# Symmetric Implicit Surface Microstructures for Hierarchical Design

RUIQI CHEN, HARDIK KABARIA, and ADRIAN J. LEW, Stanford University, USA

We present a framework for generating hierarchical structures using a palette of symmetric implicit surface microstructures. The framework utilizes an unstructured mesh composed solely of triangles, quadrilaterals, tetrahedrons, or hexahedrons to serve as a scaffold for microstructural unit cells to tile, where each unit cell is represented by an implicit surface and defined using a set of symmetric basis functions derived from Fourier series. Any arbitrary unit cell may be converted into a symmetric implicit surface using an Fast-Fourier-Transform-based method as long as containment queries (i.e. inside or outside) can be performed on the unit cell. Using our framework, we demonstrate blending together different unit cells to create new ones as well as spatial transitions between different unit cells. In addition, we present a method to constrain unit cells to intersect its bounding polygon/polyhedron at a right angle, which we postulate is necessary for $C^1$ continuity. Finally, we briefly mention some mechanical applications that may benefit from our framework.

CCS Concepts: • **Applied computing** → **Computer-aided design**; Aerospace; • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: hierarchical structure, implicit surface, lattice, microstructure, symmetry

**ACM Reference Format:**
Ruiqi Chen, Hardik Kabaria, and Adrian J. Lew. 2025. Symmetric Implicit Surface Microstructures for Hierarchical Design. *ACM Trans. Graph.* 37, 4, Article 111 (August 2025), 5 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

### 1.1 Contributions

To the best of our knowledge, this work is the first to present a framework for generating hierarchical structures using *unstructured* scaffold meshes composed solely of one element type: triangles, quadrilaterals, tetrahedrons, and hexahedrons. Furthermore, this work is the first to present a method of transitioning between arbitrarily chosen unit cells within the unstructured scaffold mesh. By supporting the most common unstructured mesh types, we ensure that our framework is compatible with a wide range of off-the-shelf mesh generators and can be used to generate hierarchical structures from complex design domains that are practically unsuitable for structured meshing.

In short, this paper makes the following contributions:

- We develop a novel method to construct hierarchical structures from an unstructured mesh scaffold comprised solely of triangles, quadrilaterals, tetrahedrons, or hexahedrons.
- Our method allows for multiple cell types to be prescribed, enabling spatially varying hierarchical structures.
- We enable constraining the unit cell to intersect its bounding polygon/polyhedron at a right angle by solving a constrained least squares problem.

## 2 Related Works

## 3 Methodology

### 3.1 Symmetric Implicit Surfaces

Implicit surfaces represent a geometric object through an equation of the form $\Phi(\mathbf{x}) = 0$ where $\Phi : \mathbb{R}^d \to \mathbb{R}$ is some function that evaluates a query point $\mathbf{x} \in \mathbb{R}^d$ and returns zero if $\mathbf{x}$ is on the surface, returns a negative value if $\mathbf{x}$ is inside the surface, or returns a positive value if $\mathbf{x}$ is outside the surface (note that this convention is not universal and some authors use the opposite convention). In this work, we are interested in implicit surface functions that represent 2D and 3D geometries, so we assume that $d = 2$ or $d = 3$. The magnitude of the return value of the implicit surface function may have some geometric meaning, but this is not required. For example, signed distance functions (SDFs) are a subset of implicit surface equations where the magnitude of the return value denotes the distance from $\mathbf{x}$ to the surface.

A symmetric implicit surface is defined in this work to be an implicit surface equation that is invariant under transformations from a chosen point group $\mathcal{S}$. In other words, $\Phi(\mathbf{x}) = \Phi(\mathbf{Qx})$, where $\mathbf{Q} : \mathbb{R}^{d\times d} \in \mathcal{S}$ is an orthonormal matrix that represents an isometry (i.e. rotation, reflection, inversion, or rotoinversion of a point $\mathbf{x}$ about the origin). To construct an implicit surface suitable for generating hierarhical structures, we want to constrain the implicit surface to match the point group of the cell type in the unstructured scaffold mesh to tile: dihedral group of degree 3 and order 6 ($D_3$) for triangle meshes, dihedral group of degree 4 and order 8 ($D_4$) for quadrilateral meshes, achiral tetrahedral symmetry ($T_d$) for tetrahedron meshes, and achiral octahedral symmetry ($O_h$) for hexahedron meshes. The list of symmetry operators for these four symmetry groups is given in Appendix A. The necessity of symmetric implicit surfaces to tile the unstructured scaffold mesh is explained in detail in Section 3.4, but in brief, it is the symmetry that ensures $C^0$ continuity at the boundaries of the hierarchical structure. Furthermore, symmetry is crucial for enabling the spatial transitions described in Section 3.7.

