r_{1,1}$, so a D block is not compatible with three of such CRS blocks. Clearly, this argument does not depend on the specific location of the D blocks, since we are free to rotate the 2×2 configuration and did not make any assumptions about the orientations of any of the building blocks. Hence, valid zero modes in any 2×2 plaquette cannot feature a single D building block [Fig. 2.3(b)].

This implies that, first, in tilings that are at least of size 2×2, D blocks cannot occur in isolation. Second, this implies that areas of CRS must always form a rectangular shape. To see this, consider zero modes with arbitrarily shaped CRS areas and consider 2×2 plaquettes near its edge [Fig. 2.3(c)]. Any concave corner would locally feature a 2×2 plaquette with a single D block, and is thus forbidden; only straight edges and convex corners are allowed. Hence, each area of CRS must be rectangular. In general, this means that in a valid zero mode the D and CRS blocks form a pattern of rectangular patches of CRS in a background of D [Fig. 2.3(d)].

Note that our considerations above only indicate which mode structures are forbidden. However, we have found that modes can take most "allowed" shapes, including 'edge'-modes where the D blocks form a strip near the boundary, 'stripe'-modes where the D blocks form system spanning strips, and 'Swiss cheese'-modes, where a background of D blocks is speckled with rectangular areas of CRS [Fig. 2.3(d)].

We associate such modes with class (ii) or (iii) mode-scaling in unit cells. We observe that most edge-modes in a unit cell persist upon tiling of the unit cell by extending in the direction of the edge, resulting in a single larger edge-mode [Fig. 2.3(e)-left]. Swiss cheese-modes can also persist upon tiling of the unit cell by deforming compatibly with itself or

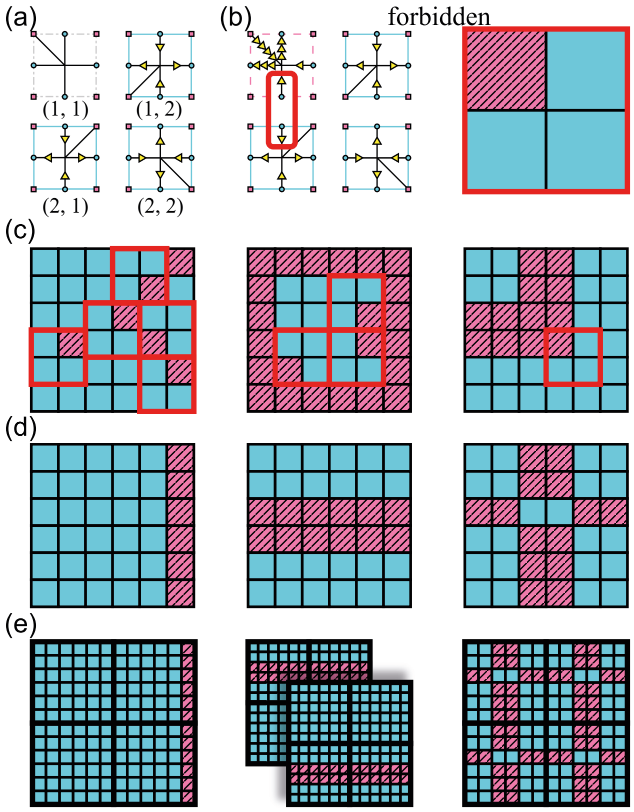


FIGURE 2.3: (a) 2×2 configuration of building blocks with open boundaries. Three building blocks deform compatibly as CRS blocks (cyan solid squares) with amplitude $\alpha = 1$, while the top left building block is undetermined (gray dash-dotted square). (b) Left: example of an invalid zero mode. The top-left building block deforms incompatibly as a D block (pink dashed square) with its CRS block neighbors (frustrated deformation is circled by thick red square). Right: we describe the structure of a mode M in CRS blocks (cyan and solid) and D blocks (pink and striped). In general, a valid zero mode cannot contain any 2×2 configurations that deform with a single D block surrounded by CRS blocks. Thus 2×2 configurations with a single D block are forbidden, which we label by a thick red square. (c) Forbidden zero mode structures for a 6×6 configuration with open boundaries. (d) Allowed zero mode structures for a 6×6 configuration with open boundaries. In App. A2.2 we show specific realizations of 'edge'-modes (left), 'stripe'-modes (middle), and 'Swiss cheese'-modes (right). (e) Zero mode structures for a 2×2 tiling of a 6×6 unit cell (thick black squares). Note that the strip of D blocks in the stripe-mode (middle) can be located in both the top and bottom row of the tiling, and therefore leads to two valid zero modes in the tiling.

another Swiss cheese-mode, creating a single larger Swiss cheese-mode [Fig. 2.3(e)-right]. Thus unit cells that support only edge-modes and Swiss cheese-modes have class (ii) mode-scaling. Moreover, we will show that a special type of stripe-mode, ‘strip’-modes, extend only along a single tiling direction, and allow for more strip modes by a translation symmetry [Fig. 2.3(e)-middle]. Here, we have found a rule on the deformations of 2×2 plaquettes of building blocks that restricts the structure of valid zero modes in larger tilings.

2.5. strip modes

We now focus on unit cells that are specifically of class (iii). We argue that a unit cell that can deform with the structure of a ‘strip’-mode is a sufficient condition for the number of modes $N_{\mathrm{ZM}}(n)$ to grow linearly with $a \geq 1$ for increasingly large $n \times n$ tilings. Here, we distinguish between stripe-modes and strip modes. We consider any zero mode that contains a deformation of non-CRS sites located in a strip enclosed by two areas of CRS a stripe-mode [Fig. 2.3(d)]. strip modes are a special case of stripe-modes: in addition to the aforementioned mode structure, we require the strip mode to deform compatibly (anti-)periodically across its lateral boundaries [Fig. 2.4(a)]. As we will show, this requirement ensures that the strip mode persists in the metamaterial upon tiling of the unit cell and in turn leads to a growing number of zero modes with n. To find rules for unit cell configuration C to support strip modes, we first in detail determine the required properties of strip modes for class (iii) mode-scaling. We then use these properties to impose additional conditions on the zero mode inside the strip of the configuration, strip conditions, and introduce a transfer matrix-based framework to find requirements on the configuration to support a strip mode.

We now consider the required properties of a strip mode for a $k \times k$ unit cell. We consider a unit cell in the center of a larger metamaterial that features a horizontal strip mode of width W [Fig. 2.4(a)]. In the strip mode, we take the areas outside the strip to deform as areas of CRS with amplitudes $\alpha = \alpha^u$ and $\alpha = \alpha^v$ for the areas above and below the strip respectively. We denote the deformation of the area inside the strip as M^{SM} and require the strip to contain at least one D block. Compatibility between our central unit cell and its neighbors requires neighboring areas of CRS to be compatible. This is easy to do, as every unit cell is free to deform with a unit cell-spanning area of CRS. Thus the unit cells above and

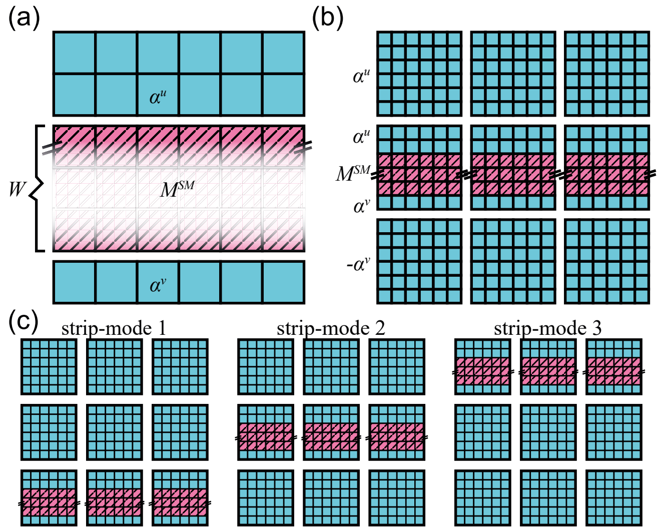


FIGURE 2.4: (a) Mode-structure of a strip mode in a 6×6 unit cell. The strip of width W deforms with strip deformation M^{SM} (pink and striped blocks) enclosed by two areas of CRS (cyan and solid blocks) above and below the strip with CRS amplitudes α^u and α^v. (b) Unit cells above and below the central unit cell deform compatibly with the strip mode as global areas of CRS. The sign of the CRS amplitudes depends on the parity of k and the size of the area of CRS above and below the strip. Unit cells to the left and right of the central unit cell deform compatibly with the strip mode as strip modes. (c) A 6×6 unit cell with a $W = 3$ strip that supports a strip mode is tiled to form a 3×3 metamaterial. This metamaterial supports a strip mode in the bottom (left), middle (middle) and top (right) rows.

below the central unit cell deform compatibly with the strip mode if they deform completely as areas of CRS with equal or staggered CRS amplitude α^u and α^v [Fig. 2.4(b)]. In addition, we require compatibility between the central unit cell and its left and right neighbors. Because the deformation in the strip M^{SM} deforms compatibly with (anti-)periodic strip conditions across its lateral boundaries, unit cells to the right and left of the central unit cell deform compatibly with the strip mode if they deform as strip modes themselves [Fig. 2.4(b)]. In an $n \times n$ tiling, all unit cells in any of the n rows deforming as strip modes is a valid zero mode in the larger metamaterial [Fig. 2.4(c)]. Therefore, we find a linearly increasing number of zero modes $N_{\text{ZM}}(n)$ for unit cells that support a strip mode.

To find conditions on unit cell configurations C to support a strip mode, we derive strip conditions from the structure of the strip mode on the strip deformation M^{SM}. Because areas of CRS are independent of the orientations of the building blocks in the area, we need only to find conditions on the configuration of building blocks in the strip C^{SM}. Without loss of generality, we focus on horizontal strip modes only. We consider a strip of building blocks C^{SM} of length k and width W and relabel the indices of our lattice such that $(i, j) = (1, 1)$ corresponds to the upper-left building block in the strip: the row index is constrained to $1 \leq i \leq k$ and the column index is constrained to $1 \leq j \leq W$. For building blocks at the top of the strip to deform compatibly with an upper CRS area we require

$$u_{i,1} = -u_{i+1,1}, \tag{2.4}$$

to hold along the entire strip. We refer to this constraint as the upper strip condition. Without loss of generality we can set $u_{i,1} = 0$ everywhere along the strip to ease computation, because we are free to add the global CRS mode with amplitude $-\alpha^u$ to the full strip mode so as to ensure that the upper deformation $u_{i,1} = 0$ for all i. Similarly, we require the building blocks at the bottom of the strip to satisfy

$$v_{i,W} = -v_{i+1,W} \tag{2.5}$$

along the entire strip. This constraint is referred to as the lower strip condition. Finally, we require the strip deformation to deform (anti-)periodically:

$$\mathbf{v}_1 = \begin{cases} (-1)^k \mathbf{v}_{k+1}, & \text{if} \quad v_{1,W} \neq 0 \\ |\mathbf{v}_{k+1}|, & \text{if} \quad v_{1,W} = 0 \end{cases} \tag{2.6}$$

where the vector $\mathbf{v}_i = (v_{i,1}, v_{i,2}, ..., v_{i,W})$ fully describes the deformation of the building blocks in column i, if all deformations in the column satisfy the vertical compatibility constraints Eq. (2.1). We refer to this condition as the periodic strip condition (PSC). We note that if the building blocks at the bottom of the strip deform as $v_{i,W} = 0$, both anti-periodic and periodic strip conditions result in a valid strip deformation.

Together with the horizontal and vertical compatibility constraints Eq. (2.1) and diagonal compatibility constraints Eq. (2.2), the strip conditions Eq. (2.4) and Eq. (2.5) allow us to check if a configuration of building blocks in strip SM can satisfy all constraints and thus allow for a strip mode.

2.6. Transfer mapping formalism

Now, we aim to derive necessary and sufficient requirements for configurations of building blocks in a strip of width W, C^{SM}, such that they allow for a valid strip deformation M^{SM}. To find such conditions, we introduce here transfer mappings that relate deformations in a column of building blocks to deformations in its neighboring columns. We will show later that these transfer mappings allow us to relate constraints and conditions on zero modes to requirements on the strip configuration.

To derive such transfer mappings, we first derive linear mappings between the pairs of degrees of freedom that characterize the zero mode m_z: the amplitudes of the CRS and D mode (α_z, β_z), the vertical edges (u_z, v_z) and horizontal edges (l_z, r_z). Subsequently, we derive a framework to construct strip modes: we fix the orientations c_z throughout the strip (C^{SM}). We first fix the (u_z, v_z) deformations for the left-most blocks in the strip [Fig. 2.5(a)]. Then, using our linear maps, we determine (l_z, r_z) for these blocks [Fig. 2.5(b)]. We use the upper strip condition [Eq. 2.4] to determine u_z of the top block in the second column, and the horizontal compatibility constraint [Eq. 2.1] to determine l_z of the second column [Fig. 2.5(c)]. Then we use a linear map to determine (v_z) of the first block in the second column, and use vertical compatibility constraint [Eq. 2.1] to determine (u_z) of the second block in the second column [Fig. 2.5(d)]. Repeating this last step, we obtain (u_z, v_z) of the second column [Figs. 2.5(e) and 2.5(f)], after which we can iterate this process to obtain (u_z, v_z, l_z, r_z) throughout the strip. While above we have worked with upper and lower vertical edges (u_z, v_z), we note that the deformations in a column follow from only the lower vertical edges v_z in a column of building blocks $\mathbf{v}_i$, where u_z follows from applying the vertical compatibility constraint [Eq. (2.1)]. Thus, the deformation of building blocks in column $i+1$ is fully determined by the deformation in column i by satisfying the vertical and horizontal compatibility constraints and the upper strip condition.

We refer to the linear mappings relating the deformations of column i, $\mathbf{v}_i$, to the deformations in adjacent column $i+1$, $\mathbf{v}_{i+1}$, as a linear transfer mapping $T(\mathbf{c}_i, \mathbf{c}_{i+1})$ which depends on the orientations of the building blocks in the two columns. Thus, by iterating this relation, the strip deformation is determined entirely by the deformations $\mathbf{v}_1$ of the left-most column.

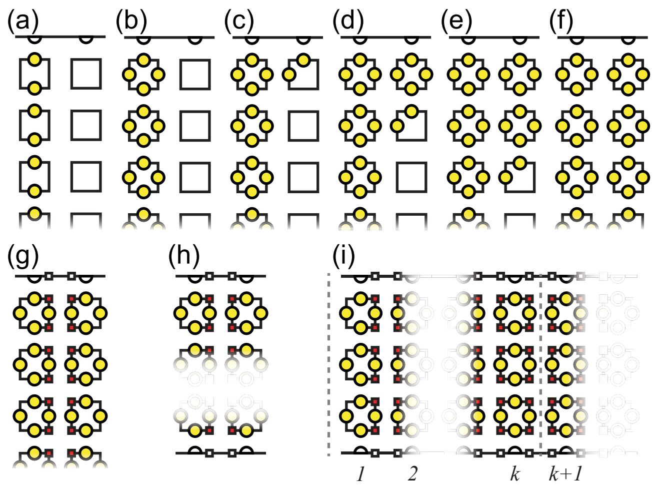


FIGURE 2.5: (a-f) Step-wise schematic illustration of our transfer mapping of deformations in a column of building blocks in the strip (white squares) to the next column in the strip, see main text. Yellow circles (light gray) indicate known deformations of the building blocks, the upper white half-circles represent the upper strip condition [Eq. 2.4]. (g-i) Schematic illustration of the constraints and conditions on the strip deformation, see main text. Red squares (dark gray) indicate known diagonal deformation d_z of the building blocks, the lower white half-circles represent the lower strip condition [Eq. (2.5)], and the lower numbers enumerate the columns for a strip of length k.

2.6.1. Linear degree of freedom transformations

To derive these transfer mappings, we require linear mappings between the pairs of degrees of freedom that characterize the zero mode m_z. For given set of orientations $\{c_z\}$, we derive linear mappings from the mode-amplitudes (α_z, β_z) to vertical edges (u_z, v_z) to horizontal edges (l_z, r_z) and find that they all are nonsingular—this implies that any of these pairs fully characterizes the local soft mode m_z.

First, we define Λ as

$$\begin{pmatrix} u_z \\ v_z \end{pmatrix} = \Lambda(c_z) \begin{pmatrix} \alpha_z \\ \beta_z \end{pmatrix} . \tag{2.7}$$

Subsequently, we express (l_z, r_z) in terms of (u_z, v_z) as

$$\begin{pmatrix} l_z \\ r_z \end{pmatrix} = \Gamma(c_z) \begin{pmatrix} \alpha_z \\ \beta_z \end{pmatrix} = \Gamma(c_z)\Lambda^{-1}(c_z) \begin{pmatrix} u_z \\ v_z \end{pmatrix} . \tag{2.8}$$

Explicit expressions for the 2×2 matrices Λ and Γ are given in App. A2.3. Finally, we rewrite this equation as (see Table. 2.1):

$$\begin{pmatrix} l_z \\ r_z \end{pmatrix} = \begin{pmatrix} L_u(c_z) & L_v(c_z) \\ R_u(c_z) & R_v(c_z) \end{pmatrix} \begin{pmatrix} u_z \\ v_z \end{pmatrix} . \tag{2.9}$$

Similarly, we can express the diagonal edge d_z^o at orientation o in terms of (u_z, v_z) as (see App. A2.3)

$$d_z^o = D^o(c_z)(-u_z + v_z) , \tag{2.10}$$

where the coefficients $D^o(c_z)$ are given in Table 2.1 for all orientations $o = \{\mathrm{NE}, \mathrm{SE}, \mathrm{SW}, \mathrm{NW}\}$. We note that for CRS blocks where $u_z = v_z$ this equation immediately gives $d_z^o = 0$ for all orientations o. Together, Eqs. (2.9) and (2.10) allow to express all building block deformations as linear combinations of the vertical deformations (u_z, v_z).

2.7. Constraints and Symmetries

Here, we define a general framework based on transfer-mappings and constraint-counting to determine if a given (strip) configuration C^{SM} supports a valid strip mode M^{SM}. The strip deformation $\mathbf{v}_1$ describes a valid strip mode only if it leads to a deformation which satisfies the diagonal compatibility constraints [Eq. 2.2] [Fig. 2.5(g)], the lower strip conditions [Eq. 2.5] [Fig. 2.5(h)] and the periodic strip condition [Eq. 2.6] [Fig. 2.5(i)] everywhere along the strip. To determine if these constraints are satisfied

TABLE 2.1: Values for the coefficients L_u, L_v, R_u, R_v for the (u_z, v_z) to (l_z, r_z) mapping [Eq. (2.9)] and the coefficient D^o for the (u_z, v_z) mapping to d_z^o for a building block of orientation $c_z = \{\mathrm{NE}, \mathrm{SE}, \mathrm{SW}, \mathrm{NW}\}$ [Eq. (2.10)].

	NE	SE	SW	NW
L_u	-1/2	-1/2	-3/2	1/2
L_v	-1/2	-1/2	1/2	-3/2
R_u	1/2	-3/2	-1/2	-1/2
R_v	-3/2	1/2	-1/2	-1/2
D^{NE}	1	0	0	0
D^{SE}	0	-1	0	0
D^{SW}	0	0	-1	0
D^{NW}	0	0	0	1

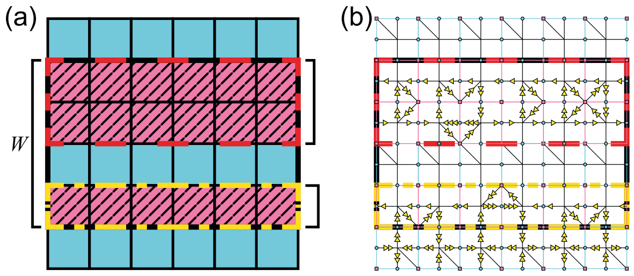


FIGURE 2.6: (a) A seemingly valid strip deformation of width $W = 4$ (thick black solid line) can be decomposed into two strips of smaller widths (thick, red dashed and yellow dash-dotted lines) if it does not satisfy the CC and NT conditions. (b) Realization of a $W = 4$ strip deformation (thick black solid line) that does not satisfy the CC and NT conditions: it can be decomposed into $W' = 2$ (enclosed in thick red dashed line) and $W' = 1$ (thick yellow dash-dotted line) strips that individually satisfy the NT and CC conditions.

by the deformation $\mathbf{v}_1$, we use the transfer mapping to map all the constraints throughout the strip to constraints on $\mathbf{v}_1$. Since each additional column yields additional constraints, we obtain a large set of constraints on $\mathbf{v}_1$, and without symmetries and degeneracies, one does not expect to find nontrivial deformations which satisfy all these constraints. However, for appropriately chosen orientations of the building blocks, many constraints are degenerate, due to the underlying symmetries. Hence, we can now formulate two conditions for obtaining a nontrivial strip mode of width W.

First, after mapping all the constraints in the strip to constraints on $\mathbf{v}_1$, and after removing redundant constraints, the number of nondegenerate constraints should equal $W - 1$ so that the strip configuration contains a single non-CRS floppy mode. We refer to this condition as the constraint counting (CC) condition. Second, we focus on irreducible strip modes of width W, and exclude strip deformations composed of strip modes of smaller width or rows of CRS blocks [Fig. 2.6(a)]. Such reducible strip deformations not only satisfy all constraints in a strip of width W, but also in an encompassing strip of width $W' < W$ [Fig. 2.6(b)]. Irreducible strip modes of width W do not satisfy all constraints for any encompassing strips of width $W' < W$. We refer to this condition as the nontrivial (NT) condition as it excludes rows of CRS from the strip mode, which are trivial solutions to the imposed constraints. Valid strip modes are those that satisfy both CC and NT conditions.

To map all constraints to $\mathbf{v}_1$, we use the linear mapping between the diagonal edge d_z and (u_z, v_z) [Eq. (2.10)] such that the diagonal compatibility constraints [Eq. (2.2)] can be expressed in v_z. The diagonal compatibility constraints, lower strip conditions [Eq. (2.5)] and periodic strip condition [Eq. (2.6)] can all be expressed in v_z and then be mapped to $\mathbf{v}_1$ by iteratively applying the set of transfer mappings $\{T(\mathbf{c}_i, \mathbf{c}_{i+1})\}$.

This constraint mapping method allows us to systematically determine if a given strip configuration C^{SM} supports a valid strip mode M^{SM}:

1. Determine the set of transfer matrices $\{T(\mathbf{c}_i, \mathbf{c}_{i+1}))\}$.
2. Express the diagonal compatibility constraints [Eq. (2.2)], lower strip conditions [Eq. (2.5)] and periodic strip condition [Eq. (2.6)] in terms of $\{\mathbf{v}_i\}$.
3. Map the set of all constraints to constraints on $\mathbf{v}_1$ using the transfer matrices.
4. Check if the CC and NT conditions are satisfied on $\mathbf{v}_1$.

In what follows, we consider the transfer mappings and constraints explicitly for strips of widths up to $W = 3$ and derive geometric necessary and sufficient rules for the orientations c_z of the building blocks to satisfy the CC and NT conditions. Finally, we consider strips of even larger width W and construct sufficient requirements on strip configurations.

2.8. Deriving rules for strip modes

Here we aim to derive design rules for strip modes. We first derive necessary and sufficient conditions on strip configurations C^{SM} of widths up to $W = 3$. Then, we use those requirements to conjecture a set of general rules for strips of arbitrary widths. We provide numerical proof that these rules are correct and use them to generate a $W = 10$ example that we would not have been able to find through Monte Carlo sampling of the design space.

2.8.1. Case 1: $W = 1$

We now derive necessary and sufficient conditions on the orientations of the building blocks for strip modes of width $W = 1$ to appear [Fig. 2.7(a)]. We show that a simple pairing rule for the orientations of neighboring building blocks gives necessary and sufficient conditions for such a configuration to support a valid strip mode, i.e., a strip deformation that satisfies the horizontal compatibility constraints [Eq. (2.1)], the diagonal compatibility

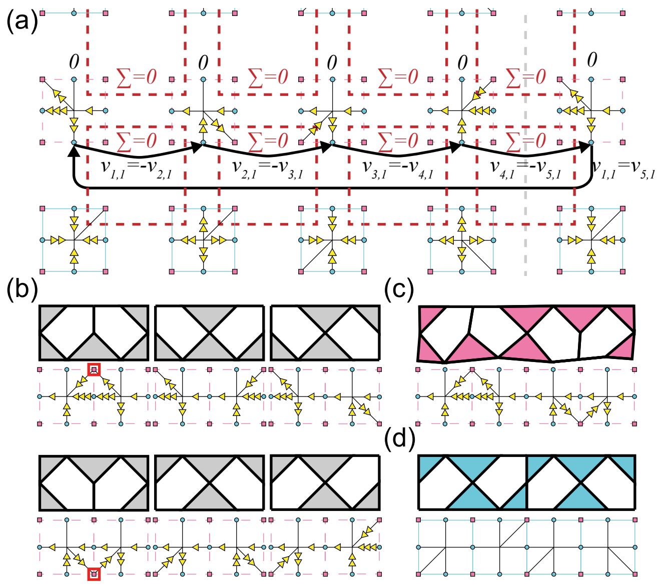


FIGURE 2.7: (a) Schematic representation of the degrees of freedom, constraints, and mapping for a $W = 1$ strip mode of length $k = 4$ in the vertex representation. The building blocks in the strip and lower CRS area are highlighted with pink (dashed) and blue (solid) boxes; the upper CRS area has amplitude zero. Applying the horizontal compatibility constraint and upper strip condition leads to a mapping from $v_{i,1}$ to $v_{i+1,1}$. We show the deformation of each building block in the strip for such a mapping with $v_{1,1} = 2$. The diagonal compatibility constraints are indicated by $\sum = 0$ in thick red dashed boxes and are all satisfied by the mapping. The lower strip condition ($v_{i,1} = -v_{i+1,1}$, arrows) and periodic strip condition ($v_{1,1} = v_{5,1}$, long arrow) are also satisfied by the mapping. The strip therefore deforms compatibly with the lower CRS area with amplitude two. (b) The six h-pairs of horizontally adjacent building blocks $(c_{i,1}, c_{i+1,1})$ that satisfy Eq. (2.12) and examples of their deformations in vertex representation obtained from the map [Eq. (2.11)] with $v_{i,1} = 2$. Note that $d^{NE}_{i,1} = -d^{NW}_{i+1,1}$ and $d^{SE}_{i,1} = -d^{SW}_{i+1,1}$ are satisfied either trivially or by the transfer mapping [Eq. (2.11)] (corner nodes highlighted with thick red squares) for all h-pairs. (c) Example of a $k = 4$ strip configuration (top) deformed as a valid strip mode M^{SM} (bottom, vertex representation) that satisfies all compatibility constraints and strip conditions. (d) Example of a $k = 4$ strip configuration (top) that can only satisfy all compatibility constraints and strip conditions by not deforming (bottom, vertex representation).

constraints [Eq. (2.2)], the upper strip conditions [Eq. (2.4)], the lower strip conditions [Eq. (2.5)], and the periodic strip condition [Eq. (2.6)[] (see Fig. 2.7(a)) in addition to the constraint counting (CC) and nontrivial (NT) conditions.

First, we derive the transfer mapping that maps the deformations of building block $(i, 1)$ to block $(i + 1, 1)$ for general orientations $(c_{i,1}, c_{i+1,1})$. Without loss of generality, we set the amplitude $\alpha^u = 0$ such that $u_{i,1} = 0$ everywhere along the strip—this trivially satisfies the upper strip condition [Eq. (2.4)] (recall that we can always do this by adding a global CRS deformation of appropriate amplitude to a given mode). The deformations of each building block are now completely determined by choosing $v_{i,1}$. However, these cannot be chosen independently due to the various constraints. Implementing the horizontal compatibility constraints and upper strip condition, we find that the $v_{i,1}$ in adjacent blocks are related via a linear mapping (see App. A2.4):

$$v_{i+1,1} = -\frac{R_v(c_{i,1})}{L_v(c_{i+1,1})} v_{i,1} \ , \tag{2.11}$$

where the values of $R_v(c)$ and $L_v(c)$ are given in Table 2.1. We interpret this mapping as a simple (scalar) version of a transfer mapping (see Fig. 2.7(a)). The idea is then that, by choosing $v_{1,1}$ and iterating the map [Eq. (2.11)], we determine a strip deformation which satisfies both the upper strip conditions and horizontal compatibility constraints. The goal is to find values for the orientations $c_{i,1}$ that produce a *valid* strip mode, i.e., a deformation which also satisfies the diagonal compatibility constraints [Eq. (2.2), red dashed boxes in Fig. 2.7(a)], lower strip conditions [Eq. (2.5), black arrows in Fig. 2.7(a)], periodic strip condition [Eq. (2.6), long black arrow in Fig. 2.7(a)], and CC and NT conditions—note that if we take $v_{1,1} = 0$, all deformations throughout the unit cell are zero and we have simply obtained a zero amplitude CRS mode, which is not a valid strip mode [see example in Fig. 2.7(d)].

To construct configurations that produce a valid strip mode, we first consider an example. In this example, we only consider orientations $(c_{i,1}, c_{i+1,1})$ that satisfy

$$R_v(c_{i,1}) = L_v(c_{i+1,1}) \ , \tag{2.12}$$

and show that this is a sufficient condition to produce a valid strip mode. We refer to the six pairs $(c_{i,1}, c_{i+1,1})$ that satisfy condition Eq. (2.12) as h-pairs (for horizontal) [Fig. 2.7(b)].

We find that configurations consisting only of h-pairs satisfy the lower and periodic strip conditions and diagonal compatibility constraints. Specifically, we find the following for h-pairs

(1) the map Eq. (2.11) simplifies to $v_{i+1,1} = -v_{i,1}$ and thus directly satisfies the lower strip condition [Eq. (2.5)] and periodic strip condition

[Eq. (2.6)] by iterating the map, see deformations in Fig. 2.7(b).

(2) the diagonal compatibility constraints are either trivially satisfied or the same as the map Eq. (2.11) and thus impose no constraints on $v_{i,1}$. To see this, note that the diagonal compatibility constraint [Eq. (2.2)] is required to be satisfied at all corner nodes in the strip (pink squares in Fig. 2.7(a)). Note that away from the strip, all diagonals are zero (recall that a CRS block always has $d^c = 0$). Thus, the diagonal compatibility constraint at the corner nodes shared between two building blocks in a pair simplifies to $d^{NE}_{i,1} = d^{NW}_{i+1,1}$ and $d^{SE}_{i,1} = d^{SW}_{i+1,1}$ (see Fig. 2.7(a)). For the six h-pairs, there are four pairs where all diagonals in the constraints are zero, i.e., trivially satisfied, and two pairs where the diagonals are nonzero (highlighted in red in Fig. 2.7(b)). For the latter case, the diagonal compatibility constraint implies that $v_{i,1} = -v_{i+1,1}$—this follows from $u_{i,1} = 0$ and the mapping [Eq. (2.10)]—which is the same as the map Eq. (2.11).

Thus, all conditions and constraints are trivially satisfied for strip configurations consisting only of h-pairs, see Fig. 2.7(c) for an example.

Such strip configurations thus impose no constraints on $\mathbf{v}_1$, thereby satisfying the constraint counting (CC) condition. Additionally, such configurations satisfy the nontrivial (NT) condition as well so long as $v_{1,1} \neq 0$. Hence, the pairing rule

(i) Every pair of horizontally adjacent building blocks in the strip must be an h-pair.

is a sufficient condition to obtain valid $W = 1$ strip modes, and thus class (iii) mode scaling. It is also a necessary condition, because any pair that does not satisfy condition Eq. (2.12) does not trivially satisfy the lower strip condition [Eq. (2.5)], breaking the CC condition, and thus only satisfies all compatibility constraints and strip conditions of a strip mode for $v_{1,1} = 0$, breaking the NT condition, see Fig. 2.7(d) for an example. Concretely, when $u_{1,1}$ and $v_{1,1}$ are both zero, the whole deformation is zero which is not a valid strip mode but rather a zero amplitude CRS mode. Hence, the pairing rule (i) is a necessary and sufficient condition to obtain $W = 1$ strip modes.

2.8.2. Case 2: $W = 2$

Now, we consider strips of width $W = 2$. strip deformations in such strips have an additional degree of freedom, $v_{i,2}$, compared to strips of width

$W = 1$. To result in a valid strip mode there must be one constraint on the strip deformation $\mathbf{v}_1$ to satisfy the constraint counting (CC) condition. We show that a simple adjustment and addition to the pairing rule results in a sufficient and necessary condition to obtain $W = 2$ strip modes.

First, we extend our transfer mapping to account for the extra row of building blocks in the strip. We again set the amplitude $\alpha^u = 0$, so that the deformations of column i are completely determined by fixing vector $\mathbf{v}_i = (v_{i,1}, v_{i,2})$ [Fig. 2.5]. We now aim to obtain a complete map from $\mathbf{v}_i$ to $\mathbf{v}_{i+1}$. Note that the map for $v_{i+1,1}$ does not depend on the extra row of building blocks and therefore follows the map [Eq. (2.11)] derived for $W = 1$ strip modes. To obtain a map for $v_{i+1,2}$, we note that for the building blocks in column $i + 1$ to deform compatibly, we require the vertical compatibility constraint [Eq. (2.1)] to be satisfied [Fig. 2.5]. Then, by implementing the horizontal and vertical compatibility constraints, we find a linear mapping for $v_{i+1,2}$ which depends on both $v_{i,1}$ and $v_{i,2}$ (see App. A2.4):

$$\begin{aligned} v_{i+1,2} &= \frac{L_u(c_{i+1,2})}{L_v(c_{i+1,2})}\left(\frac{R_u(c_{i,2})}{L_u(c_{i+1,2})} - \frac{R_v(c_{i,1})}{L_v(c_{i+1,1})}\right) v_{i,1} \\ &\quad - \frac{R_v(c_{i,2})}{L_v(c_{i+1,2})} v_{i,2} \ . \end{aligned} \tag{2.13}$$

Together, Eq. (2.11) and Eq. (2.13) form the transfer mapping from $\mathbf{v}_i$ to $\mathbf{v}_{i+1}$, which we capture compactly as $\mathbf{v}_{i+1} = T(\mathbf{c}_i, \mathbf{c}_{i+1})\mathbf{v}_i$ (see Fig. 2.8(a) for a schematic representation), where :

$$T(\mathbf{c}_i, \mathbf{c}_{i+1}) = \begin{pmatrix} -\frac{R_v(c_{i,1})}{L_v(c_{i+1,1})} & 0 \\ \frac{L_u(c_{i+1,2})}{L_v(c_{i+1,2})}\left(\frac{R_u(c_{i,2})}{L_u(c_{i+1,2})} - \frac{R_v(c_{i,1})}{L_v(c_{i+1,1})}\right) & -\frac{R_v(c_{i,2})}{L_v(c_{i+1,2})} \end{pmatrix} . \tag{2.14}$$

Note that $T(\mathbf{c}_i, \mathbf{c}_{i+1})$ is a lower-triangular transfer matrix which depends only on the orientations $\mathbf{c}_i = (c_{i,1}, c_{i,2})$ of column i and column $i + 1$.

Now, we want to find values for the orientations $\mathbf{c}_i$ that produce a valid strip mode, i.e., a deformation which satisfies all constraints: the diagonal compatibility constraints [Eq. (2.2)], the lower strip condition [Eq. (2.5)] and periodic strip condition [Eq. (2.6)]. Additionally, the strip deformation $\mathbf{v}_1$ should satisfy the CC and NT conditions. We note that $\mathbf{v}_1 = \mathbf{0}$ corresponds to the strip deforming as an area of CRS [Fig. 2.8(b)-i]. Additionally, $v_{1,1} = 0$ while $v_{1,2} \neq 0$ corresponds to the top row deforming as an area of CRS with zero amplitude [Fig. 2.8(b)-ii] and $v_{1,1} = -v_{1,2}$ corresponds to the bottom row deforming as an area of CRS with arbitrary amplitude

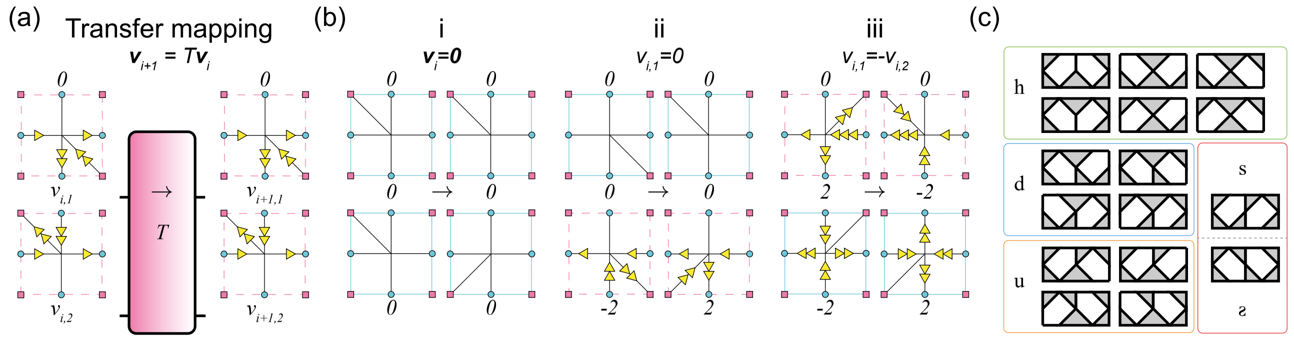


FIGURE 2.8: (a) The transfer matrix $T(\mathbf{c}_i, \mathbf{c}_{i+1})$ maps the displacements $\mathbf{v}_i = (v_{i,1}, v_{i,2})$ of the building blocks in column i, $\mathbf{c}_i$, to the displacements $\mathbf{v}_{i+1}$ of the building blocks in column $i+1$, $\mathbf{c}_{i+1}$, indicated by $\rightarrow$. (b) Three constraints on the strip deformation $\mathbf{v}_i$ that break the nontrivial (NT) condition in any 2×2 strip configuration. The strip deformation can only satisfy the constraint by deforming rows of the strip as blocks of CRS (solid cyan); either both rows (i), the top row (ii) or the bottom row (iii). (c) The 16 possible pairs of horizontally adjacent building blocks can be divided in four categories: horizontal (h, green), down (d, blue), up (u, orange), and vertical (s or ƨ, red).

[Fig. 2.8(b)-iii, see App. A2.5]. All these cases break the nontrivial (NT) condition as they describe strip deformations completely or in-part composed of rows of CRS blocks and thus do not represent valid $W = 2$ strip modes. We exclude these configurations.

To construct valid strip configurations, we consider 2×2 configurations of building blocks $(\mathbf{c}_i, \mathbf{c}_{i+1})$. We compose such 2×2 configurations by vertically stacking pairs of horizontally adjacent building blocks $(c_{i,1}, c_{i+1,1})$ and $(c_{i,2}, c_{i+1,2})$ for the top row and bottom row. There are 16 different pairs, and we note these can be grouped in four categories, depending on the corresponding values of R_u, R_v, L_u and L_v (Table 2.1):

$$\mathrm{h-pairs:} \qquad \frac{R_u(c_{i,j})}{L_u(c_{i+1,j})} = \frac{R_v(c_{i,j})}{L_v(c_{i+1,j})} = 1 \ , \tag{2.15}$$

$$\mathrm{u-pairs:} \qquad \frac{R_u(c_{i,j})}{L_u(c_{i+1,j})} = -1 \ , \tag{2.16}$$

$$\mathrm{d-pairs:} \qquad \frac{R_v(c_{i,j})}{L_v(c_{i+1,j})} = -1 \ , \tag{2.17}$$

$$\mathrm{s-pairs:} \qquad \frac{R_u(c_{i,j})}{L_v(c_{i+1,j})} = \frac{R_v(c_{i,j})}{L_u(c_{i+1,j})} = 1 \ . \tag{2.18}$$

Each of the sixteen possible pairs satisfy only one of these conditions [Fig. 2.8(c)]. We denote groups of 2×2 configurations as vertical stacks of such pairs, e.g., a (d, u)-pair obeys the condition for d-pairs [Eq. (2.17)] for $(c_{i,1}, c_{i+1,1})$ and the condition for u-pairs [Eq. (2.16)] for $(c_{i,2}, c_{i+1,2})$; see Figs. 2.9(a) and 2.9(c) for examples of (d, u)-pairs.

By stacking pairs, there are 16^2 possible 2×2 configurations. We now show that (d, u)-pairs and (h, h)-pairs are the only 2×2 configurations that make up strip configurations that support valid $W = 2$ strip modes. First, we will show that a strip composed only of (d, u)-pairs results in a valid strip mode. Second, we show that a strip composed only of (h, h)-pairs does not result in a single $W = 2$ strip mode, but in two $W = 1$ strip modes, breaking the CC condition. Finally, we show that combining (h, h)-pairs and (d, u)-pairs in a strip configuration results in a valid $W = 2$ strip mode.

First, we consider (d, u)-pairs and show that these satisfy all conditions for a valid strip mode, provided that a single constraint on $\mathbf{v}_i$ is satisfied. First, from Eq. (2.16) and Eq. (2.17) we see that such pairs satisfy the condition

$$\frac{R_u(c_{i,2})}{L_u(c_{i+1,2})} = \frac{R_v(c_{i,1})}{L_v(c_{i+1,1})} \ , \tag{2.19}$$

which implies that the transfer matrix $T((\mathrm{d}, \mathrm{u}))$ [Eq. (2.14)] is purely diagonal. The map [Eq. (2.13)] from $v_{i,2}$ to $v_{i+1,2}$ is thus independent of $v_{i,1}$. We now show that the choice $\mathbf{v}_1 = (v_{1,1}, 0)$, which satisfies the constraint

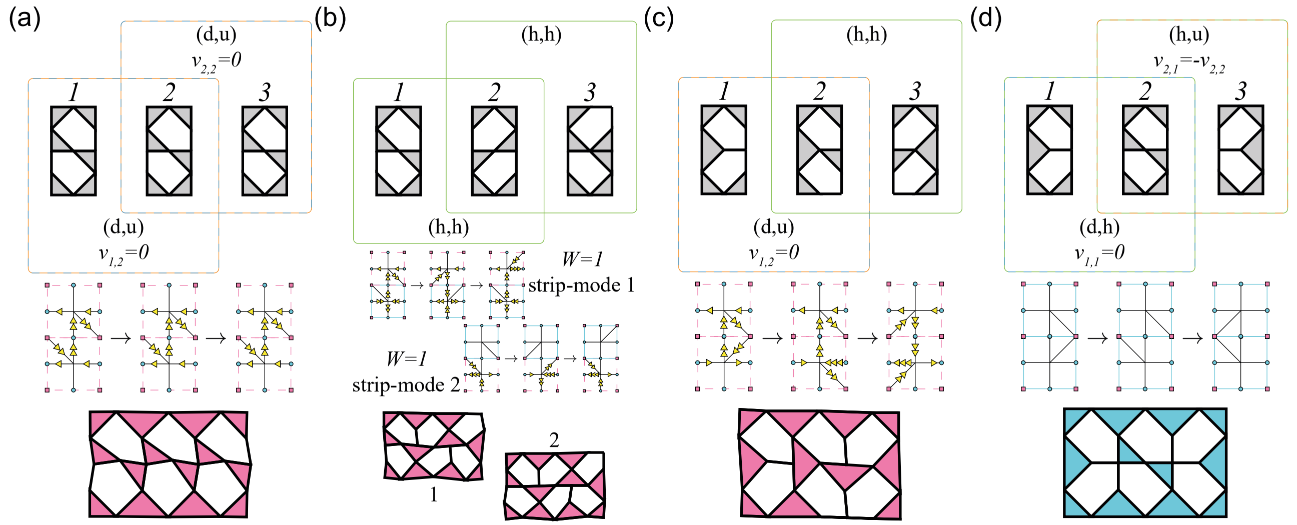


FIGURE 2.9: (d) A strip configuration (top) consisting solely of (d, u)-pairs can deform as a valid $W = 2$ strip mode. A realization of valid strip mode with $\mathbf{v}_1 = (-2, 0)$ is shown in vertex representation (middle) and schematic representation (bottom). (e) A strip configuration (top) consisting solely of (h, h)-pairs supports two $W = 1$ strip modes (middle: vertex representation, bottom: schematic representation), as (h, h)-pairs impose no constraint on the strip deformation thereby breaking the constraint counting (CC) condition. Note that only part of the strip is shown; a strip consisting only of (h, h)-pairs must always have an even strip length k. (f) A strip configuration (top) consisting of a (d, u)-pair and (h, h)-pair. The (d, u)-pair imposes the constraint $v_{1,2} = 0$ on the strip deformation $\mathbf{v}_1$. A realization of a valid strip mode with $\mathbf{v}_1 = (-2, 0)$ is shown in vertex representation (middle) and schematic representation (bottom). (g) An invalid strip configuration (top), consisting of a (d, h)- and (h, u)-pair. The constraints imposed on the strip deformation, $v_{1,1} = 0$ and $v_{2,1} = -v_{2,2}$, result in the strip being unable to deform, i.e., $\mathbf{v}_1 = (0, 0)$ (middle: vertex representation, bottom: schematic representation).

$v_{1,2} = 0$, produces a valid strip mode for $v_{1,1} \neq 0$, see Fig. 2.9(a) for an example strip deformation. This choice clearly satisfies the lower strip condition [Eq. (2.5)]. Moreover, the diagonal compatibility constraints [Eq. (2.2)] on corner nodes between the two columns i and $i+1$ are also satisfied by the constraint $v_{i,2} = 0$, regardless of the precise orientations of the building blocks as can be shown (see App. A2.6.1). Finally, by iterating the transfer map [Eq. (2.14)] for a strip that consists only of (d, u)-pairs, we find that $v_{1,1} = v_{k+1,1}$ and $v_{1,2} = v_{k+1,2} = 0$, i.e., the periodic strip condition [Eq. (2.6)] is satisfied. Thus, a strip consisting only of (d, u)-pairs satisfies all constraints in the strip by imposing a single constraint on $\mathbf{v}_1$, satisfying the CC condition, and satisfies the NT condition so long as $v_{1,1} \neq 0$. The resulting strip deformation is characterized by the choices of $c_{i,j}$ and $\mathbf{v}_1 = (v_{1,1}, 0)$.

Second, we consider (h, h)-pairs and show that, while satisfying the diagonal compatibility constraints [Eq. (2.2)], lower strip conditions [Eq. (2.5)] and periodic strip conditions [Eq. (2.6)], they in fact lead to two adjacent $W = 1$ strip modes, breaking the CC condition. Using Eq. (2.15) and the definition of the transfer matrix, we find that $T((\mathrm{h}, \mathrm{h})) = -I$, where I is the identity matrix. Thus, (h, h)-pairs trivially satisfy the lower strip condition and diagonal compatibility constraints (see App. A2.7, see Fig. 2.9(b) for examples of strip deformations). Additionally, a strip that consists only of (h, h)-pairs maps $v_{1,j} = (-1)^k v_{k+1,j}$ by iterating the transfer mapping [Eq. (2.14)] and thus satisfies the periodic strip condition. However, a strip which consists only of (h, h)-pairs does not place any constraints on $\mathbf{v}_1$ and retains the two degrees of freedom that each can describe valid $W = 1$ strip modes [Fig. 2.9(b)], breaking the CC condition. Thus, a strip composed only of (h, h)-pairs does not support one $W = 2$ strip mode, but two $W = 1$ strip modes.

We now consider combining (h, h)-pairs and (d, u)-pairs in a single strip and show that such a strip supports a valid $W = 2$ strip mode. We note that for both pairs, the transfer matrix [Eq. (2.14)] is diagonal. Thus, the constraint from a (d, u)-pair anywhere in the strip, $v_{i,2} = 0$, to satisfy the diagonal compatibility constraints [Eq. (2.2)] and lower strip condition [Eq. (2.5)] locally maps to the constraint $v_{1,2} = 0$ on $\mathbf{v}_1$. Both (h, h)-pairs and (d, u)-pairs satisfy the diagonal compatibility constraints and lower strip condition locally with this constraint, see Fig. 2.9(c) for an example strip deformation. To result in valid strip mode, we also require the periodic strip condition [Eq. (2.6)] to be satisfied. We find that $v_{1,2} = v_{k+1,2} = 0$ and $v_{1,1} = (-1)^{\mathrm{No.(h,h)}} v_{k+1,1}$, where $\mathrm{No.(h, h)}$ is the number of (h, h)-pairs in the strip with periodic boundary conditions, thereby satisfying the periodic strip condition [Eq. (2.6)].

Thus, a strip that consists of any number of (h, h)-pairs and at least one (d, u)-pair satisfies all constraints as well as the CC and NT conditions when $\mathbf{v}_1 = (v_{1,1}, 0)$ with $v_{1,1} \neq 0$, thereby resulting in a valid $W = 2$ strip mode. Hence, the pairing rules for configurations that support valid $W = 2$ strip modes are the following:

(i) Every 2×2 configuration of building blocks in the strip must be an (h, h)-pair or (d, u)-pair.

(ii) There must be at least a single (d, u)-pair in the strip.

These are sufficient conditions to obtain $W = 2$ strip modes. They can also be shown to be necessary conditions, because any pair that is not a (h, h)-pair or (d, u)-pair constrains the strip deformation $\mathbf{v}_1$ to $v_{1,1} = 0$, or $v_{1,1} = -v_{1,2}$, or both (see App. A2.6.1), thereby breaking the nontrivial (NT) condition and therefore does not result in a valid $W = 2$ strip mode [Fig. 2.9(d)]. Hence, these pairing rules are necessary and sufficient conditions on the strip configuration to obtain $W = 2$ strip modes.

2.8.3. Case 3: $W = 3$

Finally, we consider strips of width $W = 3$. We show that in addition to simple adjustments to the pairing rules, we require an additional rule restricting the ordering of pairs in the strip configuration. This ordering rule highlights that the problem of constructing configurations that support valid strip modes is not reducible to a tiling problem which relies on nearest-neighbor interactions, but rather requires information of the entire strip configuration. This is surprising, as these rules emerge from local compatibility constraints. The new set of rules that we obtain are necessary and sufficient conditions to obtain $W = 3$ strip modes.

First, we extend our transfer mapping to account for the extra row of building blocks in the strip. As in the previous two cases, we set the amplitude $\alpha^u = 0$ such that the deformations of column i are completely determined by fixing vector $\mathbf{v}_i = (v_{i,1}, v_{i,2}, v_{i,3})$. We again want to obtain a complete map from $\mathbf{v}_i$ to $\mathbf{v}_{i+1}$. The maps for $v_{i+1,1}$ and $v_{i+1,2}$ do not depend on the extra row of building blocks and therefore follow Eq. (2.11) and Eq. (2.13) respectively. To obtain a map for $v_{i,3}$, we implement the horizontal and vertical compatibility constraints [Eq. (2.1)] and find a linear mapping for $v_{i+1,3}$ (see App. A2.4):

$$\begin{aligned} v_{i+1,3} &= \frac{L_u(c_{i+1,2})}{L_v(c_{i+1,2})}\frac{L_u(c_{i+1,3})}{L_v(c_{i+1,3})}\left(\frac{R_u(c_{i,2})}{L_u(c_{i+1,2})} - \frac{R_v(c_{i,1})}{L_v(c_{i+1,1})}\right) v_{i,1} \\ &\quad + \frac{L_u(c_{i+1,3})}{L_v(c_{i+1,3})}\left(\frac{R_u(c_{i,3})}{L_u(c_{i+1,3})} - \frac{R_v(c_{i,2})}{L_v(c_{i+1,2})}\right) v_{i,2} \\ &\quad - \frac{R_v(c_{i,3})}{L_v(c_{i+1,3})} v_{i,3} \,. \end{aligned} \tag{2.20}$$

Together, Eq. (2.11), Eq. (2.13) and Eq. (2.20) form the transfer mapping from $\mathbf{v}_i$ to $\mathbf{v}_{i+1}$, which we capture compactly as $\mathbf{v}_{i+1} = T(\mathbf{c}_i, \mathbf{c}_{i+1})\mathbf{v}_i$. Note that the transfer matrix $T(\mathbf{c}_i, \mathbf{c}_{i+1})$ is now a 3×3 lower-triangular matrix that depends on the orientations $\mathbf{c}_i = (c_{i,1}, c_{i,2}, c_{i,3})$ of the building blocks in column i and column $i+1$.

Now, we want to find values for the orientations $\mathbf{c}_i$ that produce a valid strip mode, i.e., a deformation $\mathbf{v}_1$ which satisfies all constraints: the diagonal compatibility constraints [Eq. (2.2)], lower strip condition [Eq. (2.5)] and periodic strip condition [Eq. (2.6)]. Additionally, the strip deformation $\mathbf{v}_1$ should satisfy the CC and NT conditions. We note that $\mathbf{v}_1 = 0$ corresponds to the strip deforming as an area of CRS with zero amplitude, i.e., not deforming at all. Additionally, $v_{1,1} = 0$ with $v_{1,2} \neq 0$ and $v_{1,3} \neq 0$ corresponds to the top row not deforming at all and $v_{1,2} = -v_{1,3}$ with $v_{1,1} \neq 0$ corresponds to the bottom row deforming as an area of CRS with arbitrary amplitude. All these cases break the nontrivial (NT) condition as they describe strip deformations completely or in-part composed of rows of CRS blocks and thus do not describe valid $W = 3$ strip modes. We exclude these configurations.

To construct valid strip configurations, we consider 2×3 configurations of building blocks $(\mathbf{c}_i, \mathbf{c}_{i+1})$. Again, we compose such configurations by vertically stacking pairs of horizontally adjacent building blocks $(c_{i,j}, c_{i+1,j})$ for the top row $j = 1$, middle row $j = 2$ and bottom row $j = 3$, e.g., a triplet of d-, u-, and h-pairs, which we denote as a (d, u, h)-pair, satisfies condition [Eq. (2.17)] for $(c_{i,1}, c_{i+1,1})$, satisfies condition [Eq. (2.16)] for $(c_{i,2}, c_{i+1,2})$ and satisfies condition [Eq. (2.15)] for $(c_{i,3}, c_{i+1,3})$, see Fig. 2.10(a) for an example of a (d, u, h)-pair. Additionally, we now distinguish between the s-pair $(c_{i,j}, c_{i+1,j}) = (\mathrm{NE}, \mathrm{SW})$ and the ƨ-pair $(c_{i,j}, c_{i+1,j}) = (\mathrm{SE}, \mathrm{NW})$ [Fig. 2.8(c)] despite both pairs satisfying condition [Eq. (2.18)] as configurations composed of such pairs impose distinct constraints on the local strip deformation $\mathbf{v}_i$.

In what follows, we will show that a valid strip configuration consists only of (h, h, h), (d, u, h), (h, d, u), (d, s, u) and (d, ƨ, u)-pairs. Specifically, we will show the following for strip configurations composed of such pairs:

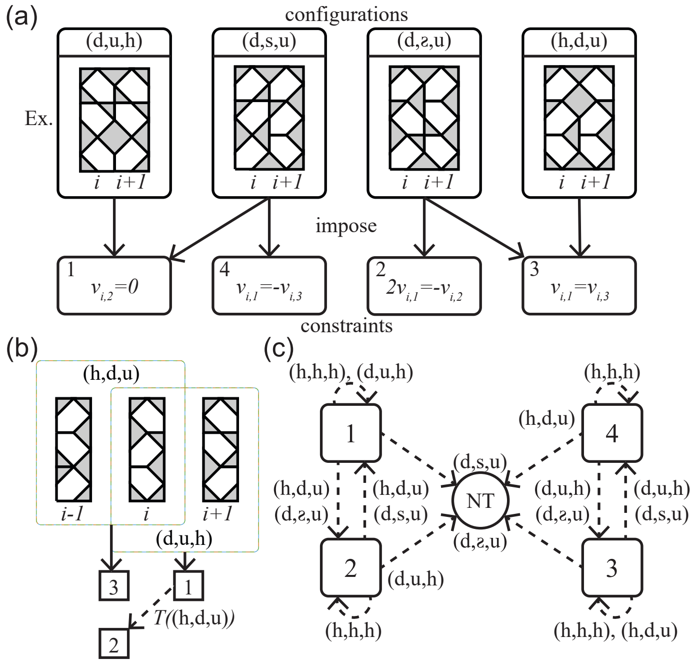


FIGURE 2.10: (a) 2×3 configurations $(\mathbf{c}_i, \mathbf{c}_{i+1})$ consisting of triplets of (d, u, h)-pairs, (d, s, u)-pairs, (d, ƨ, u)-pairs and (h, d, u)-pairs, impose one or two of four constraints (Eqs. (2.21)-(2.24), labeled 1 to 4 respectively) on the local strip deformation $\mathbf{v}_i$. We indicate this by black solid arrows. Note that the shown configurations are examples of the indicated pair type; other configurations that belong to the same type are possible. (b) A tiling of a (h, d, u)-pair and (d, u, h)-pair. Both pairs impose a constraint on their local strip deformation, $\mathbf{v}_{i-1}$ and $\mathbf{v}_i$ respectively, indicated by solid black arrows pointing to squares with numbers corresponding to the constraints as indicated in (a). Additionally, the constraint on $\mathbf{v}_i$ can be mapped using the transfer matrix $T((\mathrm{h}, \mathrm{d}, \mathrm{u}))$ to a constraint on $\mathbf{v}_{i-1}$. This imposes constraint 2 with $i \mapsto i-1$ on $\mathbf{v}_{i-1}$, the transfer mapping is indicated by the dashed arrow. (c) The constraints map from a constraint on strip deformation $\mathbf{v}_i$ to a constraint on $\mathbf{v}_{i-1}$ under application of the transfer mapping $T(\mathbf{c}_{i-1}, \mathbf{c}_i)$ (dashed arrows) for the configurations $(\mathbf{c}_{i-1}, \mathbf{c}_i)$ as indicated next to the arrows. Note that here we only consider (h, h, h)-pairs, (d, u, h)-pairs, (h, d, u)-pairs, (d, s, u)-pairs and (d, ƨ, u)-pairs. A constraint maps to the indicated constraint with $i \mapsto i-1$. Constraints are labeled by number as indicated in (a). Every constraint can map to a constraint that breaks the NT condition.

(1) each of these configurations except (h, h, h)-pairs imposes one or two constraints out of a set of four possible constraints on the deformation $\mathbf{v}_i$ to satisfy the diagonal compatibility constraints [Eq. (2.2)] and lower

strip condition [Eq. (2.5)] locally.

(2) upon applying the transfer mapping $T(\mathbf{c}_{i-1}, \mathbf{c}_i)$, each of the four possible constraints on $\mathbf{v}_i$ map to constraints on $\mathbf{v}_{i-1}$ that are degenerate to the four possible constraints that can be imposed by the $(\mathbf{c}_{i-1}, \mathbf{c}_i)$-pair locally on $\mathbf{v}_{i-1}$ for most valid 2×3 configurations. For the other valid configurations, the mapped constraints and local constraints imposed by the $(\mathbf{c}_{i-1}, \mathbf{c}_i)$-pair on $\mathbf{v}_{i-1}$ together break the CC or NT conditions and do not result in a valid $W = 3$ strip mode. We exclude such combinations.

(3) constraints on the configurational ordering of $(\mathbf{c}_i, \mathbf{c}_{i+1})$-pairs are captured with a simple additional rule.

We now consider these points one-by-one.

First, we find that for (d,u,h)-pairs, (h, d, u)-pairs, (d, s, u)-pairs, and (d, ƨ, u)-pairs the diagonal compatibility constraints [Eq. (2.2)] and lower strip condition [Eq. (2.5)] are satisfied locally by satisfying one or two of four different constraints on $\mathbf{v}_i$. These four different constraints are (see App. A2.6.2):

$$v_{i,2} = 0 \ , \tag{2.21}$$

$$2v_{i,1} = -v_{i,2} \ , \tag{2.22}$$

$$v_{i,1} = v_{i,3} \ , \quad \text{and} \tag{2.23}$$

$$v_{i,1} = -v_{i,3} \ . \tag{2.24}$$

We find that a (d, u, h)-pair imposes constraint [Eq. (2.21)], a (h, d, u)-pair imposes constraint [Eq. (2.23)], a (d, s, u)-pair imposes constraints [Eq. (2.21)] and [Eq. (2.24)], and a (d, ƨ, u)-pair imposes constraints [Eq. (2.22)] and [Eq. (2.23)] on $\mathbf{v}_i$ [Fig. 2.10(a)]. An (h, h, h)-pair trivially satisfies the diagonal compatibility constraints and lower strip condition and does not place any constraints on $\mathbf{v}_i$.

Now, we combine the valid 2×3 configurations (h, h, h)-pairs, (h, d, u)-pairs, (d, u, h)-pairs, (d, s, u)-pairs and (d, ƨ, u)-pairs in a strip configuration, see Fig. 2.10(b) for an example. We find that most combinations of these configurations result in a valid strip mode, but there are exceptions for which we devise a rule. First, we consider each of the four constraints [Eqs. (2.21)-(2.24)] on $\mathbf{v}_i$ and use the transfer mapping $T(\mathbf{c}_{i-1}, \mathbf{c}_i)$ to transform each constraint to a constraint on $\mathbf{v}_{i-1}$ for each valid 2×3 configuration $(\mathbf{c}_i, \mathbf{c}_{i+1})$ (see App. A2.8). The total set of constraints on $\mathbf{v}_{i-1}$ then consists

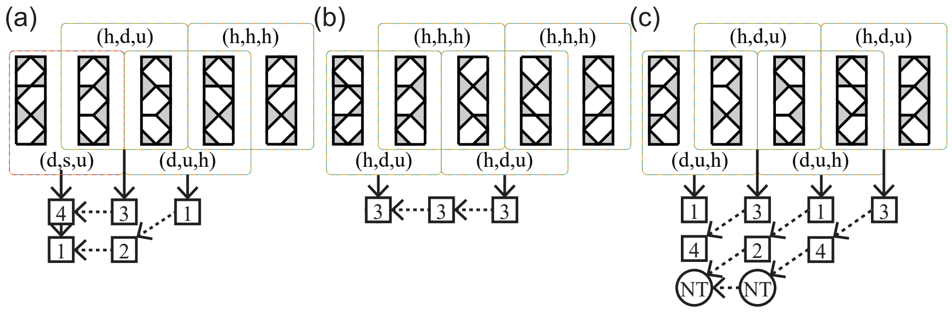


FIGURE 2.11: (a) Example of a valid $k = 4$ strip configuration supporting a $W = 3$ strip mode. Notice that we take periodic boundary conditions. There are two mapped and local constraints on $\mathbf{v}_1$, thereby satisfying the CC condition, and both results do not break the NT condition, resulting in a valid $W = 3$ strip mode. (b) Example of an invalid strip configuration that does not support a valid $W = 3$ strip mode. There is only one mapped and local constraint on $\mathbf{v}_1$, breaking the CC condition. The strip deformation can be decomposed into a $W = 1$ strip mode and $W = 2$ strip mode. (c) Example of an invalid strip configuration that does not support a valid $W = 3$ strip mode. Some constraints map to a constraint that breaks the NT condition, resulting in an invalid strip deformation.

of the mapped constraint(s) and local constraints imposed by the configuration $(\mathbf{c}_{i-1}, \mathbf{c}_i)$ [Fig. 2.10(b)]. To have a valid strip mode, the total number of constraints must equal two to satisfy the CC condition. Additionally, none of the constraints may result in a strip deformation that does not satisfy the NT condition.

We find that the four constraints on $\mathbf{v}_i$ [Eqs. (2.21)-(2.24)] map within the set of these same four constraints with index $i \mapsto i-1$ on $\mathbf{v}_{i-1}$ for most configurations $(\mathbf{c}_{i-1}, \mathbf{c}_i)$ (see App. A2.9, Fig. 2.10(c)). However, for some configurations, the mapped constraints, when taken together with the local constraints imposed by the configuration on $\mathbf{v}_i$, result in a strip deformation $\mathbf{v}_i$ that breaks the NT condition [Fig. 2.11(c)]. To construct strip configurations that result in a valid $W = 3$ strip mode we exclude combinations of valid configurations that result in such constraints.

We now aim to find what combinations of valid configurations do not result in a valid $W = 3$ strip mode. The constraint mapping [Fig. 2.10(c)] prohibits certain combinations of valid configurations. In general, for a given strip configuration C^{SM} each $(\mathbf{c}_i, \mathbf{c}_{i+1})$-pair imposes one or two constraints [Eqs. (2.21)-(2.24)] on the local deformation $\mathbf{v}_i$. These constraints then need to be iteratively mapped to $\mathbf{v}_1$, starting from $\mathbf{v}_k$ (Fig. 2.11). If at any point in the strip configuration the CC or NT conditions on $\mathbf{v}_i$ are not satisfied, the strip configuration does not support a valid $W = 3$ strip mode [Fig. 2.11(c)]. To find which sets of pairs result in invalid strip modes, we look for combinations of pairs that lead to a constraint on $\mathbf{v}_{i+1}$ that will get mapped to a constraint that breaks the NT condition on $\mathbf{v}_i$ using the constraint map [Fig. 2.10(c)]. We find that there are sets of pairs in either the top two rows or bottom two rows of the strip that are not allowed to occur in order anywhere in the strip (see App. A2.10). Moreover, this set of pairs can be freely padded with (h, h, h)-pairs as such pairs do not add any constraints of their own and act as an identity mapping for the constraints [Fig. 2.10(c)]. Thus, to determine if a strip configuration supports a valid strip mode requires knowledge of the entire strip configuration.

We observe that the combinations of valid configurations that result in an invalid strip mode all follow a simple configurational rule. To formulate this rule, we note that the nontrivial diagonal edge d^c of each building block in a strip composed of valid configurations meets at a vertex with a single other nontrivial diagonal edge of a building block in the strip. We refer to such pairs of building blocks as *linked*. Linked building blocks can be oriented either horizontally, vertically or diagonally with respect to each other [Fig. 2.12(a)]. We observe that sequences of valid configurations that result in an invalid strip mode always contain both vertically linked and diagonally linked building blocks. Thus we can formulate a simple rule to exclude invalid sequences: all building blocks linked together in two adjacent rows can only be linked vertically or diagonally, never both.

We capture these necessary requirements in a compact set of design rules:

(i) Every 2×3 configuration of building blocks in the strip must be a (h, h, h)-pair, (d, u, h)-pair, (h, d, u)-pair, (d, s, u)-pair or (d, ƨ, u)-pair.

(ii) There must be at least a single d-pair in the top row and at least a single u-pair in the bottom row.

(iii) All linked building blocks in two adjacent rows can only be linked vertically and horizontally or diagonally and horizontally.

Rule (ii) is required to satisfy the constraint counting (CC) condition and result in a single $W = 3$ strip mode, rather than multiple smaller strip modes [Fig. 2.11(b)]. Rule (iii) is added to exclude invalid sequences of configurations that do not result in a valid $W = 3$ strip mode [Fig. 2.11(c)]. Note that this rule is global—checking it requires knowledge of the entire strip. This is because the CC condition now permits two constraints, both

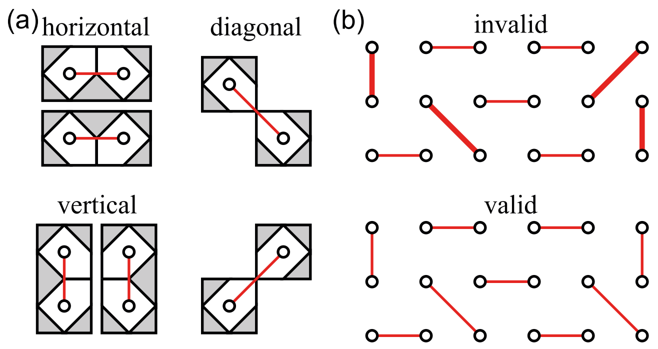


Figure 2.12: (a) Linked pairs of building blocks are marked by a white circle in the center of each building block and a red solid line that connects the circles. Linked building blocks are labeled by orientation. (b) $k = 6$ strip configurations are represented as collections of linked building blocks. The invalid configuration (top) breaks rule (ii) as it contains both vertically and diagonally linked building blocks in both pairs of adjacent rows (thick red solid lines). The valid configuration (bottom) describes a strip configuration that supports a $W = 3$ strip mode.

of which can potentially map to a constraint that breaks the nontrivial (NT) condition. A constraint introduced at the very end of the strip can be mapped throughout the entire strip and only encounter an incompatible configuration at the beginning of the strip. These are sufficient conditions to obtain $W = 3$ strip modes. They can also be shown to be necessary conditions, because other 2×3 configurations constrain the strip deformation to $v_{1,1} = 0$, $v_{1,1} = -v_{1,2}$ or $v_{1,2} = -v_{i,3}$ or combinations, thereby breaking the NT condition and therefore do not result in a valid $W = 3$ strip mode (see App. A2.6.2). Hence, these pairing rules are necessary and sufficient conditions on the strip configuration to support a $W = 3$ strip mode.

2.8.4. Towards general design rules

Now we discuss how these design rules generalize to larger width W strip configurations. We have proven that the rules we found for strip modes of width $W = 1$, $W = 2$ and $W = 3$ are necessary and sufficient requirements on a strip configuration to support a valid strip mode. Based on these rules, we formulate a general set of rules that we conjecture are, at the least, also sufficient requirements for larger width W strip modes. We formulate these rules completely in terms of linked building blocks [Fig. 2.12(a)]:

(i) Every building block in the strip must be linked with a single other building block in the strip

(ii) All linked building blocks in two adjacent rows must only be linked vertically and horizontally or diagonally and horizontally, never vertically and diagonally.

The smallest width W and irreducible strip in the unit cell for which these rules hold supports a strip mode of width W. Rule (ii) is a global rule; checking it requires knowledge of the entire strip [Fig. 2.12(b)]. We find a perfect agreement of our rules for $\sim 10^6$ randomly generated $k \times k$ unit cell designs to be of class (iii) or not (see App. A2.11). We therefore have strong numerical evidence that our rules are not only necessary to have a strip mode, but also that strip modes are the only type of zero mode that result in class (iii) mode-scaling. As a final indication that these rules are sufficient for a strip configuration to support a strip mode, we use the rules to design a strip mode of width $W = 10$ [Fig. 2.13].

2.9. Discussion

The rational design of multiple soft modes in aperiodic metamaterials is intrinsically different from tiling- or spin-ice based design strategies for a single soft mode [10, 11, 17, 19, 77, 79]. The key challenge is to precisely control the balance between the kinematic degrees of freedom with the kinematic constraints. For increasing sizes, these constraints proliferate though the sample, and to obtain multiple soft modes, the spatial design must be such that many of these constraints are degenerate. What is particularly vexing is that these constraints act on a growing set of local kinematic degrees of freedom, so that checking for degenerate constraints is cumbersome. As a consequence, current design strategies for multimodal metamaterials rely on computational methods, in either continuous systems [85–87] or discrete systems [88].

Here, we introduced a general transfer matrix-like framework for mapping the local constraints to a small, pre-defined subset of kinematic degrees of freedom, and use this framework to obtain effective tiling rules for a combinatorial multimodal metamaterial. Strikingly, beside the usual local rules which express constraints on pairs of adjacent building blocks, we find nonlocal rules that restrict the types of tiles that are allowed to appear together *anywhere* in the metamaterial. These kind of nonlocal rules are unique to multimodal metamaterials.

More broadly, our work is a first example where metamaterial design leads to complex combinatorial tiling problems that are beyond the limitations of Wang tilings. It is complementary to combinatorial computational

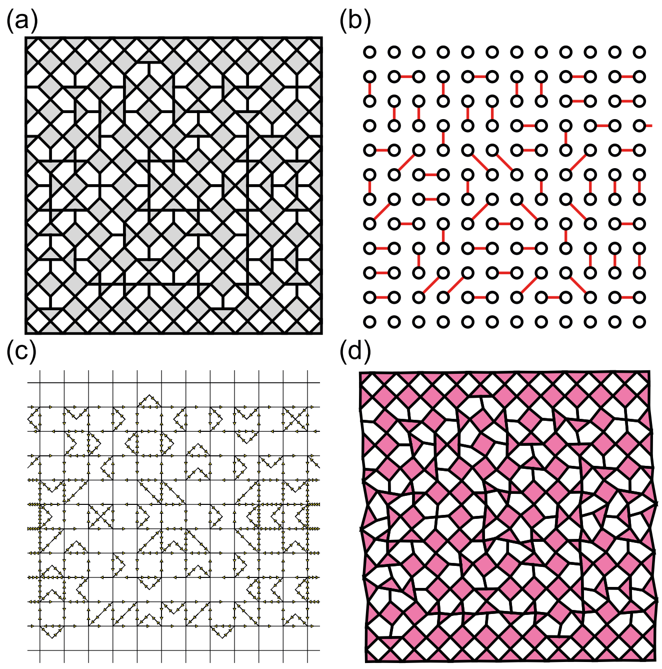


Figure 2.13: Realization of a 12 × 12 unit cell that supports a $W = 10$ strip mode. To better illustrate the kinematics of the strip mode, we have restrained the top and bottom layers of building blocks to deform solely with the CRS mode. (a) Schematic representation of the unit cell. (b) Linked pairs of the configuration. Note that the horizontal strip between rows 2 and 11 satisfies the general design rules. (c) Vertex representation of the strip mode. The number of arrows on horizontal and vertical edges connecting two building blocks is reduced by half for clearer visualization. (d) Schematic representation of the unit cell deforming as the strip mode.

methods used in design of irregular architectured materials [89] or computer graphics [90] that use local tiling rules to fabricate complicated spatial patterns.

Conversely, instead of clear-cut local rules that state which tiles fit together, our method requires careful bookkeeping of local constraints imposed by placed tiles and propagation of these constraints through all previously placed tiles to a single set of degrees of freedom. As a result, knowledge of a tile's neighbors is no longer sufficient information to determine if that tile can be placed. Instead, one requires knowledge of most, if not all, previously placed tiles. We believe our method is well-suited to tackle tiling problems beyond Wang tiles. Several open questions remain:

are nonlocal rule generically emerging in multimodal metamaterials? How does our method relate to other emergent nonlocal tiling constraints that arise, for example, in the fields of computer graphics [32–34] and chip design [35, 36]? Additionally, our method is limited to design of zero modes and thus may be insufficient when designing for larger deformations. How to adjust our method to include nonlinear kinematic constraints is an open question.

Our framework opens up a new route for rational design of spatially textured soft modes in multimodal metamaterials, which we demonstrate by designing metamaterials with strip modes of targeted width and location. Such strip modes can be utilized to control buckling and energy-absorption under uniaxial compression perpendicular to the orientation of the strip [18]. Our method can readily be extended to edge-modes, by considering, e.g., horizontal edge strips, imposing the upper strip condition and periodic strip condition and taking into account open boundary conditions at the bottom of the strip. Similarly, Swiss cheese-modes can be modeled by imposing upper and lower strip conditions horizontally and vertically at appropriate locations in the metamaterial. Additionally, our method can be extended to design three dimensional metamaterials by constructing an additional transfer matrix that propagates local degrees of freedom (dof) along the newly added spatial dimension. To ensure kinematic compatibility, additional constraints may need to be introduced to ensure different dof propagation paths result in the same final deformation. We hope our work will push the interest in multimodal metamaterials whose mechanical functionality is selectable through actuation, with potential applications in programmable materials, soft robotics, and computing *in materia*.

Data availability statement.—The code supporting the findings reported in this chapter is publicly available on GitLab [1]—the data on Zenodo [91].

Acknowledgments.—We thank David Dykstra and Marjolein Dijkstra for discussions. C.C. acknowledges funding from the European Research Council under Grant Agreement No. 852587.

[1]See `https://uva-hva.gitlab.host/published-projects/CombiMetaMaterial` for code to calculate zero modes and numerically check design rules.

Appendix

In this appendix, we provide more details on deformations in our metamaterial, explicit derivations of transfer matrices and kinematic constraints, and numerical proof of our conjectured general strip mode rules.

A2.1. Open boundary conditions

Here, we show that angles located at an open boundary can deform unconstrained, both at the faces of the building blocks (u, v, l, r) and corners (d^c). First, we consider angles at the face of each building block. If the face of the building block is located at an open boundary, the angle can deform freely as there is no competing adjacent angle. For example, if the top face of a building block z is located at an open boundary, there are no constraints placed upon deformation u_z.

Second, we consider the nontrivial corner angle d^c of a building block with orientation c where the corner is located at an open boundary. Here, there can be an adjacent diagonal angle of a neighboring building block. However, the diagonal angle is still unconstrained in its deformation at the open boundary, regardless if it is adjacent to a diagonal angle of another building block. To see this, note that for the two neighboring building blocks at an open boundary to be kinematically compatible, only the angles at the shared face between the two building blocks are constrained with the horizontal or vertical compatibility constraint. For example, two horizontally neighboring building blocks at locations z and $z + 1$ with their top face at the open boundary deform compatibly only if the right and left angles satisfy $r_z = -l_{z+1}$. More formally, this can be shown by composing the compatibility matrix for these two building blocks in all possible orientations and determining the dimension of the matrix' null space [40, 42]. This dimension is always equal to six, which corresponds to three floppy modes and three trivial modes: rotation and translation. As each building block has two zero modes, there must only be one constraint placed on their deformations: the horizontal compatibility constraint. As there are no states of self-stress in this structure, the number of floppy modes also follows from a simple Maxwell counting argument [41]. Thus, nontrivial diagonal corners d^c located at the open boundary can deform unconstrained.

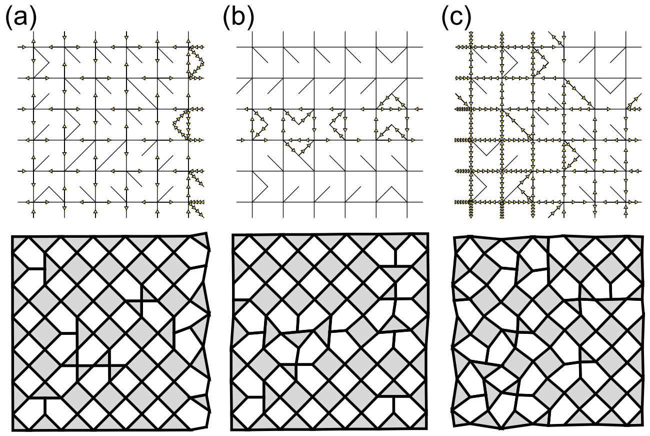


Figure A2.1: Vertex (top) and schematic (bottom) representations of an edge-mode (a), strip mode (b) and Swiss cheese-mode (c). Note that we replaced rigid pentagons with a reentrant edge with rigid diamonds (rotated squares) that are kinematically equivalent in the schematic representation for ease of interpretation.

A2.2. Realizations mode structure

Here we show explicit realizations of unit cells that support an edge-mode [Fig. A2.1(a)], a strip mode [Fig. A2.1(b)] and a Swiss cheese-mode [Fig. A2.1(c)] as described in Sec. 2.4 and Fig. 2.3.

A2.3. Linear coordinate transformations

To find conditions on the strip configuration C^{SM}, we change to a more convenient basis where instead of mode amplitudes α_z and β_z, deformations u_z and v_z are the two degrees of freedom for each building block. We find

$$\begin{pmatrix} u_z \\ v_z \end{pmatrix} = \begin{bmatrix} 1 & u_D(c_z) \\ 1 & v_D(c_z) \end{bmatrix} \begin{pmatrix} \alpha_z \\ \beta_z \end{pmatrix} = \Lambda(c_z) \begin{pmatrix} \alpha_z \\ \beta_z \end{pmatrix}, \tag{A2.1}$$

where $u_D(c_z)$ and $v_D(c_z)$ are the u- and v-components of the basis zero mode $m_D(c_z)$. Subsequently, we invert $\Lambda(c_z)$ to find the change of basis matrix

$$\Lambda^{-1}(c_z) = \frac{1}{v_D(c_z) - u_D(c_z)} \begin{bmatrix} v_D(c_z) & -u_D(c_z) \\ -1 & 1 \end{bmatrix}, \tag{A2.2}$$

which is well-defined since $u_D(c_z) \neq v_D(c_z)$ for all orientations c_z. We can express deformations l_z, r_z, and d_z^o in terms of (u_z, v_z) using the change of basis matrix:

$$\begin{pmatrix} l_z \\ r_z \end{pmatrix} = \Gamma(c_z)\Lambda^{-1}(c_z)\begin{pmatrix} u_z \\ v_z \end{pmatrix} \tag{A2.3}$$

and

$$d_z^o = \zeta^o(c_z)\Lambda^{-1}(c_z) \cdot \begin{pmatrix} u_z \\ v_z \end{pmatrix}, \tag{A2.4}$$

where

$$\Gamma(c_z) = \begin{pmatrix} -1 & l_D(c_z) \\ -1 & r_D(c_z) \end{pmatrix} \quad \text{and} \tag{A2.5}$$

$$\zeta^o(c_z) = (0, d_D^o(c_z)) \tag{A2.6}$$

depend on the orientation c_z of the building block. Note that o denotes the orientation of diagonal deformation d_z^o, which is independent from the building block orientation c_z. The equation for l_z, r_z and d_z^o further simplify to

$$l_z = L_u(c_z)\, u_z + L_v(c_z)\, v_z, \tag{A2.7}$$

$$r_z = R_u(c_z)\, u_z + R_v(c_z)\, v_z \tag{A2.8}$$

and

$$d_z^o = D^o(c_z)(-u_z + v_z). \tag{A2.9}$$

Values of the coefficients L_u, L_v, R_u, R_v, D^o for the four orientations c_z are given in Table 2.1.