To construct a symmetric implicit surface $\Phi(\mathbf{x})$, we first construct a symmetric orthonormal basis function $\phi(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$. The symmetric basis function is constructed from some other conventional (i.e. non-symmetric) orthonormal basis function $\psi(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ by iterating over the symmetry operators $\mathbf{Q}$ and summing together $\psi$ composed with the linear transformation of $\mathbf{x}$ by $\mathbf{Q}$:

$$\phi(\mathbf{x}) = K \sum_{\mathbf{Q}\in\mathcal{S}} \psi(\mathbf{Qx}) \tag{1}$$

Authors' Contact Information: Ruiqi Chen, rchensix@alumni.stanford.edu; Hardik Kabaria, hardikk@alumni.stanford.edu; Adrian J. Lew, lewa@stanford.edu, Stanford University, Stanford, California, USA.

ACM 1557-7368/2025/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

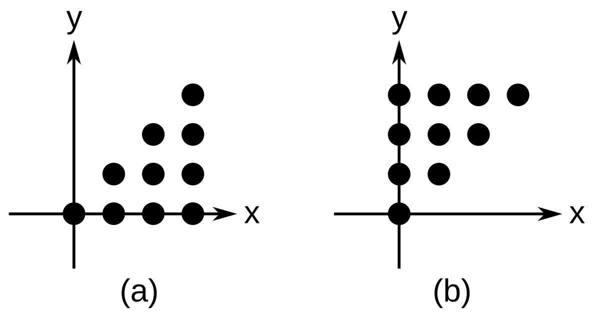

Fig. 1. A subset $\mathcal{F}$ of the $\mathbb{Z}^2$ lattice defines unique spatial frequencies that can be used to generate unit cells with $D_4$ symmetry. Both (a) and (b) are valid representations for $\mathcal{F}$, and it is a matter of user preference to choose.

where $K$ is a normalization factor to make the basis function orthonormal. In this work, the $d$-dimensional Fourier series

$$\psi_{\mathbf{f}}(\mathbf{x}) = e^{2\pi i \mathbf{f}^T \mathbf{x}} \tag{2}$$

is used, where $\mathbf{f} : \mathbb{R}^d \in \mathcal{F}$ is an integer-valued vector that represents spatial frequencies in the Cartesian directions, and $\mathcal{F} \subset \mathbb{Z}^d$ is a subset of the $d$-dimensional integer lattice that is specific to the chosen scaffold mesh type. In other words, $\mathcal{F}$ contains all of the *unique* spatial frequencies that may be used to construct a symmetric basis functions. Any spatial frequencies not in $\mathcal{F}$ do not need to be considered due to symmetry. For example, suppose the scaffold mesh is a quadrilateral mesh, then one representation for $\mathcal{F}$ is given by

$$\mathcal{F} = \left\{(f_x, f_y) \in \mathbb{Z}^2 | f_y \geq 0 \text{ and } f_x - f_y \geq 0\right\} \tag{3}$$

This representation of $\mathcal{F}$ for a quadrilateral mesh is depicted in Figure 1(a). Note that if all of the symmetry operators in $S$ are applied to $\mathcal{F}$, then we would recover $\mathbb{Z}^2$. Also note that the representation of $\mathcal{F}$ is not unique; Figure 1(b) shows an alternative representation that is equally valid. Representations of $\mathcal{F}$ used in this work for triangles, quadrilaterals, tetrahedrons, and hexahedrons are given in Appendix A. Additionally, note that by writing the Fourier series using complex exponentials (as expressed in Equation 2), Equation 1 is, in general, a function that takes in real-valued $d$-D points and returns complex valued scalars, i.e. $\phi_{\mathbf{f}}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{C}$. However, for the purpose of generating implicit surfaces, only the real or imaginary part (but not both) of the final expression for $\Phi(\mathbf{x})$ will be used.

There are many reasons, both mathematical and practical, as to why the Fourier series is an excellent choice for the foundational basis function, but the primary reasons are: Fourier series are an orthonormal basis, Fourier series have a very simple expression for the gradient (this is used in Section 3.3); there are fast, robust algorithms available–most notably, the fast Fourier transform (FFT)–for computing the cofficients of a Fourier expansion; and finally, several famous triply periodic minimal surfaces (TPMS) commonly used to construct hierarchical lattice structures (e.g. Schwarz primitive, gyroid) may be approximated using Fourier series [Wohlgemuth et al. 2001].

Substituting Equation 2 into Equation 1, the symmetric basis function may be expressed as

$$\phi_{\mathbf{f}}(\mathbf{x}) = K_{\mathbf{f}} \sum_{\mathbf{Q} \in \mathcal{S}} e^{2\pi i \mathbf{f}^T \mathbf{Q} \mathbf{x}} \tag{4}$$

where $K_{\mathbf{f}}$ is the normalization factor given by

$$K_{\mathbf{f}} = \left( \int_{\Omega} \sum_{\mathbf{Q}_1, \mathbf{Q}_2 \in \mathcal{S}} e^{2\pi i \mathbf{f}^T (\mathbf{Q}_1 - \mathbf{Q}_2) \mathbf{x}} d\mathbf{x} \right)^{-1/2} \tag{5}$$

In the integrand in the denominator of Equation 5, due to the orthonormal property of Fourier series, only terms where $\mathbf{f}^T (\mathbf{Q_1} - \mathbf{Q_2}) = \mathbf{0}$ contribute to the integral.

Just like the Fourier series, the resulting basis function in Equation 4 is orthonormal, i.e.

$$\begin{aligned} \int_{\Omega} \phi_{\mathbf{f}}(\mathbf{x}) \overline{\phi_{\mathbf{g}}(\mathbf{x})} d\mathbf{x} &= 0 \\ \int_{\Omega} \phi_{\mathbf{f}}(\mathbf{x}) \overline{\phi_{\mathbf{f}}(\mathbf{x})} d\mathbf{x} &= 1 \end{aligned} \tag{6}$$

for some $\mathbf{f}, \mathbf{g} \in \mathcal{F}$ and $\mathbf{f} \neq \mathbf{g}$. The symmetric implicit surface $\Phi(\mathbf{x})$ may now be expressed in terms of a sum of the basis functions:

$$\Phi(\mathbf{x}) = \sum_{\mathbf{f} \in \mathcal{F}} c_{\mathbf{f}} \phi_{\mathbf{f}}(\mathbf{x}) \tag{7}$$

where $c_{\mathbf{f}}$ are the weight coefficients for each basis function.

Given an arbitrary input implicit surface $g(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, which may or may not be symmetric, a symmetric approximation to $g(\mathbf{x})$ may be computed by projecting $g(\mathbf{x})$ onto the basis functions to recover the weight coefficients:

$$c_{\mathbf{f}} = \int_{\Omega} g(\mathbf{x}) \phi_{\mathbf{f}}(\mathbf{x}) dV \tag{8}$$

In this case, because the input function $g(\mathbf{x})$ is real-valued, the output function $\Phi(\mathbf{x})$ will also be real-valued. In practice, Equation 8 will be evaluated numerically, and the unit cell to approximate does not have to be an implicit surface equation, as demonstrated in Section 3.2.

*3.1.1 Choice of* $\Omega$. Before proceeding, we need to clarify what the domain $\Omega$ is. Recall that the goal of defining these symmetric orthonormal basis functions is to use them to approximate an input unit cell. Thus, $\Omega$ should be a (non-strict) superset of the domain of the unit cell. However, $\Omega$ should also ideally be carefully chosen such that the $d$-dimensional Fourier series is an orthonormal basis in $\Omega$; while this is not strictly necessary for the purposes of approximating the unit cell, it greatly simplifies things and enables us to utilize efficient algorithms such as the FFT and existing software packages (e.g. numpy.fft [Harris et al. 2020] and scipy.fft [Virtanen et al. 2020]) to compute the approximation. For unit cells defined in the square or cube, the choice of $\Omega$ is trivial: $\Omega$ will be the square or cube, respectively. For unit cells defined in the regular triangle or tetrahedron, we choose $\Omega$ to be an *extended domain* that is a superset of the regular triangle or tetrahedron, and which the Fourier series is orthonormal.