A2.4. Deriving the transfer mapping

Here we derive the linear transfer mapping that maps the vertical deformations $\mathbf{v}_i = (v_{i,1}, v_{i,2}, ..., v_{i,W})$ of row i in a strip configuration of width W to the vertical deformations $\mathbf{v}_{i+1}$ of the neighboring row $i+1$ such that $\mathbf{v}_{i+1} = T(\mathbf{c}_i, \mathbf{c}_{i+1})\mathbf{v}_i$. We consider a $2 \times W$ strip configuration of unspecified orientations $(\mathbf{c}_i, \mathbf{c}_{i+1})$. To derive this transfer matrix, we solve the horizontal and vertical compatibility constraints [Eq. (2.1)] and upper boundary condition [Eq. (2.4)] iteratively for the vertical deformations $\mathbf{v}_i$ [Fig. 2.5(a)-(f)]. We first consider the first row in the strip such that $j = 1$. From the upper boundary conditions and setting $u_{i,1} = 0$ without loss of generality, we find $u_{i,1} = -u_{i+1,1} = 0$ such that the horizontal compatibility constraint reduces to

$$v_{i+1,1} = -\frac{R_v(c_{i,1})}{L_v(c_{i+1,1})} v_{i,1}. \tag{A2.10}$$

We now consider a general row j and find that the horizontal compatibility condition reduces to

$$\begin{aligned} v_{i+1,j} = {} & \frac{R_u(c_{i,j})}{L_v(c_{i+1,j})} v_{i,j-1} - \frac{R_v(c_{i,j})}{L_v(c_{i+1,j})} v_{i,j} \\ & + \frac{L_u(c_{i+1,j})}{L_v(c_{i+1,j})} v_{i+1,j-1}. \end{aligned} \tag{A2.11}$$

Thus we can solve for $v_{i+1,j}$ in terms of $\mathbf{v}_i$ by recursively applying this equation. We find

$$\begin{aligned} v_{i+1,j} = {} & \sum_{a=1}^{j-1} \left(\frac{R_u(c_{i,a+1})}{L_u(c_{i+1,a+1})} - \frac{R_v(c_{i,a})}{L_v(c_{i+1,a})} \right) \\ & \times \prod_{b=a+1}^{j} \frac{L_u(c_{i+1,b})}{L_v(c_{i+1,b})} v_{i,a} - \frac{R_v(c_{i,j})}{L_v(c_{i+1,j})} v_{i,j}, \end{aligned} \tag{A2.12}$$

for $j \geq 2$. The linear map from $\mathbf{v}_i$ to $\mathbf{v}_{i+1}$ is captured in the transfer matrix $T(\mathbf{c}_i, \mathbf{c}_{i+1})$, which we can now define explicitly:

$$T(\mathbf{c}_i, \mathbf{c}_{i+1})_{a,b} = \begin{cases} \left(\frac{R_u(c_{i,b+1})}{L_u(c_{i+1,b+1})} - \frac{R_v(c_{i,b})}{L_v(c_{i+1,b})} \right) \prod_{j=b+1}^{a} \frac{L_u(c_{i+1,j})}{L_v(c_{i+1,j})}, & \text{if} \quad b < a \\ -\frac{R_v(c_{i,b})}{L_v(c_{i+1,b})}, & \text{if} \quad b = a \\ 0, & \text{if} \quad b > a. \end{cases} \tag{A2.13}$$

A2.5. nontrivial conditions

Here we show that a CRS site, which has deformations $u_{i,j} = v_{i,j}$, in the top row of the strip SM or bottom row of the strip leads to that entire strip deforming with the CRS mode, breaking the nontrivial (NT) condition. This already follows from the restriction on the mode structure described in Sec. 2.4, but for completeness we derive it here using the transfer-matrix formalism. Moreover, we show that a CRS block in the second row of the strip in a $W = 3$ strip breaks the NT condition. We first consider a

CRS site in the top row, such that $j = 1$, and use the upper boundary conditions [Eq. (2.4)] and transfer matrix [Eq. (A2.13)] to show that the left-most vertical deformation $v_{1,1} = 0$. Second, we consider a CRS site in the bottom row, such that $j = W$, and use lower boundary conditions [Eq. (2.5)] and transfer matrix [Eq. (A2.13)] to show that the left-most vertical deformations $v_{i,W-1} = -v_{i,W}$. Finally, we consider a CRS site in the second row, such that $j = 2$, and use the transfer matrix [Eq. (A2.13)] to show that such a block results in a constraint on $\mathbf{v}_1$ that is incompatible with the four $W = 3$ constraints [Eqs. (2.21)-(2.24)] and breaks the NT condition.

First, we consider a general strip configuration C^{SM}. Suppose the building block at site $(i, 1)$ can only deform with the CRS mode, such that deformations $u_{i,1} = v_{i,1}$. Without loss of generality, we set the left-most upper deformation $u_{1,1} = 0$. From the upper boundary condition [Eq. (2.4)] we find that upper deformation $u_{i,1} = 0$, such that the vertical deformation becomes $v_{i,1} = 0$. The transfer matrix [Eq. (A2.13)] is lower triangular, thus $v_{i,1}$ only depends on the upper left-most vertical deformation $v_{1,1}$ by a factor consisting of the product of the diagonal transfer matrix components $T(\mathbf{c}_i, \mathbf{c}_{i+1})_{1,1}$ of the building block pairs between $(1, 1)$ and $(i, 1)$. Therefore, if $v_{i,j} = 0$, $v_{1,1} = 0$ must be true as well. Moreover, $v_{a,1} = 0$ for all columns a in the strip. Thus all building blocks in the strip are CRS sites and deform with $u_{i,1} = v_{i,1} = 0$, resulting in the entire top row of the strip deforming as a CRS mode compatibly with area of CRS with amplitude $\alpha^u = 0$.

Second, we consider a general strip configuration C^{SM} where the building block at site (i, W) deforms with the CRS mode, such that $u_{i,W} = v_{i,W}$. From the vertical compatibility constraint [Eq. (2.1)] we know that $u_{i,W} = -v_{i,W-1}$, such that $v_{i,W-1} = -v_{i,W}$. To find the deformations of the left neighbor at site $(i + 1, W)$, we plug the map [Eq. (A2.12)] into the lower boundary condition [Eq. (2.5)] to find

$$v_{i+1,W} \frac{L_v(c_{i+1,W}) - R_u(c_{i,W}) - R_v(c_{i,W})}{L_u(c_{i+1,W})} = -v_{i+1,W-1}. \tag{A2.14}$$

The fraction reduces to 1 for all possible configuration pairs $(c_{i,W}, c_{i+1,W})$ (see Table 2.1), such that the vertical deformations of the neighboring building block deform as $v_{i+1,W} = -v_{i+1,W-1}$ and the block is a CRS site. Similarly, we can do the same calculation for the right neighbor at site $(i - 1, W)$ to find

$$v_{i-1,W-1} = v_{i-1,W} \frac{R_v(c_{i-1,W}) - L_u(c_{i,W}) - L_v(c_{i,W})}{R_u(c_{i-1,W})} \tag{A2.15}$$

which also reduces to $v_{i-1,W-1} = -v_{i-1,W}$ for all possible configuration pairs $(c_{i-1,W}, c_{i,W})$. Thus we find that a CRS site in the bottom of the strip results in CRS sites to its left and right neighbor upon requiring the lower boundary condition [Eq. (2.5)] to be satisfied. In conclusion, we find that a single CRS site in the top or bottom row of the strip SM results in that entire row deforming as an area of CRS, breaking the NT condition.

Next, we consider how a CRS site in row $j = 2$, where $v_{i,1} = -v_{i,2}$, maps to a constraint on $\mathbf{v}_{i-1}$. This mapping depends on the configuration of columns $(\mathbf{c}_i, \mathbf{c}_{i+1})$. In general, we find it maps to

$$v_{i-1,1} = -\frac{T_{2,2}}{T_{1,1} + T_{2,1}} v_{i-1,2}, \tag{A2.16}$$

where $T_{a,b}$ is the (a,b)-th component of the transfer matrix $T(\mathbf{c}_i, \mathbf{c}_{i+1})$. This mapping depends only on the first two rows of the $2 \times W$ configuration. If the first two rows are (h, h)-pairs, the constraint maps to itself. If the column configuration has any other type, the constraint maps to a new constraint. For $W = 3$ configurations, this new constraint is not one of the four constraints [Eqs. (2.21)-(2.24)] we find in the main text. When this mapped constraint is taken together with one of the four constraints, the mapped constraint results in either the top two rows, or bottom two rows to deform as an area of CRS, breaking the NT condition. This is not a valid $W = 3$ strip mode. Thus the constraint $v_{i,1} = -v_{i,2}$ is incompatible with a valid strip deformation for $W = 3$ strip modes.

A2.6. Diagonal compatibility constraints

Here we derive the diagonal compatibility constraint [Eq. (2.2)] for all 2×2 configurations $(c_{i,j}, c_{i,j+1}, c_{i+1,j}, c_{i+1,j+1})$. We consider all configurations one-by-one. We first consider the configurations for which every diagonal edge d in the diagonal compatibility constraint [Eq. (2.2)] is trivially zero. The diagonal compatibility constraint is then trivially satisfied and imposes no condition on $\mathbf{v}_i$.

Subsequently, we consider the case where $d^{\mathrm{SE}}_{i,j}$ is the only nontrivial diagonal edge in the diagonal compatibility constraint. To satisfy this constraint, we require $d^{\mathrm{SE}}_{i,j} = 0$ to hold. From Eq. (2.10) we find that this constraint only holds if (i,j) is a CRS site, such that $u_{i,j} = v_{i,j}$. Similarly, if $d^{\mathrm{SW}}_{i+1,j}$ is the only nontrivial diagonal edge, we find $u_{i+1,j} = v_{i+1,j}$, if $d^{\mathrm{NE}}_{i,j+1}$ is the only nontrivial diagonal edge, we find $u_{i,j+1} = v_{i,j+1}$, and if $d^{\mathrm{NW}}_{i+1,j+1}$ is the only nontrivial diagonal edge, we find $u_{i+1,j+1} = v_{i+1,j+1}$. Thus a

2×2 configuration where a single building block is oriented such that its nontrivial diagonal edge d^c is part of the diagonal compatibility constraint [Eq. (2.2)] must deform that single building block as a CRS site to satisfy the diagonal compatibility constraint.

Now we consider configurations that contain horizontally paired building blocks which have nontrivial diagonal edges d^c that are both part of the diagonal compatibility constraint [Eq. (2.2)]. The two other building blocks' diagonal edges in the diagonal compatibility constraint are trivial. We first consider configurations where such a pair of building blocks is in the top row, such that $(c_{i,j}, c_{i+1,j}) = (SE, SW)$. The diagonal compatibility constraint reduces to

$$\begin{aligned} d^{\mathrm{SE}}_{i,j} + d^{\mathrm{SW}}_{i+1,j} &= 0 \\ u_{i,j} - v_{i,j} + u_{i+1,j} - v_{i+1,j} &= 0, \end{aligned} \tag{A2.17}$$

where we used Eq. (2.10) to replace d^c. We can simplify this further by replacing $v_{i+1,j}$ using the map Eq. (A2.12):

$$\left(1 + \frac{R_u(c_{i,j})}{L_v(c_{i+1,j})}\right) v_{i,j-1} + \left(1 - \frac{R_v(c_{i,j})}{L_v(c_{i+1,j})}\right) v_{i,j} + \left(1 + \frac{L_u(c_{i+1,j})}{L_v(c_{i+1,j})}\right) v_{i+1,j-1} = 0, \tag{A2.18}$$

where we have also used the horizontal compatibility constraint to replace u with v. Replacing the components R_u, L_u, L_v by their explicit values for $(c_{i,j}, c_{i+1,j}) = (SE, SW)$ (Table 2.1) we finally find the constraint

$$v_{i,j-1} = -v_{i+1,j-1}. \tag{A2.19}$$

Similarly, we can derive the constraint for when the horizontally paired building are located in the bottom row, such that $(c_{i,j+1}, c_{i+1,j+1}) = (NE, NW)$. We find the constraint

$$v_{i,j} = -v_{i+1,j}. \tag{A2.20}$$

Thus we find that such horizontally paired building blocks must deform their upper edges anti-symmetrically to satisfy the diagonal compatibility constraint. We can write Eq. (A2.20) as a constraint on $v_{i,j-1}$ by replacing $v_{i+1,j}$ using map [Eq. (A2.12)] and find

$$v_{i,j-1} + v_{i+1,j-1} = 0, \tag{A2.21}$$

$$2v_{i,j} + v_{i,j-1} - v_{i+1,j-1} = 0, \tag{A2.22}$$

$$\frac{2}{3} v_{i,j} - v_{i,j-1} + v_{i+1,j-1} = 0 \tag{A2.23}$$

for (h, h)-pairs and (u, h)-pairs with $c_{i+1,j} = NW$ or $c_{i,j} = NE$ respectively.

Now we consider configurations which contain vertically paired building blocks which have nontrivial diagonal edges that both are part of the same diagonal compatibility constraint [Eq. (2.2)]. The other two building blocks' diagonal edges in the diagonal compatibility constraint are trivial. We first consider configurations where this vertical pair of building blocks is located on the left, such that $(c_{i,j}, c_{i,j+1}) = (SE, NE)$. The diagonal compatibility constraint reduces to

$$\begin{aligned} d^{SE}_{i,j} + d^{NE}_{i,j+1} &= 0 \\ u_{i,j} - v_{i,j} - u_{i,j+1} + v_{i,j+1} &= 0 \\ -v_{i,j-1} + v_{i,j+1} &= 0, \end{aligned} \tag{A2.24}$$

where we used the horizontal compatibility constraint [Eq. (2.1)] to replace u with v. Now we consider configurations where this vertical pair is located on the right, such that $(c_{i+1,j}, c_{i+1,j+1}) = (SW, NW)$. The diagonal compatibility constraint reduces to

$$-v_{i+1,j-1} + v_{i+1,j+1} = 0 \tag{A2.25}$$

We now try to map this constraint on $\mathbf{v}_{i+1}$ to a constraint on $\mathbf{v}_i$ using map [Eq. (A2.12)]. We find that the constraint maps to

$$\begin{aligned} \left(\frac{L_u(c_{i+1,j+1})L_u(c_{i+1,j})}{L_v(c_{i+1,j+1})L_v(c_{i+1,j})} - 1\right) v_{i+1,j-1} \\ + \left(\frac{R_u(c_{i,j+1})}{L_v(c_{i+1,j+1})} - \frac{L_u(c_{i+1,j+1})R_v(c_{i,j})}{L_v(c_{i+1,j+1})L_v(c_{i+1,j})}\right) v_{i,j} \\ - \frac{R_v(c_{i,j+1})}{L_v(c_{i+1,j+1})} v_{i,j+1} \\ + \frac{L_u(c_{i+1,j+1})R_u(c_{i,j})}{L_v(c_{i+1,j+1})L_v(c_{i+1,j})} v_{i,j-1} = 0 \end{aligned} \tag{A2.26}$$

Depending on the precise orientations of the building blocks in the 2×2 configuration, this constraint reduces to two different constraints (Table 2.1)

$$v_{i,j+1} - v_{i,j-1} = 0 \tag{A2.27}$$

$$\frac{2}{3} v_{i,j} + \frac{1}{3} v_{i,j+1} + \frac{1}{3} v_{i,j-1} = 0. \tag{A2.28}$$

The first constraint [Eq. (A2.27)] must hold to satisfy the diagonal compatibility constraint for (d, u)-pairs and (s, ƨ)-pairs. The second constraint

[Eq. (A2.28)] must hold to satisfy the diagonal compatibility constraint for (d, ɘ)-pairs and (s, u)-pairs.

Next we consider configurations which contain diagonally paired building blocks which have nontrivial diagonal edges that are part of the same diagonal compatibility constraint. The other two building block's diagonal edges in the diagonal compatibility constraint are trivial. We first consider the case where $(c_{i,j}, c_{i+1,j+1}) = (SE, NW)$. The diagonal compatibility constraint [Eq. (2.2)] reduces to

$$d^{SE}_{i,j} + d^{NW}_{i+1,j+1} = 0 \tag{A2.29}$$

$$-v_{i,j-1} + v_{i,j} + v_{i+1,j} + v_{i+1,j+1} = 0, \tag{A2.30}$$

where we used Eq. (2.10) and the horizontal compatibility constraint to simplify the constraint. We use the map [Eq. (A2.12)] to replace $\mathbf{v}_{i+1}$ by $\mathbf{v}_i$. We find four different constraints depending on the orientations of the building blocks in the 2×2 configuration:

$$v_{i,j-1} - \frac{1}{3}v_{i,j+1} + \frac{2}{3}v_{i+1,j-1} = 0, \tag{A2.31}$$

$$\frac{1}{3}v_{i,j-1} + \frac{4}{9}v_{i,j} + \frac{1}{3}v_{i,j+1} + \frac{2}{9}v_{i+1,j-1} = 0, \tag{A2.32}$$

$$v_{i,j-1} + \frac{2}{3}v_{i,j} + \frac{1}{3}v_{i,j+1} + \frac{2}{3}v_{i+1,j-1} = 0, \tag{A2.33}$$

$$-\frac{1}{3}v_{i,j-1} + \frac{2}{9}v_{i,j} + \frac{1}{3}v_{i,j+1} - \frac{2}{9}v_{i+1,j-1} = 0, \tag{A2.34}$$

for (d, u)-pairs, (ɘ, u)-pairs, (d, ɘ)-pairs, and (ɘ, ɘ)-pairs respectively. Second, we consider the case where $(c_{i+1,j}, c_{i,j+1}) = (SW, NE)$. The diagonal compatibility constraint reduces to

$$\left(1 + \frac{R_v(c_{i,j})}{L_v(c_{i+1,j})}\right) v_{i,j} + v_{i,j+1} - \frac{R_u(c_{i,j})}{L_v(c_{i+1,j})} v_{i,j-1} - \left(1 + \frac{L_u(c_{i+1,j})}{L_v(c_{i+1,j})}\right) v_{i+1,j-1} = 0. \tag{A2.35}$$

Depending on the orientations of the 2×2 configuration, we find two different constraints:

$$v_{i,j+1} + v_{i,j-1} + 2v_{i+1,j-1} = 0 \tag{A2.36}$$

$$-2v_{i,j} + v_{i,j+1} - v_{i,j-1} + 2v_{i+1,j-1} = 0 \tag{A2.37}$$

for (d, u)-pairs and (d, s)-pairs, and (s, u)-pairs and (s, s)-pairs respectively.

Next, we consider configurations where three building blocks have a nontrivial diagonal edge that is part of the diagonal compatibility constraint [Eq. (2.2)]. We first consider $(c_{i,j}, c_{i+1,j}, c_{i,j+1}) = (\mathrm{SE}, \mathrm{SW}, \mathrm{NE})$. The diagonal compatibility constraint reduces to

$$v_{i,j-1} - v_{i,j+1} + v_{i+1,j-1} + v_{i+1,j} = 0. \tag{A2.38}$$

We replace $v_{i+1,j}$ using map [Eq. (A2.12)] for $(c_{i,j}, c_{i+1,j}) = (SE, SW)$ to find the constraint

$$-2v_{i,j-1} - v_{i,j+1} - v_{i,j} - 2v_{i+1,j-1} = 0, \tag{A2.39}$$

regardless of the orientation $c_{i+1,j+1}$. Next we consider $(c_{i,j}, c_{i+1,j}, c_{i+1,j+1}) = (SE, SW, NW)$. The diagonal compatibility constraint reduces to

$$v_{i,j-1} + v_{i,j} + v_{i+1,j-1} - v_{i+1,j+1} = 0. \tag{A2.40}$$

We replace $v_{i+1,j+1}$ using the map [Eq. (A2.12)] and find

$$\begin{aligned} v_{i,j-1} + \left(1 - \frac{R_u(c_{i,j+1})}{L_v(c_{i+1,j+1})}\right) v_{i,j} \\ + \frac{R_v(c_{i,j+1})}{L_v(c_{i+1,j+1})} v_{i,j+1} + v_{i+1,j-1} \\ + \frac{L_u(c_{i+1,j+1})}{L_v(c_{i+1,j+1})} v_{i+1,j} = 0 \end{aligned} \tag{A2.41}$$

We again apply map [Eq. (A2.12)] to replace $v_{i+1,j}$ and find the same constraint regardless of the orientation $c_{i,j+1}$:

$$v_{i,j} + v_{i,j+1} = 0. \tag{A2.42}$$

This constraint requires the building block at site $(i, j+1)$ to deform as a CRS block to satisfy the diagonal compatibility constraint. Next we consider $(c_{i,j}, c_{i,j+1}, c_{i+1,j+1}) = (\mathrm{SE}, \mathrm{NE}, \mathrm{NW})$. The diagonal compatibility constraint reduces to

$$v_{i,j-1} - v_{i,kj+1} - v_{i+1,j} - v_{i+1,j+1} = 0. \tag{A2.43}$$

We replace $\mathbf{v}_{i+1}$ using the map [Eq. (A2.12)] and find one of two constraints

$$-v_{i,j-1} + \frac{1}{3} v_{i,j} - \frac{2}{3} v_{i+1,j-1} = 0 \tag{A2.44}$$

$$v_{i,j-1} + \frac{1}{3} v_{i,j} + \frac{2}{3} v_{i+1,j-1} = 0, \tag{A2.45}$$

for a (d, h)-pair or (ɘ, h)-pair respectively. Finally, we consider the configuration $(c_{i+1,j}, c_{i,j+1}, c_{i+1,j+1}) = (\mathrm{SW}, \mathrm{NE}, \mathrm{NW})$. The diagonal compatibility constraint reduces to

$$-v_{i,j} - v_{i,j+1} + v_{i+1,j-1} - v_{i+1,j+1} = 0. \tag{A2.46}$$

We replace $\mathbf{v}_{i+1}$ using map [Eq. (A2.12)] and find the constraint

$$v_{i,j-1} + v_{i,j} = 0 \tag{A2.47}$$

regardless of the orientation of $c_{i,j}$. This constraint corresponds to site (i, j) deforming as a CRS site to satisfy the diagonal compatibility constraint.

Finally, we consider the last 2×2 configuration: $(c_{i,j}, c_{i+1,j}, c_{i,j+1}, c_{i+1,j+1}) = (\mathrm{SE}, \mathrm{SW}, \mathrm{NE}, \mathrm{NW})$. The diagonal compatibility constraint reduces to

$$v_{i,j-1} - v_{i,j+1} + v_{i+1,j-1} - v_{i+1,j+1} = 0. \tag{A2.48}$$

We replace $\mathbf{v}_{i+1}$ using map [Eq. (A2.12)] to find

$$v_{i,j-1} + v_{i+1,j-1} - v_{i,j-1} - v_{i+1,j-1} = 0, \tag{A2.49}$$

which is trivially true. Thus this 2×2 configuration does not impose any additional constraints on $\mathbf{v}_i$ to satisfy the diagonal compatibility constraint.

A2.6.1. Diagonal constraints for $W = 2$ configurations

These equations simplify further for specific cases. We first consider $W = 2$ valid configurations and then $W = 3$ valid configurations. For $W = 2$ strip configurations, $1 \leq j \leq 2$ and $v_{i,0} = -v_{i+1,0} = 0$. Moreover, the lower boundary condition dictates $v_{i,2} = -v_{i+1,2}$. We further simplify the diagonal compatibility constraint [Eq. (2.2)] by assuming the lower and upper boundary conditions are satisfied.

We find for (h, h)-pairs that constraint [Eq. (A2.20)] is trivially satisfied by the top boundary condition. Thus (h, h)-pairs do not impose any additional constraints on the strip deformation. Additionally, (d, u)-pairs can impose the constraints [Eqs. (A2.27), (A2.31), and (A2.36)], which simplify to

$$v_{i,2} = 0 \tag{A2.50}$$

for all cases.

Now, we consider (h, u)-pairs and (d, h)-pairs. The constraints [Eqs. (A2.39), (A2.42), (A2.44), and (A2.47)] reduce to

$$v_{i,1} + v_{i,2} = 0, \quad \text{or} \tag{A2.51}$$
$$v_{i,1} = 0 \tag{A2.52}$$

for (h, u)-pairs and (d, h)-pairs respectively. Both of these constraints are not compatible with a strip deformation as they break the nontrivial (NT) condition.

A2.6.2. Diagonal constraints for $W = 3$ configurations

Here, we consider the diagonal constraints for $W = 3$ configurations. Valid 3×2 configurations are (h, h, h)-pairs, (d, u, h)-pairs, (h, d, u)-pairs, (h, s, u)-pairs and (h, ƨ, u)-pairs. The upper boundary condition implies $v_{i,0} = -v_{i+1,0} = 0$ and the bottom boundary condition implies $v_{i,3} = -v_{i+1,3} = 0$. We consider each configuration one-by-one.

We first consider (h, h, h)-pairs, which can only impose constraint [Eq. (A2.19)] for all j. It is straightforward to check that the map [Eq. (A2.12)] for (h, h, h)-pairs trivially satisfies all possible diagonal compatibility constraints.

Second, we consider (d, u, h)-pairs, which can impose the constraints [Eqs. (A2.23), (A2.22), (A2.36), and (A2.31)]. Combining the possible constraints together with the upper and bottom boundary conditions and map [Eq. (A2.12)] impose the constraint

$$v_{i,2} = 0. \tag{A2.53}$$

Thus (d, u, h)-pairs impose constraint [Eq. (2.21)] on $\mathbf{v}_i$ to satisfy the diagonal compatibility constraints.

Third, we consider (h, d, u)-pairs. Such pairs can impose the constraints [Eqs. (A2.19), (A2.36), and (A2.31)], which simplify to

$$v_{i,1} = v_{i,3} \tag{A2.54}$$

using the upper and lower boundary conditions. Thus (h, d, u)-pairs impose constraint [Eq. (2.23)] to satisfy the diagonal compatibility constraints.

Fourth, we consider (d, s, u)-pairs. Such pairs impose the constraints [Eqs. (A2.28), (A2.36), and (A2.37)], which simplify to the constraints

$$v_{i,2} = 0, \quad \text{and} \tag{A2.55}$$
$$v_{i,1} = -v_{i,3}. \tag{A2.56}$$

Thus (d, s, u)-pairs impose constraints [Eqs. (2.21) and (2.24)] to satisfy the diagonal compatibility constraints.

Finally, we consider (d, ƨ, u)-pairs. Such pairs impose the constraints [Eqs. (A2.28), (A2.33), and (A2.32)], which simplify to the constraints

$$2v_{i,1} = -v_{i,2}, \quad \text{and} \tag{A2.57}$$

$$v_{i,1} = v_{i,3}. \tag{A2.58}$$

Thus (d, ƨ, u)-pairs impose constraints [Eqs. (2.22) and (2.23)] to satisfy the diagonal compatibility constraints.

In summary, we find that (h, h, h)-pairs do not impose any constraints on the strip deformation $\mathbf{v}_i$, (d, u, h)-pairs impose constraint [Eq. (2.21)], (h, d, u)-pairs impose constraint [Eq. (2.23)], (d, s, u)-pairs impose constraints [Eqs. (2.21) and (2.24)], and (s, ƨ, u)-pairs impose constraints [Eqs. (2.22) and (2.23)].

Now we consider invalid configurations that contain diagonal compatibility constraints with three nontrivial diagonal edges. First, we consider (h, u, h)-pairs. Such pairs impose the constraints [Eqs. (A2.42) and (A2.39)], which simplify to

$$v_{i,1} = -v_{i,2}. \tag{A2.59}$$

This corresponds to the block at site $(i, 1)$ deforming as a CRS block, which is incompatible with a $W = 3$ strip deformation. Second, we consider (d, h, h)-pairs. Such pairs impose the constraints [Eqs. (A2.47) and (A2.44)], which simplify to

$$v_{i,1} = 0, \tag{A2.60}$$

which is incompatible with a valid strip deformation. Third we consider (h, s, h)-pairs. Such pairs impose the constraints [Eqs. (A2.39) and (A2.47)], which simplify to

$$v_{i,2} = -v_{i,1}, \tag{A2.61}$$

which is incompatible with a valid strip deformation. Fourth, we consider (h, ƨ, h)-pairs. Such pairs impose the constraints [Eqs. (A2.42) and (A2.45)], which simplify to

$$v_{i,1} = -v_{i,2}, \tag{A2.62}$$

which is incompatible with a valid strip deformation. (h, s, u)-pairs, (h, ƨ, u)-pairs, (d, s, h)-pairs and (d, ƨ, h)-pairs all impose the same constraint on the strip deformation. Finally, we consider (h, h, u)-pairs. Such pairs impose the constraints [Eqs. (A2.42) and (A2.39)], which simplify to

$$v_{i,2} = -v_{i,3}. \tag{A2.63}$$

This is incompatible with a valid strip deformation.

A2.7. lower strip condition

In this appendix we find constraints that need to be satisfied in order to satisfy the lower boundary condition [Eq. (2.5)]. We first consider $W = 2$ configurations and then $W = 3$ configurations. We find that for valid configurations the lower boundary condition is satisfied by the same constraints that arise from the diagonal compatibility constraints [Eq. (2.2)].

First, we consider $W = 2$ configurations. In general, the map [Eq. (2.13)] does not satisfy the lower boundary condition $v_{i+1,2} = -v_{i,2}$. We first consider (h, h)-pairs. Here the map [Eq. (2.13)] reduces to $v_{i+1,2} = -v_{i,2}$, and the lower boundary condition is trivially satisfied. Next, we consider (d, u)-pairs. The map [Eq. (2.13)] together with the lower boundary condition reduces to the constraint

$$\left(1 - \frac{R_v(c_{i,2})}{L_v(c_{i+1,2})}\right) v_{i,2} = 0. \tag{A2.64}$$

For (d, u)-pairs, $R_v(c_{i,2})/L_v(c_{i+1,2})$ is either equal to 3 or 1/3. This constraint is thus satisfied if $v_{i,2} = 0$, which is the same as the constraint [Eq. (A2.50)] needed to satisfy the diagonal compatibility constraint. Thus the lower boundary condition and diagonal compatibility constraint for (d, u)-pairs both are satisfied if constraint $v_{i,2} = 0$ is satisfied.

Now, we consider $W = 3$ configurations. First, we consider (h, h, h)-pairs. The map [Eq. (2.20)] reduces to $v_{i+1,3} = -v_{i,3}$, thus trivially satisfying the lower boundary condition [Eq. (2.2)]. Second, we consider (d, u, h)-pairs. The map [Eq. (2.20)] together with the lower boundary condition [Eq. (2.2)] reduces to the constraint

$$\frac{L_u(c_{i+1,3})}{L_v(c_{i+1,3})}\left(1 - \frac{R_v(c_{i,2})}{L_v(c_{i+1,2})}\right) v_{i,2} = 0, \tag{A2.65}$$

which is satisfied by the constraint [Eq. (2.21)], just like the diagonal compatibility constraint for (d, u, h)-pairs. Third, we consider (h, d, u)-pairs. The map [Eq. (2.20)] together with the lower boundary condition reduces to the constraint

$$\begin{aligned}\frac{L_u(c_{i+1,2})L_u(c_{i+1,3})}{L_v(c_{i+1,2})L_v(c_{i+1,3})}&\left(\frac{R_u(c_{i,2})}{L_v(c_{i+1,2})} - 1\right) v_{i,1}\\ &+ \left(1 - \frac{R_v(c_{i,3})}{L_v(c_{i+1,3})} v_{i,3}\right) v_{i,3} = 0.\end{aligned} \tag{A2.66}$$

This constraint reduces to $v_{i,1} = v_{i,3}$ by filling in the possible explicit values of $(c_{i,2}, c_{i+1,2}, c_{i,3}, c_{i+1,3})$. Thus the lower boundary condition and diagonal compatibility constraint are satisfied by satisfying constraint [Eq. (2.23)] for (h, d, u)-pairs.

Now, we consider (d, s, u)-pairs. The map [Eq. (2.20)] together with the lower boundary condition [Eq. (2.5)] reduces to

$$-v_{i,1} + v_{i,2} - v_{i,3} = 0. \tag{A2.67}$$

This equation is satisfied by constraints [Eqs. (2.21) and (2.24)], just like the diagonal compatibility constraints [Eq. (2.2)] for (d, s, u)-pairs.

Finally, we consider (d, ƨ, u)-pairs. The map [Eq. (2.20)] together with the lower boundary condition [Eq. (2.5)] reduces to the constraint

$$v_{i,1} - v_{i,2} - 3v_{i,3} = 0. \tag{A2.68}$$

This constraint is satisfied by constraints [Eqs. (2.22) and (2.23)]. Thus the lower boundary condition and diagonal compatibility constraints for (d, ƨ, u)-pairs are satisfied by constraints [Eqs. (2.22) and (2.23)]. In conclusion, we find that satisfying the lower boundary constraints does not require any additional constraints than required to satisfy the diagonal compatibility constraints for valid $W = 3$ configurations: (h, h, h)-pairs, (d, u, h)-pairs, (h, d, u)-pairs, (d, s, u)-pairs and (d, ƨ, u)-pairs.

A2.8. Constraint mapping

In general, a linear constraint $f(\mathbf{v}_i, \mathbf{v}_{i+1}) = g(\mathbf{v}_i) + h(\mathbf{v}_{i+1}) = 0$ depends on the deformations of two adjacent columns: $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$. To map this constraint to a constraint on $\mathbf{v}_1$, we iteratively apply the transfer matrix to obtain a constraint on $\mathbf{v}_1$

$$\begin{aligned} &\prod_{a=1}^{i-1} T(\mathbf{c}_{i-a}, \mathbf{c}_{i+1-a}) g(\mathbf{v}_1) \\ &+ \prod_{a=1}^{i} T(\mathbf{c}_{i+1-a}, \mathbf{c}_{i+2-a}) h(\mathbf{v}_1) = 0 \end{aligned} \tag{A2.69}$$

A2.9. $W = 3$ constraint mapping

To find which combinations of valid 2×3 configurations lead to $W = 3$ strip modes, we consider how the four constraints [Eqs. (2.21)-(2.24)] on $\mathbf{v}_i$ map to $\mathbf{v}_{i+1}$ upon application of the transfer matrix $T(\mathbf{c}_{i-1}, \mathbf{c}_i)$ for all possible valid 2×3 configurations $(\mathbf{c}_{i-1}, \mathbf{c}_i)$. We first derive the transfer matrices for each valid configuration, which we subsequently apply tot the four constraints to see how they map.

First, we consider the transfer matrix for (h, h, h)-pairs: $T((h,h,h))$. We find that $T((h,h,h)) = -I$, where I is the 3×3 identity matrix. Thus the four constraints on $\mathbf{v}_i$ map to the same constraints on $\mathbf{v}_{i-1}$.

Second, we consider the transfer matrix for (d, u, h)-pairs. We find

$$T((d,u,h)) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{R_v(c_{i-1,2})}{L_v(c_{i,2})} & 0 \\ 0 & \frac{L_u(c_{i,3})}{L_v(c_{i,3})}\left(1 - \frac{R_v(c_{i-1,2})}{L_v(c_{i,2})}\right) & -1 \end{pmatrix}, \tag{A2.70}$$

where the precise orientations of $(c_{i-1,2}, c_{i,2})$ determine the explicit form of the matrix. Now we apply this transfer matrix on the four constraints and rewrite the constraints together with constraint [Eq. (2.21)] on $\mathbf{v}_{i-1}$ imposed by the (d, u, h)-pair itself. We find that constraint [Eq. (2.21)] maps to itself with $i \mapsto i-1$. Constraint [Eq. (2.22)] maps to $v_{i-1,1} = 0$, which is incompatible with a $W = 3$ strip deformation as it breaks the nontrivial (NT) condition. Finally, constraint [Eq. (2.23)] maps to constraint [Eq. (2.24)] with $i \mapsto i-1$ and similarly constraint [Eq. (2.23)] maps to constraint [Eq. (2.24)] with $i \mapsto i-1$.

Third, we consider the transfer matrix for (h, d, u)-pairs:

$$T((h,d,u)) = \begin{pmatrix} -1 & 0 & 0 \\ 2 & 1 & 0 \\ \frac{L_u(c_{i,3})}{L_v(c_{i,3})}2 & 0 & -\frac{R_v(c_{i-1,3})}{L_v(c_{i,3})} \end{pmatrix}, \tag{A2.71}$$

which depends on the precise orientations of $(c_{i-1,3}, c_{i,3})$. If we apply this transfer matrix on the four constraints and take the mapped constraint together with the constraint [Eq. (2.23)[] imposed by the (h, d, u)-pair on $\mathbf{v}_{i-1}$ we find that constraint [Eq. (2.21)] maps to constraint [Eq. (2.22)] with $i \mapsto i-1$. Constraint [Eq. (2.22)] maps to constraint [Eq. (2.21)] with $i \mapsto i-1$. Constraint [Eq. (2.23)] maps to itself with $i \mapsto i-1$. Constraint [Eq. (2.24)] maps to $v_{i-1,1} = v_{i-1,3} = 0$ when taken together with constraint [Eq. (2.23)] with $i \mapsto i-1$ imposed by the (h, d, u)-pair, which is incompatible with a $W = 3$ strip deformation as it breaks the NT condition.

Fourth, we consider the transfer matrix for (d, s, u)-pairs:

$$T((d,s,u)) = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 3 & 0 \\ -2\frac{L_u(c_{i,3})}{L_v(c_{i,3})} & 2\frac{L_u(c_{i,3})}{L_v(c_{i,3})} & -\frac{R_v(c_{i-1,3})}{L_v(c_{i,3})} \end{pmatrix}, \tag{A2.72}$$

which depends on the orientations of $(c_{i-1,3}, c_{i,3})$. We find that constraint [Eq. (2.21)] maps to the constraint $v_{i-1,1} = 0$ when taken together with constraint [Eq. (2.21)] with $i \mapsto i-1$ imposed by the (d, s, u)-pair itself. This constraint $v_{i-1,1} = 0$ is incompatible with a valid strip deformation. Constraint [Eq. (2.22)] maps to constraint [Eq. (2.22)] with $i \mapsto i-1$. Constraint [Eq. (2.23)] maps to constraint [Eq. (2.24)] with $i \mapsto i-1$ when considered together with constraint [Eq. (2.21)] with $i \mapsto i-1$ imposed by the (d, s, u)-pair. Finally, constraint [Eq. (2.24)] maps to $v_{i-1,1} = v_{i-1,3} = 0$ when taken together with constraints [Eqs. (2.21) and (2.24)] with $i \mapsto i-1$ imposed by the (d, s, u)-pair. This mapped constraint is not compatible with a valid strip deformation.

Finally, we consider the transfer matrix for (d, ƨ, u)-pairs:

$$T((d,\text{ƨ},u)) = \begin{pmatrix} 1 & 0 & 0 \\ \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{2}{3}\frac{L_u(c_{i,3})}{L_v(c_{i,3})} & -\frac{2}{3}\frac{L_u(c_{i+1,3})}{L_v(c_{i+1,3})} & -\frac{R_v(c_{i-1,3})}{L_v(c_{i,3})} \end{pmatrix}, \tag{A2.73}$$

which depends on the orientations of $(c_{i-1,3}, c_{i,3})$. We find that constraint [Eq. (2.21)] maps to constraint [Eq. (2.22)] with $i \mapsto i-1$. Constraint [Eq. (2.22)] maps to $v_{i-1,1} = 0$ when taken together with constraint [Eq. (2.22)] with $i \mapsto i-1$ imposed by the (d, ƨ, u)-pair itself. This mapped constraint is incompatible with a valid strip deformation. Constraint [Eq. (2.23)] maps to $v_{i-1,1} = v_{i-1,3} = 0$ when taken together with constraints [Eqs. (2.22) and (2.23)] with $i \mapsto i-1$ imposed by the (d, ƨ, u)-pair. This mapped constraint is incompatible with a valid strip deformation. Finally, constraint [Eq. (2.24)] maps to constraint [Eq. (2.23)] with $i \mapsto i-1$ using constraint [Eq. (2.22)] with $i \mapsto i-1$.

A2.10. Invalid $W = 3$ sequences

We now explicitly state the combinations of valid $(\mathbf{c}_i, \mathbf{c}_{i+1})$ configurations that do not result in a valid $W = 3$ strip mode. The constraint mapping

[Fig. 2.10(c)] and constraints imposed by the valid configurations prohibit certain combinations of valid configurations. In particular, we find for the top two rows in the strip that (d, u)-pairs or (d, s)-pairs cannot be preceded by a (d, s)-pair. Similarly, (d, ƨ)-pairs cannot be preceded by a (d, ƨ)-pair or (d, u)-pair. Additionally, a (d, u)-pair or (d, s)-pair preceded by a (h, d)-pair cannot be preceded by a (d, ƨ)-pair. Finally, a (d, ƨ)-pair preceded by a (h, d)-pair cannot be preceded by a (d, s)-pair. In the bottom two rows similar rules apply. There, we find that a (d, u)-pair or (ƨ, u)-pair cannot be preceded by a (ƨ, u)-pair. Similarly, a (s, u)-pair cannot be preceded by a (d, u)-pair or (s, u)-pair. Additionally, a (d, u)-pair or (d, ƨ)-pair preceded by a (u, h)-pair cannot be preceded by a (d, u)-pair or (s, u)-pair. Finally, a (s, u)-pair preceded by a (u, h)-pair cannot be preceded by a (ƨ, u)-pair. Sequences of such pairs in either the top or bottom two rows of the strip will not result in a valid strip mode. Note that such invalid sequences can be freely padded with (h, h)-pairs, as these map all the constraints to themselves and do not place any additional constraints on the strip deformation themselves.