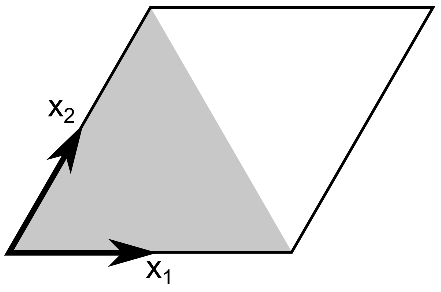

Fig. 2. A regular triangle (grey), its extended domain Ω (black outline), and skew coordinate system defined by directions $x_1$ and $x_2$.

*3.1.2 Extended Domain for Triangle and Tetrahedron.* For a regular triangle, Ω is chosen to be a rhombus with 60° and 120° interior angles (i.e. the union of two regular triangles). In addition, a skew coordinate system where $\mathbf{x_1}$ and $\mathbf{x_2}$ are 60° apart is used so that the Fourier series expressed in this skew coordinate system is orthogonal. Note that we can easily make the Fourier series orthonormal in Ω by scaling by the square root of the Jacobian determinant:

$$\sqrt{\begin{vmatrix}\mathbf{x_1} & \mathbf{x_2}\end{vmatrix}} = \sqrt{\begin{vmatrix}1 & \cos(60^\circ) \\ 0 & \sin(60^\circ)\end{vmatrix}} = \sqrt{\frac{2}{\sqrt{3}}} \tag{9}$$

A regular triangle, its extended domain Ω, and the skew coordinate system are depicted in Figure 2. For a regular tetrahedron, Ω is chosen to be the unit cube because, intuitively, both the cube and tetrahedron have $T_d$ symmetry (i.e. $T_d$ is a subgroup of $O_h$). A regular tetrahedron and its extended domain Ω are depicted in Figure **??**.

In addition to defining the extended domains, we also have to define how to extend the original unit cell into the extended domains. In this work, we extend the unit cell geometry by mirroring the unit cell about the edges (faces) of the triangle (tetrahedron) until the entire extended domain is populated. This space-filling process intuitively mimics how hierarchical structures are constructed from a unit cell. Without this step, the approximate implicit surface geometries may "disappear" at the boundaries of the original unit cell. We note that the extended domains and methods of extending the original unit cell we have chosen are not unique.

In addition, it is worth stepping back to understand *why* we choose to use the method of extended domains rather than, say, some other orthonormal basis function that is orthonormal in the *original* unit cell domain. The main reason is due to practicality: we want to use the Fourier series as our starting basis function to take advantage of the FFT algorithm and existing software libraries. If we were to use some other orthonormal basis, we would first have to *find* some other orthonormal basis (via Gram-Schmidt or similar algorithms), then numerically evaluate the integrals to compute the coefficients for approximating the unit cell. While all this is technically feasible, we suspect there may be some challenges with numerical stability and computational efficiency. Thus, we choose to use the method of extended domains in this paper instead.

## 3.2 Converting Geometry Representations to Implicit Surfaces

Frequently, a user will choose a unit cell based on some criteria (e.g. aesthetics, porosity, mechanical performance, etc.) to create the hierarchical structure. The unit cell to tile may not be an implicit surface; furthermore, it may not have *any* symmetry either. To accomodate nearly arbitrary incoming geometry representations (e.g. triangle meshes, NURBS, voxel grids), the only requirement is that the input representation has a method to determine whether a query point is inside or outside the geometry. For example, containment queries on oriented point clouds and triangle meshes/soups can be computed extremely robustly using the fast winding number method [Barill et al. 2018]. However, practically speaking, better results are obtained if the input representation has a method to query the signed distance instead of just the containment because a signed distance query will generally be smoother than a binary containment query, which helps reduce artifacts associated with the Gibbs phenomenon. In addition, if the unit cell is not symmetric, a preprocessing step to force the unit cell to be symmetric may yield better results. This forced symmetricization may be performed using various methods and the methods will depend on the input unit cell representation. For example, if the unit cell is a triangle mesh, then one may apply the transformations from $\mathcal{S}$ onto copies of the unit cell and union together the transformed copies. Note that this forced symmetricization preprocessing step is not necessary–fitting an implicit surface approximation using symmetric basis functions to a non-symmetric input unit cell will still yield a symmetric result–however, the results may be better and helps avoid cases where the implicit surface approximation "disappears". Finally, for triangle and tetrahedron based unit cells, the unit cell should be extended into Ω.

To numerically compute the implicit surface approximation $\Phi(\mathbf{x})$, first, the (extended) unit cell is positioned such that it is fully contained by Ω. The unit cell is then sampled by querying the containment status (or, better yet, the signed distance) on an $N^d$ grid of equispaced query points to yield a sampling of $g(\mathbf{x})$. Next, the fast Fourier transform (FFT) is applied to the sampling of $g(\mathbf{x})$. Finally, the weight coefficients $c_\mathbf{f}$ are computed from the FFT result by noting that the right hand side of Equation 8 is simply a summation of FFT coefficients due to the fact that $\phi_\mathbf{f}(\mathbf{x})$ is itself a summation of 3D complex exponentials.

TODO: give an example? Or an algorithm?

It is worth noting that if the input geometry is band limited (i.e. the input geometry is generated from a 3D Fourier series implicit equation with a finite set of spatial frequencies), then using the FFT to compute the weight coefficients $c_\mathbf{f}$ will be exact up to limitations in floating point computations, as long as $N$ is large enough to satisfy the Nyquist-Shannon sampling theorem, i.e. $h < \frac{N}{2}$ where $h \in \mathbb{Z}$ is the maximum magnitude integer frequency in any basis direction of the input geometry. However, if the input geometry is not band limited (e.g. there are sharp edges and corners), then this method will only yield approximate values of $c_\mathbf{f}$ due to aliasing. In practice, given some arbitrary input geometry that may or may not be band limited, one can choose some reasonable starting value for $N$ such that the FFT can compute relatively quickly (for example

$N = 2^7$), compute the implicit surface approximation to the input, and use some stopping criteria to determine if $N$ is large enough. One example of stopping criteria is the Hausdorff distance between the input geometry and output approximation. Furthermore, choosing powers of two for values of $N$ is practical due to exploiting efficiencies in the FFT algorithm. The topic of convergence of the Fourier series is well studied in literature. Since the main purpose of this paper is to develop methods to generate hierarchical structures, i.e. the main purpose is *not* about the convergence of Fourier series, most examples in this paper will use some "large-enough", arbitrarily chosen value for $N$.

As previously mentioned, better results are achieved if the signed distance of the unit cell is used instead of the binary containment query. Another reason to use the signed distance is because the implicit surface approximation is effectively an approximation of the signed distance function of the unit cell, which means performing certain downstream unit cell modifications such as erosion and dilation is trivial. This is incredibly important and useful for generating hierarchical structures with spatially varying thicknesses, as depicted in Section TODO.

## 3.3 Face Normal Surfaces

In addition to symmetry constraints, the implicit surface unit cell can be further constrained to intersect with its bounding polygon/polyhedron at a right angle. We postulate this is an important property to have in order to generate hierarchical structures with $C^1$ continuity in higher order meshes with $C^1$ continuity constraints applied to control points (see Section 3.4.1). To apply this face normal constraint, we first note that the outward normal vector of an implicit surface is given by its gradient:

$$\mathbf{N}(\mathbf{x}) = \nabla\Phi(\mathbf{x}) \tag{10}$$

To constrain the implicit surface to intersect its bounding polygon/polyhedron at a right angle, we need its outward normal to be at a right angle to the outward normal $\mathbf{N}_p$ of the polygon/polyhedron face. Thus, the dot product between these two vectors must be zero when evaluated on the polygon/polyhedron faces:

$$\nabla\Phi(\mathbf{x}) \cdot \mathbf{N}_p |\mathbf{x} \rightarrow \mathbf{x}_p = 0 \tag{11}$$

where $\mathbf{x}_p$ is a point on the bounding polygon/polyhedron face. Note that due to the symmetry constraint already imposed from our usage of symmetric basis functions, we only need to satisfy Equation 11 on one face of the bounding polygon/polyhedron, as it will be automatically applied to the other faces. We can rewrite $\mathbf{x}_p$ as

$$\mathbf{x}_p = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{12}$$

where $\mathbf{A} : \mathbb{R}^{d\times d}$ is a projection matrix and $\mathbf{b} : \mathbb{R}^d$ is an offset term. Equation 12 can be geometrically interpreted as projecting a point $\mathbf{x}$ onto the extended plane of the chosen face of the bounding polygon/polyhedron. Using Equation 4 and 7, the gradient term can be written as:

$$\begin{aligned} \nabla\Phi(\mathbf{x}) &= \sum_{\mathbf{f}\in\mathcal{F}} c_{\mathbf{f}} \nabla\phi_{\mathbf{f}}(\mathbf{x}) \\ &= 2\pi i \sum_{\mathbf{f}\in\mathcal{F}} c_{\mathbf{f}} K_{\mathbf{f}} \sum_{\mathbf{Q}\in\mathcal{S}} \mathbf{Q}^T \mathbf{f} e^{2\pi i \mathbf{f}^T \mathbf{Q}\mathbf{x}} \end{aligned} \tag{13}$$