We now translate the invalid sequences of pairs to conditions on sequences of linked building blocks [Fig. 2.12(a)]. First, we consider all invalid sequences of pairs in the top two rows. We note that all realizations of invalid sequences must feature a vertical and diagonal linked pair in the top two rows. To see this, note that (d, u)-pairs have one building block unlinked in the top row and one building block unlinked in the middle row, where these blocks are either both left or right if the two linked building blocks are linked vertically or the blocks are left and right or right and left if the two linked building blocks are linked diagonally. Similarly, (d, s)-pairs or (d, ƨ)-pairs have one building block unlinked within the two top rows: either to the left or right in the top row depending on if the two linked building blocks are diagonally or vertically linked ((d, s)-pairs), or vertically or diagonally linked ((d, ƨ)-pairs) respectively. Additionally, (h, d)-pairs have one unlinked building block in the top two rows have one unlinked building block depending on the orientation of the d-pair. To satisfy the strip rules, every building block needs to be linked. To construct a linked representation that features vertical and diagonal linked building blocks necessarily requires an invalid sequence of pairs. Inversely, every realization of an invalid sequence of pairs necessarily requires vertical and diagonal linked building blocks. The same holds for the invalid sequences in the bottom two rows. Thus we capture the exclusion of invalid sequences of pairs in a simple rule on linked building blocks in rule (iii).

A2.11. Numerical proof strip mode rules

Here we show numerical proof that to have class (iii) mode-scaling, a unit cell should support at least one strip mode. To show this, we use publicly available [91] randomly generated $k \times k$ unit cell designs ranging from size $k = 2$ to $k = 8$ generated in earlier work [79]. We check for each generated unit cell if it obeys the strip mode rules as formulated in Sec. 2.8.4 using simple matrix operations and checks [2]. We denote the number of strips in a unit cell that satisfy the strip mode rules as a_{rule} and compare this to the slope a we find from the scaling of $N_{\mathrm{ZM}} = an + b$. We do this for all possible $k = 3$ unit cells and approximately one million $k = 4, 5, 6$ unit cells, two million $k = 7$ unit cells and 1.52 million $k = 8$ unit cells.

We find that the true slope a and the number of strip modes supported by the unit cell according to the strip mode rules a_{rule} have perfect agreement [Fig. A2.2]. The numerical results thus strongly suggest that supporting a strip mode is a necessary requirement for a unit cell to have class (iii) mode-scaling and that the conjectured general strip mode rules dictate when a unit cell supports such a strip mode.

[2]See `https://uva-hva.gitlab.host/published-projects/CombiMetaMaterial` for code to check the strip mode rules.

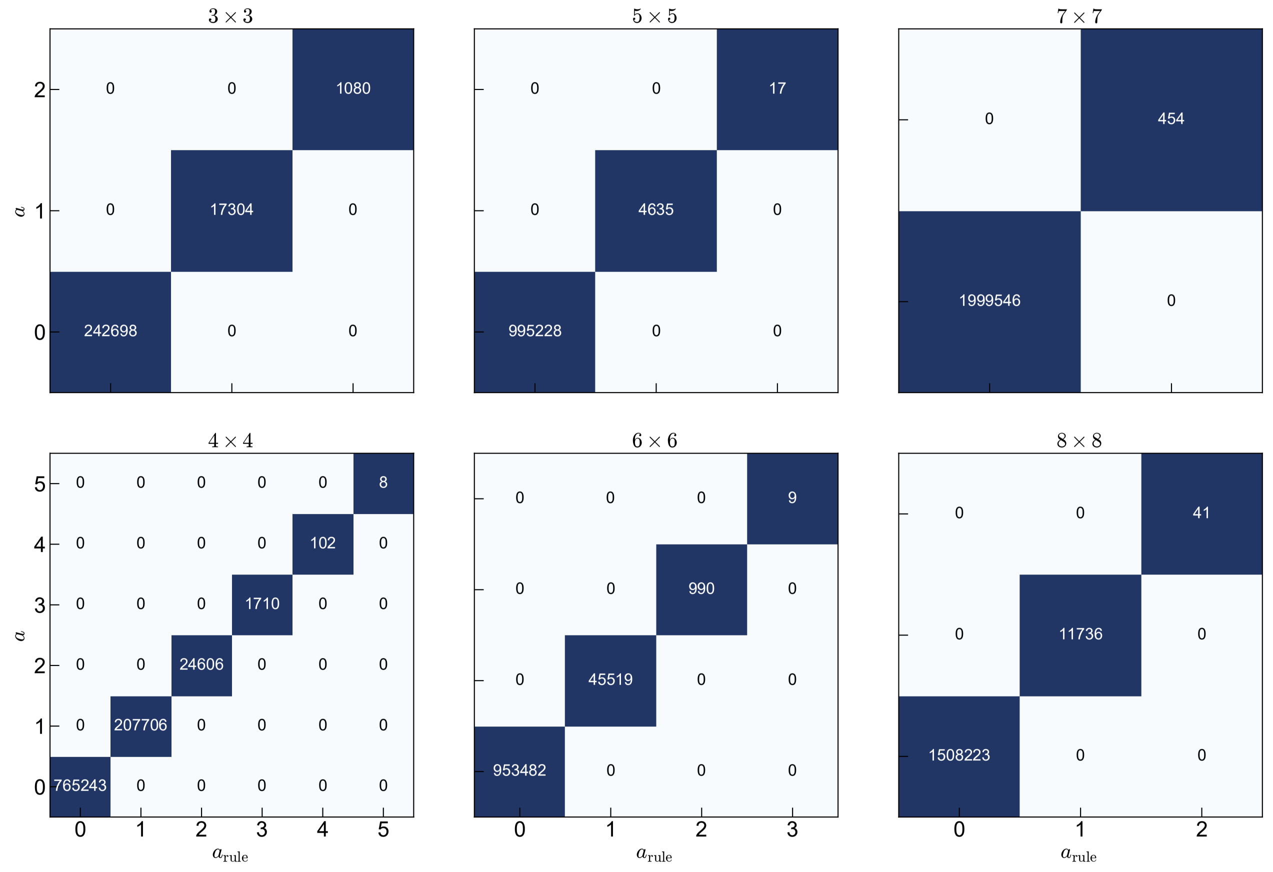


FIGURE A2.2: Confusion matrices comparing the slope a based on mode-scaling $N_{\text{ZM}}(n)$ to the number of strips that obey the strip mode rules a_{rule} as formulated in Sec. 2.8.4. The $k \times k$ unit cell size is indicated on top of each matrix.

3 Machine learning implicit combinatorial rules

Adapted from "Machine Learning Implicit Combinatorial Rules in Mechanical Metamaterials" published in Physical Review Letters, November 2022

Combinatorial problems arising in puzzles, origami, and (meta)material design have rare sets of solutions, which define complex and sharply delineated boundaries in configuration space. These boundaries are difficult to capture with conventional statistical and numerical methods. Here we show that convolutional neural networks can learn to recognize these boundaries for combinatorial mechanical metamaterials, down to finest detail, despite using heavily undersampled training sets, and can successfully generalize. This suggests that the network infers the underlying combinatorial rules from the sparse training set, opening up new possibilities for complex design of (meta)materials.

3.1. Introduction

From proteins and magnets to metamaterials, all around us systems with emergent properties are made from collections of interacting building blocks. Classifying such systems—do they fold, are they magnetized, do they have a target property—normally involves calculating these properties from their structure. This is often straightforward in principle, yet computationally expensive in practice, *e.g.* requiring the diagonalization of large matrices. Machine learning algorithms such as neural networks (NNs) forgo the need for such calculations by "learning" the classification of structures. In particular, machine learning has proven successful to find patterns in crumpling [92], active matter [93–95] and hydrodynamics [96], photonics [97–99], predict structural defects and plasticity [100, 101], design metamaterials [49, 51, 102–107], determine order parameters [108–115], identify phase transitions [50, 116–132] , and predict protein structure [133]. In these examples, the relevant property typically varies smoothly and there is no sharp boundary separating classes in configuration space. NNs are thought to be successful because they interpolate these blurred boundaries, even when the configuration space is heavily undersampled.

In contrast, combinatorial problems, *viz.* those where building blocks have to fit together as in a jigsaw puzzle, feature a sharp boundary between

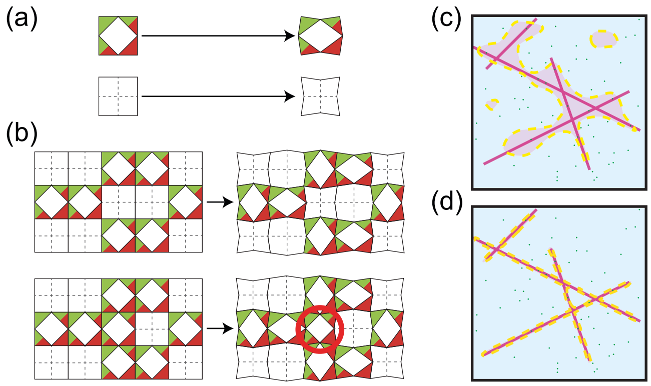


Figure 3.1: (a) The building block of [11] can be tiled in two orientations (left) that have a distinct deformation in two dimensions (right). (b) The building blocks of (a) combine into larger designs (structures) that are either C (top) or I (bottom). A change of a single building block can frustrate the deformation (red circle) and change the structure from one that hosts a zero mode (a deformation that costs no energy) (C) to one that does not host a zero mode (I). A set of rules can be formulated for a unit cell design to have a zero mode [11]. (c, d) Conceptual configuration spaces of a discrete combinatorial metamaterial problem. Class C (pink lines) exists in a background of class I (blue), is sensitive to perturbations, and has a complex filamentous structure. Distinguishing between a network with a "coarse" decision boundary (purple dashed line) (c) versus a network with a "fine" decision boundary (d) is not possible with the test set (green dots) due to the undersampled C-I boundary.

compatible (C) and incompatible (I) configurations. Such problems arise in self-assembly [20, 22], folding [17, 25], tiling problems [134] and combinatorial mechanical metamaterials [10, 11, 19, 65]. The latter are created by tiling different unit cells and are restricted by kinematic compatibility. A simple example is that of structures that can be either floppy (zero mode) or frustrated (no zero mode) [Fig. 3.1(a, b)]. The floppy structures require a specific arrangement of building blocks where all the deformations fit together compatibly (C), and therefore are rare and very sensitive to small perturbations. These perturbations are likely to induce frustrated incompatible (I) configurations [Fig. 3.1(b)]. The space of C designs can be pictured as needles in a haystack [Fig. 3.1(c, d)] and crucially is determined by a set of implicit combinatorial rules. Unless we know these rules, these problems are typically computationally intractable.

Here we show that convolutional neural networks (CNNs) are able to accurately perform three distinct classifications of combinatorial mechanical metamaterials and to generalize to never-before-seen configurations. Crucially, we find that well-trained CNNs can capture the fine structure of the boundary of C, despite being trained on sparse datasets. These results suggest that CNNs implicitly learn the underlying rule-based structure of combinatorial problems. This opens up the possibility for using NNs for efficient exploration of the design space and inverse design when the combinatorial rules are unknown.

Coarse vs. fine boundaries— The boundary between C and I configurations has the shape of needles in a haystack. Therefore, in a randomly sampled training set, this boundary will be typically undersampled, *e.g.* the training set will contain few I close to C (see App. A3.2.4 for more details on the undersampled C-I boundary in the training sets.). We argue that a NN simply interpolating the training data will misclassify most I configurations close to C, resulting in a "coarse" decision boundary around C [Fig. 3.1(c)]. Instead, an ideal NN should approximate the fine structure of the needles more closely, resulting in a "fine" decision boundary around C [Fig. 3.1(d)]. While this may sound impossible, let's recall that this fine structure ultimately arises from combinatorial rules. These rules are in principle much simpler than the myriad of compatible configurations C they can generate. Hence, the question is whether NNs could implicitly learn these rules and finely approximate the fine boundary with great precision. Although a NN can classify perfectly the metamaterial M1 of Fig. 3.1(a, b) (Tab. 3.1), this is not sufficient to address this question because the data set is too small and the C configurations are too rare to consider larger configurations (see App. A3.1.1 for more details on the design rules and rarity of metamaterial M1).

Table 3.1: Confusion matrices of trained CNNs with the lowest validation loss over the test set for the classification problems of Fig. 3.1(b) (M1), Fig. 3.2(d) (M2.i), and Fig. 3.2(e) (M2.ii).

		M1 predicted		M2.i predicted		M2.ii predicted	
		C	I	C	I	C	I
actual	C	19	0	685	1	43418	750
	I	0	4896	29	149265	453	105361

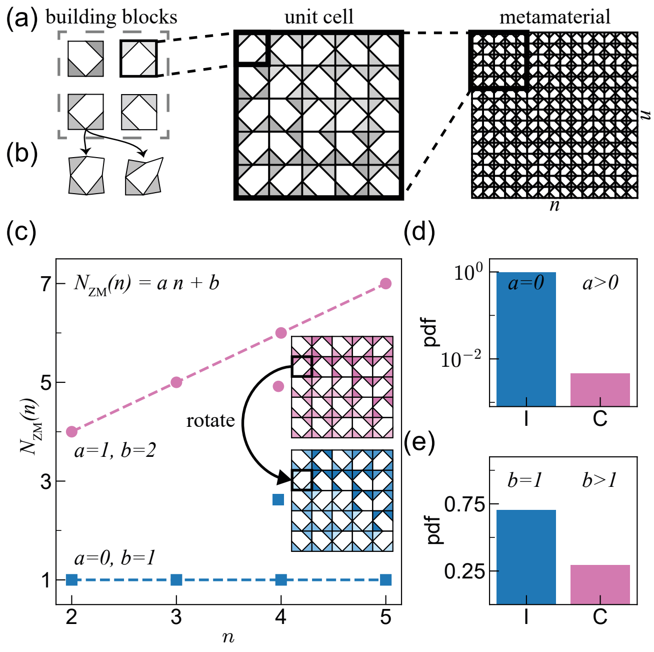


FIGURE 3.2: (a) Four two-dimensional building blocks (left), combined into a square 5×5 unit cell (middle), which is tiled on a $n = 3$ grid, form a combinatorial metamaterial (right). (b) The building blocks feature two zero modes and four orientations with distinct deformations. (c) The number of zero modes $N_{\text{ZM}}(n)$ as function of n for two unit cells. The pink unit cell (circles) differs by a point mutation from the blue unit cell (squares), yet the pink unit cell has $a = 1$ and $b = 2$ and the blue unit cell has $a = 0$ and $b = 1$. Thus the pink unit cell is classified as class C for both classification problems while the blue unit cell is classified as class I for both problems. (d) Probability density function (pdf) for classification problem (ii). Class C is more rare than class I. (e) Probability density function (pdf) for classification problem (i). Class C is much rarer than class I.

3.2. Metamaterial Classification

Therefore, to see if NNs are still able to learn the structure of C if the C-I boundary is undersampled, we consider another combinatorial metamaterial M2 [19] [for details on how we define it, see Sec. 1.2.2 and Fig. 3.2(a, b)]. While metamaterial M1 had a unit cell of size $k \times k$ with $k = 1$, metamaterial M2 has larger unit cell size—we focus on $k = 5$ in the Main Text and cover the cases $k = 3$ to 8 in the appendix. For such a metamaterial, the design space is too large to fully map and class C is rare, yet class C is abundant enough that we can create sufficiently large training sets to train

NNs.

The number of zero modes $N_{\mathrm{ZM}}(n)$ of a metamaterial consisting of $n \times n$ unit cells depends on the design of the unit cell: when the linear size n is increased, the number of zero modes $N_{\mathrm{ZM}}(n)$ either grows linearly with n or saturates at a non-zero value [Fig. 3.2(c)] as $N_{\mathrm{ZM}}(n) = an + b$, where a and b are positive integers. Accordingly, we can now specify two well-defined binary classification problems, which each feature a rare "compatible" (C) class and frequent "incompatible" (I) class [Fig. 3.2(d, e)]: (i) $a > 0$ (C) vs. $a = 0$ (I). The metamaterial with $a > 0$ hosts zero modes that are organized along strips, for which one can formulate combinatorials rules [see Ch. 2 for a detailed description of and numerical evidence for combinatorial rules of classification problem (i)]; (ii) $b > 1$ (C) vs. $b = 1$ (I). The metamaterial with $b > 1$ hosts additional zero modes—up to 6—that typically span the full structure and for which the rules still remain unknown despite our best efforts. In both classification problems, a single rotation of one building block in the unit cell can be sufficient to change class [Fig. 3.2(c)]. Hence, the boundary between classes C and I is sharp and sensitive to minimal perturbations as in the case of metamaterial M1 [Fig. 3.1(c)].

If the rules are unknown, the classification of this metamaterial requires the determination of $N_{\mathrm{ZM}}(n)$—via rank-revealing QR factorization [44]—as function of the number of unit cells n, which is computationally demanding. For $k \times k$ unit cells, the time it takes to compute this brute-force classification scales nearly cubically with input size k^2. In contrast, classification with NNs scales linearly with input size and is readily parallelizable. In practice this makes NNs invariant to input size due to computational overhead (see App. A3.4 for a more detailed description of the computational time comparison). Hence a trained NN allows for much more time-efficient exploration of the design space.

To train our NNs, we generate labeled data through Monte Carlo sampling the design space to generate 5×5 unit cells designs and explicitly calculate $N_{\mathrm{ZM}}(n)$ for $n \in \{2, 3, 4\}$ to determine the classification. We do this for a range of $k \times k$ unit cells with $3 \leq k \leq 8$. We focus on 5×5 but the results hold for other unit cell sizes (see App. A3.3.2 for CNN results of more unit cell sizes). The generated data is subsequently split into training (85%) and test (15%) sets. As our designs are spatially structured and local building block interactions drive compatible deformations, we ask whether convolutional neural networks (CNNs) are able to distinguish between class C and I. The input of our CNNs are pixelated representations of our designs. This approach facilitates the identification of neighboring building blocks that are capable of compatible deformations (see App. A3.2.1 for a more detailed description of the pixel representation). The CNNs are trained using 10-fold stratified cross-validation. Crucially, we use a balanced training set, where the proportion of class I has been randomly undersampled such that classes C and I are equally represented (see App. A3.2.3 for more details about the training sets for each metamaterial).

Despite the complexity of the classification problems, we find that the CNNs perform very well (Tab. 3.1). In particular, the CNNs correctly classify most class C unit cells as class C, and most class I unit cells as class I. However, the test set is likely to contain few examples of class I close to the C-I boundary, especially as C becomes more rare [Fig. 3.1(c), see App. A3.2.4]. Hence, whether our CNNs capture the complex boundary of C cannot be deduced from the test set alone. In other words, the CNNs find the needles in the haystack but it remains unclear whether the needles are approximated finely [Fig. 3.1(c)] or coarsely [Fig. 3.1(d)] [1].

3.3. Combinatorial structure

To probe the shape of both the true set of C configurations and the set of classified C configurations, we start from a true class C configuration, perform random walks in configuration space, and at each step probe the probabilities to be in the set of true class C [Fig. 3.3(a)]. We randomly change the orientation of a single random building block at each step $s \mapsto s+1$ and average over 1000 realizations (see App. A3.3.4 for a more detailed description of the random walks) The probability to remain in true class C, $\rho_{\mathrm{C}\to\mathrm{C}}(s)$, decreases with s and saturates to the class C volume fraction β for classification (i) and (ii) [Fig. 3.3(b)]. We note that we can fit this decay by a simple model, where we assume that subspace C is highly complex, so that the probabilities to leave it are uncorrelated. For every step, there is a chance α to remain C. Once in class I, we assume any subsequent steps are akin to uniformly probing the design space such that the probability to become C is equal to the C volume fraction β. Thus the probability to become C can be modeled as

$$\rho_{\mathrm{C}\to\mathrm{C}}(s) = \alpha^s + \beta\left(1-\alpha^{s-1}\right). \tag{3.1}$$

[1]We have observed from qualitative analysis of the 29 falsely classified C unit cells of M2.i that these unit cells can be transformed to true class C by changing a single or a few building blocks, thus they are close to the C-I boundary in design space.

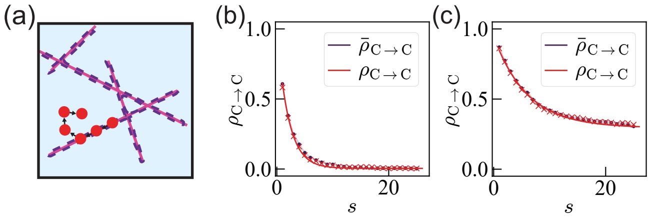


FIGURE 3.3: (a) Example of a 6-step random walk through design space (red dots) and sketch of the decision boundary of trained CNNs that has learned the combinatorial rules (purple dashed line). (b) Probabilities to remain in true and predicted class C under random walks of s steps, $\rho_{C\to C}(s)$ (red crosses) and fold-averaged $\langle\bar{\rho}_{C\to C}\rangle(s)$ (purple circles) with standard deviation (purple area), for classification (i) (left) and (ii) (right). The red continuous line is a least-squares fit to $\rho_{C\to C}(s)$ using Eq. (3.1).

The uncorrelated nature of the steps are consistent with a random needle structure [Fig. 3.1(c)], where the coefficient $\alpha \times 4^{5\times5}$ corresponds to the average dimensionality of the needles and β corresponds to their volume fraction. We can interpret α as the probability to not break the combinatorial rules when we randomly rotate a building block.

To see whether the CNNs are able to capture these key features of space C, we repeat our random walk procedure using the CNNs' classification instead, starting from true and classified C configurations, and obtain the probability $\bar{\rho}_{C\to C}(s)$. The decay of the fold-averaged $\langle\bar{\rho}_{C\to C}\rangle(s)$ closely matches that of the true class C for classification problems (i) and (ii) [Fig. 3.3(b, c)]. By fitting the predicted probability $\bar{\rho}_{C\to C}(s)$ for each fold to Eq. (3.1), using measurements of the CNN's predicted volume fraction $\bar{\beta}$ over the test set to constrain the fit, we obtain the fold-averaged dimensionality $\bar{\alpha}$. For classification (i) we find $\bar{\alpha} \approx 0.632 \pm 0.001$ closely matches the true $\alpha \approx 0.612 \pm 0.001$. In practice, α corresponds to the fraction of building blocks that are outside the relevant combinatorial strip. Using a simple counting argument, we find good agreement with the lower-bound of $\alpha \simeq 3/5$ (see App. A3.3.4). Similarly, for classification (ii) we find $\bar{\alpha} \approx 0.8514 \pm 0.0005$ closely matches $\alpha \approx 0.846 \pm 0.002$. Our results thus demonstrate that CNNs successfully capture on average the complex local shape of the combinatorial space C. Even though during learning the algorithm 'sees' very few class I unit cells that are close the C-I boundary, the decision boundary still captures on average the sparsity and fine structure of the class C subset. Thus we conclude that the CNNs infer the combinatorial rules [Fig. 3.1(c)], rather than interpolate the shape in high dimensional design space [Fig. 3.1(d)]. In other words, CNNs are able not only to capture accurately the volume fraction of the needles, but also to finely distinguish between needle and hay.

3.4. Volume before structure

But what happens with smaller CNNs? We focus on classification (i) and probe how well our CNNs—which consist of a single 20 filters convolution layer, a single n_h neurons hidden layer, and a two neurons output layer—capture the sparsity and structure of class C. First we compare their true and predicted volumes β and $\bar{\beta}(n_h)$ as a function of the number of hidden neurons n_h. The CNNs' predicted class C volume fraction $\bar{\beta}$ approaches the true class C volume fraction β as the number of hidden neurons n_h increases sufficiently, despite their balanced training set [Fig. 3.4(a)]. Next we compare the true and predicted dimensionality α and $\bar{\alpha}(n_h)$. While for small values of n_h, $\bar{\alpha}$ overestimates α, $\bar{\alpha}$ closely matches α for large n_h [Fig. 3.4(b)]. For small number of hidden neurons n_h, the CNNs overestimate both the probability to remain in class C and the rarity of class C; in other words, small CNNs coarsen the complex shape of C [Fig. 3.1(c)]. As seen above, for larger number of hidden neurons n_h both the probability and rarity of C are closely approximated, thus large CNNs finely capture the complex shape of C [Fig. 3.1(d)].

Strikingly, we observe that the predicted class C volume $\bar{\beta}$ more quickly reaches its asymptotic value than the dimensionality $\bar{\alpha}$. To see this, we plot $\bar{\beta}(n_h) - \beta$ versus $\bar{\alpha}(n_h) - \alpha$, which demonstrates that after $\bar{\beta}$ closely approximates β, increasing the number of hidden neurons n_h improves $\bar{\alpha}(n_h)$ towards its asymptote α [Fig. 3.4(c)]— this observation is also present for other unit cell sizes, see App. A3.3.5 for measurements of $\bar{\alpha}$ and $\bar{\beta}$ of classification problem (i) for more unit cell sizes. Thus, further increasing the size of the CNN beyond the point of marginal gain of test set performance results in a significantly more closely captured fine structure of C. In other words, to correctly capture the average dimensionality of the needles requires more neurons than to capture their volume.

3.5. Discussion

NNs are known to be universal approximators [45] and efficient classifiers. They often generalize well when the training data samples representative portions of the input space sufficiently, even for non-smooth [135] or noisy

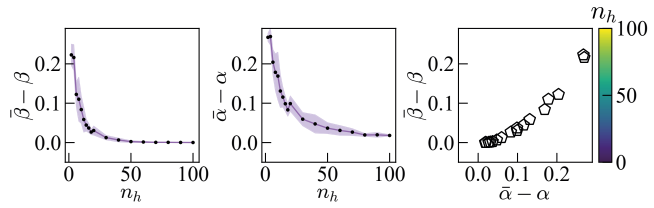


FIGURE 3.4: (a) Difference between predicted class C volume $\bar{\beta}(n_h)$ and true class C volume β as a function of number of hidden neurons n_h shows that $\bar{\beta}(n_h)$ approaches β for increasing n_h. (b) Difference between predicted dimensionality $\bar{\alpha}(n_h)$ and true dimensionality α obtained through least-squares fits to Eq. (3.1) as a function of the number of hidden neurons n_h shows that $\bar{\alpha}(n_h)$ approaches α for increasing n_h. (c) Scatter plots of class volumes $\bar{\beta}(n_h) - \beta$ versus dimensionality $\bar{\alpha}(n_h) - \alpha$ shows that the latter asymptotes later than the former (n_h indicated by colorbar). We use CNNs with a single convolution layer of 20 2×2 filters, which are spatially offset with respect to the unit cell and subsequently flattened. The flattened feature maps are fully-connected to a layer of n_h hidden neurons, which itself is fully-connected to two output neurons that correspond to class C and I. The CNNs are systematically trained using 10-fold stratified cross-validation for varying number of hidden neurons n_h.

data [136]. As combinatorial problems are sharply delineated and severely class-imbalanced, one expects that the fine details of an undersampled complex boundary would be blurred by NNs. Surprisingly, we have shown that CNNs will closely approximate such a complex combinatorial structure, despite being trained on a sparse training set. We attribute this to the underlying set of rules which govern the complex space of compatible configurations—in simple terms, the CNN learns the combinatorial rules, rather than the geometry of design space, which is the complex result of those rules [2].

Recognizing NNs' ability to learn these rules from a sparse representation of the design space opens new strategies for design. For instance, our CNNs could be readily used as surrogate models within a design algorithm to save computational time. Alternatively, one could instead devise a design algorithm based on generative adversarial NNs [137] or variational auto-encoders [138]. It is an open question whether and how such generative models could successfully leverage the learning of combinatorial

[2]We expect NNs to work beyond combinatorial metamaterials for a wide range of combinatorial problems in physics, such as spin-ice. These combinatorial rules in such problems can typically be translated to matrix operations, NNs naturally capture such matrix operations, and therefore we expect them to perform well.

rules [139].

Our work shows that metamaterials provide a compelling avenue for machine learning combinatorial problems, as they are straightforward to simulate yet exhibit complex combinatorial structure [Fig. 3.1(c)]. More broadly, applying neural networks to combinatorial problems opens many exciting questions. What is the relation between the complexity of the combinatorial rules and that of the networks? Can unsolved combinatorial problems be solved by neural networks? Can neural networks learn size-independent combinatorial rules? Conversely, can these problems help us understand why neural networks work so well [140]? Can they provide insight in how to effectively overcome strong data-imbalance [141]? We believe combinatorial metamaterials are well suited to answer such questions.

Data availability statement.—The code supporting the findings reported in this chapter is publicly available on GitLab [3] [4]—the data on Zenodo [91, 142].

Acknowledgements.—We thank David Dykstra, Marc Serra-Garcia, Jan-Willem van de Meent, and Tristan Bereau for discussions. This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. C.C. acknowledges funding from the European Research Council under Grant Agreement 852587.

[3]See `https://uva-hva.gitlab.host/published-projects/CombiMetaMaterial` for code to calculate zero modes and numerically check design rules.

[4]See `https://uva-hva.gitlab.host/published-projects/CNN_MetaCombi` for code to train and evaluate convolutional neural networks.

Appendix

In this appendix, we derive design rules for metamaterial M1, provide more details on our CNNs, analyze our CNNs' performances for different unit cell sizes of metamaterial M2, and show that our CNNs are more computationally efficient than directly calculating the mode scaling $N_{\mathrm{ZM}}(n)$.

A3.1. Floppy and frustrated structures

In this section, we discuss in more detail the metamaterial M1 of Fig. 3.1. We first derive the design rules that lead to floppy structures, then we discuss the rarity of such structures.

A3.1.1. Design rules for floppy structures

Here we provide a brief overview of the rules that lead to floppy structures for the combinatorial metamaterial M1 of Fig. 3.1. The three-dimensional building block of this metamaterial can deform in one way that does not stretch any of the bonds: it has one zero mode (see Sec. 1.2.1 and [11] for details of the unit cell). In two dimensions, there are two orientations of the building block that deform differently in-plane. We label these two orientations as green/red and white [Fig. 3.1(a)].

We can formulate a set of rules for configurations of these building blocks in two dimensions. Configurations of only green/red building blocks or white building blocks deform compatibly (C): the configuration is floppy. A single horizontal or vertical line of white building blocks in a configuration filled with green/red building blocks also deforms compatibly. More lines (horizontal or vertical) of white blocks in a configuration filled with green/red blocks deform compatibly if the building block at the intersection of the lines is of type green/red [Fig. 3.1(b)].

In summary, we can formulate a set of rules:

i All white building blocks need to be part of a horizontal or vertical line of white building blocks.

ii At the intersection of horizontal and vertical lines of white building blocks there needs to be a green/red building block.

If these rules are met in a configuration, the configuration will be floppy (C). A single change of building block is sufficient to break the rules, creating an incompatible (I) frustrated configuration [Fig. 3.1(b)].

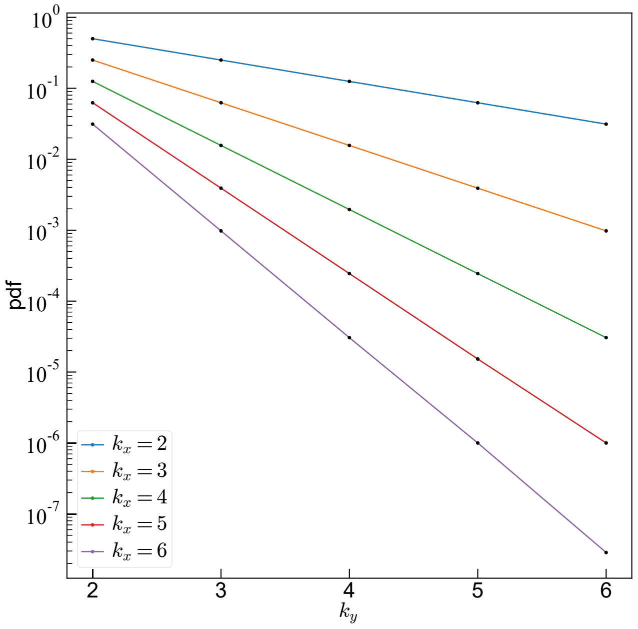


FIGURE A3.1: Probability density function (pdf) of $k_x \times k_y$ class C configurations.

A3.1.2. Rarity of floppy structures

Here we show how the rarity of class C in combinatorial metamaterial M1 depends on the size of the $k_x \times k_y$ configuration. To show this, we simulate configurations with varying $k_x, k_y \in \{2, 3, 4, 5, 6\}$. The size of the design space grows exponentially as $2^{k_x k_y}$, yet the fraction of class C configurations decreases exponentially with unit cell size (Fig. A3.1). Thus the number of C configuration scale with unit cell size at a much slower rate than the number of total configurations. For large configuration size, the number of C configurations is too small to create a sufficiently large class-balanced training set to train neural networks on.

A3.2. Constructing and Training Convolutional Neural Networks for metamaterials

In this section, we describe in detail how we construct and train our convolutional neural networks (CNNs) for classifying unit cells into class I and C. We first transform our unit cells to a CNN input, second we establish

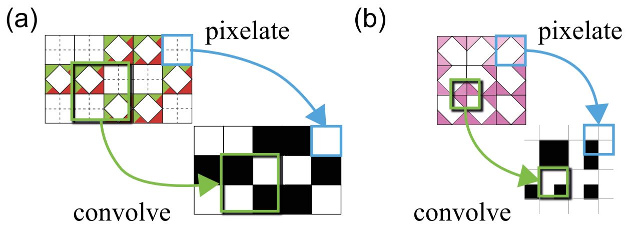


Figure A3.2: Unit cell designs of the combinatorial metamaterials in Fig. 3.1(a)-(b) and Fig. 3.2(a), and their respective pixel representations. The blue squares indicates how the building blocks are transformed to pixels, the green squares show which part of the unit cell is convolved by the first convolution layer.

the architecture of our CNNs. Next, we obtain the training set, and finally we train our CNNs.

A3.2.1. Pixel Representation

To feed our design to a neural network, we need to choose a representation a neural network can understand. Since we aim to use convolutional neural networks, this representation needs to be a two-dimensional image. For our classification problem, the presence or absence of a zero mode ultimately depends on compatible deformations between neighboring building blocks. As such, the representation we choose should allow for an easy identification of the interaction between neighbors.

In addition to being translation invariant, the classification is rotation invariant. While we do not hard code this symmetry in the convolutional neural network, we do choose a representation where rotating the unit cell should still yield a correct classification. For example, this excludes a representation where each building block is simply labeled by a number corresponding to its orientation. For such a representation, rotating the design without changing the numbers results in a different interplay between the numbers than for the original design. Thus, we cannot expect a network to correctly classify the rotated design.

For both metamaterials, we introduce a *pixel* representation. We represent the two building blocks of metamaterial featured in Fig. 3.1 as either a black pixel (1) or a white pixel (0) [Fig. A3.2(a)]. A $k_x \times k_y$ unit cell thus turns into a $k_x \times k_y$ black-and-white image.

Likewise, we introduce a *pixel* representation for the metamaterial M2 of Fig. 3.2(a) which naturally captures the spatial orientation of the building blocks, and emphasizes the interaction with neighboring building blocks. In this representation, each building block is represented as a 2×2 matrix, with one black pixel (1) and three white (0) pixels, see Fig. A3.2(b). The black pixel is located in the quadrant where in the bars-and-hinges representation the missing diagonal bar is. Equivalently, this is the quadrant where in the directed graph representation the diagonal edge is located. Moreover, in terms of mechanics, this quadrant can be considered floppy, while the three others are rigid.

This representation naturally divides the building blocks into 2×2 *plaquettes* in which paired building blocks are easily identified, see Fig. A3.2(b). Building blocks sharing their black pixel in the same plaquette are necessarily paired, and thus allow for deformations beyond the counter-rotating squares mode. Note that this includes diagonally paired building blocks as well. By setting the stride of the first convolution layer to $(2, 2)$, the filters only convolve over the plaquettes and not the building blocks, which do not contain any extra information for classification.

A3.2.2. CNN architecture details

To classify the unit cells into class I and C, we use a convolutional neural network (CNN) architecture. We first discuss the architectures used to obtain the results of Tab. 3.1. Then we discuss the architecture used to obtain the results of Fig. 3.4.

For the metamaterial M1 of Fig. 3.1, the CNN consists of a single convolution layer with 20 2×2 filters with bias and ReLu activation function. The filters move across the input image with stride $(1, 1)$ such that all building block interactions are considered. Subsequently the feature maps are flattened and fully-connected to a hidden layer of 100 neurons with bias and ReLu activation function. This layer subsequently connected to 2 output neurons corresponding to C and I with bias and softmax activiation function. The input image is not padded. Since a network of this size was already able to achieve perfect performance, we saw no reason to go to a bigger network.

For the metamaterial M2 of Fig. 3.2 and classification problem (i) we first periodically pad the input image with a pixel-wide layer, such that a $2k \times 2k$ image becomes a $2k + 2 \times 2k + 2$ image. This image is then fed to a convolutional layer, consisting of 20 2×2 filters with bias and ReLu activation function. The filters move across the input image with stride $(2, 2)$, such that the filters always look at the parts of the image showing the interactions between four building blocks [Fig. A3.2(b)]. Subsequently the 20 $k + 1 \times k + 1$ feature maps are flattened and fully-connected to a hidden layer of 100 neurons with bias and ReLu activation function. This layer is then fully-connected to 2 output neurons corresponding to the two classes with bias and softmax activation function. From the hyperparameter grid search (see Sec. A3.2.5) we noted that this n_f and n_h were sufficiently large for good performance.

For classification (ii) we again pad the input image with a pixel-wide layer. The CNN now consists of three sequential convolutional layers of increasing sizes 20, 80, and 160 filters with bias and ReLu activation function. The first convolution layer moves with stride $(2,2)$. The feature maps after the last convolutional layer are flattened and fully-connected to a hidden layer with 1000 neurons with bias and ReLu activation function. This layer is fully-connected to two output neurons with bias and softmax activation function. This network is larger than for classification (i); we saw noticeable improvements over the validation set when we considered larger networks. This is most likely a result of the (unknown) rules behind classification (ii) being more complex.

The networks are trained using a cross-entropy loss function. This loss function is minimized using the Adam optimization algorithm [143]. This algorithm introduces additional parameters to set before training compared to stochastic gradient descent. We keep all algorithm-specific parameters as standard ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\mathrm{e}-07$), and only vary the learning rate η from run to run. The network for the classification problem of Fig. 3.1 uses a weighted cross-entropy loss function, where examples of C are weighted by a factor 200 more than examples of I.

To obtain the results of Fig. 3.4, we use the architecture of classification (i) and vary the number of neurons n_h in the hidden layer. We keep the number of filters the same. To obtain this architecture, we performed a hyperparameter grid search, where we varied the number of filters n_f of the convolution layer and the learning rate η as well. The details are discussed in the Sec. A3.2.5. The total number of parameters for this network with n_f filters and n_h neurons is

$$N_{\text{params}} = (4+1)n_f + ((k+1)^2 n_f + 1)n_h + (n_h+1)2. \tag{A3.1}$$

A3.2.3. Training set details

Each classification problem has its own training set. For the classification problem of Fig. 3.1, the networks are trained on a training set D_t of size $|D_t| = 27853$ that is artificially balanced 200-to-1 I-to-C. Classification problem (i) has a class balanced training set size of $|D_t| = 793200$. Problem

(ii) has a training set size of $|D_t| = 501850$. For the classification problems (i) and (ii), the class is determined through the total number of modes $N_{\mathrm{ZM}}(n)$ as described in Sec. 3.2. For the metamaterial M1 of Fig. 3.1, we determine the class through the rules as described in Sec. A3.1.

Since there is a strong class-imbalance in the design space, for the network to learn to distinguish between class I and C, the training set is class-balanced. If the training set is not class-balanced, the networks tend to learn to always predict the majority class. The training set is class-balanced using random undersampling of the class I designs. For problem (i), with the strongest class-imbalance, the number of class C designs is artificially increased using translation and rotation of class C designs. We then use stratified cross-validation over 10 folds, thus for each fold 90% is used for training and 10% for validation. The division of the set changes from fold to fold. To pick the best performing networks, we use performance measures measured over the validation set.

To show that our findings are robust to changes in unit cell size, we also train CNNs on classification problem (i) for different $k \times k$ unit cell sizes. The size of the training set D_{t} for each unit cell size k is shown in Tab. A3.1. Increasing the unit cell size increases the rarity of C and the size of the design space. This leads a more strongly undersampled C-I boundary as we will show in the next section.

A3.2.4. Sparsity of the training set

To illustrate how sparse the training set is for classification problem (i), we divide the number of training unit cells per class, $|D_t(\mathrm{Class})|$ over the estimated total number of $k \times k$ unit cells of that class, $|\Omega_D(\mathrm{Class})|$. We estimate this number for class C through multiplying the volume fraction of class C β in a uniformly generated set of unit cells with the total

Table A3.1: Details of the hyperparameter grid search.

k	size of D_t	size of D_{test}
3	31180	39321
4	397914	150000
5	793200	149980
6	1620584	150000
7	292432	600000
8	1619240	144000

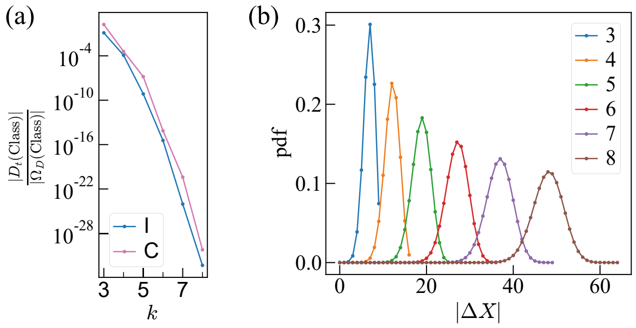


FIGURE A3.3: Training set details for classification problem (i) of metamaterial M2. (a) Fraction of the total unit cells of class C that are in the training set. (b) Average absolute distance $|\Delta X|$ in number of building blocks between class C and class I unit cells in the training set.

number of possible unit cells $|\Omega_D| = 4^{k^2}$: $|\Omega_D(\text{C})| \approx \beta|\Omega_D|$. Likewise, we determine the ratio for class I. The resulting ratio for class C and I is shown in Fig. A3.3(a). Clearly, for increasing unit cell size k, the class sparsity in the training set increases exponentially. Consequently, the neural networks get relatively fewer unit cells to learn the design rules bisecting the design space for increasing unit cell size.