## 3.4 Tiling Mesh Scaffolds

### 3.4.1 *Higher Order Mesh Scaffolds.*

## 3.5 Dilation, Erosion, and Inversion

Dilating, eroding, and inverting the implicit surface unit cell is trivial.

## 3.6 Hybridization

### 3.6.1 *Weighted Averaging.*

### 3.6.2 *Minkowski Sum and Difference.*

## 3.7 Spatial Transitions

## 3.8 Generalizing to Other Symmetry Groups

## 4 Discussion

## 4.1 Limitations

TODO continuity of implicit surface equations, enforcing continuity, etc.

TODO continuity of hybridization and transitions, how to pick transition path to enforce continuity, etc.

## 4.2 Future Work

TODO optimizing weight coefficients to achieve some target, i.e. porosity, stiffness, etc.

## 5 Conclusion

A publically available implementation of this work is available at https://github.com/rchensix/tetrahedral_implicit_lattice under an MIT License. However, please note that while the implementation itself falls under an MIT License, methods and ideas mentioned in this paper are, at the time of writing, under consideration for a patent [Chen and Kabaria 2022].

## Acknowledgments

TODO

## References

Mois I. Aroyo (Ed.). 2016. *International Tables for Crystallography Volume A: Space-group symmetry.* John Wiley & Sons, Inc., New York, NY, USA. doi:10.1107/97809553602060000114

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics* 37, 4 (July 2018), 1–12. doi:10.1145/3197517.3201337

Ruiqi Chen and Hardik Kabaria. 2022. Systems and methods for constructing lattice objects for additive manufacturing. https://patents.google.com/patent/WO2022212472A1 Patent Application No. WO2022212472A1, Filed March 30, 2022, Published October 6, 2022.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. doi:10.1038/s41586-020-2649-2

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors.

2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. doi:10.1038/s41592-019-0686-2

Meinhard Wohlgemuth, Nataliya Yufa, James Hoffman, and Edwin L. Thomas. 2001. Triply periodic bicontinuous cubic microdomain morphologies by symmetries. *Macromolecules* 34, 17 (July 2001), 6083–6089. doi:10.1021/ma0019499

## A Symmetry Groups

All symmetry operators here are written in crystallographic notation, composed of a $d$-tuple representing the result of applying a linear transformation to a point $\mathbf{x} \in \mathbb{R}^d$ (affine symmetry operators are not used in this work). An overbar denotes negation. For example, $x, \overline{z}, y$ represents the linear transformation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ -z \\ y \end{bmatrix} \tag{14}$$

An authoritative source enumerating all space groups is the International Tables of Crystallography [Aroyo 2016].

### A.1 Dihedral Group of Degree 3 and Order 6

The dihedral group of degree 3 and order 6 ($D_3$) is the symmetry group of an equilateral triangle.

### A.2 Dihedral Group of Degree 4 and Order 8

The dihedral group of degree 4 and order 8 ($D_4$) is the symmetry group of a square.

### A.3 Achiral Tetrahedral Symmetry

Achiral tetrahedral symmetry ($T_d$) is the symmetry group of a regular tetrahedron.

$$\mathcal{F} = \left\{ \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \in \mathbb{Z}^3 | f_y + f_z \geq 0 \text{ and} f_y - f_z \geq 0 \text{ and} f_x - f_y \geq 0 \right\} \tag{15}$$

### A.4 Achiral Octahedral Symmetry

Achiral tetrahedral symmetry ($O_h$) is the symmetry group of a cube (as well as that of a regular octahedron, hence the name). There are 48 symmetry operators.

$$\begin{gathered} x, y, z \\ x, \overline{y}, \overline{z} \end{gathered} \tag{16}$$

Received DD MONTH YYYY; revised DD MONTH YYYY; accepted DD MONTH YYYY