Moreover, the training set unit cells of different class are, on average, farther removed from one another for increasing unit cell size k. The distance between two unit cells $|\Delta X|$ is defined as the number of building blocks with a different orientation compared to their corresponding building block at the same spatial location in the other unit cell. So two $k \times k$ unit cells can at most be k^2 building blocks removed from one another, if every single building block has a different orientation compared to its corresponding building block at the same spatial location in the other unit cell. Note that we only consider *different* orientations in this definition, we do not define an additional notion of distance between orientations of building blocks.

By measuring the distance in number of different building block orientations $|\Delta X|$ between every class C to every class I unit cell, we obtain the probability density function of distance in number of different building blocks between two unit cells of different class in the training set, see Fig. A3.3(b). Consequently, if k increases, the networks are shown fewer examples of unit cells similar to each other, but of different class. Thus the boundary between C and I is undersampled in the training set, with few I designs close to the boundary.

A3.2.5. CNN hyperparameter grid search details

To see how convolutional neural network (CNN) size impacts classification performance, a hyperparameter grid search is performed. We focus on classification problem (i), which features a shallow CNN with a single convolution layer and single hidden layer as described in Sec. A3.2.2. This search varied three hyperparameters: the number of filters n_f, the number of hidden neurons n_h, and the learning rate η. The number of filters n_f runs from 2 to 20 in steps of 2, the number of hidden neurons n_h first runs from 2 to 20 in steps of 2, then from 20 to 100 in steps of 10. The learning rate ranges from $\eta \in \{0.0001, 0.001, 0.002, 0.003, 0.004, 0.005\}$. For each possible hyperparameter combination, a 10-fold stratified cross validation is performed on a class-balanced training set. Early stopping using the validation loss is used to prevent overfitting.

To create the results of Fig. 3.4, n_f has been fixed to 20 since most of the performance increase seems to come from the number of hidden neurons n_h after reaching a certain treshhold for n_f as we will show in Sec. A3.3. The best η is picked by selecting the networks with the highest fold-averaged accuracy over the validation set.

A3.3. Assessing the performances of CNNs

In this section, we describe in detail how we assess the performance of our trained convolutional neural networks (CNNs). We first quantify performance over the test set, then we define our sensitivity measure. Finally, we apply this sensitivity measure to the CNNs.

A3.3.1. Test set results

After training the CNNs on the training sets, we test their performance over the test set. The test set consists of unit cells the networks have not seen during training, and it is not class-balanced. Instead, it is highly class-imbalanced, since the set is obtained from uniformly sampling the design space. In this way, the performance of the network to new, uniformly generated designs is fairly assessed.

For the classification problem of metamaterial M1, the test set D_{test} has size $|D_{\text{test}}| = 4915$. Classification problem (i) for metamaterial M2 has test set size $|D_{\text{test}}| = 149982$. Problem (ii) for M2 has test set size $|D_{\text{test}}| = 149980$.

Precisely because the test set is imbalanced, standard performance measures, such as the accuracy, may not be good indicators of the actual performance of the network. There is a wide plethora of measures to choose from [144]. To give a fair assessment of the performance, we show the confusion matrices over the test sets for the trained networks with the lowest loss over the validation set in Tab. 3.1.

A3.3.2. Varying the unit cell size

To see how the size of the unit cell impacts network performance, we performed a hyperparameter grid search as described in Sec. A3.2.5 for $k \times k$ unit cells ranging from $3 \leq k \leq 8$. We focus on classification problem (i). The size of the test set D_{test} is shown in Tab. A3.1.

To quantify the performance of our networks in a single measure, we use the Balanced Accuracy:

$$\begin{aligned} \text{BA} &= \left\langle \frac{1}{2}\left(\frac{V_{\text{TC}}}{V_{\text{TC}}+V_{\text{FI}}}+\frac{V_{\text{TI}}}{V_{\text{TI}}+V_{\text{FC}}}\right)\right\rangle && \text{(A3.2)} \\ &= \left\langle \frac{1}{2}\left(\text{TCR}+\text{TIR}\right)\right\rangle, && \text{(A3.3)} \end{aligned}$$

where V_{TC}, V_{TI}, V_{FC}, and V_{FI} are the volumes of the subspaces true class C TC, true class I TI, false class C FC, and false class I FI [Fig. 3.1(c, d)]. We do not consider other commonly used performance measures for class-imbalanced classification, such as the F_1 score, since they are sensitive to the class-balance.

The BA can be understood as the arithmetic mean between the true class C rate TCR (sensitivity), and true class I rate TIR (specificity). As such, it considers the performance over all class C designs and all class I designs separately, giving them equal weight in the final score. Class-imbalance therefore has no impact on this score.

Despite the complexity of the classification problem, we find that, for sufficiently large n_f and n_h, the balanced accuracy BA approaches its maximum value 1 for every considered unit cell size k (Fig. A3.4). Strikingly, the number of filters n_f required to achieve large BA does not vary with k. This is most likely because the plaquettes encode a finite amount of information—there are only 16 unique 2×2 plaquettes. This does not change with unit cell size k, thus the required number of filters n_f is invariant to the unit cell size. The number of required hidden neurons n_h increases with k, but not dramatically, despite the combinatorial explosion of the design space. To interpret this result, we note that a high BA

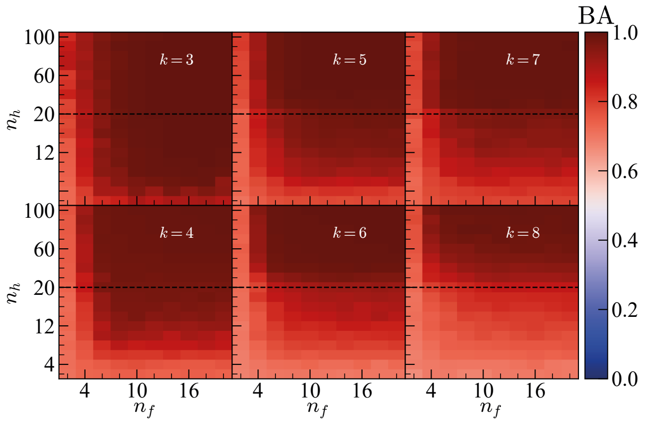


FIGURE A3.4: Heatmaps of the fold-averaged balanced accuracy BA for CNNs with n_f filters and n_h hidden neurons trained on $k \times k$ unit cells indicated on top of each heatmap.

corresponds to correctly classifying most class C unit cells as class C, and most class I unit cells as class I. Hence, sufficiently large networks yield decision boundaries such that most needles are enclosed and most hay is outside [Fig. 3.1(c, d)]. However, whether this decision boundary coarsely [Fig. 3.1(c)] or finely [Fig. 3.1(d)] approximates the structure close to the needles cannot be deducted from a coarse measure such as the BA over the test set.

The usage of BA to show trends between neural network performance and hyperparameters is warranted, since no significant difference between the true class C rate TCR and true class I rate TIR appears to exist, see Fig. A3.5. Evidently TCR and TIR depend similarly on the number of filters n_f and number of hidden neurons n_h. This is to be expected, since the networks are trained on a class-balanced training set.

The effect of class-imbalance on CNN performance can be further illustrated through constructing the confusion matrices [Fig. A3.6(b)]. Though all CNNs show high true C and I rates, the sheer number of falsely classified C unit cells can overtake the number of correctly classified C unit cells if the class-imbalance is sufficiently strong, as for the 7×7 unit cells.

A3.3.3. Increasing the size of the training set

To illustrate how the size of the training set D_t influences the performance over the test set, we compare CNNs trained on two training sets of different

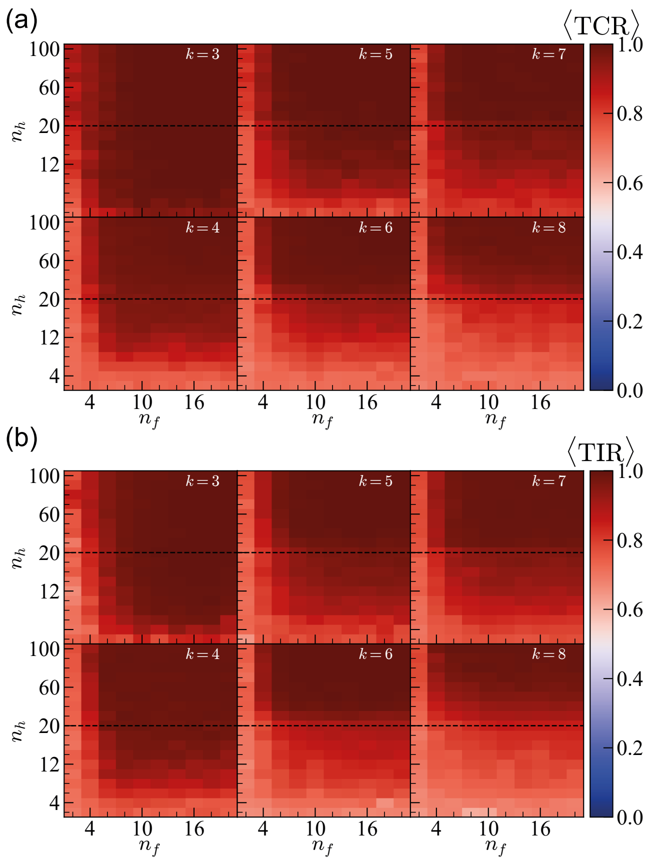


FIGURE A3.5: (a) Heatmaps of the fold-averaged true class C rate ⟨TCR⟩. (b) Heatmaps of the fold-averaged true class I rate ⟨TIR⟩.

size consisting of 7×7 unit cells—the unit cell size with the strongest class-imbalance. We use the fold-averaged balanced accuracy BA to quantify the performance. The training sets are obtained from 1M and 2M uniformly sampled unit cells respectively, and the number of class C unit cells is artificially increased using translation and rotation to create class-balanced training sets. The best BA is more than a factor 2 smaller for CNNs trained on the larger training set, compared to the smaller training set (Fig. A3.7). Thus, lack of performance due to a strong data-imbalance can be improved through increasing the number of training samples.

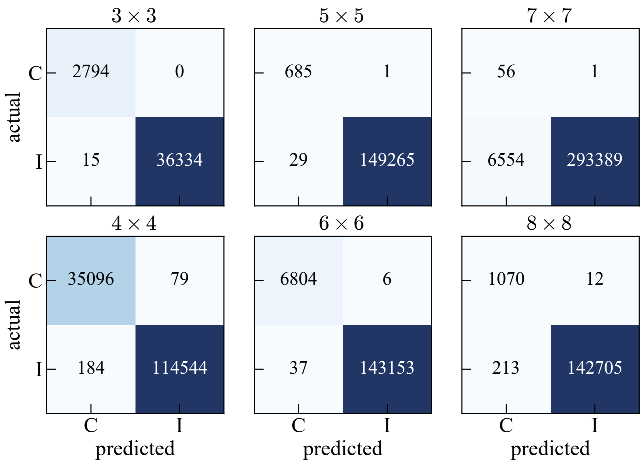


Figure A3.6: Confusion matrices over the test set for trained CNNs with the highest accuracy over the class-balanced validation set. The $k \times k$ unit cell size is indicated on top of each matrix.

A3.3.4. Random walk near the class boundary

To better understand the complexity of the classification problem, we probe the design space near test set unit cell designs. Starting from a test set design X_0 with true class C, we rotate a randomly selected unit cell to create a new unit cell design X_1. We do this iteratively up to a given number of steps s to create a chain of designs. For each generated design, we assess the new true class using the design rules for classification (i) and through calculating $N_{\mathrm{ZM}}(n)$ for $n \in \{3, 4\}$ for classification (ii).

For each unit cell size k, we take $s = k^2$ steps in design space. The probability to transition from an initial 5×5 design X_0 of class C to another design X_s of class C as a function of s random walk steps in design space $p_{\mathrm{C}\to\mathrm{C}}(s)$, is shown in Fig. 3.3(b, c) for classification problems (i) and (ii).

We repeat the random walks for other $k \times k$ unit cells for problem (i). A clear difference between the different unit cell sizes is visible. Both the rate at which the probability decreases initially, and the value to which it saturates differs per unit cell size (Fig. A3.8).

For even unit cell size, the dominant strip mode width is $W = 1$ (Fig. A3.9) and each class C design is most likely to just have a single strip mode. Thus, the probability to transition from C to I relies on the probability to rotate a unit cell inside the strip of the strip mode, which is $1/k$, so $\alpha_t \approx 1/k$. For odd unit cell sizes, the dominant strip mode width is

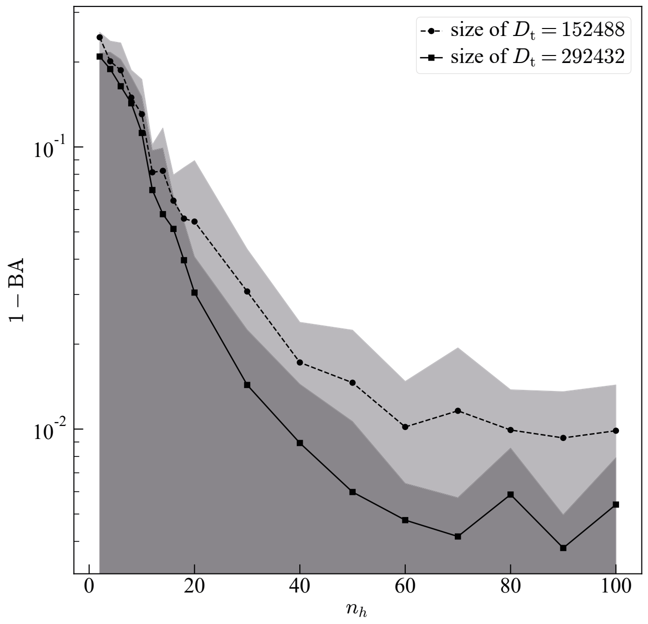


FIGURE A3.7: Balanced accuracy BA for CNNs with $n_f = 20$ trained on a smaller training set (circles) and larger training set (squares). The size of the training set is indicated in the legend.

$W = 2$, such that $\alpha_t \approx 2/k$.

To understand the asymptotic behavior, we note that for large s the unit cells are uncorrelated to their original designs. Thus, the set of unit cells are akin to a uniformly sampled set of unit cells. Consequently, the probability to transition from C to C for large s is approximately equal to the true class C volume fraction β.

A3.3.5. Random walk near the decision boundary

In addition to the true class, we can assess the predicted class by a given network for each unit cell in the random walk. This allows us to probe the decision boundary, which is the boundary between unit cells that a given network will classify as C and those it will classify as I. By comparing the transition probabilities for given networks to the true transition probability we get an indication of how close the decision boundary is to the true class boundary.

To quantitatively compare the true class boundary with the decision

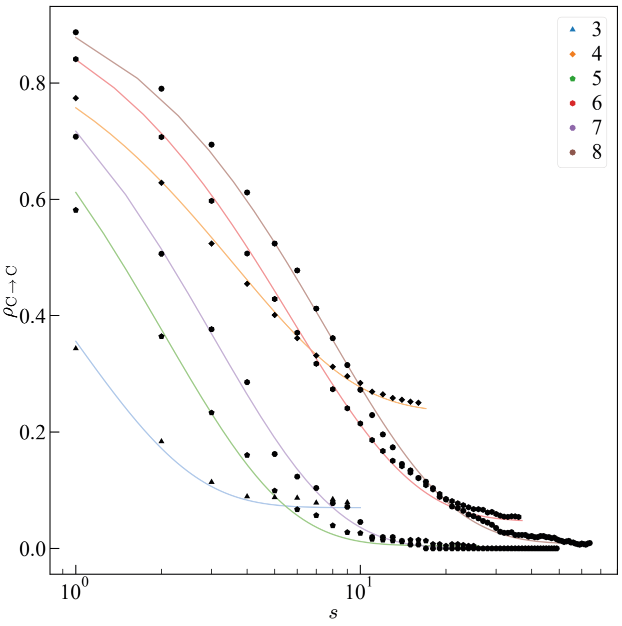


FIGURE A3.8: Probability $\rho_{\mathrm{C}\to\mathrm{C}}$ (polygons) to transition from initial design X_0 of class C to another design X_s of class C as a function of s random walk steps in design space starting from the initial design. The legend indicates the polygon and color for each unit cell size k. The continuous lines are obtained from a least-squares fit using Eq. (3.1).

boundaries, we fit the measured transition probability for each network to Eq. (3.1) with $\bar{\alpha}$ as fitting parameter. We start from designs with true and predicted class C, and track the predicted class for the random walk designs. We set the asymptotic value to the predicted class C volume fraction $\bar{\beta}$ [Fig. A3.10(a)] for each network. From this we obtain a 10-fold averaged estimate of $\bar{\alpha}$.

Additionally, we do this for varying unit cell size k for classification problem (i) using the hyper parameter grid search networks. We use CNNs with fixed number of filters $n_f = 20$ and varying number of hidden neurons n_h. We select the networks with the best-performing learning rate η over the validation set, and obtain a 10-fold averaged estimate of $\bar{\alpha}$ for each n_h [Fig. A3.10(b)].

Small networks tend to overestimate the class C dimensionality α [Fig. A3.10(b)]. Larger networks tend to approach the true α for increasing number of hidden neurons n_h. For large data-imbalance, as is the case for

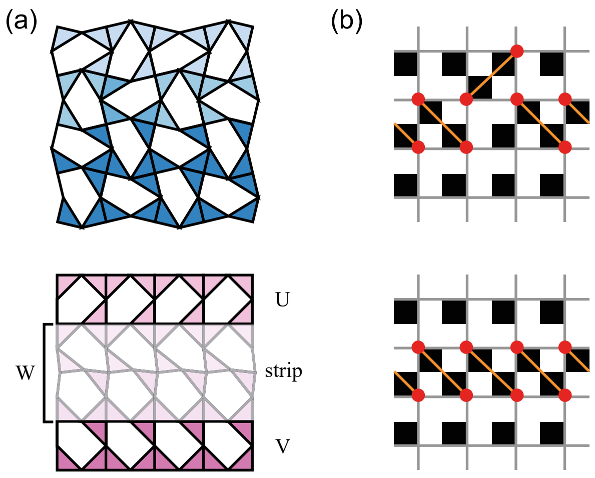


Figure A3.9: Schematic and pixel representation of modes in a 4×4 unit cell. (a) Schematic deformation of counter-rotating squares mode (top unit cell, blue) and a strip mode (bottom unit cell, pink). The strip mode spans the entire area of the strip (white) of width $W = 2$, while the areas U and V do not deform. (b) Respective pixel representations of the left unit cells. Paired unit cells are highlighted through red dots connected by orange lines. Note that the top unit cell does not contain a strip that meets the strip mode rules, while the bottom unit cell does.

$k = 7$ and $k = 8$, even the larger networks overestimate α. This is not a fundamental limitation, and can most likely be improved by increasing the size of the training set, see Sec. A3.3.3. We conjecture that this is due to the higher combinatorial complexity of the C subspace for larger unit cells, which requires a larger number of training samples to adequately learn the relevant features describing the subspace. The trend shown in Fig. 3.4(c) holds across all unit cell sizes [Fig. A3.10(c)].

A3.4. Computational time analysis

In this section we discuss the computational time it takes to classify a $k \times k$ unit cell design by calculating the number of zero modes $N_{\text{ZM}}(n)$ for $n \in \{2, 3, 4\}$ using rank-revealing QR (rrQR) decomposition. The first algorithm takes as input a unit cell design, creates compatibility matrices $\mathcal{C}$ for each n, and calculates the dimension of the null space for each matrix using rrQR decomposition. The classification then follows from the determination of a and b in $N_{\text{ZM}}(n) = an + b$ as described in Ch. 3.

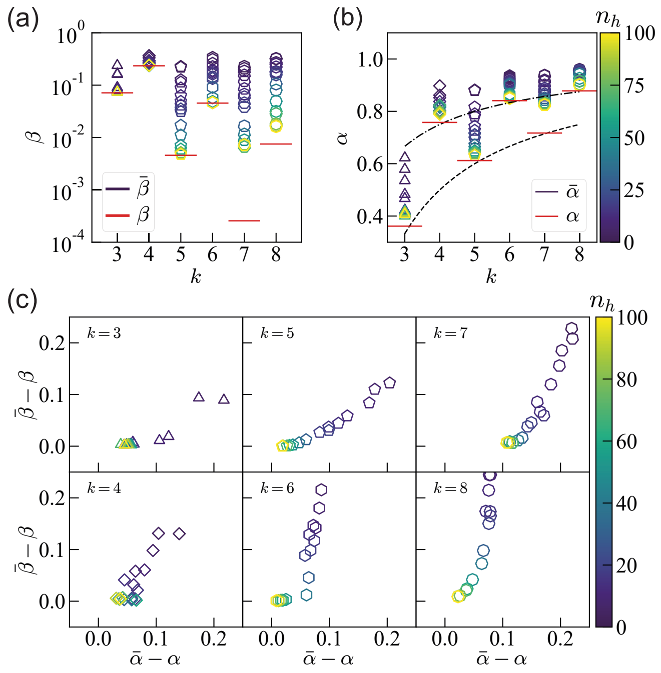


FIGURE A3.10: (a) Classification problem (i) Class C volume fraction β (red) as a function of unit cell size k. The predicted class C volume fraction $\bar{\beta}(n_h)$ (for $n_f = 20$) approaches β for increasing number of hidden neurons n_h (colorbar). (b) True dimensionality α (red) and predicted dimensionality $\bar{\alpha}(n_h)$ (colorbar) obtained through least-squares fits to data as in Fig. 3.3(b) for all k. The estimated α for both odd (dashed line) and even (dashdotted line) k agree well with α. (c) Scatter plots of class volume fractions $\bar{\beta}(n_h) - \beta$ versus dimensionality $\bar{\alpha}(n_h) - \alpha$ shows that the latter asymptotes later than the former (n_h indicated by a colorbar, and unit cell size k indicated on top of each graph)

We contrast this brute-force calculation of the class with a trained neural networks time to compute the classification. We consider a shallow CNN with a single convolution layer of $n_f = 20$ filters, a single hidden layer of $n_h = 100$ hidden neurons and an output layer of 2 neurons. The network takes as input a $k \times k$ unit cell design in the pixel representation (with padding) and outputs the class. The time complexity of a forward evaluation of the network is of order $\mathcal{O}(N_{\text{params}})$, since only the number of parameters changes under an increase of input size (the number of neurons in subsequent layers remain fixed) and each extra parameter constitutes an extra multiplication operation. The number of parameters of these CNNs grows linearly with input size k^2, see Eq. (A3.1). We focus on networks trained on classification problem (i).

The brute-force calculation scales nearly cubically with input size k^2, while the neural network's computational time remains constant with unit cell size k. This highlights the advantage of using a neural network for classification: it allows for much quicker classification of new designs. In addition, the neural network is able to classify designs in parallel extremely quickly: increasing the number of unit cells to classify from 1 to 1000 only increased the computational time by a factor ≈ 1.33.

Please note that this analysis does not include the time to train such neural networks, nor the time it takes to simulate a large enough dataset to train them. Clearly there is a balance, where one has to weigh the time it takes to compute a sufficiently large dataset versus the number of samples that they would like to have classified. For classification problems (i) and (ii) it did not take an unreasonable time to create large enough datasets, yet brute-forcing the entire design space would take too much computational time. Our training sets are large enough to train networks on—of order 10^5—but are still extremely small in comparison to the total design space, such that the time gained by using a CNN to classify allows for exploring a much larger portion of the design space as generating random designs is computationally cheap.

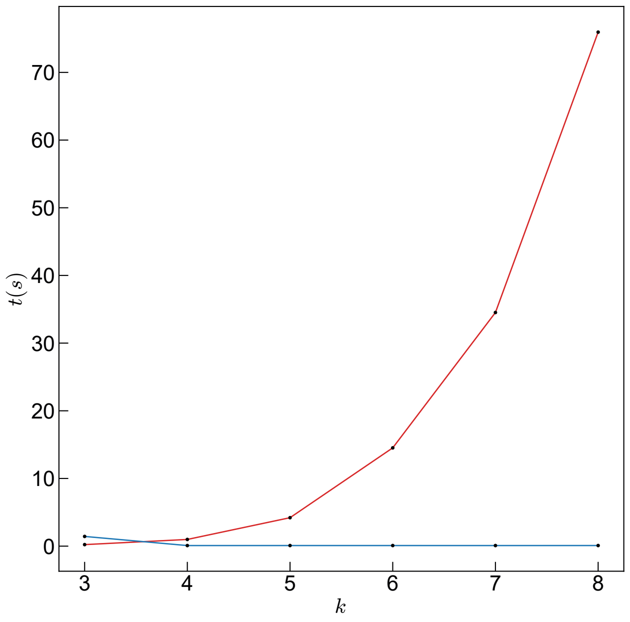


Figure A3.11: Computation time t measured in seconds s to classify $k \times k$ unit cells by modescaling (red) versus using neural network (blue).

4 Data-driven design

From self-assembly and protein folding to combinatorial metamaterials, a key challenge in material design is finding the right combination of interacting building blocks that yield targeted properties. Such structures are fiendishly difficult to find—not only are they rare, often the design space is so rough that direct optimization is hopeless. Here, we design ultra-rare combinatorial metamaterials capable of multiple desired deformation modes by introducing a two-fold strategy which avoid the drawbacks of direct optimization. We first focus on optimizing for pluripotency, which we define as a statistical measure for potential performance; for our specific case, pluripotency is determined by the number of spatially extended deformations of a candidate design. Subsequently, we select and refine high-potential designs by strategically placing defects, thus obtaining designs that match the challenging target properties. Specifically, we use a combination of convolutional neural networks and genetic algorithms to effectively explore the design space. Our design approach enables us to obtain designs surpassing those attainable through standard optimization, thereby facilitating the discovery and design of pluripotent (meta)materials. In general, our approach represents a new paradigm for systematic and data-driven design within large, intractable design spaces, and is readily applicable to a broad spectrum of combinatorial problems beyond metamaterial design.

4.1. Introduction

Data-driven methods are revolutionizing the way we design materials, particularly in guiding self-assembly [145, 146], designing soft materials [104, 147], engineering proteins [148, 149] and designing metamaterials [48, 52, 150–153]. In most cases, the predicted property is continuous, and straightforward optimization of an objective function suffices to find a design with the desired properties. In contrast, for combinatorial designs with discrete properties that are sensitive to changes in each building block, the design space is large, high-dimensional and discontinuous [79]. Typically, targeted designs are rare exceptions in a sea of random, failed designs and cannot be found through optimization of a direct objective function. Such jagged combinatorial spaces are ubiquitous, for example, in design for ground states in self-assembly [21, 23, 24], transition graphs in amorphous matter [7, 26–29], isomers in molecular design [30, 31], and deformation modes in mechanical metamaterials [14, 19, 79, 154] The latter are particularly interesting because multiple distinct deformations enable exceptional functionalities, such as selective mechanical responses [19, 154], nonlocal resonances [14], multi-shape folding [15–17], and sequential energy-absorption [18].

Achieving multiple specific target deformations is an extremely challenging design problem. In principle, multiple deformations enable multiple on-demand properties; yet, in practice, increasing the number of deformation modes too much renders the material floppy and hinders control. For example, designing metamaterials with multiple target deformations is extremely challenging due to the risk of generating superfluous undesired deformations that impede the actuation of the desired deformations [88]. Moreover, deformations are sensitive to changes in the design—changing a single building block may prohibit one or multiple deformation modes. Consequently, targeted designs with multiple desired deformations are extremely rare and direct optimization is unfeasible. It remains an open question how to systematically find rare designs with multiple desired properties in large, jagged combinatorial spaces.

Here, we introduce a new design approach that embraces superfluous deformations to find such rare designs. We consider a family of mechanical metamaterials with multiple ordered deformation modes [Fig. 4.1(a)-(b)]. To find rare designs with desired deformations, we propose to initially search for designs with high *pluripotency*, rather than high fitness. We define pluripotency P as a statistical measure that quantifies the performance of a design over a class of properties—of which the desired property constitutes a much smaller subset. In the context of our metamaterials, pluripotency P is positively correlated with the number of intensive zero modes b [Fig. 4.1(c)]. To illustrate our proposed approach, we draw an analogy to the process of finding, extracting, and refining rare metals such as gold, which are scattered across the earth in trace amounts. Because we focus on pluripotency rather than fitness, our design approach consists of two steps: (i) *prospecting* for highly pluripotent candidate designs—high-grade soil—using a genetic algorithm (GA) guided by a convolutional neural network (CNN) [Fig. 4.2(a)] and (ii) *extracting* and *refining* candidate designs to find a design with targeted properties [Fig. 4.2(b)].

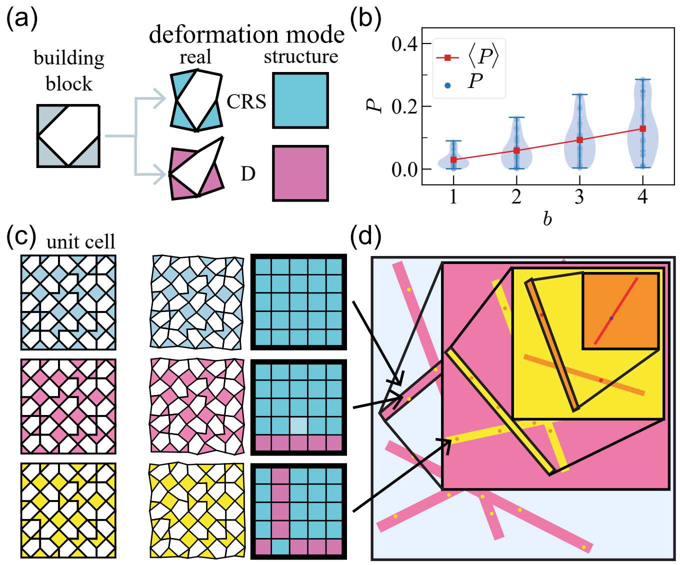


FIGURE 4.1: **Hierarchical structure of pluripotent metamaterials.** (a) The building block (left, gray) can be tiled in four orientations and features two distinct zero-energy deformations, zero modes, (middle) in two dimensions, which we label CRS (top, cyan) and D (bottom, pink). We visualize the modal structure of a deformed building block as a cyan or pink square (right). (b) The building blocks of (a) combine into larger 5×5 unit cells (left) that feature intensive zero modes (middle) upon tiling the unit cell. The mode structure of such intensive modes is shown on the right. The three unit cells differ only by a single building block to their neighbors, yet each unit cell supports the zero modes of their top neighbors in addition to the zero mode directly right to the unit cell. The number of intensive modes b thus increases from top to bottom and such designs occupy subspaces of increasing codimension in the design space of (d) as indicated by the arrows. (c) An increasing number of intensive modes b correlates to a higher average capacity [Eq. (4.3)] or pluripotency P per design (blue circles). The pluripotency averaged over random designs (red squares) increases with b. The blue shaded area is a violin plot to visualize the distribution of P values. (d) Conceptual configuration space of a discrete combinatorial multimodal metamaterial problem. Random point-mutations of the design—e.g. rotating or modifying a single building block—generally lower the number of intensive modes b, most designs have a low b and we have few examples of rare designs with higher values of b. Here, satisfying a hierarchy of conditions leads to increasingly low-dimensional subspaces where b has larger values (areas of the same color). We represent such a structure as needles (solid lines) in hay (square background), where each needle in turn contains higher codimension needles (circles). Such a structure forms a hierarchy (insets).

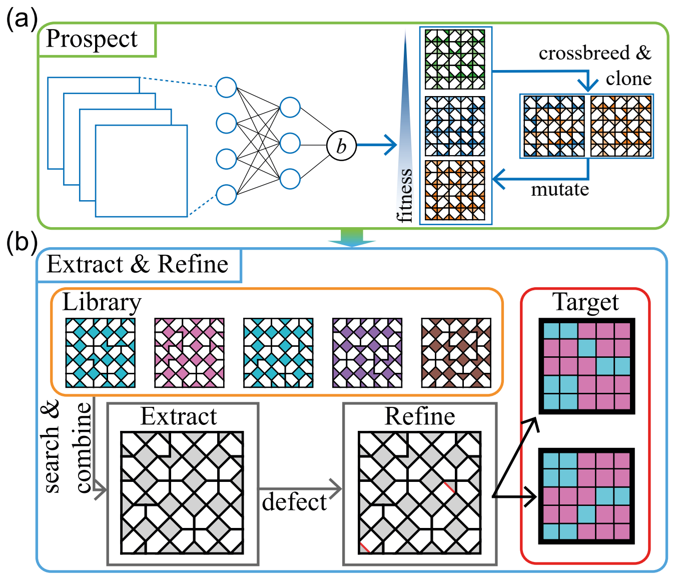


Figure 4.2: **Design strategy for combinatorial multimodal metamaterials.** (a) Step (i) of our design approach uses a convolutional neural network (CNN) to predict the number of intensive modes b to guide a genetic algorithm (GA) to efficiently generate high b (highly pluripotent) designs. (b) The generated highly pluripotent designs are added to a library. We then search and combine designs from this library to find a design that closely matches a set of target deformations (red): This process is called extraction. However, our extracted design often requires further refinement. We achieve this by strategically introducing defects to the building blocks, a step we refer to as refining. This yields a final refined design that features the desired target deformation modes while minimizing undesired superfluous modes.

In step (i), we aim to find highly pluripotent designs by increasing the number of intensive modes b. We have indications for the existence—but no examples—of designs with high values of b. Crucially, highly pluripotent designs are not randomly distributed in design space; instead they follow a structure we describe as 'needles-within-needles-within-needles in a haystack' [Fig. 4.1(d)]. This hierarchical structure is characteristic of pluripotency and stands in stark contrast to the more discontinuous, jagged structure of the direct objective function in design space. Analogous to prospecting for high-grade gold, we aim to exploit this structure to find ultra-rare highly pluripotent designs. To achieve this, our CNN must extrapolate and identify designs with a high number of intensive modes b outside the scope of the training set. However, it remains an open question whether CNNs are suitable for such a task.

In step (ii), we leverage our highly pluripotent candidate designs to create a design that supports the desired target deformation modes. Specifically, we begin by filling a library with these highly pluripotent candidate designs, similar to extracting gold from various sources. Next, we combine these candidate designs to form a new design that satisfies the target property. However, this new design also exhibits many other undesired, superfluous modes, akin to imperfections in gold. Crucially, the process of removing such undesired modes—refining—is much easier than finding them initially: while a single building block can break a mode, supporting a mode requires an intricate configuration.

In brief, we demonstrate how a combination of a CNN and a GA can discover previously unseen, rare, highly pluripotent (high b) designs for combinatorial metamaterials with multiple targeted deformation modes. Additionally, we illustrate how these discovered designs can be systematically combined to generate new designs with desired deformation modes. First, we observe that our CNN provides reasonable predictions for designs with large b, even when its training set comprises designs with lower b. Second, we show that a GA using such a CNN can efficiently identify numerous designs with large b—facilitating the creation of a library of highly pluripotent designs. Third, we use this library to rationally select and combine designs that feature the desired targeted modes. Finally, we introduce directed defects to the constituent building blocks to prohibit superfluous modes, resulting in designs that feature only the targeted modes. As a final proof of concept, we employ our methodology to design larger metamaterials that feature complex spatially textured modes, such as those resembling a smiley and a frowny face or the letters A and U. Previously, designing for targeted deformations was limited to a single target mode for specific monomodal metamaterials governed by established design rules [11, 17, 37, 154]. Now, employing our data-driven approach we can design for multiple target deformations in multimodal metamaterials without the necessity of knowing the design rules. Thus, our two-step approach opens up the possibility to use machine learning to efficiently design metamaterials possessing ultra-rare properties that were previously unattainable. We anticipate that our approach will be applicable to a wide range of combinatorial problems characterized by multiple, independent sets of constraints to satisfy. Such problems can readily be found in, e.g., protein folding [155, 156], self-assembly [21, 24], computer graphics [32, 34], and molecular design [157].

4.2. Multimodal metamaterial

To test our design approach, we consider the spatial structure of zero modes—infinitesimal deformations that do not stretch any bonds to first order—in a multimodal combinatorial metamaterial [Fig. 4.1(a)- (b)] [19, 79, 154]. In Ch. 3, we focused on the scaling behavior of the number of zero modes, $N_{\text{ZM}}(n)$, for an $n \times n$ tiling of identical unit cells, each comprising a $k \times k$ tiling of building blocks. In particular, we found that

$$N_{\text{ZM}} = an + b\,, \tag{4.1}$$

and we developed CNNs capable of distinguishing between so-called incompatible (I) designs with $a = 0$ and compatible (C) designs with $a \geq 1$. Furthermore, we identified that type C designs require the presence of a specific type of mode we referred to as 'strip modes', and demonstrated how these modes can be used to sequentially absorb energy [18]. In Ch. 2, we have also described a set of combinatorial rules that delineate these modes. However, for the design of zero modes with complex geometrical structures strip modes are limiting due to their deformations being localized to one-dimensional strips. Hence, in this chapter, we focus on spatially extended, intensive modes, the number of which is denoted by b. Since design rules for these more spatially complex intensive modes are not known, the construction of an appropriate search algorithm is necessary. We note that our metamaterial—regardless of the configuration of building blocks—always supports an intensive, global mode equivalent to the well-known counter-rotating squares (CRS) mode [19, 38, 158], ensuring that $b \geq 1$ [Fig. 4.3(a)].

We aim to design such intensive modes and set out on finding a design that features the set of target zero modes $\{\hat{M}\}$. Rather than designing for the exact deformations of every kinematic degree of freedom, we focus on the higher-level *structure* of the mode. To define this structure, we decompose the zero modes into a contribution from the trivial global CRS mode, and non-trivial deformations—we focus on the spatial structure of the latter. To do so, we characterize the structure of a zero mode M by writing the deformations of the constituent building blocks m_i as $m_i = \alpha_i m_{CRS} + \beta_i m_D(c_i)$, where i labels the building block. Here α and β are (rational) coefficients for the trivial zero mode m_{CRS} and non-trivial modes $m_D(c_i)$, respectively, where c_i denotes the orientation of the building block i— m_{CRS} remains independent of the building block orientation c. We classify building block deformations with $\beta = 0$ as CRS blocks and those with $\beta \neq 0$ as D blocks, and characterize the structure of a target zero mode $\hat{M}$ by the spatial distribution of CRS and D blocks. For more details on the zero mode structure of our metamaterial, we refer to [154].

To quantify the pluripotency of a given $k \times k$ design, we first define the similarity of a mode $\{\alpha_i, \beta_i\} \in M$ to the target mode $\{\hat{\alpha}_i, \hat{\beta}_i\} \in \hat{M}$ as

$$S(M, \hat{M}) = \frac{1}{k^2} \sum_{i=1}^{k^2} \begin{cases} \delta(\beta_i, 0), & \text{if } \hat{\beta}_i = 0 \\ 1 - \delta(\beta_i, 0), & \text{if } \hat{\beta}_i \neq 0 \end{cases} \tag{4.2}$$

where δ represents the Kronecker delta. We divide by the number of building blocks k^2, such that the maximum capacity is 1. Thus, the similarity $S(M, \hat{M})$ denotes the fraction of matching CRS and D blocks between mode M and the target $\hat{M}$.

The similarity $S(M, \hat{M})$ is defined for a single given deformation mode M, while a design usually supports a set of basis zero modes $\{M^B\}$ of size N_B rather than a single mode. We define the capacity of a set of modes as the maximum similarity of all linear combinations of the modes in the set, i.e.:

$$C(\{M^B\}, \hat{M}) = \max_{\mathbf{w}} \left[S(\sum_i w_i M_i^B, \hat{M}) \right] - \hat{N}_{\mathrm{CRS}}, \tag{4.3}$$

where $\mathbf{w}$ is the vector of basis-mode weights w_i of dimension N_B. The maximum $S(\sum_i w_i^* M_i^B, \hat{M})$ is characterized by the weights $\mathbf{w}^*$ which we compute using constraint programming (see App. A4.1.3). To ensure that the lowest capacity is 0, we subtract the fraction of CRS blocks in the target mode $\hat{N}_{\mathrm{CRS}} = \sum_{i=1}^{k^2} \delta(\hat{\beta}_i, 0)/k^2$. This is because the trivial solution $\mathbf{w} = \mathbf{0}$ corresponds to a mode M composed solely of CRS blocks, resulting in a complete overlap of CRS blocks in mode M and the target $\hat{M}$. We define the pluripotency P of a design as the average capacity C computed over a randomized set of target modes (see App. A4.1.4).

We find that pluripotency averaged over randomly selected designs $\langle P \rangle$ steadily increases with the number of intensive modes b of those designs [Fig. 4.1(c))]. Thus, we take b as a proxy for pluripotency P. Intuitively, a design that features more zero modes has more deformational degrees of freedom and thus, with the appropriate combination of modes, it is more likely to deform close to a randomized target mode.

Yet, we note that simply increasing the deformational freedom of a metamaterial is not sufficient—such an approach may indeed enhance the pluripotency, but at the expense of introducing a large number of superfluous modes. Such a floppy material would be practically useless.

Balancing pluripotency with the number of modes poses a significant challenge [88]. By increasing the number of intensive modes b, which are likely to span the entire structure, we effectively increase the pluripotency without introducing too many superfluous modes. Therefore, our focus lies on the number of intensive modes b.

4.3. Predicting intensive modes

First, we consider step (i) of our design approach: prospecting for highly pluripotent designs. Generally, our aim is to find designs that satisfy a large, independent number of conditions—such designs are more likely to satisfy randomized target properties. This set of conditions in combinatorial configuration spaces describes a hierarchy of embedded sets with increasing codimension that resembles a needles-within-needles structure [Fig. 4.1(d)]. Thus, finding needles near the top of the hierarchy is improbable without informed navigation of the configuration space.

To find these rare needles, we aim to use combinatorial optimization algorithms that iteratively explore the design space. However, the computational bottleneck of such algorithms lies in evaluating the property to steer navigation. Calculating this property, the number of intensive modes b, requires determining $N_{\mathrm{ZM}}(n)$–using rank-revealing QR decomposition [44, 159]—as a function of the number of unit cells n, which is computationally demanding. In previous work, we demonstrated that convolutional neural networks (CNNs) are adapt at classifying combinatorial problems, even in undersampled regions [79]. Moreover, a trained CNN significantly improves the computational time complexity by two orders of magnitude and is readily parallelizable [79], providing an immense speed-up for population-based combinatorial optimization. Consequently, a trained CNN allows for a more efficient exploration of the design space. Therefore, we use CNNs to provide an effective approximate calculation of the property b. In contrast to our previous work, we now ask the network to extrapolate. In other words, we investigate whether the CNN can discern an hierarchical structure within the property and identify regions of the configuration space that exhibit ultra-rare properties outside the range of the training set.

To address this question, we explore the design space of 5×5 unit cells, as this size offers a good balance between spatial complexity, the size of the design space, and the rarity of the number of intensive modes b. First, we need to generate training data for the CNN. We generate this data by Monte Carlo sampling of the design space for 5×5 unit cells, and subsequently, calculate $N_{\mathrm{ZM}}(n)$ for $n \in \{2, 3, 4\}$ to determine b. The generated data is then divided into a training (85%) and a test (15%) set. Because our designs are spatially structured and local interactions between building blocks drive compatible deformations, CNNs are well-suited for predicting the number of intensive modes b. We transform our designs into pixelated black-and-white images as input for our CNNs, facilitating straightforward identification of neighboring building blocks capable of deforming compatibly (see App. A4.2.3).

Our CNNs are trained on imbalanced datasets—designs with large values of b are increasingly rare [Fig. 4.3(a)]. Crucially, there is structure within the design space, which we explore by characterizing the changes in b under point mutations of the design, defined as a random rotation of one of the building blocks [Fig. 4.3(b)]. Comparing the number of intensive modes of the initial design, b_I, to the number of modes of the mutated design, b_{NN}, we find that the most likely scenario is that $b_{NN} = b_I$, showing that designs with the same value of b are interconnected. The next most likely scenario is that $b_{NN} = b_I - 1$, showing that designs with b zero modes are surrounded by designs with $b - 1$ zero modes. To gain further insight, we have explored the ratio r of neighbors that have $b_{NN} < b_I$, and find that this ratio increases with b_I, showing that subspaces with large b have increasingly small dimensionality [Fig. 4.3(c)]. This means that the subspaces of the design space with constant b become progressively sparse and low-dimensional with increasing b. Consequently, the structure of b in design space resembles a hierarchy of needles-within-needles [Fig. 4.1(d)], which the CNN may infer. The CNNs are trained using 10-fold cross validation, and we use the CNN with the lowest loss over the validation set.

Despite the sparsity of designs with a high number of intensive modes b in the training set, we find that the CNN remains accurate for high b in the test set (Tab. 4.1). In particular, the output of the CNN, b_{CNN}, is always close to the actual number of intensive modes b, albeit with a slight underestimation bias. Thus, the CNN is able to accurately delineate the needles within the bounds of the training data. However, whether the CNNs can identify needles beyond the range of the training set and can help in finding ultra-rare designs with $b > 6$ cannot be deduced from the test set alone. In other words, it remains an open question at this point whether the trained CNN can infer the hierarchical structure of needles-within-needles and extrapolate accordingly.

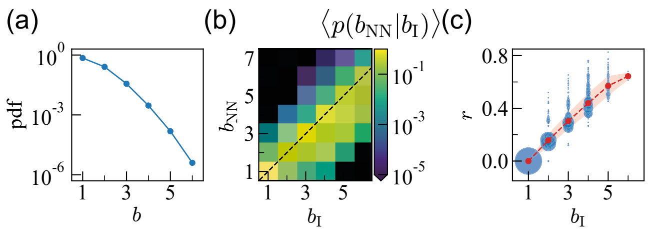


FIGURE 4.3: **Statistical characterization of the design space.** (a) Probability density function (pdf) for the number of intensive modes b obtained through Monte Carlo sampling of 5×5 unit cells. (b) The average probability $\langle p(b_{\mathrm{NN}}|b_{\mathrm{I}})\rangle$ of finding a nearest neighbor unit cell with b_{NN} intensive modes by changing a single building block for a given initial unit cell with b_{I} intensive modes. To highlight the low probability of transitioning to a higher b, a logarithmic colormap (colorbar) is used, with zero (measured) probability denoted in black. (c) The ratio r (red dashed line, shaded area indicates the standard deviation) of nearest neighbor unit cells with $b_{\mathrm{NN}} \leq b_{\mathrm{I}}$ averaged over random unit cells with b_{I} intensive modes and increases with b_{I}. The ratio varies for individual unit cells (blue circles, where the size indicates the number of unit cells with the same r) and appears multimodal in structure. This is likely a consequence of the different types of intensive modes: edge and global (App. A4.1.5).

4.4. Extrapolation

To probe the rare regions of the design space characterized by a high number of intensive modes b, we employ a genetic algorithm (GA) to iteratively progress towards designs with higher b (see App. A4.3). We observe that the combination of nonlocal and local exploration facilitated by the GA ensures efficient exploration of the design space. In short, the GA is able to combine designs to generate new designs with an average

TABLE 4.1: Confusion matrix for the test set of the CNN with the lowest validation loss.

		predicted b_{CNN}				
		1	2	3	4	5
actual b	1	105767	47	0	0	0
	2	418	37877	38	0	0
	3	1	242	5101	16	0
	4	0	0	66	383	3
	5	0	0	1	5	17

b, after which random mutations eventually allow the GA to reach a high b. However, generating high b designs with the GA requires a significant number of calls to the evaluation function. Without a fast approximate evaluation function, such as a neural network, generating high b designs would be unfeasible (see App. A4.3). Specifically, we use the GA to find designs with a target number of intensive modes b_T by maximizing the fitness

$$f = \frac{1}{1 + (b_{CNN} - b_T)^2}. \tag{4.4}$$

The fitness f is maximal when the CNN's prediction equals the targeted number of intensive modes, i.e., when $b_{CNN} = b_T$.

Surprisingly, starting from a Monte Carlo sampled set of unit cell designs, the GA consistently reaches its maximum fitness within a finite number of generations, even when the number of target intensive modes exceeds the range of the training data, i.e., $b_T > 6$ [Fig. 4.4(a)]. Increasing b_T requires, on average, more generations to converge; this is expected as the fraction of designs in the design space decreases exponentially with b [Fig. 4.3(a)]. Crucially, we find that a significant fraction of the GA-generated designs indeed feature $b = b_T = b_{CNN}$ intensive modes [Fig. 4.4(b)]. The CNN systematically overestimates the number of intensive modes, but remains close in prediction to the true number. Thus, our CNN is able to extrapolate and predict quantities beyond the range of the training data, specifically, our CNN can identify designs with $b > 6$. Thanks to this extrapolation, the CNN-guided GA successfully discovers extremely rare designs that feature up to $b = 9$ intensive modes. We estimate that such designs represent only a fraction of $\mathcal{O}(10^{-8})$ of the total design space and can be found within minutes on a desktop computer. Thus, we conclude that CNNs can extrapolate to properties outside the range of the training data by inferring the hierarchical structure in combinatorial problems. In other words, CNNs can accurately identify not only observed needles but also extrapolate the hierarchical structure of needles-in-needles to identify unseen needles.

4.5. Designing for target deformations

Now, we proceed to step (ii) of our design method: transforming our GA-generated high-pluripotency candidate designs into a design with targeted deformation modes. We accomplish this in three substeps. First, we *organize*: starting from a library of highly pluripotent (high b) initial

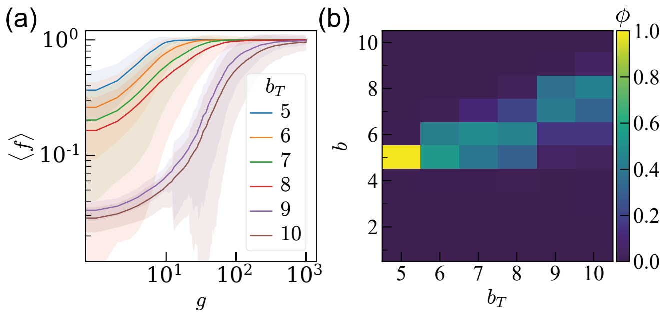


Figure 4.4: **Extrapolation of the trained CNN.** (a) Average (solid line) and standard deviation (shaded area) fitness f [Eq. (4.4)] of the fittest design per generation g over two hundred GA runs per target number of intensive modes b_T (legend). (b) Fraction $\phi(b)$ of fittest designs with b intensive modes for a given target b_T across two hundred GA runs per b_T.

designs [Fig. 4.5(a)], we define cliques (sets of compatible deformation modes derived from initial designs) and list all maximal cliques: the largest sets of modes that can be supported by the same design [Fig. 4.5(b)]. Second, we *extract*: we evaluate the capacity C [Eq. (4.3)] of these maximal cliques with respect to a target deformation mode, select the clique with the highest capacity, and consolidate the constituent designs of the clique into a new, highest-capacity, single candidate design that supports all modes in the clique [Fig. 4.5(c)]. Third, we *refine*: generally, the candidate designs feature not only the target modes but also superfluous modes, which we eliminate by introducing point defects in our design [Fig. 4.5(d)-(e)]. Below, we demonstrate that this strategy successfully finds 5×5 designs that satisfy randomized target deformation modes with little superfluous modes. In addition, we show that these 5×5 candidate designs can be combined into 10×10 designs with specific target deformation modes. Finally, we design a metamaterial that features deformation modes whose mode structures resemble complex spatial structures, such as a smiley and frowny face [Fig. 4.7 (b)-(c)] or the letters A and U [Fig. 4.7(d)-(e)].

To start, we first employ a GA, guided by our trained CNN, to generate a library of 1000 designs with a target number of intensive modes of $b_T = 7$. The distribution of the actual values of b for these 1000 designs is centered around $b = 6$ [Fig. 4.6(a)]. Next, we explicitly compute all zero modes for each design (see App. A4.1.2). As our target modes span the entire structure, we neglect edge modes where the D sites are all located on the edge of the unit cell. This reduces the number of modes in the library

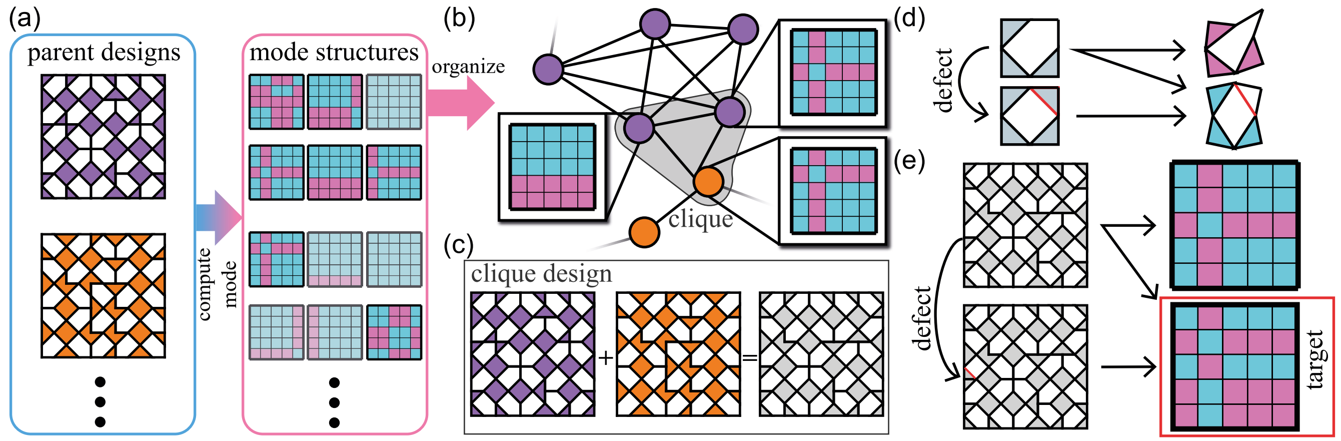


FIGURE 4.5: **Extracting and refining highly pluripotent designs.** (a) The basis of our library is composed of GA-generated designs (left), which we organize based on their mode structures (right). We discard the common CRS mode (all blue) and simple edge modes (single-block-wide strip(s) of D blocks located at the edge), as indicated by the grayed-out mode structures. (b) Individual modes from distinct parent designs (indicated by color) are represented as nodes in a graph. Nodes are connected by edges if the corresponding modes are compatible. Each maximal clique (fully-connected nodes enclosed in gray area) corresponds to a new design [see (d)] that supports all constituent modes. (c) The clique in (b) comprises modes from two parent designs (purple and orange) which we combine into a new design (gray) that supports all three modes in the clique. (d) Introducing a defect (extra rigid bar, red) to a building block (bottom left) prohibits the D mode (top right, pink) while retaining the CRS mode (bottom right, cyan). Arrows indicate the modes supported by the configuration. (e) Strategic placement of an extra rigid bar (red), or defect, can prohibit an undesired mode structure (top right) while retaining the desired target mode structure (bottom right).

significantly [Fig. 4.6(a)].

To effectively search this library, we set out on identifying large groups of compatible modes—such groups correspond to highly pluripotent designs. First, we structure our library of (mostly) bulk modes[1] as a graph, where nodes represent individual modes, and edges indicate pairs of modes that are "compatible": the modes have no D blocks with different constituent building block orientations at the same site [Fig. 4.5(b)]. By this definition, all modes in the library originating from the same design are compatible. Surprisingly, there is a significant number of pairs of modes from different designs that are also compatible. For such compatible modes, we can rationally combine both parent designs to create a new design that supports both zero modes (see App. A4.4.1) [Fig. 4.5(c)]. We find the maximal cliques using the Bron-Kerbosch algorithm [160].

Searching over maximal cliques instead of the initial designs improves the chances of finding a design that features a desired target mode in two ways. First, searching over maximal cliques significantly expands the search space compared to the initial designs: the number of maximal cliques exceeds the number of initial designs by a factor 70. Second, many of these cliques combine a large number of bulk modes and thus correspond to designs with higher pluripotency [Fig. 4.6(b)]. To illustrate the advantage of searching over maximal cliques, we conduct a search to find designs that satisfy a single randomized target deformation (see App. A4.1.4). Starting from the largest cliques, we successfully find a set of modes—and thus a design—that satisfies a single randomized target deformation for all 100 randomized targets. Moreover, out of one hundred sets of two randomized target deformations, we find designs for 82 sets that fully satisfy the desired target deformations. Thus, our approach allows us to find designs that satisfy multiple target deformations with a high probability.

The downside of this approach is that most successfully found designs originate from large cliques which feature many modes, most of which are superfluous and should be removed. Crucially, removing undesired modes is much easier than adding desired modes: a mode requires a careful selection of building blocks to deform compatibly while changing a single building block can be sufficient to prohibit the mode. Thus, we turn to the third substep of our approach—refining the design to eliminate superfluous modes. To achieve this, we introduce an additional diagonal

[1]Our library also contains strip-modes as we do not actively exclude them. See Fig. 4.6(a) for the distribution of the number of extensive modes a in the library prior to discarding the edge modes.

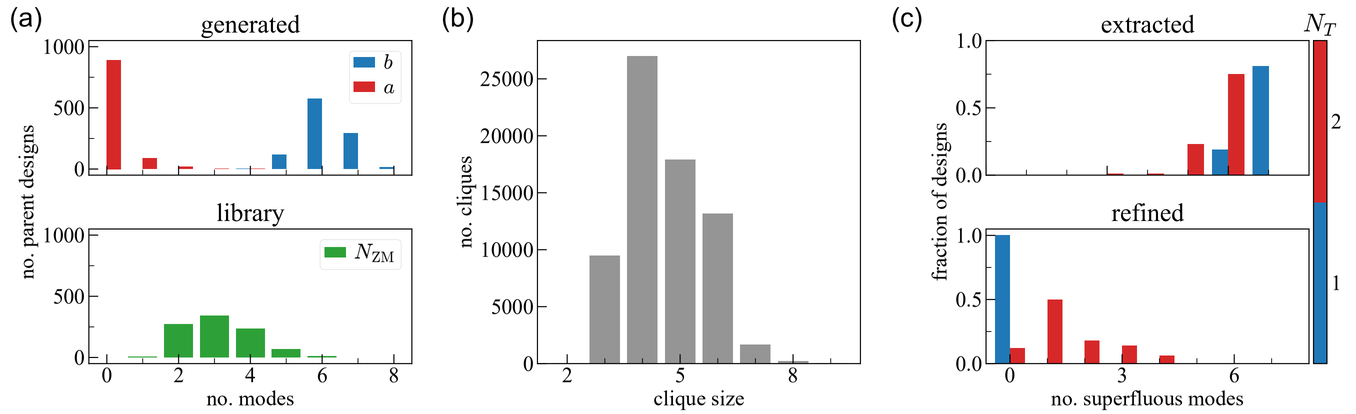


Figure 4.6: **Distribution of extracted and refined designs.** (a) Top: histogram of the true number of intensive modes b (blue left bars) and extensive modes a (red right bars) for 1000 GA-generated parent designs with a target of $b_T = 7$. Bottom: histogram of the number of modes N_{ZM} (green bars) after discarding the trivial CRS mode and edge modes (see App. A4.4.1) per parent design in the library. The total number of modes is significantly reduced after discarding. (b) Histogram of maximal clique sizes (number of nodes) in the library. (c) Histograms of the fraction of designs that have a number of superfluous modes out of 100 designs found in our library with the highest cumulative capacity [Eq. (4.3)] for N_T (colorbar) randomized target deformations before (top) and after (bottom) adding defects[2]. Most designs found by searching maximal cliques for target modes feature a large number of superfluous modes. Strategically introducing defects to these designs significantly lowers the number of modes in most cases—we refer to such designs as refined.

rigid bar to strategically selected building blocks. Recall that CRS blocks are independent of the orientation c of the building block, while D blocks are dependent on the orientation. Thus, by adding an extra diagonal bar, the building block is restricted from deforming in the D mode m_D and can only deform as a CRS block [Fig. 4.5(d)].

To prohibit superfluous modes, we use these extra rigid bars by placing them in building blocks where undesired modes contain a D block, while desired modes feature a CRS block [Fig. 4.5(e)] (see App. A4.4.2). We iterate this procedure until we can no longer add any CRS blocks or only the desired modes are left. This approach drastically reduces the number of superfluous modes [Fig. 4.6(c)]. In fact, for a single target deformation mode, we are able to completely eliminate any superfluous modes[2]. For two target deformations, most refined designs feature a single superfluous mode, and the number of designs with a higher number of intensive modes rapidly tapers off. Thus, we are able to find extremely rare designs that exhibit target deformations while minimizing the number of superfluous deformations.

4.6. Combining unit cells

To illustrate the effectiveness of our sequential method of prospecting, extracting, and refining, we apply our method to find multiple 5×5 tilings that can be combined into larger metamaterials that feature complex, spatially-textured target mode structures. Specifically, we aim to create two 10×10 metamaterials that support two distinct modes with spatially complex mode structures: one that resembles a smiley and frowny face and another that resembles the letters A and U. We stress that finding designs that feature such modes, especially within the design space of 10×10 tilings, is intractable with direct optimization methods.

We illustrate our method for combining 5×5 "unit cells" by an example. First, we use symmetry to our advantage and reduce the two desired 10×10 target mode structures into two 5×10 target deformations: a half-smiley and half-frowny (Fig. 4.7(a)-i). Next, we divide the deformations into three 5×5 parts: the eye, sad mouth, and happy mouth (Fig. 4.7(a)-ii). For each of these target deformations, we search over *all* maximal cliques and compute the capacity [Eq. (4.3)] of each clique. Subsequently, we rank the best cliques—and thus unit cell designs—for the top left and bottom left

[2]Note that we neglect the trivial global CRS mode, as we are free to prohibit this mode by adding a single additional horizontal or vertical rigid bar.

unit cells by the averaged capacity over the target deformations: eye, and sad and happy mouth, respectively (Fig. 4.7(a)-iii). Finally, we combine the unit cell designs to create a 5×10 tiling and calculate the tiling's capacity with respect to the half-smiley and half-frowny modes. In general, combining two unit cells may prohibit certain modes due to kinematic constraints at the interface. Thus, combining two unit cells with high individual capacity is not guaranteed to result in a high averaged capacity. Therefore, we iterate over the highest ranked combinations if the result is not satisfactory (Fig. 4.7(a)-iv). Using this approach, we find a 5×10 design which we mirror to create the 10×10 design [Fig. 4.7(b)] that features a mode structure which resembles a smiley and frowny face [Fig. 4.7(c)].

In addition to the smiley and frowny modes, we find seven superfluous modes—we neglect the trivial global CRS mode that is always present, as we can remove this mode while retaining all other modes by adding a single extra horizontal or vertical rigid bar to the design. This large number of superfluous modes is because our method combines highly pluripotent designs. Hence, we turn to the final step of our method: refining. Again, we strategically add extra rigid bars to prohibit undesired modes [Fig. 4.5(e)]. By placing 5 extra rigid bars, we reduce the number of superfluous modes in our design to two. Thus, we have found, in the intractable design space of 10×10 designs, a design that features two desired, spatially textured deformation modes that represent a smiley and frowny face.

To demonstrate the generality of this approach, we follow the same procedure to find a design that features modes whose structures resemble the letters A and U, spelling Au for gold. Once again, we leverage symmetry to our advantage and find a 5×10 design which we then mirror to form our 10×10 design [Fig. 4.7(d)]. This design features 11 superfluous modes in addition to the two desired modes [Fig. 4.7(e)]. By strategically placing 5 rigid bars, we manage to reduce this to 6 superfluous modes. Thus, our sequential method allows us to find extremely rare designs with multiple desired mode structures within an otherwise intractable design space.

4.7. Discussion

Combining building blocks to create structures with desired properties is notoriously difficult—the problem is often ill-posed, and the design space is too vast to fully explore. Without access to underlying design rules, directly navigating such spaces using a strongly discontinuous objective function to find designs with desired properties is hopeless. In this chap-

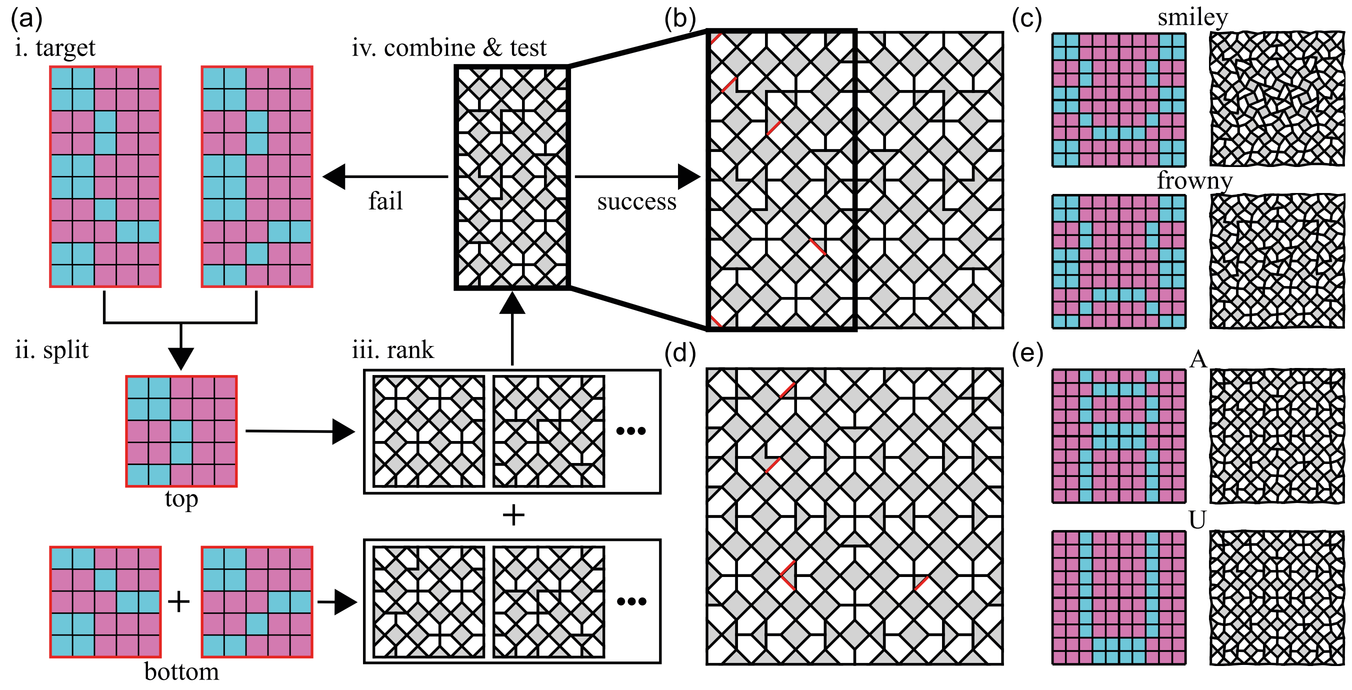


Figure 4.7: **Combining designs for target deformations.** (a) Strategy for combining designs in four substeps. (b) The found and refined 10×10 design for the target smiley and frowny deformation modes. The defects added during the refining are highlighted in red. (c) The design of (b) features two deformation modes whose structure, which we visualize by CRS blocks (cyan) and D blocks (pink), resembles a smiley and frowny face. (d) Found and refined 10×10 design for the target A and U deformation modes. (e) The design of (d) features two deformation modes whose structure resembles the letters A and U.

ter, we introduced a general strategy to find such ultra-rare designs using pluripotency: a statistical measure that quantifies performance over a class of problems. In short, our approach exploits the hierarchical structure of pluripotency in design space to generate a library of many highly pluripotent designs. Subsequently, we select and refine from this library to reach the final, ultra-rare design that satisfies the desired properties.

Our approach opens up new exciting avenues for combinatorial design. For example, our approach could be readily applied to metamaterial designs for a plethora of different building block designs, allowing for much faster exploration of the vast design space of metamaterial geometries that remain largely unexplored. Moreover, we foresee applications beyond the field of metamaterials. For example, self-assembling systems require the right set of building blocks to achieve the desired end geometry—this is hard without access to assembly rules [21, 23, 161]. Additionally, information processing in designer matter can be described by a set of hysteretic elements—how to order and tune interactions between these elements to achieve complex memory properties is an open challenge [7, 28, 29].

Data availability statement.—The codes to calculate zero modes and their structures [162], and to design [163] are freely available. Additionally, the data used to train our neural networks [91] is also freely available.

Acknowledgments.—This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. C.C. acknowledges financial support from the European Research Council under Grant Agreement 852587. M.D. acknowledges financial support from the European Research Council under Grant Agreement 884902.

Appendix

In this appendix we provide an extended description of our metamaterial, convolutional neural networks (CNNs), genetic algorithm (GA), and design approach. Additionally, we provide details on obtaining and preprocessing the training data for our CNNs. Moreover, we show that the combination of local and nonlocal exploration allows our GA to efficiently explore the design space.

A4.1. The metamaterial

In this chapter, we focus on a combinatorial metamaterial built by tiling building blocks to form $k \times k$ unit cells which are periodically repeated to tile a larger $n \times n$ metamaterial [Fig. A4.1(a)]. This metamaterial is composed of a collection of rigid bars and hinges. The building block features two zero modes—infinitesimal deformations that do not stretch any of the bonds up to first order—that we label m_{CRS} and $m_D(c)$, where c is the orientation of the building block (see Ch. 2) [154].

In chapter 2, we have showed that there are limitations on the structure of zero modes [154]. Most importantly, we have showed that areas of adjacent CRS blocks must be rectangular of shape, that is their boundaries feature only convex corners. This constraint strongly limits the possible mode structures in our metamaterial. In particular, we distinguish between three types of zero modes: (i) strip-modes, (ii) edge-modes, and (iii) global modes [Fig. A4.1(b)]. Each of these mode-types are defined by the spatial ordering of CRS and D blocks. Specifically, strip-modes feature a horizontal or vertical strip of D blocks that spans the entire material sandwiched between two areas of CRS. Edge-modes feature a strip of D blocks that borders the edge(s) of the material, the bulk are CRS blocks. Global modes feature D blocks throughout the entire material.

Additionally, these types of zero modes correspond to a change in the scaling of the number of non-trivial zero modes $N_{\text{ZM}}(n) = an+b$. Note that we exclude the three trivial zero modes corresponding to translation and rotation of rigid bodies in two dimensions. Strip-modes are translationally invariant in one direction, resulting in a linear increase of N_{ZM} and a contribution to the slope a. In contrast, edge-modes and global-modes are not translationally invariant: such modes correspond to an offset of N_{ZM} and thus contribute to the number of intesive modes b.

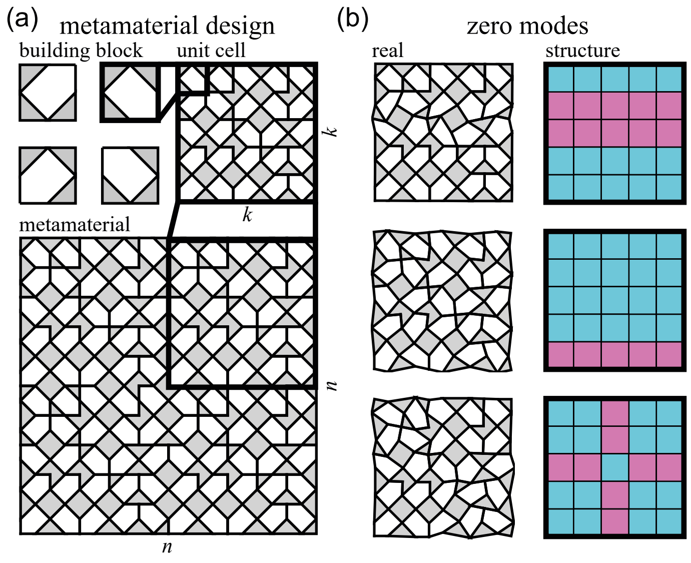


FIGURE A4.1: **Metamaterial design and zero modes.** (a) Building blocks (top left) combine into a $k = 5$ unit cell (top right) to form a $n = 2$ metamaterial (bottom). (b) Examples of the three types of zero mode structures: (i) strip-modes (top); (ii) edge-modes (middle); (iii) global modes (bottom).

A4.1.1. Calculating the number of zero modes

To calculate the coefficients a and b from our mode-scaling relation [Eq. (4.1)], we use rrQR to calculate the number of modes $N_{\mathrm{ZM}}(n)$ for $n \times n$ tilings of unit cells with open boundary conditions in the range $n \in \{1, 2, 3, 4\}$. We use $N_{\mathrm{ZM}}(3)$ and $N_{\mathrm{ZM}}(4)$ to determine the slope a and offset b. We use 1M 5×5 unit cells drawn from an uniform discrete distribution to obtain the distribution of b shown in Fig. 4.3(a).

To obtain Fig. 4.1(b), we select 100 designs per number of intensive modes b from the set of Monte Carlo sampled designs and generate 100 randomized target deformations. For each design, we calculate the capacity [(4.3)] with respect to the target deformations. The average of these capacities gives the pluripotency P per design. The average pluripotency $\langle P \rangle(b)$ is simply the pluripotency P averaged over all the designs with b intensive modes.

To obtain Fig. 4.3(b)-(c), we selected at most 1000 designs per initial number of intensive modes b_{I} from the set of Monte Carlo sampled designs. For $b_{\mathrm{I}} = 5$ and $b_{\mathrm{I}} = 6$, there are fewer than 1000 designs available, instead we use only 311 and 8 designs, respectively. For each of the selected designs, we infer the number of strip modes a and intensive modes b for the 75 designs that differ from the selected design by a single building block mutation. We determine the ratio of neighboring designs with b_{NN} intensive modes to the total number of neighboring designs for each design and average over all designs for a given initial number of intensive modes b_{I} to obtain the probability density function $p(b_{\mathrm{NN}}|b_{\mathrm{I}})$. Similarly, we define the normalized codimension for a design with b_{I} modes as the fraction of all neighboring designs with $b_{\mathrm{NN}} \leq b_{\mathrm{I}}$.

A4.1.2. Determining the structure of a zero mode

The structure of zero modes can be defined in terms of individual building block deformations. As described in Ch. 2, our building blocks feature two zero modes: the CRS mode m_{CRS} and D mode m_D. A building block can deform with any linear combination of these two independent modes: $m = \alpha m_{CRS} + \beta m_D$. We classify building blocks by these zero modes: if a block deforms with $\beta = 0$, it is a CRS block, and if a block deforms with $\beta \neq 0$, it is a D block. In chapter 2, we showed that there are limitations on the spatial structure of zero modes in tilings of these building blocks: regions of adjacent CRS blocks must always be rectangular of shape [154].

To determine the structure of zero modes supported by a given design in terms of CRS and D blocks, a simple calculation of the null space of the compatibility matrix $\mathcal{C}$ is no longer sufficient. Instead, we directly solve for the kinematic degrees of freedom of the building block: α and β. These kinematic degrees of freedom effectively describe the infinitesimal change in each of the five free interior angles of the building block to first order. The kinematic constraints between neighboring building blocks can be described as constraints of adjacent angles of building blocks. For more technical details on this representation of modes, we refer to Ch. 2.

In short, we solve for each building block's kinematic constraints by composing a large integer matrix of all kinematic constraints between building blocks and use the Python package sympy [164] to find the null space of this matrix. This yields a set of rational vectors that form a basis for all valid zero modes in the design and translate directly to the kinematic degrees of freedom α and β. Thus, this method allows us to determine the structure of zero modes for a given metamaterial design in terms of CRS and D blocks. The code for this is freely available on our public GitLab [162].

A4.1.3. Capacity of a set of modes

The capacity $C(\{M^B\}, \hat{M})$ [Eq. (4.3)] of a set of basis zero modes $\{M^B\}$ with respect to a target mode $\hat{M}$ is $S(M^*, \hat{M})$ [Eq. (4.2)], where $M^* = \sum w_j^* M_j^B$ is a linear combination of the basis modes, and the set of weights $\mathbf{w}^*$ maximizes S. To find this set of weights, we use constraint programming [165].

Specifically, we consider the D mode coefficients β_i^j at site i corresponding to the basis mode labeled by j. We define a set of integer variables $\mathbf{w}$ such that $M = \sum_j w_j M_j^B$. The goal is to find an M that is close to the target mode $\hat{M}$. For every D site i in the target mode, if any of the basis modes $\{M^B\}$ also has a D block at that site, we add a constraint $\sum_j w_j \beta_i^j \neq \delta_i$, where δ_i is a non-negative integer variable. Similarly, at every CRS site i in the target mode, if any of the basis modes $\{M^B\}$ also has a D block at that site, we add the constraints $\sum_j w_j \beta_i^j \geq -\delta_i$ and $\sum_j w_j \beta_i^j \leq \delta_i$. The constraint programming solver then attempts to find a solution to all the variables that minimizes $\sum \delta_i$. The underlying idea is that by minimizing this sum, the program tries to find a set $\mathbf{w}^*$ for which most variables satisfy $\delta_i = 0$. When $\delta_i = 0$ for all i, the mode M has the same kind of block, CRS or D, at site i as the target mode $\hat{M}$. The final similarity is then $S(M^*, \hat{M})$ [Eq. (4.2)] with $M^* = \sum_j w_j^* M_j^B$. The code we used to calculate the capacity is publicly available [163].

A4.1.4. Randomized target deformations

To determine the pluripotency P of a design or a set of zero modes, we calculate the average capacity C [Eq. (4.3)] for a set of randomized target modes $\hat{M}$. We define our target modes $\hat{M}$ in terms of their structure: the spatial distribution of CRS and D blocks. However, our metamaterial has limitations regarding the mode structures it can achieve. In chapter 2, we demonstrated that any valid mode must comprise rectangular patches of adjacent CRS blocks [154]. Thus, to assess our designs fairly, we cannot simply generate any random distribution of CRS and D blocks.

Instead, we generate viable target modes by generating a horizontal and vertical strip of D blocks within a background of CRS blocks. The position and width of these strips are randomly drawn from a uniform distribution. Where the two strips overlap, we replace the D blocks with CRS blocks. This approach ensures that there are only rectangular patches of adjacent CRS blocks present. Using this approach, we generate 200 unique target modes (the 200 possible target deformations are shown in Fig. A4.2).

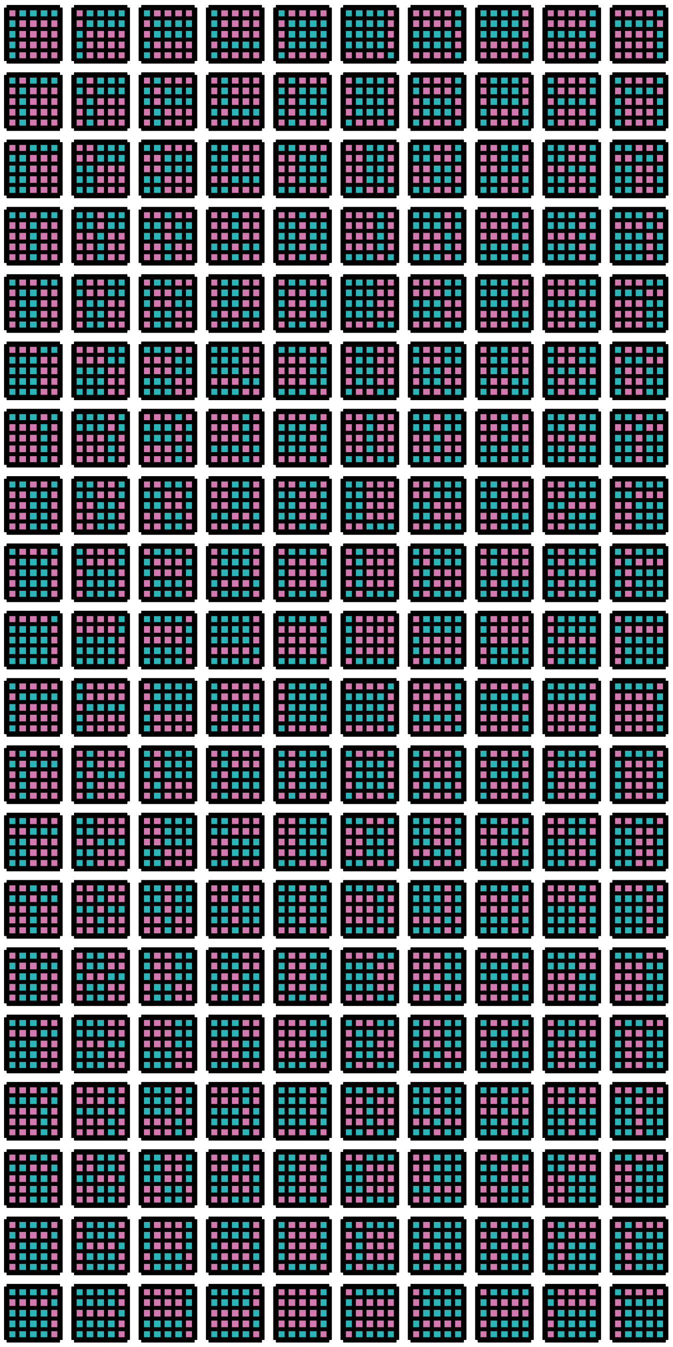


Figure A4.2: All 200 possible global target deformations generated using the method as described in the Materials and Methods.

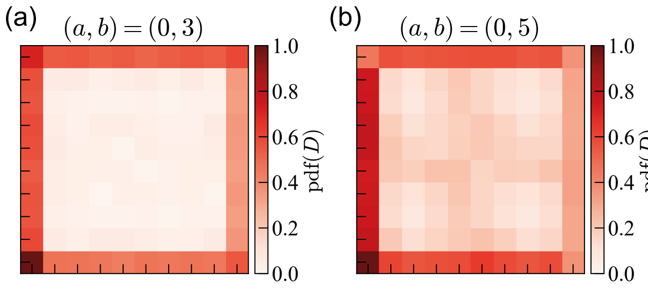


FIGURE A4.3: Spatial distribution of the probility density function (pdf) of D blocks for 2×2 tilings of random 5×5 unit cells with $a = 0$ extensive modes and $b = 3$ (a) or $b = 5$ (b) intensive modes.

A4.1.5. Distribution of edge-modes and global modes.

The codimension of the subspace of designs with b intensive modes appears multimodal in distribution, especially for larger b. This is most likely due to the different types of modes that contribute to b: edge-modes and global modes. For low b, most intensive modes are edge-modes [Fig. A4.3(a)]. As b increases, global modes become more prevalent [Fig. A4.3(b)]. The reason for this is twofold. First, edge-modes consisting of a single line of D blocks require less building blocks with specific orientations than global modes that contain more D blocks (recall that the CRS mode is independent of the building block orientation). Thus, random unit cells are more likely to contain edge-modes than global modes. Second, the number of edge-modes is limited by the number of edges of the metamaterial. Thus, for sufficiently large number of intensive modes b there is a higher probability of global modes. The codimension is related to the probability to prohibit an intensive mode by changing the orientation of a randomly selected building block. The type of mode influences this probability. In general, modes with more D blocks are easier to prohibit as the D mode is sensitive to the orientation of the building block. Additionally, building blocks at the edge are less kinematically restricted than building blocks in the bulk (see Ch. 2) [154], making them more robust to changes of orientation.

A4.2. Convolutional neural networks

A4.2.1. Training the CNNs

We train our convolutional neural network (CNN) on 5×5 designs obtained from Monte Carlo sampling of the space. Before training, we preprocess the data.

The configurational data of the unit cell designs is preprocessed to a pixelated representation (see Supplemental Information) for input to the neural network. Additionally, we use periodic padding to add an extra one-pixel-wide layer to the designs. This allows the network to capture the interactions between building blocks with periodic boundary conditions.

To train the CNNs to predict the number of intensive modes b_{CNN}, we require a set of unit cell designs $\mathbf{X} = \{X_i\}$ and their respective number of intensive modes $\mathbf{b} = \{b_i\}$. The combination of these designs and their corresponding numbers of intensive modes is referred to as the training set $\mathbf{D}_t = (\mathbf{X}, \mathbf{b})$. We split our original data into a training and test set; the training set contains 848898 samples, and the set set contains 149982 samples. Both the training and test set follow the same distribution of the number of intensive modes b; the majority of samples have $b = 1$, while there are only 4 samples with $b = 6$ in the training set.

We train the CNN to minimize the mean squared error (MSE) between the CNN's prediction b_{CNN} and the true b using the Adam optimization algorithm [143]. We use 10-fold cross-validation to validate the robustness of our network architecture and training procedure. The network with the lowest MSE over the validation set is selected to be the primary network to use. Specifically, we use a learning rate of 0.0005, train the network for 100 epochs, and use a batch size of 256. Additionally we use L2-regularization on the weights and biases to reduce overfitting.

A4.2.2. Neural network architecture

The CNNs used in this chapter are composed of three convolution layers for feature extraction, which are connected to a fully-connected hidden layer, which in turn is connected to a single-neuron output layer. Specifically, the first convolution layer consists of twenty 2×2 filters with a stride of $(2, 2)$. As the convolution starts in the upper left corner of the input image, the network convolves only 2×2 *plaquettes* between four building blocks. Each plaquette contains a black pixel if one of the four constituent building blocks is oriented such that it has its diagonal interior angle within that plaquette (see Supplemental Information). As such, each plaquette contains information on which building blocks share adjacent diagonal corners and allow for possible compatible deformations of those corners. We conjecture that restricting the network to see only these plaquettes helps achieve a better and more robust performance.

The second and third convolution layers have 80 and 160 2×2 filters, respectively, with a stride of $(1, 1)$. After each convolution operation, we add a trainable bias vector and apply a ReLu activation function on each element of the convolved images. Note that we do not use pooling operations in-between convolution layers. After the third convolution layer, we flatten the convolved images and fully-connect this vector to a hidden layer of 1000 neurons. Again we add a bias vector and apply the ReLu activation function. The final layer consists of only a single neuron, and we do not apply an activation function. We take the single output neuron to be b_{CNN}, which we aim to be as close to the true b of any input design as possible. We use Jax [166] to code our networks.

We use 10-fold cross-validation to validate the robustness of our network architecture and training procedure. The network with the lowest mean squared error (MSE) over the validation set is selected to be the primary network to use. Specifically, we use a learning rate of 0.0005, train the network for 100 epochs and use a batch size of 256. Additionally we use L2-regularization on the weights and biases. The architecture of our CNN is shown schematically in Fig. A4.4(b). The training process for each fold is shown in Fig. A4.4(c).

A4.2.3. Preprocessing

The configurational data of the unit cell designs is preprocessed to a pixelated representation (see Fig. A4.4(a)) for input to the neural network. We found that representing the unit cell designs in this pixelated representation improved convergence during training and better performance of the trained networks over the validation sets. We believe that this is due to the orientations of the building blocks being clearly represented visually as opposed to simply representing the orientations as integers in a matrix. Because each building block is represented as a 2×2 square, we can choose where the convolutional layer applies its filters more finely. As discussed in the Methods and Materials section, this allows us to let the networks 'see' only the interactions between building blocks. Additionally, we use periodic padding to pad the designs with an extra one pixel wide layer to allow the network to see interactions between building blocks with

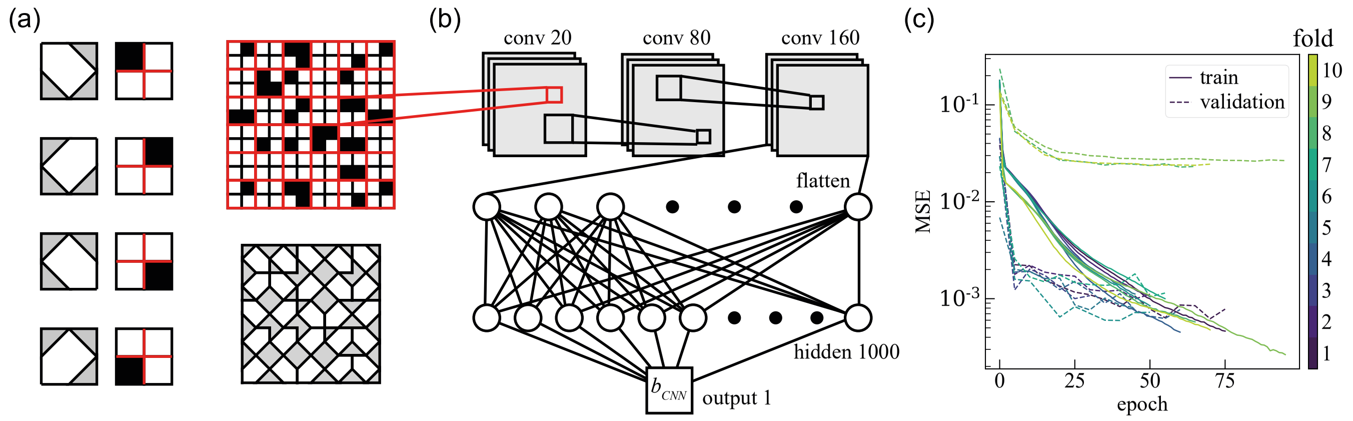


Figure A4.4: **Convolutional neural network (CNN) for metamaterial prediction.** (a) To feed our metamaterial designs into a neural network, we represent our designs as a black-and-white image. Building blocks are represented as 2×2 plaquettes of pixels, one black and three white (left). 5×5 designs (bottom right) translate to 10×10 pixel images. Additionally, we pad these images using periodic boundary conditions with one additional layer of pixels, so that we end up with a 12×12 pixel image (top right). (b) The pixel image of (a) forms the input for our CNN, which consists of three convolutional layers, a single hidden layer and a single output node. (c) The training (solid line) and validation (dashed line) mean squared error (MSE) over the training epochs for each fold (colorbar) in our 10-fold cross-validation. Note that we used an early stopping condition to prevent overfitting, resulting in different number of training epochs for the folds.

periodic boundary conditions.

A4.3. Genetic algorithm

We employ a genetic algorithm (GA) to explore our CNN beyond its training scope and to prospect highly pluripotent designs (see Fig. A4.5 for a schematic overview). Specifically, we utilize a GA where the fitness function [Eq. (4.4)] is estimated by a trained CNN. Without such an approximate fitness function, the computational time increases by a factor 390 and exploration of the design space is unfeasible. The goal of the GA is to achieve a target number of intensive zero modes b_T. To achieve this goal, the GA iteratively generates a population of designs (the generation) in three steps: (i) sampling, (ii) fitness evaluation, and (iii) update. Below, we give an overview of these three steps.

In the first step, a fixed number of candidate designs is randomly drawn from the discrete design space. In GA terminology, this set of designs (population) is referred to as generation 0. Only the very first generation is generated in this manner. Second, we score and rank the designs based on their fitness f [Eq. (4.4)]. This fitness is maximal when the CNN's prediction is equivalent to the target number of intensive modes b_T. Finally, we use the ranking of designs based on the fitness to generate a new population of designs—the next generation. To efficiently explore the design space, the GA combines and mutates designs. To this end, we employ several standard GA techniques. We select a group of designs from the initial population based on a three-way random tournament selection, and we always include the design with the highest fitness (elitism). This group of designs forms the "parents". From this set of parents, we combine designs using a custom crossbreeding scheme (see below), and we clone to create an additional set of designs. This set of designs also undergoes random mutations of building blocks and produces the "children". The combination of parents and children then forms the next generation, maintaining the same size as the previous generation. This procedure of fitness evaluation and generation of a new population of designs is repeated until the fitness is maximized or a predefined criterion is met.

Specifically, our genetic algorithm has a population size of 100, of which 29 are selected to be parents using a three-way tournament selection. Additionally, the fittest candidate in the population is automatically selected to be one of the parents, so that we have a total of 30 parents. To create a new generation of 100 designs, we require 70 children. Of those 70

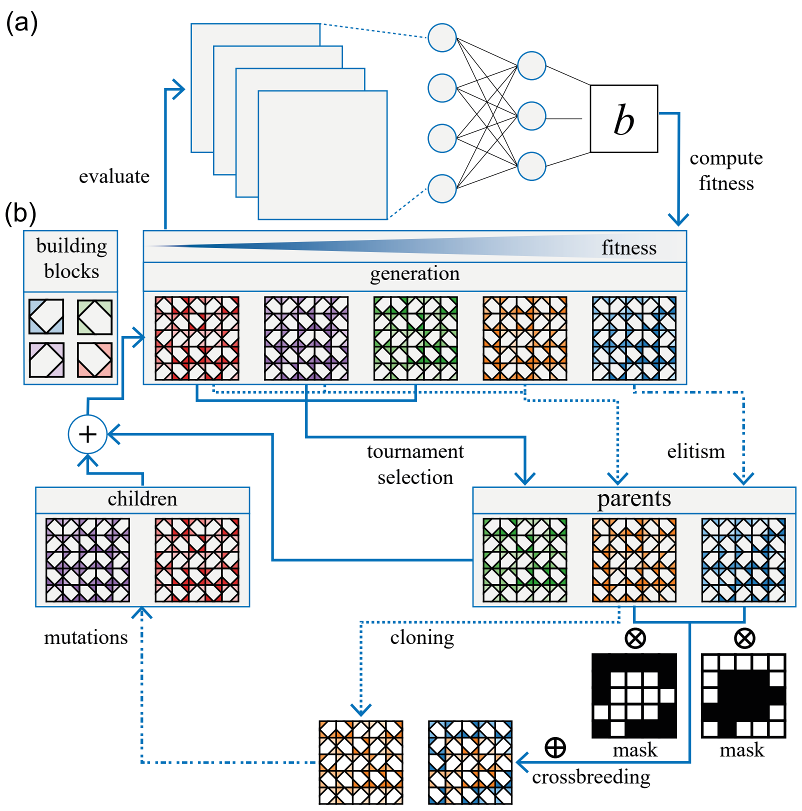


FIGURE A4.5: **Schematic representation of our genetic algorithm.** (a) A CNN calculates the number of intensive modes b_{CNN} for each design in the generation. (b) A generation of designs (top) is ranked based on the fitness f. Using a three-way tournament selection the algorithm selects designs from the generation that, together with the fittest design in the generation, form the parents (right). From the parents, we generate new designs in two ways: cloning and crossbreeding (bottom). In cloning, we simply make a copy of the parent design chosen at random. In crossbreeding, we randomly select two parents that we combine using two-dimensional masks (black-and-white matrices). Additionally, building blocks in these newly generated designs have the chance to mutate into new orientations. The mutated designs form the children (left) who, together with the parents, form the next generation. This iterative procedure continues until a sufficiently high fitness or stopping condition is reached.

children, half are created using cross-breeding of randomly selected pairs of parents. To combine the design of two parents to create a new child, we use a Gaussian filter to filter a random binary 5×5 mask, which we then again binarize to create a mask where half of the elements are 0 and half are 1. This mask is then used to take part of the two parents and combine them to create a new child [Fig. A4.5(a)]. The reason we use this method over more standard methods, such as k-point crossover or uniform crossover, is that we believe local clusters of building blocks are important for intensive zero modes. The other 35 children are taken by cloning randomly selected parents. All building blocks in the children designs have a chance of 10% to mutate to a different orientation. Each of these different orientations are equally likely to be selected. The combination of the parents and children after mutation form the new generation of designs.

A4.3.1. GA exploits nonlocal structure

To understand how the GA explores design space, we investigate the two key exploration techniques of our GA. Underlying the evolution of generations of designs are two main processes: (i) cloning and (ii) crossbreeding [Fig. A4.5(b)]. Our GA thus explores the design space in two ways: (i) the cloned designs undergo local mutations. This is local exploration of the design space surrounding the cloned parent design. (ii) The crossbred designs are combinations of two parent designs on top of local mutations. This is non-local exploration of the design space.

The combination of both local and non-local exploration are crucial for the success of the GA. To illustrate this, we determine the minimal distance between between the design X with the highest fitness of generation $g+1$ and every other design Y in generation g as

$$d_{min} = \min_{\{Y\}} \left(\sum_{i,j} 1 - \delta(X_{i,j}, Y_{i,j}) \right), \tag{A4.1}$$

where $\delta(x,y)$ is the Kronecker delta function and $X_{i,j}$ is the orientation of the building block at site (i,j) in design X. (A4.1) thus captures the number of different building blocks between design X of generation $g+1$ and the design Y in generation g that is closest to X. Non-local exploration primarily plays a role for early generations when $b_{CNN}(g)$ is likely to be small, which allows the GA to find designs of a higher b_{CNN} (Fig. A4.6(c)). For larger b_{CNN}, in later generations, local exploration is key.

Intuitively, the GA is able to efficiently explore the design space by first crossbreeding designs to quickly find designs with a reasonably high number of intensive modes b_{CNN}. This is most likely a consequence of the fact that most designs with a low number of intensive modes feature

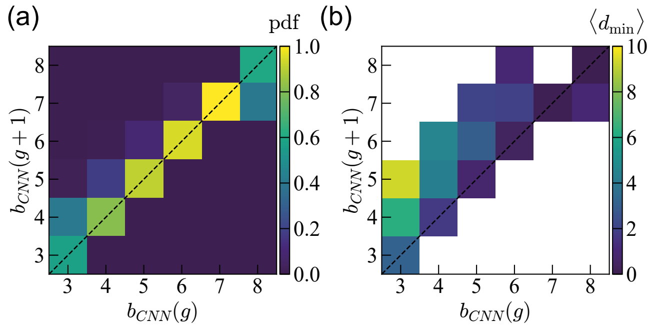


FIGURE A4.6: **GA design space exploration.** (a) Pdf for the design with the highest fitness in a GA run to transition from $b_{CNN}(g)$ to $b_{CNN}(g+1)$ for iterative generations. For low $b_{CNN}(g)$ the GA is more likely to transition to a higher b_{CNN}. For high $b_{CNN}(g)$ the GA struggles to quickly find designs with a larger $b_{CNN}(g+1)$. Note that $b_{CNN}(g)$ can never transition to a lower $b_{CNN}(g+1)$ as the GA always takes the design with the highest b_{CNN} to the next generation. (b) The average minimal distance $\langle d_{min} \rangle$ [Eq. (A4.1)], where $\langle . \rangle$ denotes the average over an ensemble of GA runs, for the fittest design in generation $g+1$ with $b_{CNN}(g+1)$ as a function of $b_{CNN}(g)$ of the fittest design in generation g. As $b_{CNN}(g)$ increases, the distance $\langle d_{min} \rangle$ between the fittest design in the next generation $g+1$ and all designs in generation g decreases.

deformations that are localized on the edge of the material. Such edge modes are less sensitive to building block mutations than global modes and thus crossbreeding is more likely to combine designs that feature edge modes. Moreover, as the number of intensive modes b increases the probability to inhibit any mode by changing orientations increases, such that the crossbreeding is more likely to result in a net conservation or decrease of b. After $b = 5$, the GA appears to rely on cloning to locally explore around an ensemble of designs with high b to find rare $b + 1$ designs. Thus, the GA is able to efficiently generate designs that feature a large number of intensive modes b thanks to both non-local and local exploration, resulting in successful runs that generally converge.

A4.3.2. Random walks

To both obtain starting designs for the random walks and compare the evolution of GA designs, we perform and keep track of a hundred GA runs with $b_T = 7$. We start a hundred random walks from the final hundred designs with the highest fitness, thus the starting $b(s = 0)$ varies from $5 \leq b \leq 7$.

A4.3.3. Comparison to other methods.

Our GA is able to find ultra-rare designs with a large number of intensive modes b, that is not possible using random search as designs with high b become exponentially more rare with b. Alternatively, one could try to exploit the hierarchical design space structure through a hill-climbing method. While this approach succeeds sometimes, it fails an exponentially larger fraction of the time for increasing target number of intensive modes b_T [Fig. A4.7(a)-(b)]. This results in a larger average number of evaluations of the fitness function f for large b_T to make a successful run [Fig. A4.7(c)-(d)]. Compared to state-of-the-art generative methods, such as variational autoencoders (VAE), generative adversarial neural networks (GANs), and normalizing flows, our method allows for extrapolation outside the scope of the training set. These generative methods all aim to approximate the (unknown) underlying probability distributions of the training set—without examples such methods are unable to generate designs with the desired property.

A4.4. Design approach

Here, we describe in more detail the second step of our design approach: extracting and refining designs.

A4.4.1. Combining and selecting designs for target deformations

The goal of extracting is to combine and select from a set of 5×5 designs to form larger larger metamaterial that deform close to desired target deformations. We start from a set of designs generated using our design method and compute the mode structures. We aim to throw away edge modes. We do this by checking the location of CRS sites; if the entire 4×4 inner square of the 5×5 mode consists of CRS blocks, we assume it is an edge mode and disregard it.

Next, we determine which modes are *compatible* with each other. Here, compatible means that there exists a design that can feature both modes. Modes are compatible if (1) there are no overlapping D sites or (2) if for all overlapping D sites, the building block orientations from the original designs are the same. This means that all modes that originate from the same design are compatible, as they should be. Surprisingly, modes that originate from different designs can also be compatible. By taking the building block orientations of the D sites for both modes, we can create

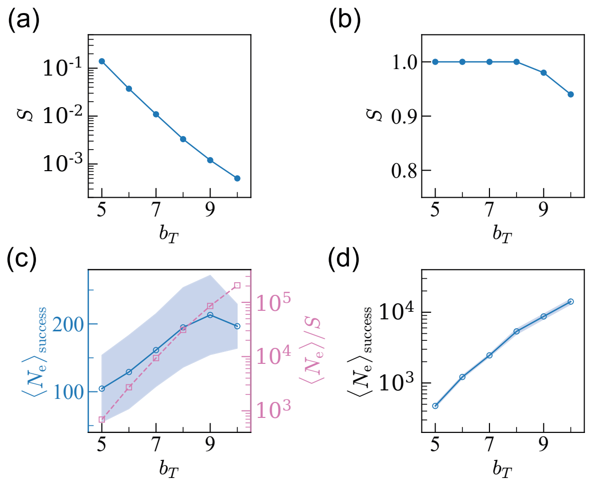


Figure A4.7: **Comparison of hill-climbing and genetic algorithm.** (a) Success rate S of 10000 hill-climbing runs to reach target number of intensive modes b_T decreases exponentially. (b) The success rate S of 10000 GA-runs to reach b_T. (c) The average (blue circles) and standard deviation (blue shaded area) of the number of evaluations of the fitness function f for successful runs $\langle N_e \rangle_{\text{success}}$ to reach b_T using hill-climbing. The average number of evaluations per successive run $\langle N_e \rangle$ S (pink squares) increases exponentially with b_T. (d) The average (blue circles) and standard deviation (shaded area) of the number of evaluations of the fitness function f for successful runs $\langle N_e \rangle_{\text{success}}$ increases exponentially with b_T.

a design that features both modes, as the orientation of CRS blocks is irrelevant. This procedure yields a set of modes and pairs of modes that we label compatible. We represent this as a graph with nodes (modes) and undirected edges (compatible).

To explore our set of modes for combinations that deform close to a target deformation, we search the space of maximal cliques. Such cliques hold the largest set of compatible modes, thus providing the design with the most deformational freedom. We calculate the set of maximal cliques using the Bron-Kerbosch algorithm with vertex ordering by first calculating the degeneracy ordering of the graph [160, 167] . For any given set of designs, we need to perform this calculation only once and save the graph and maximal clique for subsequent searches for target deformations.

To determine if a maximal clique of modes deforms closely to target deformations, we calculate the capacity C [Eq. (4.3)]. This score is calculated using a constraint programming solver that uses satisfiability methods [165] as described in Materials and Methods C.

To design larger metamaterials, such as the 10×10 designs of Fig. 4.7(b) and 4.7(d), we split the larger target deformation into local 5×5 deformations. We rank all maximal cliques based on their cumulative capacity C for each 5×5 subset. Then, we take a greedy search approach and combine designs with the highest capacity for each 5×5 subset and calculate the total capacity for the combined design. If the total score in unsatisfactory (below a predefined threshold), we iteratively try the next best designs for each 5×5 subset until we find a satisfactory total score.

A4.4.2. Prohibiting undesired zero modes

To prohibit undesired zero modes, we strategically introduce additional rigid bars to the metamaterial structure. To determine the placement of these bars, we analyze the modal structure of the desired and undesired zero modes. In particular, if the undesired mode contains a D block where the desired modes feature CRS blocks, we add a rigid bar across the diagonal to transform the pentodal building block shape to a square shape. This prohibits the building block from deforming with the D mode, effectively prohibiting the undesired mode without introducing any new (undesired) zero modes.

5 Discussion

In this thesis, we set out to find systematic design strategies for multimodal metamaterials. To this end, we have focused on one family of combinatorial metamaterials that are emblematic of multimodal metamaterial design—designs with desired properties are extremely rare, mechanical properties are sensitive to minute changes, and the design space is vast. Without design rules, finding designs with desired mechanical properties is intractable.

Central to combinatorial design problems is the notion of an underlying structure in design space. At first glance, the design problem seems intractable: the number of possible building block combinations is too large to fully enumerate and tilings with desired properties are too rare to find through Monte Carlo sampling of the design space. However, there is an implicit, lower dimensional order to the design space. In other words, properties are not randomly dispersed throughout design space but form structured sets. Precisely because such an order exists, we can use techniques, rational or computational, to uncover a set of simpler descriptors that delineate these structured sets—the design rules. Below, we summarize how we uncovered this order in each chapter.

In chapter 2, this structure emerges through kinematic constraints linked to pairs of neighboring building blocks. Both the pairs and constraints can be grouped into distinct types—revealing a first lower order to the structure-property map of extensive modes. Remarkably, these constraints, by large, map within the set of constraints induced by other types of pairs. Thus, kinematic constraints can be captured in a directed graph dictating how constraints map to new constraints under a given configuration of building blocks. Configurations resulting in a mapping of kinematic constraints across the entire configuration that do not exhaust the deformational degrees of freedom support an extensive mode. Thus, we can derive an explicit set of (nonlocal) conditions that configurations need to satisfy to support an extensive mode, allowing for rational design of such configurations.

In chapter 3, the underlying structure to the design space becomes apparent in the success of the neural network. By virtue of the accuracy of the trained network to new, previously unseen designs, the design space must have some lower dimensional structure that is captured by the trained network. Intuitively, we describe this structure as needles-in-hay—rare designs form a sparse filamentous network of hyperplanes (needles) of lower dimensionality than the encompassing design space (hay). This intuitive picture helps interpret the characteristic response of statistical probes of the boundary between classes in the average dimensionality of these needles. In comparing this average dimensionality to that of the internal representation of the trained networks, we found a remarkable agreement. We conclude that the trained neural networks infer (approximate) design rules that delineate the design space to a higher accuracy than expected through simple interpolation of the space.

In chapter 4, we train a neural network for regression of the number of intensive modes instead of classification. The structure of this number in design space is similar to the needles-in-hay structure of chapter 3. However, the number's structure is hierarchical—needles contain more needles themselves, and an intuitive picture of needles-within-needles-within-needles emerges. Needles higher in the hierarchy become exponentially more rare and are sparsely found through Monte Carlo sampling of the design space. As such, the network is not trained on designs in ultra-rare needles high in the hierarchy. Surprisingly, a trained neural network is able to accurately predict, for some designs, a number of intensive modes beyond the range in the training set. Thus, we conclude that the network has learned the implicit hierarchical structure to the design space. This allows us to efficiently generate, using a genetic algorithm, ultra-rare design with a high number of intensive modes. Due to the nature of these modes, such designs are likely to feature any desired deformation. Using a library of such designs, we find designs that deform close to a set of targeted deformations. Finally, we use directed defects to prohibit undesired modes and create designs with desired modes and few superfluous modes.

Recognizing that design rules can be systematically derived or learned using neural networks opens new strategies for metamaterial design. For instance, our rational approach in chapter 2 can be readily applied to combinatorial metamaterials with different building block designs. Such designs will alter the vertex model, yet the kinematic constraints can be derived in the same way. Using these constraints, another transfer mapping can be made, and constraints can be mapped and tracked across configurations. Alternatively, neural networks can be trained to classify designs and predict their number of deformation modes. For these networks to be successful, there should be order to the structure-property map present in the data set. Moreover, our strategy to design metamaterials by first increasing their deformational freedom before prohibiting undesired modes is generally applicable to multimodal metamaterials with spatially extended modes. We hope our work will push the interest in such multimodal metamaterials which have potential applications in programmable materials, soft robotics, and computing *in materia*.

More broadly, our work is relevant to combinatorial design problems that are ubiquitous in science. In chapter 2, we present a systematic strategy to solve tiling problems with nonlocal constraints, beyond Wang tiles. In chapter 3, we show how well neural networks can capture combinatorial rules in a strongly class-imbalanced design space. In chapter 4, we show that a two-step strategy of first finding designs that satisfy many independent sets of conditions before changing the design to violate selected undesired conditions is preferable to direct optimization for the desired sets of conditions. We believe that combinatorial metamaterials are well suited to probe open questions in combinatorial design, as they are straightforward to simulate yet exhibit complex combinatorial structure. Some open questions include, for instance, how can we learn size-independent combinatorial rules from data? What is the relation between the complexity of the combinatorial rules and that of neural networks? How do we best overcome a strong data-imbalance? How do we infer interpretable combinatorial rules using machine learning? What are good strategies for exploring a vast combinatorial space when simulations are computationally expensive?

Bibliography

1. Reis, P. M., Jaeger, H. M. & Van Hecke, M. Designer matter: A perspective. *Extreme Mechanics Letters* **5,** 25–29 (2015).
2. Bertoldi, K., Vitelli, V., Christensen, J. & Van Hecke, M. Flexible mechanical metamaterials. *Nature Reviews Materials* **2,** 1–11 (2017).
3. Florijn, B., Coulais, C. & van Hecke, M. Programmable mechanical metamaterials. *Physical review letters* **113,** 175503 (2014).
4. Silverberg, J. L. *et al.* Using origami design principles to fold reprogrammable mechanical metamaterials. *Science* **345,** 647–650. eprint: https://www.science.org/doi/pdf/10.1126/science.1252876. https://www.science.org/doi/abs/10.1126/science.1252876 (2014).
5. Medina, E., Farrell, P. E., Bertoldi, K. & Rycroft, C. H. Navigating the landscape of nonlinear mechanical metamaterials for advanced programmability. *Physical Review B* **101,** 064101 (2020).
6. Mueller, J., Lewis, J. A. & Bertoldi, K. Architected multimaterial lattices with thermally programmable mechanical response. *Advanced Functional Materials* **32,** 2105128 (2022).
7. Bense, H. & van Hecke, M. Complex pathways and memory in compressed corrugated sheets. *Proceedings of the National Academy of Sciences* **118,** e2111436118 (2021).
8. Kwakernaak, L. J. & van Hecke, M. Counting and sequential information processing in mechanical metamaterials. *Physical Review Letters* **130,** 268204 (2023).
9. Guo, X., Guzmán, M., Carpentier, D., Bartolo, D. & Coulais, C. Non-orientable order and non-commutative response in frustrated metamaterials. *Nature* **618,** 506–512 (2023).
10. Meeussen, A. S., Oğuz, E. C., Shokef, Y. & Hecke, M. v. Topological defects produce exotic mechanics in complex metamaterials. *Nature Physics* **16,** 307–311 (2020).
11. Coulais, C., Teomy, E., De Reus, K., Shokef, Y. & Van Hecke, M. Combinatorial design of textured mechanical metamaterials. *Nature* **535,** 529–532 (2016).
12. Jiang, C., Rist, F., Wang, H., Wallner, J. & Pottmann, H. Shape-morphing mechanical metamaterials. *Computer-Aided Design* **143,** 103146. ISSN: 0010-4485. https://www.sciencedirect.com/science/article/pii/S0010448521001573 (2022).
13. Meeussen, A. & van Hecke, M. Multistable sheets with rewritable patterns for switchable shape-morphing. *Nature* **621,** 516–520 (2023).
14. Bossart, A. & Fleury, R. Extreme Spatial Dispersion in Nonlocally Resonant Elastic Metamaterials. *Physical Review Letters* **130,** 207201 (2023).
15. Cho, Y. *et al.* Engineering the shape and structure of materials by fractal cut. *Proceedings of the National Academy of Sciences* **111,** 17390–17395. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1417276111. https://www.pnas.org/doi/abs/10.1073/pnas.1417276111 (2014).
16. Sussman, D. M. *et al.* Algorithmic lattice kirigami: A route to pluripotent materials. *Proceedings of the National Academy of Sciences* **112.** Publisher: Proceedings of the National Academy of Sciences, 7449–7453. https://www.pnas.org/doi/10.1073/pnas.1506048112 (2024) (June 2015).
17. Dieleman, P., Vasmel, N., Waitukaitis, S. & van Hecke, M. Jigsaw puzzle design of pluripotent origami. *Nature Physics* **16,** 63–68 (2020).

18. Liu, W., Janbaz, S., Dykstra, D., Ennis, B. & Coulais, C. *Leveraging yield buckling to achieve ideal shock absorbers* 2023. arXiv: 2310.04748 [cs.CE].
19. Bossart, A., Dykstra, D. M., Van der Laan, J. & Coulais, C. Oligomodal metamaterials with multifunctional mechanics. *Proceedings of the National Academy of Sciences* **118,** e2018610118 (2021).
20. Gazit, E. Self-assembled peptide nanostructures: the design of molecular building blocks and their technological utilization. *Chemical Society Reviews* **36,** 1263–1269 (2007).
21. Zeravcic, Z., Manoharan, V. N. & Brenner, M. P. Size limits of self-assembled colloidal structures made using specific interactions. *Proceedings of the National Academy of Sciences* **111,** 15918–15923 (2014).
22. Levin, A. *et al.* Biomimetic peptide self-assembly for functional materials. *Nature Reviews Chemistry* **4,** 615–634 (2020).
23. Gartner, F. M., Graf, I. R. & Frey, E. The time complexity of self-assembly. *Proceedings of the National Academy of Sciences* **119,** e2116373119 (2022).
24. Evans, C. G., O'Brien, J., Winfree, E. & Murugan, A. Pattern recognition in the nucleation kinetics of non-equilibrium self-assembly. *Nature* **625,** 500–507 (2024).
25. Hull, T. *The combinatorics of flat folds: a survey* in *Origami3: Proceedings of the 3rd International Meeting of Origami Science, Math, and Education* (2002), 29–38.
26. Keim, N. C., Paulsen, J. D., Zeravcic, Z., Sastry, S. & Nagel, S. R. Memory formation in matter. *Reviews of Modern Physics* **91,** 035002 (2019).
27. Mungan, M., Sastry, S., Dahmen, K. & Regev, I. Networks and hierarchies: How amorphous materials learn to remember. *Physical review letters* **123,** 178002 (2019).
28. Van Hecke, M. Profusion of transition pathways for interacting hysterons. *Physical Review E* **104,** 054608 (2021).
29. Shohat, D., Hexner, D. & Lahini, Y. Memory from coupled instabilities in unfolded crumpled sheets. *Proceedings of the National Academy of Sciences* **119,** e2200028119 (2022).
30. Rouvray, D. Isomer enumeration methods. *Chemical Society Reviews* **3,** 355–372 (1974).
31. Chhabra, N., Aseri, M. L. & Padmanabhan, D. A review of drug isomerism and its significance. *International journal of applied and basic medical research* **3,** 16–18 (2013).
32. Merrell, P. *Example-based model synthesis* in *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (2007), 105–112.
33. Merrell, P. & Manocha, D. in *ACM SIGGRAPH Asia 2008 papers* 1–7 (2008).
34. Merrell, P. & Manocha, D. Model synthesis: A general procedural modeling algorithm. *IEEE transactions on visualization and computer graphics* **17,** 715–728 (2010).
35. Adya, S. N. & Markov, I. L. Combinatorial Techniques for Mixed-Size Placement. *ACM Trans. Des. Autom. Electron. Syst.* **10,** 58–90. ISSN: 1084-4309. https://doi.org/10.1145/1044111.1044116 (Jan. 2005).
36. Oh, C. *et al.* Bayesian Optimization for Macro Placement. *arXiv preprint arXiv:2207.08398* (2022).
37. Pisanty, B., Oğuz, E. C., Nisoli, C. & Shokef, Y. Putting a spin on metamaterials: Mechanical incompatibility as magnetic frustration. *SciPost Physics* **10,** 136 (2021).
38. Resch, R. D. *Geometrical device having articulated relatively movable sections* United States of America Patent 3201894 (1965).
39. Evans, K. E. & Alderson, A. Auxetic Materials: Functional Materials and Structures from Lateral Thinking! *Advanced Materials* **12,** 617–628 (2000).

40. Lubensky, T., Kane, C., Mao, X., Souslov, A. & Sun, K. Phonons and elasticity in critically coordinated lattices. *Reports on Progress in Physics* **78,** 073901 (2015).
41. Maxwell, J. C. L. on the calculation of the equilibrium and stiffness of frames. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **27,** 294–299 (1864).
42. Calladine, C. R. Buckminster Fuller's "tensegrity" structures and Clerk Maxwell's rules for the construction of stiff frames. *International journal of solids and structures* **14,** 161–172 (1978).
43. Isakov, S. V., Moessner, R. & Sondhi, S. L. Why spin ice obeys the ice rules. *Physical review letters* **95,** 217201 (2005).
44. Hong, Y. P. & Pan, C.-T. Rank-revealing QR Factorizations and the singular value decomposition. *Mathematics of Computation* **58,** 213–232 (1992).
45. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural Networks* **2,** 359–366 (1989).
46. Schmidt, J., Marques, M. R., Botti, S. & Marques, M. A. Recent advances and applications of machine learning in solid-state materials science. *npj computational materials* **5,** 83 (2019).
47. Merchant, A. *et al.* Scaling deep learning for materials discovery. *Nature* **624,** 80–85 (2023).
48. Zheng, X., Zhang, X., Chen, T.-T. & Watanabe, I. Deep learning in mechanical metamaterials: from prediction and generation to inverse design. *Advanced Materials* **35,** 2302530 (2023).
49. Bessa, M. A., Glowacki, P. & Houlder, M. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. *Advanced Materials* **31,** 1904845 (2019).
50. Coli, G. M. & Dijkstra, M. An Artificial Neural Network Reveals the Nucleation Mechanism of a Binary Colloidal AB_{13} Crystal. *ACS Nano* **15,** 4335–4346 (2021).
51. Bastek, J.-H., Kumar, S., Telgen, B., Glaesener, R. N. & Kochmann, D. M. Inverting the structure–property map of truss metamaterials by deep learning. *Proceedings of the National Academy of Sciences* **119,** 1. ISSN: 0027-8424 (2022).
52. Deng, B. *et al.* Inverse design of mechanical metamaterials with target nonlinear response via a neural accelerated evolution strategy. *Advanced Materials* **34,** 2206238 (2022).
53. Alexander, S. Amorphous solids: their structure, lattice dynamics and elasticity. *Physics Reports* **296,** 65–236. ISSN: 0370-1573. https://www.sciencedirect.com/science/article/pii/S0370157397000690 (1998).
54. Van Hecke, M. Jamming of soft particles: geometry, mechanics, scaling and isostaticity. *Journal of Physics: Condensed Matter* **22,** 033101 (2009).
55. Liu, A. J. & Nagel, S. R. The jamming transition and the marginally jammed solid. *Annu. Rev. Condens. Matter Phys.* **1,** 347–369 (2010).
56. Nicolas, A., Ferrero, E. E., Martens, K. & Barrat, J.-L. Deformation and flow of amorphous solids: Insights from elastoplastic models. *Reviews of Modern Physics* **90,** 045006 (2018).
57. Sigmund, O. Materials with prescribed constitutive parameters: an inverse homogenization problem. *International Journal of Solids and Structures* **31,** 2313–2329 (1994).
58. Sigmund, O. On the design of compliant mechanisms using topology optimization. *Journal of Structural Mechanics* **25,** 493–524 (1997).
59. Zadpoor, A. A. Mechanical meta-materials. *Mater. Horiz.* **3,** 371–381. http://dx.doi.org/10.1039/C6MH00065G (5 2016).

60. Ion, A. *et al. Metamaterial mechanisms* in *Proceedings of the 29th annual symposium on user interface software and technology* (2016), 529–539.
61. Zhu, B. *et al.* Design of compliant mechanisms using continuum topology optimization: A review. *Mechanism and Machine Theory* **143,** 103622 (2020).
62. Dykstra, D. M., Janbaz, S. & Coulais, C. The extreme mechanics of viscoelastic metamaterials. *APL Materials* **10,** 080702 (2022).
63. Coulais, C., Kettenis, C. & van Hecke, M. A characteristic length scale causes anomalous size effects and boundary programmability in mechanical metamaterials. *Nature Physics* **14,** 40–44 (2018).
64. Fan, H., Tian, Y., Yang, L., Hu, D. & Liu, P. Multistable mechanical metamaterials with highly tunable strength and energy absorption performance. *Mechanics of Advanced Materials and Structures* **29,** 1637–1649 (2022).
65. Coulais, C., Sabbadini, A., Vink, F. & van Hecke, M. Multi-step self-guided pathways for shape-changing metamaterials. *Nature* **561,** 512–515 (2018).
66. Kane, C. & Lubensky, T. Topological boundary modes in isostatic lattices. *Nature Physics* **10,** 39–45 (2014).
67. Paulose, J., Chen, B. G.-g. & Vitelli, V. Topological modes bound to dislocations in mechanical metamaterials. *Nature Physics* **11,** 153–156 (2015).
68. Ghatak, A., Brandenbourger, M., Van Wezel, J. & Coulais, C. Observation of non-Hermitian topology and its bulk–edge correspondence in an active mechanical metamaterial. *Proceedings of the National Academy of Sciences* **117,** 29561–29568 (2020).
69. Mullin, T., Deschanel, S., Bertoldi, K. & Boyce, M. C. Pattern transformation triggered by deformation. *Physical review letters* **99,** 084301 (2007).
70. Doškář, M. *et al.* Wang tiles enable combinatorial design and robot-assisted manufacturing of modular mechanical metamaterials. *arXiv preprint arXiv:2305.09280* (2023).
71. Shim, J., Perdigou, C., Chen, E. R., Bertoldi, K. & Reis, P. M. Buckling-induced encapsulation of structured elastic shells under pressure. *Proceedings of the National Academy of Sciences* **109,** 5978–5983 (2012).
72. Waitukaitis, S., Menaut, R., Chen, B.-g. & van Hecke, M. Origami multistability: From single vertices to metasheets. *Physical review letters* **114,** 055503 (2015).
73. Silverberg, J. L. *et al.* Origami structures with a critical transition to bistability arising from hidden degrees of freedom. *Nature materials* **14,** 389–393 (2015).
74. Dudte, L. H., Vouga, E., Tachi, T. & Mahadevan, L. Programming curvature using origami tessellations. *Nature materials* **15,** 583–588 (2016).
75. Overvelde, J. T. *et al.* A three-dimensional actuated origami-inspired transformable metamaterial with multiple degrees of freedom. *Nature communications* **7,** 10929 (2016).
76. Overvelde, J. T., Weaver, J. C., Hoberman, C. & Bertoldi, K. Rational design of reconfigurable prismatic architected materials. *Nature* **541,** 347–352 (2017).
77. Nežerka, V. *et al.* A jigsaw puzzle metamaterial concept. *Composite Structures* **202,** 1275–1279 (2018).
78. Hu, Z. *et al.* Engineering zero modes in transformable mechanical metamaterials. *Nature Communications* **14,** 1266 (2023).
79. Van Mastrigt, R., Dijkstra, M., van Hecke, M. & Coulais, C. Machine learning of implicit combinatorial rules in mechanical metamaterials. *Physical Review Letters* **129,** 198003 (2022).

80. The kink in the probability to find class (i) unit cells through Monte Carlo sampling of the design space is most likely due to the change in how we calculate the the number of zero modes $\#M(n)$ for $k \geq 6$ onward. See [79] for details on this calculation.
81. Czajkowski, M., Coulais, C., van Hecke, M. & Rocklin, D. Conformal elasticity of mechanism-based metamaterials. *Nature communications* **13,** 1–9 (2022).
82. Grima, J. N. & Evans, K. E. Auxetic behavior from rotating squares. *Journal of materials science letters* **19,** 1563–1565 (2000).
83. Bertoldi, K., Reis, P. M., Willshaw, S. & Mullin, T. Negative Poisson's ratio behavior induced by an elastic instability. *Advanced materials* **22,** 361–366 (2010).
84. Coulais, C., Overvelde, J. T. B., Lubbers, L. A., Bertoldi, K. & van Hecke, M. Discontinuous buckling of wide beams and metabeams. *Physical review letters* **115,** 044301 (2015).
85. Kim, J. Z., Lu, Z., Strogatz, S. H. & Bassett, D. S. Conformational control of mechanical networks. *Nature Physics* **15,** 714–720 (2019).
86. Choi, G. P., Dudte, L. H. & Mahadevan, L. Programming shape using kirigami tessellations. *Nature materials* **18,** 999–1004 (2019).
87. Choi, G. P. T., Dudte, L. H. & Mahadevan, L. Compact reconfigurable kirigami. *Physical Review Research* **3,** 043030 (2021).
88. Dykstra, D. M. & Coulais, C. Inverse design of multishape metamaterials. *arXiv preprint arXiv:2304.12124* (2023).
89. Liu, K., Sun, R. & Daraio, C. Growth rules for irregular architected materials with programmable properties. *Science* **377,** 975–981 (2022).
90. Bian, X., Wei, L.-Y. & Lefebvre, S. Tile-based pattern design with topology control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* **1,** 1–15 (2018).
91. Van Mastrigt, R., Dijkstra, M., van Hecke, M. & Coulais, C. *Zero Modes and Classification of Combinatorial Metamaterials* 2022.
92. Hoffmann, J. *et al.* Machine learning in a data-limited regime: Augmenting experiments with synthetic data uncovers order in crumpled sheets. *Science advances* **5,** eaau6792 (2019).
93. Colen, J. *et al.* Machine learning active-nematic hydrodynamics. *Proceedings of the National Academy of Sciences* **118,** 10 (2021).
94. Falk, M. J., Alizadehyazdi, V., Jaeger, H. & Murugan, A. Learning to Control Active Matter. *arXiv preprint arXiv:2105.04641* (2021).
95. Dulaney, A. R. & Brady, J. F. Machine learning for phase behavior in active matter systems. *Soft Matter* **17,** 6808–6816 (2021).
96. Bar-Sinai, Y., Hoyer, S., Hickey, J. & Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences* **116,** 15344–15349 (2019).
97. Wiecha, P. R., Arbouet, A., Girard, C. & Muskens, O. L. Deep learning in nanophotonics: inverse design and beyond. *Photonics Research* **9,** B182–B200 (2021).
98. Ma, W. *et al.* Deep learning for the design of photonic structures. *Nature Photonics* **15,** 77–90 (2021).
99. Xu, Y., Zhang, X., Fu, Y. & Liu, Y. Interfacing photonics with artificial intelligence: an innovative design strategy for photonic structures and devices based on artificial neural networks. *Photonics Research* **9,** B135–B152 (2021).

100. Harrington, M., Liu, A. J. & Durian, D. J. Machine learning characterization of structural defects in amorphous packings of dimers and ellipses. *Physical Review E* **99,** 022903 (2019).
101. Mozaffar, M. *et al.* Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences* **116,** 26414–26420 (2019).
102. Gu, G. X., Chen, C.-T. & Buehler, M. J. De novo composite design based on machine learning algorithm. *Extreme Mechanics Letters* **18,** 19–28 (2018).
103. Wang, L. *et al.* Deep Generative Modeling for Mechanistic-based Learning and Design of Metamaterial Systems. *arXiv preprint arXiv:2006.15274* (2020).
104. Coli, G. M., Boattini, E., Filion, L. & Dijkstra, M. Inverse design of soft materials via a deep learning–based evolutionary strategy. *Science Advances* **8,** eabj6731 (2022).
105. Shin, D. *et al.* Spiderweb nanomechanical resonators via Bayesian optimization: inspired by nature and guided by machine learning. *Advanced Materials,* 2106248 (2021).
106. Hanakata, P. Z., Cubuk, E. D., Campbell, D. K. & Park, H. S. Forward and inverse design of kirigami via supervised autoencoder. *Physical Review Research* **2,** 042006 (2020).
107. Forte, A. E. *et al.* Inverse Design of Inflatable Soft Membranes Through Machine Learning. *Advanced Functional Materials* **32,** 2111610 (2022).
108. Cubuk, E. D. *et al.* Identifying structural flow defects in disordered solids using machine-learning methods. *Physical Review Letters* **114,** 108001 (2015).
109. Bapst, V. *et al.* Unveiling the predictive power of static structure in glassy systems. *Nature Physics* **16,** 448–454 (2020).
110. Schoenholz, S. S., Cubuk, E. D., Sussman, D. M., Kaxiras, E. & Liu, A. J. A structural approach to relaxation in glassy liquids. *Nature Physics* **12,** 469–471 (2016).
111. Hsu, Y.-T., Li, X., Deng, D.-L. & Sarma, S. D. Machine learning many-body localization: Search for the elusive nonergodic metal. *Physical Review Letters* **121,** 245701 (2018).
112. Venderley, J., Khemani, V. & Kim, E.-A. Machine learning out-of-equilibrium phases of matter. *Physical Review Letters* **120,** 257204 (2018).
113. Swanson, K., Trivedi, S., Lequieu, J., Swanson, K. & Kondor, R. Deep learning for automated classification and characterization of amorphous materials. *Soft matter* **16,** 435–446 (2020).
114. Van Damme, R., Coli, G. M., van Roij, R. & Dijkstra, M. Classifying Crystals of Rounded Tetrahedra and Determining Their Order Parameters Using Dimensionality Reduction. *ACS Nano* **14,** 15144–15153 (2020).
115. Miles, C. *et al.* Correlator convolutional neural networks as an interpretable architecture for image-like quantum matter data. *Nature Communications* **12,** 1–7 (2021).
116. Andrejevic, N., Andrejevic, J., Rycroft, C. H. & Li, M. Machine learning spectral indicators of topology. *arXiv preprint arXiv:2003.00994* (2020).
117. Bohrdt, A. *et al.* Classifying snapshots of the doped Hubbard model with machine learning. *Nature Physics* **15,** 921–924 (2019).
118. Carrasquilla, J. Machine learning for quantum matter. *Advances in Physics: X* **5,** 1797528 (2020).
119. Van Nieuwenburg, E., Bairey, E. & Refael, G. Learning phase transitions from dynamics. *Physical Review B* **98,** 060301 (2018).
120. Bohrdt, A. *et al.* Analyzing nonequilibrium quantum states through snapshots with artificial neural networks. *Physical Review Letters* **127,** 150504 (2021).

121. Sigaki, H. Y., Lenzi, E. K., Zola, R. S., Perc, M. & Ribeiro, H. V. Learning physical properties of liquid crystals with deep convolutional neural networks. *Scientific Reports* **10,** 1–10 (2020).
122. Geiger, P. & Dellago, C. Neural networks for local structure detection in polymorphic systems. *The Journal of Chemical Physics* **139,** 164105 (2013).
123. Dietz, C., Kretz, T. & Thoma, M. Machine-learning approach for local classification of crystalline structures in multiphase systems. *Physical Review E* **96,** 011301 (2017).
124. Zhang, L.-F. *et al.* Machine learning topological invariants of non-Hermitian systems. *Physical Review A* **103,** 012419 (2021).
125. Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. *Nature Physics* **13,** 431–434 (2017).
126. Van Nieuwenburg, E. P., Liu, Y.-H. & Huber, S. D. Learning phase transitions by confusion. *Nature Physics* **13,** 435–439 (2017).
127. Deng, D.-L., Li, X. & Sarma, S. D. Machine learning topological states. *Physical Review B* **96,** 195145 (2017).
128. Zhang, Y. & Kim, E.-A. Quantum loop topography for machine learning. *Physical Review Letters* **118,** 216401 (2017).
129. Zhang, P., Shen, H. & Zhai, H. Machine learning topological invariants with neural networks. *Physical Review Letters* **120,** 066401 (2018).
130. Zhang, Y. *et al.* Machine learning in electronic-quantum-matter imaging experiments. *Nature* **570,** 484–490 (2019).
131. Ch'Ng, K., Carrasquilla, J., Melko, R. G. & Khatami, E. Machine learning phases of strongly correlated fermions. *Physical Review X* **7,** 031038 (2017).
132. Rem, B. S. *et al.* Identifying quantum phase transitions using artificial neural networks on experimental data. *Nature Physics* **15,** 917–920 (2019).
133. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583–589 (2021).
134. Demaine, E. D. & Demaine, M. L. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics* **23,** 195–208 (2007).
135. Imaizumi, M. & Fukumizu, K. *Deep Neural Networks Learn Non-Smooth Functions Effectively* in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics* (eds Chaudhuri, K. & Sugiyama, M.) **89** (PMLR, Apr. 2019), 869–878. `https://proceedings.mlr.press/v89/imaizumi19a.html`.
136. Rolnick, D., Veit, A., Belongie, S. & Shavit, N. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694* (2017).
137. Goodfellow, I. *et al.* Generative adversarial nets. *Advances in Neural Information Processing Systems* **27** (2014).
138. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
139. Bengio, Y., Lodi, A. & Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research* **290,** 405–421 (2021).
140. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM* **64,** 107–115 (2021).
141. Johnson, J. M. & Khoshgoftaar, T. M. Survey on deep learning with class imbalance. *Journal of Big Data* **6,** 1–54 (2019).
142. Van Mastrigt, R., Dijkstra, M., van Hecke, M. & Coulais, C. *Convolutional Neural Networks for Classifying Combinatorial Metamaterials* 2022.

143. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
144. Hossin, M. & Sulaiman, M. N. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process* **5,** 1 (2015).
145. Dijkstra, M. & Luijten, E. From predictive modelling to machine learning and reverse engineering of colloidal self-assembly. *Nature materials* **20,** 762–773 (2021).
146. Vargo, E. *et al.* Using Machine Learning to Predict and Understand Complex Self-Assembly Behaviors of a Multicomponent Nanocomposite. *Advanced Materials* **34,** 2203168 (2022).
147. Kim, B., Lee, S. & Kim, J. Inverse design of porous materials using artificial neural networks. *Science advances* **6,** eaax9324 (2020).
148. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nature methods* **16,** 687–694 (2019).
149. Goverde, C. A., Wolf, B., Khakzad, H., Rosset, S. & Correia, B. E. De novo protein design by inversion of the AlphaFold structure prediction network. *Protein Science* **32,** e4653 (2023).
150. Guo, K. & Buehler, M. J. A semi-supervised approach to architected materials design using graph neural networks. *Extreme Mechanics Letters* **41,** 101029 (2020).
151. Mao, Y., He, Q. & Zhao, X. Designing complex architectured materials with generative adversarial networks. *Science advances* **6,** eaaz4169 (2020).
152. Brown, N. K., Garland, A. P., Fadel, G. M. & Li, G. Deep reinforcement learning for the rapid on-demand design of mechanical metamaterials with targeted nonlinear deformation responses. *Engineering Applications of Artificial Intelligence* **126,** 106998 (2023).
153. Ha, C. S. *et al.* Rapid inverse design of metamaterials based on prescribed mechanical behavior through machine learning. *Nature Communications* **14,** 5765 (2023).
154. Van Mastrigt, R., Coulais, C. & van Hecke, M. Emergent nonlocal combinatorial design rules for multimodal metamaterials. *Phys. Rev. E* **108,** 065002. `https://link.aps.org/doi/10.1103/PhysRevE.108.065002` (6 Dec. 2023).
155. Huang, P.-S., Boyken, S. E. & Baker, D. The coming of age of de novo protein design. *Nature* **537,** 320–327 (2016).
156. Lovelock, S. L. *et al.* The road to fully programmable protein catalysis. *Nature* **606,** 49–58 (2022).
157. Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R. & Jensen, K. F. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **12,** e1608 (2022).
158. Grima, J. N., Alderson, A. & Evans, K. Auxetic behaviour from rotating rigid units. *Physica Status Solidi (b)* **242,** 561–575 (2005).
159. Gu, M. & Eisenstat, S. C. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing* **17,** 848–869 (1996).
160. Bron, C. & Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM* **16,** 575–577 (1973).
161. Hormoz, S. & Brenner, M. P. Design principles for self-assembly with short-range interactions. *Proceedings of the National Academy of Sciences* **108,** 5193–5198 (2011).
162. Van Mastrigt, R. *Code for calculating zero modes* `https://uva-hva.gitlab.host/published-projects/CombiMetaMaterial`. 2023.
163. Van Mastrigt, R. *Code for inverse design* `https://uva-hva.gitlab.host/published-projects/inversecombimat`. 2024.

164. Meurer, A. *et al.* SymPy: symbolic computing in Python. *PeerJ Computer Science* **3,** e103. ISSN: 2376-5992. https://doi.org/10.7717/peerj-cs.103 (Jan. 2017).
165. Perron, L. & Didier, F. *CP-SAT* version v9.8. Google, Nov. 15, 2023. https://developers.google.com/optimization/cp/cp_solver/.
166. Bradbury, J. *et al. JAX: composable transformations of Python+NumPy programs* version 0.3.13. 2018. http://github.com/google/jax.
167. Eppstein, D., Löffler, M. & Strash, D. *Listing all maximal cliques in sparse graphs in near-optimal time* in *Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I 21* (2010), 403–414.

Summary

Understanding how simple building blocks conspire to create larger structures with emergent properties is key to design functional matter. Think of molecular design, which aims to arrange atoms in molecular structures with desired properties; of protein engineering, which aims to sequence amino acids in a chain to fold into a functional protein; of self-assembly, which aims to combine building blocks with distinct shapes and interactions to spontaneously assembly into larger structures. In all these examples, building blocks with relatively simple local interactions combine into larger spatial structures with new, emergent properties. Underlying this process is the abstract notion of a combinatorial design (or configuration) space that contains all possible arrangements of building blocks, i.e., the structures. In turn, each of these structures can be dressed with an emergent property. For example, the stability of the molecule; the binding affinity of the protein; the yield of assembled shapes. Crucially, there is an order to this dressed design space that can be captured in so-called design rules: a set of conditions that a structure should satisfy to yield the desired property.

In this thesis, we aim to utilize such design rules to design mechanical metamaterials: engineered compound materials that leverage geometric effects to achieve unusual mechanical properties beyond the mechanical properties of the constituent materials. For example, an ordinary block of rubber surrounded by air expands laterally under uniaxial compression. However, if this block of rubber is perforated with carefully placed holes, it will contract laterally under uniaxial compression—in other words, the metamaterial is auxetic. Crucially, the mechanical properties of the rubber and air have not changed, but the arrangement (or geometry) of these materials has, resulting in a diametric mechanical response.

Specifically, we focus on so-called multimodal metamaterials. Such metamaterials utilize multiple soft deformation pathways to achieve exceptional properties, such as selectable mechanical responses, sequential energy-absorption, nonlocal resonances, and multi-shape folding. The key challenge is to design metamaterials with multiple, spatially extended desired deformations and few undesired deformations. However, designing multimodal metamaterials is fiendishly difficult: deformations are sensitive to minute changes in the design; the design space is too vast to enumerate; and superfluous deformations proliferate rapidly and impede actuation of desired deformations. A systematic approach to design such metamaterial is lacking. Here, we set out to find systematic design strategies.

To achieve this goal, we take a combinatorial approach to design: we restrict ourselves to a discrete set of building blocks. This combinatorial approach is ideal to design metamaterials with spatially textured deformations for, e.g., shape-morphing. The challenge is to find tilings of these building blocks that yield metamaterials with desired deformation pathways. For combinatorial metamaterials with a single deformation, targeted designs follow simple tiling rules. However, for multimodal metamaterials the challenge is exacerbated as we aim to control both the structure and enumeration of the deformation pathways. Targeted tilings are rare exceptions in a vast sea of failed designs, and exploration of this design space is unfeasible without design rules. This is characteristic of combinatorial design problems that are ubiquitous in science, for example in the fields mentioned before. As such, many of the strategies we develop in this thesis are of relevance beyond the field of metamaterial design. Below follows a summary of the main results and insights per chapter.

In **chapter 1**, we provide a brief background to mechanism-based metamaterial design, combinatorial metamaterials, and machine learning. Specifically, we take a mechanism-based approach to metamaterial design that uses a frame of rigid bars and hinges to derive and design deformations. We show that the structural integrity of such frames, captured in the Maxwell-Calladine count, is crucial to prevent the rapid proliferation of undesired, spatially localized deformations and we provide a condition on the integrity of unit cell designs based on this count. We show that this condition is satisfied in earlier work of a shape-morphing combinatorial metamaterial and how, for this metamaterial, one can use local tiling rules to design desired spatially textured deformations. However, that particular metamaterial design is limited to a single deformation at most. Thus, we focus on another family of metamaterials comprised of bimodal building blocks which feature two deformations. Using random sampling of the design space, we show that this metamaterial features intensive, spatially extended deformations and extensive, spatially localized deformations. How to design this metamaterials to control these deformations remains an open question at this point. Finally, we discuss machine learning, in particular neural networks, and how application of such algorithms to combinatorial problems is relatively unexplored.

In **chapter 2**, we derive a set of design rules for the extensive deformations. To do so, we first set-up a mathematical framework for the

deformations in a tiling of building blocks. Using this framework, we get expressions for kinematic constraints between neighboring building blocks. For any deformation to be supported by a configuration of building blocks, all these constraints should be satisfied. This requirement allows us to define a transfer mapping, that maps kinematic degrees of freedom (dof) of a set of building blocks to the dof of neighboring building blocks that satisfy the kinematic constraints between the two sets. However, there are additional local kinematic constraints that stem from constraints to the structure of extensive deformations. We explicitly derive conditions on the configurations of building blocks to satisfy these additional constraints for so-called ’strip’ modes. These strip modes are localized along horizontal or vertical strips in the configuration and comprise the extensive deformations of the metamaterial. Surprisingly, we find emergent nonlocal constraints for the configuration—this is exclusive to multimodal metamaterials and exemplifies the challenge in designing such metamaterials. Finally, we conjecture a set of general design rules for strip modes and verify them numerically with complete agreement. More broadly, our strategy is applicable to combinatorial problems with emergent nonlocal constraints beyond Wang tiles.

In **chapter 3**, we show that convolutional neural networks (CNNs) are remarkably adapt at learning combinatorial rules from data, despite heavily undersampled training sets. While our strategy in Ch. 2 is successful, it is rather strenuous. Instead, here we ask if a neural network might be able to infer the design rules from data alone. We show that CNNs are accurate at classifying combinatorial metamaterials into rare compatible (C) and abundant incompatible (I) designs. We ascribe this success to the CNNs’ ability to capture local spatial correlations, because local kinematic constraints are key to determine the compatibility of deformation with a given design. Moreover, we note that the structure of C designs in design space delineate a filamentous set of hyperplanes that span the space, which we intuitively picture as needles (C) in a haystack (I). In this picture, it is clear that the boundaries of these needles are relatively undersampled—I designs are likely to be far removed from any needles. On the one hand, if the network is naively interpolating the data, it would overestimate the width of the needles. On the other hand, the design space is structured by combinatorial rules, which the network might be able to infer. Thus, we ask whether the trained CNN is able to accurately delineate these undersampled boundaries. To answer this question, we probe the boundary using random walks, starting from C designs. We find a characteristic response of the probability to transition to an I design, which we neatly fit with a single free parameter that represents the dimensionality of the hyperplanes (the width of the needles). We find that the CNN's classification agrees remarkably well with the true dimensionality. This suggests the CNN infers the underlying combinatorial rules from the sparse training set, rather than doing simple interpolation. This opens up new possibilities for design of complex metamaterials and application to other combinatorial problems.

In **chapter 4**, we devise a two-step design strategy for metamaterials with multiple desired deformations. The goal is to create a metamaterial with multiple spatially patterned deformations. However, direct optimization for such designs is unfeasible—the design space is jagged and vast. Instead, we first create a library of high *pluripotency* designs which are likely to deform close to randomized target deformations, rather than specific deformations. In our metamaterial, the pluripotency is positively correlated to the number of intensive modes. However, designs with a high number of intensive modes are very rare. Crucially, there is structure to this number in the design space: designs with the same number form connected sets in design space and are surrounded by sets of designs with one less intensive mode. Intuitively, we represent this space as 'needles-within-needles-within-needles in a haystack', where each needle corresponds to sets of designs with the same number of intensive modes. To find such designs, we train a CNN to predict the number of intensive modes and couple it to a genetic algorithm (GA) to generate designs with a high number of intensive modes. We find that the CNN is able to extrapolate to designs beyond the range of its training set, which allows us to find high pluripotency designs using the GA. We ascribe the success of this approach to the underlying structure of pluripotency in the design space. In the second step, we focus on specific target deformations. From our library, we select and combine designs to create a new design that features the desired deformations. However, this design also feature multiple undesired deformations. We remove these superfluous deformations by strategically placing defects in the design. Finally, we show the success of our approach by designing two metamaterials that feature deformations that resemble a smiley and frowny face, and that resemble the letters A and U. Our design strategy opens up systematic design of complex metamaterials with multiple spatially textured deformation modes that are relevant to programmable materials, soft robotics, and computing *in materia*. More broadly, our strategy is relevant to combinatorial problems with jagged design spaces where direct optimization is unfeasible.

In conclusion, we have derived both rational and computational design strategies for multimodal combinatorial metamaterials. Crucial to the success of these methods is the notion of an underlying structure in the design space. By capturing this structure, either through a mathematical framework or using machine learning, we have shown that the previously intractable design space can be effectively explored to find metamaterials with desired properties. This notion is present in all combinatorial design problems, which makes our strategies relevant beyond metamaterial design to fields such as self-assembly, molecular design, protein engineering, computer graphics, and chip design. Even more broadly, this work is a small contribution to a long standing effort to better understand and control emergent behavior in complex systems. We have put special effort in contextualizing our approaches and results beyond metamaterial design and hope that a broader spectrum of scientists find interest in our work.

Samenvatting

Voor het ontwerp van functionele materialen is het cruciaal om te begrijpen hoe simpele bouwstenen samenwerken in grotere structuren om nieuwe eigenschappen tot uiting te brengen. Denk hierbij bijvoorbeeld aan het ontwerpen van moleculen, waarbij atomen juist geplaatst moeten worden in een moleculaire structuur; het ontwerpen van eiwitten waarin de volgorde van de aminozuren het vouwen en de uiteindelijke functie van het eiwit bepaalt; en aan zelfassemblage waarbij bouwstenen met specifieke vormen en interacties zo gecombineerd worden dat ze spontaan samenkomen in een grotere structuur.

In al deze voorbeelden worden bouwstenen met relatief simpele lokale interacties gecombineerd tot een grotere ruimtelijke structuur wat nieuwe, emergente eigenschappen voortbrengt. Ten grondslag aan dit ontwerpproces ligt het concept van een ontwerpruimte (of configuratieruimte) die alle mogelijke rangschikkingen van de bouwstenen omvat, d.w.z., alle mogelijke structuren. Vervolgens kan aan elk van deze structuren een eigenschap worden toegekend. Bijvoorbeeld de stabilititeit van het molecuul; de bindingsaffiniteit van het eiwit; de gemiddelde opbrengst van geassembleerde vormen. Om te ontwerpen is het cruciaal dat er een (onbekende) ordening zit in deze ontwerpruimte die kan worden beschreven met zogeheten ontwerpregels: een verzameling aan condities waaraan een structuur moet voldoen om de gewenste eigenschap te hebben.

In dit proefschrift wordt gebruik gemaakt van zulke regels voor het ontwerpen van mechanische metamaterialen: samengestelde materialen die geometrische effecten gebruiken om zo ongebruikelijke mechanische eigenschappen te verkrijgen die meer zijn dan de som van de mechanische eigenschappen van de betreffende materialen. Neem bijvoorbeeld een regulier blok rubber omsloten met lucht. Dit blok wordt dikker als je het indrukt. Echter, als dit blok rubber op de juiste manier met gaten wordt geperforeerd, dan zal het blok op zichzelf inklappen en dus dunner worden als je het indrukt. In andere woorden, het geperforeerde blok rubber is auxetisch en een metamateriaal. Hierbij is het cruciaal dat de mechanische eigenschappen van het rubber en de lucht onveranderd blijven, alleen de ruimtelijke verdeling (geometrie) van de betreffende materialen (rubber en lucht) is veranderd. Het veranderen van de geometrie resulteert dus in een diametrale mechanische respons van het metamateriaal.

In het bijzonder richt dit proefschrift zich op zogeheten multimodale metamaterialen. Zulke metamaterialen maken gebruik van meerdere vervomingspaden om zo uitzonderlijke eigenschappen zoals een selecteerbare mechanische respons, opeenvolgende absorptie van energie, niet-lokale resonanties, en meervorming vouwen te verkrijgen. De kern uitdaging is om metamaterialen te ontwerpen met meerdere ruimtelijk gespreide gewenste vervormingen en weinig ongewenste vervormingen. Echter is het ontwerpen van zulke multimodale metamaterialen erg uitdagend. Vervormingen zijn gevoelig voor kleine veranderingen in het ontwerp, de ontwerpruimte is te groot om helemaal te benoemen, en overbodige vervormingen vermenigvuldigen zich snel en hinderen de aandrijving van de gewenste vervormingen. Een systematische methode om zulke metamaterialen te ontwerpen ontbreekt. Dit proefschrift probeert zo'n systematische methode op te stellen.

Om dit doel te bereiken gebruiken we een combinatorische aanpak voor het ontwerpen. Dat wil zeggen, we beperken onszelf tot een discrete verzameling bouwstenen. Deze combinatorische aanpak is ideaal voor het ontwerp van metamaterialen met ruimtelijke en getextureerde vervormingen voor, bijvoorbeeld, vormveranderende materialen. De uitdaging is om legpatronen van deze bouwstenen te vinden die leiden tot metamaterialen met gewenste vervormingen. Voor enkele vervormingen kunnen combinatorische metamaterialen ontworpen worden met behulp van simpele betegelingsregels. Voor multimodale metamaterialen is deze uitdaging moeilijker, omdat we zowel de structuur en het aantal van de vervormingen willen bepalen. Succesvolle legpatronen zijn zeldzame uitzonderingen in een zee van gefaalde ontwerpen en het verkennen van de ontwerpruimte is niet mogelijk zonder ontwerpregels. Dit is typerend voor combinatorische ontwerpproblemen die alom vertegenwoordigd zijn in de wetenschap, bijvoorbeeld in de eerder genoemde velden. Veel van de ontwerpstrategieën die we ontwikkelen zijn daarom ook relevant buiten het veld van metamaterialen. Hieronder volgt een samenvatting van de belangrijkste resultaten en inzichten per hoofdstuk.

In **hoofdstuk 1** geven wij wat achtergrondinformatie over mechanisch metamateriaal ontwerp, combinatorische metamaterialen en *machine learning*. In het bijzonder gebruiken wij een mechanistische aanpak voor het ontwerp van metamaterialen, waarbij we gebruik maken van een frame staven en scharnieren om vervormingen af te leiden en te ontwerpen. We laten zien dat de structurele integriteit van zulke frames, die we vangen in de Maxwell-Calladine formule, cruciaal is om de spreiding van ongewenste, ruimtelijk gelokaliseerde vervormingen te voorkomen. Daarnaast geven wij een voorwaarde aan de structurele integriteit van eenheidscellen gebaseerd op deze formule. We laten zien dat aan deze voorwaarde wordt voldaan in eerder onderzoek naar vormveranderende combinatorische metamaterialen en hoe, voor dit specifieke metamateriaal, men lokale betegelingsregels kan afleiden en zo gewenste ruimtelijk getextureerde vervormingen kan ontwerpen. Echter is dit specifieke ontwerp gelimiteerd tot enkele vervormingen. Daarom richten we ons daarna op een andere familie van metamaterialen bestaande uit bimodale bouwstenen, d.w.z, bouwstenen die op twee manieren kunnen vervormen. Metamaterialen opgebouwd uit deze bouwstenen zijn in staat om op meerdere manieren te vervormen. Aan de hand van willekeurige legpatronen laten we zien dat dit metamateriaal beschikt over intensieve ruimtelijk uitgebreide vervormingen en extensieve ruimtelijk gelokaliseerde vervormingen. Op dit punt blijft het een open vraag hoe we dit metamateriaal moeten ontwerpen om deze soort vervormingen te bepalen. We sluiten af met een korte introductie over machine learning, in het bijzonder neurale netwerken, en hoe toepassingen van zulke algoritmes om combinatorische problemen op te lossen relatief onontgonnen is.

In **hoofdstuk 2** verzamelen we ontwerpregels voor extensieve vervormingen. Hiervoor formuleren we eerst een wiskundig kader om de vervorming te bepalen van een legpatroon bouwstenen. Met dit kader leiden we formules af voor kinematische beperkingen tussen burende bouwstenen. Een configuratie bouwstenen moet aan al deze formules voldoen om een bepaalde vervorming te ondersteunen. Met deze eis aan de configuratie kunnen we een overdrachtsafbeelding definiëren die de kinematische vrijheidsgraden (vg) van een verzameling bouwstenen afbeeldt naar de vg van burende bouwstenen zodat deze voldoen aan de kinematiche beperkingen tussen de twee verzamelingen. Er zijn echter extra lokale kinematische beperkingen die voortkomen uit de structuur van extensieve vervormingen. We leiden voorwaarden af voor configuraties bouwstenen om aan deze extra voorwaarden te voldoen voor zogeheten *strip modes*. Deze strip modes zijn vervormingen die zich lokaliseren in horizontale of verticale stroken in de configuratie en omvatten de extensieve vervormingen van het metamateriaal. Verrassend genoeg vinden we dat deze extra beperkingen resulteren in emergente, niet-lokale beperkingen aan het legpatroon — dit soort niet-lokale beperkingen zijn typerend voor multimodale metamaterialen en illustreert waarom zulke metamaterialen ontwerpen zo uitdagend is. Als laatste postuleren wij een verzameling algemene ontwerpregels voor strip modes en testen deze regels numeriek met volledige overeenkomst. In het algemeen is onze strategie toepasbaar op combinatorische problemen met emergente, niet-lokale beperkingen die voorbij de restricties van Wang legpatronen gaan.

In **hoofdstuk 3** laten we zien dat convolutionele neurale netwerken (CNNs) opmerkelijk goed geschikt zijn voor het leren van combinatorische regels uit data, ondanks dat de *training set* onderbemonsterd is. Alhoewel onze methode in hfst. 2 succesvol is, is deze erg inspannend om toe te passen. Daarom vragen we ons in dit hoofdstuk af of een neuraal netwerk in staat is om de ontwerpregels uit alleen de data af te leiden. We laten zien dat CNNs heel accuraat zijn in het classificeren van combinatorische metamaterialen in zeldzame compatibele (C) en overvloedige incompatibele (I) ontwerpen. We onderschrijven dit succes aan de vaardigheid van de CNNs om lokale ruimtelijke correlaties te herkennen, omdat lokale kinematische beperkingen cruciaal zijn voor de compatibiliteit van een vervorming voor een gegeven ontwerp. Daarnaast merken we op dat de structuur van C ontwerpen een draadvormige verzameling hypervlakken afbakent die de ontwerpruimte opspannen. We beschrijven deze structuur intuïtief als naalden (C) in een hooiberg (I). Met dit beeld in gedachte is het duidelijk dat de grenzen van deze naalden relatief onderbemonsterd zijn — het is waarschijnlijk dat de meeste I ontwerpen ver zijn verwijderd van naalden. Aan de ene kant is het te verwachten dat als het netwerk simpelweg de data naïef interpoleert, dat de CNN dan de wijdte van de naalden zou overschatten. Aan de andere kant, precies omdat de ontwerpruimte een structuur heeft door de combinatorische regels, zou het netwerk deze regels kunnen afleiden en de structuur van de naalden goed kunenn vangen. Wij vragen ons dus af of een getrainde CNN in staat is om deze onderbemonsterde grenzen accuraat af te bakenen. Om deze vraag te beantwoorden onderzoeken we de grens met behulp van willekeurige wandelingen in de ontwerpruimte beginnende bij C ontwerpen. We vinden een karakteristieke respons in de kans om tijdens deze wandeling in een I ontwerp te veranderen, wat we kunnen passen aan een functie met een enkele vrije parameter. Deze parameter representeert de dimensionaliteit van de hypervlakken, d.w.z., de wijdte van de naalden. We vinden dat de CNN's classificatie opmerkelijk goed overeenkomt met de werkelijke dimensionaliteit van de naalden. Dit suggereert dat de CNN de onderliggende combinatorische regels weet af te leiden uit de schaarse training set, in plaats van dat het simpelweg de data interpoleert. Dit opent nieuwe mogelijkheden tot het ontwerp van complexe metamaterialen en toepassing to andere combinatorische problemen.

In **hoofdstuk 4** introduceren we een tweetakt ontwerpstrategie voor metamaterialen met meerdere gewenste vervormingen. Het doel is om een metamateriaal te ontwerpen met meerdere ruimtelijke gepatroneerde vervormingen. Direct optimaliseren van zulke ontwerpen is in de praktijk echter niet haalbaar — de ontwerpruimte is grillig en uitgestrekt. In plaats daarvan genereren we eerst een verzameling aan ontwerpen met een hoge pluripotentie. Zulke ontwerpen hebben een grote kans om zich te kunnen zo te kunnen vervormen dat het lijkt op willekeurig gegenereerde vervormingen. Pluripotentie is positief gecorreleerd aan het aantal intensieve vervormingen in ons metamateriaal. Helaas zijn ontwerpen met een hoog aantal intensieve vervormingen erg zeldzaam. Gelukkig heeft de ontwerpruimte een structuur wat betreft dit aantal vervormingen: ontwerpen met hetzelfde aantal vormen verbinden verzamelingen in de ontwerpruimte en zijn omgeven met verzamelingen van ontwerpen met een enkele intensieve vervorming minder. Deze ruimte omschrijven we intuïtief als ‘naalden-in-naalden-in-naalden in een hooiberg’, waar elke naald correspondeert tot een verzameling ontwerpen met hetzelfde aantal intensieve vervormingen. Om zulke ontwerpen te vinden, trainen we een CNN om het aantal intensieve vervormingen te voorspellen en koppelen de CNN aan een genetisch algoritme (GA) om zo efficiënt ontwerpen met een hoog aantal intensieve vervormingen te genereren. We vinden dat de CNN in staat is om te extrapoleren naar ontwerpen met aantallen voorbij de limieten van de *training set*. Dit stelt ons in staat om het GA te gebruiken om ontwerpen met een hoge pluripotentie te vinden. Het succes van deze methode kennen we toe aan de onderliggende structuur van pluripotentie in de ontwerpruimte. In de tweede ontwerpstap richten we ons op specifieke doelvervormingen. Uit onze verzameling ontwerpen met een hoge pluripotentie selecteren en combineren we ontwerpen, en maken zo een nieuw ontwerp met de gewenste vervormingen. Echter bevat dit ontwerp ook ongewenste, overbodige vervormingen. De ongewenste vervormingen halen we weg door defecten strategisch te plaatsen in het ontwerp. Tot slot laten we zien dat onze methode in staat is om twee grotere metamaterialen te ontwerpen die vervormingen hebben die lijken op een *smiley* en een *frowny* gezichtje, en op de letters A en U. Onze ontwerpstrategie opent de deur tot systematisch ontwerp van complexe metamaterialen met meerdere ruimtelijke getextureerde vervormingen. Zulke metamaterialen zijn relevant voor programmeerbare materialen, zachte robots, en berekeningen doen in materialen. In het algemeen is onze strategie relevant voor combinatorische problemen met grillige ontwerpruimtes waarbij directe optimalisatie niet mogelijk is.

Samengevat, we hebben zowel rationele als computationele ontwerpstrategieën afgeleid voor multimodale combinatorische metamaterialen. De kern van het succes van deze methodes ligt in het idee van een onderliggende structuur in de ontwerpruimte. Door deze structuur te vangen, zij het door een wiskundig kader of door machine learning, laten we zien dat dat de voorheen onhandelbare ontwerpruimte efficiënt kan worden verkend om metamaterialen met gewenste eigenschappen te vinden. Dit idee is algemeen voor alle combinatorische ontwerpproblemen, wat ons onderzoek relevant maakt, naast alleen metamateriaalontwerp, voor velden als zelfassemblage, molecuulontwerp, eiwitontwerp, computer graphics en chipontwerp. Nog breder gezien is dit werk een kleine contributie aan een langlopende poging om emergent gedrag in complexe systemen beter te begrijpen en te controleren. Wij hebben extra moeite gestoken in het contextualiseren van onze methodes en resultaten voorbij metamateriaalontwerp en hopen dat het interesse wekt bij een breed spectrum aan wetenschappers.

Publications

This thesis is based on the following publications.

1. **van Mastrigt, R.**, Dijkstra, M., van Hecke, M., Coulais, C. Machine Learning of Implicit Combinatorial Rules in Mechanical Metamaterials. *Phys. Rev. Lett.* **129**, 198003 (2022).

2. **van Mastrigt, R.**, Coulais, C., van Hecke, M. Emergent nonlocal combinatorial design rules for multimodal metamaterials. *Phys. Rev. E* **108**, 065002 (2023).

3. **van Mastrigt, R.**, Dijkstra, M., van Hecke, M., Coulais, C. Prospecting for Pluripotency in Metamaterial Design. *Manuscript in preparation* (2024).

Other Publications by the author

4. **van Mastrigt, R.**, Coulais, C., van Hecke, M., Dijkstra, M. Machine Learning of Mechanisms in Combinatorial Metamaterials. *2021 Fifteenth International Congress on Artificial Materials for Novel Wave Phenomena (Metamaterials)*, 442-444 (2021).

Word of thanks

Right at the start of my PhD, the pandemic and first lockdown happened. Many of the joys of doing a PhD—summer schools, conferences, brainstorming sessions—were no longer possible, and work sometimes felt isolating. Luckily, I was blessed with fun colleagues, wonderful supervisors, and loving friends and family that pulled me through difficult times in myriad ways. It goes without saying that I could not have finished my PhD without the support from many wonderful people. It is not possible to capture my gratitude for you all in words, but I will try nonetheless.

First of all, I want to thank my supervisors. Marjolein, your sharp comments always managed to pinpoint exactly the weak points of my work. You inspire me to stay critical and to continuously work on improving my research. Martin, your ability to look for the bigger picture is inspiring and helped me abstract and conceptualize my work. I also want to thank you for your mentoring, helping me find my next challenge, and taking time out of your busy schedule to chat and joke around. Corentin, your seemingly unlimited enthusiasm, creativity and energy is a great motivator and helped me keep the joy of doing research alive. I hope to take some of your energy and creativity with me going forward. These past years were some of the most interesting, challenging and fun of my life (so far) and I feel privileged to have had you as my supervisors.

Secondly, I want to thank my peers for all the support and fun over the past years. Despite splitting my time between two groups, I always felt welcome at both—for that I am truly thankful. Especially, I want to thank my wonderful colleagues Amitesh, Ananya, Anne, Bernat, Caroline, Colin, Daan, Daniel, David, Ellen, Freek, Hadrien, Jack, Jonas, Lennard, Lishuai, Lucie, Margot, Martin, Paul, Rupesh, Shahram, Wenfeng, Xiaofei, Yao and Yiangnan for all the wonderful coffee chats, lunches, dinners, *borrels*, and outings. A special shout-out goes out to those that joined me for the GeoGuessr sessions during Covid. I also want to thank all the softies for the fun lunches and engaging Friday talks. Without such a fun, open, and diverse set of people this journey would have been a whole lot more boring.

Last but certainly not least, I want to thank my family and friends for supporting me. Work is only one part of life, and I thank you all for providing me with welcome distractions, joy, laughter, and love. To my parents, without whom I wouldn't be who I am today, for your love and continuous support in my ambitions. To Kevin and Imre for your willingness to listen to me complain and keeping me grounded. To my wonderful friends, Jan Willem, Maarten, Nick, Niels, Pim, Ruben, Thomas, and Tristan, for all the fun, and drinks, and games, and music, and late night walks and chats—I am beyond grateful to have such a loving group of close friends. To Anna and Scott, for always making me feel welcome. To my dear Doris, for holding me together throughout this ordeal. I am extremely grateful for your support and love, I will spend the rest of my days trying to reciprocate even just a fraction of all you have so generously given me. I love you all.