Skeletal-based microstructure representation and convolution reconstruction

Devyani Jivani[a], Rahul Rai[b], Olga Wodo[a,*]

[a]*Materials Design and Innovation Department, University at Buffalo, New York, United States*
[b]*Department of Automotive Engineering, Department of Mechanical Engineering, Clemson University, Clemson University International Center for Automotive Research (CU-ICAR), Greenville, South Carolina, United States*

Abstract

This paper introduces the skeletal-based microstructure representation. The skeletal representation allows for compact storage while capturing the main characteristics of the microstructure; and the separation of geometry and topology. The skeletal representation consists of two elements: primitive-based skeleton, representing topology, and a kernel representing the shape of the primitives in the skeleton. We supplement the representation with three operations: skeletonization, primitive identification, and convolution of the skeleton with a kernel. To demonstrate the robustness of the representation, we perform a quantitative study to show the decrease in the number of parameters needed to represent the several types of microstructure with respect to the voxel-based representation. We report a low error of reconstruction of the proposed representation method.

Keywords: Microstructure representation, skeletonization, convolution surfaces, microstructure informatics.

1. Introduction

The microstructure is a complex hierarchical and spatial internal organization of a material system which controls the system's properties. Microstructure images are collected through tedious imaging processes [1, 2] or via simulations [3, 4] with the goal to provide detailed, high-resolution, raw data. The raw data can be stored as structured data (e.g., micrograph from SEM [1] and phase field simulations [3]), or unstructured data (e.g., point cloud data from APT [5] and molecular dynamic simulations [4]). The size of the raw data depends on the resolution and the size of the sample, with even modest samples having sizes in the range of 100^2 to 100^3. In this context, imaging techniques inevitably produce high-dimensional datasets. This high dimensionality poses the challenge of extracting a small set of physically meaningful descriptors (to establish process structure property [PSP] maps) that explain the most of the variability observed in material properties. Moreover, when

*olgawodo@buffalo.edu

approaching the inverse design of a microstructure with desired properties, the high dimensionality significantly limits the search for optimal design due to the high number of design variables to be explored. In both cases, the capabilities heavily depend on the microstructure representation.

Significant foundational work has been done in defining various representations to gain a fundamental understanding of material behavior and for inverse design. The microstructure representation methods include voxel-based representations, characterization via physical descriptors [6] [7], statistical functions [8, 9], spectral density functions (SDF) [10, 11] and machine learning methods [12, 13]. Machine learning methods, have been adopted only recently, while the other methods have been advancing over the last decade, for example, statistical functions that include two-point correlation functions, lineal-path functions, and cluster correlation functions [14, 15]. More details on various representations can be found in the review papers [6, 16].

The choice of representation is directly linked to the task at hand. For example, when constructing a PSP link is of interest, microstructure characterization using physical descriptors provides interpretable features. Descriptors include volume fractions of constituents, total surface area, pore sizes, tortuosity and aspect ratio, among others [7, 17]. Descriptors also include the information about the geometry of the microstructure with limited topological information. Another class of microstructure representation is spectral density function (SDF) that treats the microstructure as a realization of a random field. The SDF of a microstructure is defined as the squared magnitude of its Fourier transform and is sufficient to characterize some microstructures [18]. It can take a simple parametric form and use it for inverse design through a phase recovery technique [19]. In this sense, SDF provides a low-dimensional representation for different microstructure types. This is efficient in capturing individual features and randomness in the microstructure; however, it cannot explicitly capture topological features such as type and degree of branching or intersections.

Although various methods of microstructure representation exist, none of these explicitly capture or characterize the topology of the microstructure. The need for topological descriptors is reaffirmed by recent studies on the mechanical properties of nano-porous gold. Mangipudi et.al. [20] showed that the prefactor in the scaling law of stiffness and strength in nano-porous gold scales linearly with the genus (topological descriptor) of the cellular solid topology. This is particularly significant because, by considering the topology, a more general law that spans across several materials has been proposed. This example demonstrates a lack of a topology-centric representation which further limits the knowledge discovery and materials design at the microstructural scale.

In this paper, we introduce a topology-centric alternative for microstructure representation. The skeletal representation explicitly captures the topological features of the microstructure. The representation requires significantly fewer variables to preserve the micrographs, while providing basic operations for the microstructure reconstruction as well as structural feature characterization. The reduction of the variables is favourable for efficient microstructure optimization and inverse design. Microstructure reconstruction, on the other hand, is critical for synthetic microstructure generation that aids the consideration of

topology and geometry separately. For example, the proposed representation will allow for explicit access to the topological features and the domain sizes independently.

We formulate the skeletal representation for microstructural data and describe three operations associated with the representation: skeletonization, primitive extraction, and convolution. We demonstrate the capabilities of our approach by performing two-way transformation of a two-phase microstructure from a voxel-based representation to our representation and vice versa. We show the performance of the representation for spinodal decomposition structures generated using the Cahn-Hilliard equation. We quantify the error of reconstruction for the two-way transformation for several microstructures that vary in terms of topological features (i.e., droplet-like structures vs. interpenetrated structures) and domain sizes. We report a low number of primitives required to represent the microstructure topology and a low error rate in reconstruction. The reported results demonstrate that the representation not only decreases the size of the feature space, but it is also flexible enough to model shapes for a wide range of microstructures.

2. Methodology

In this section, we provide the mathematical formalism for the skeletal representation of the microstructure. We detail three algorithms that we use to convert the array of voxels into a skeletal-based representation and then revert it back to the voxel-based representation. Figure 1 depicts the steps for the two-way transformation and the operations associated with the intermediate steps. The forward transformation involves the skeletonization operation to identify the skeleton (symmetry axis). In our approach, we represent the skeleton in two ways: as a set of voxels and as a set of primitives (e.g., line segments, arcs). We use the terms voxel skeleton and primitive skeleton, respectively, to distinguish the two forms of skeletons. The inverse transformation convolutes the primitive skeleton with potential function or kernel function. The convolution operation is applied to every primitive to produce a microstructure field. One iso-surface is selected from that field to segment the microstructure into two phases. The parameters of the primitives and the kernel function are optimized to minimize errors in reconstruction between the input microstructure and the convolution result.

The input to the workflow is a segmented micrograph obtained from simulations or experiments. For simplicity, we will consider two-phase microstructure, M. The segmented microstructure is formally defined through a local state that can take values as defined in Equation 1:

$$M_{i,j} = \begin{cases} 0, & \text{if } (x_{i,j}, y_{i,j}) \in \text{ phase one} \\ 1, & \text{if } (x_{i,j}, y_{i,j}) \in \text{ phase two} \end{cases} \tag{1}$$

where $M_{i,j}$ corresponds to the local state at the location i, j in the input array M. The local state can take one of two values, $\{0, 1\}$ to denote phase one or two, respectively. We perform all the operations on phase one (encoded in black). As mentioned previously, the procedure is general and can be applied to the second phase, or it can be extended to multi-phase material systems.

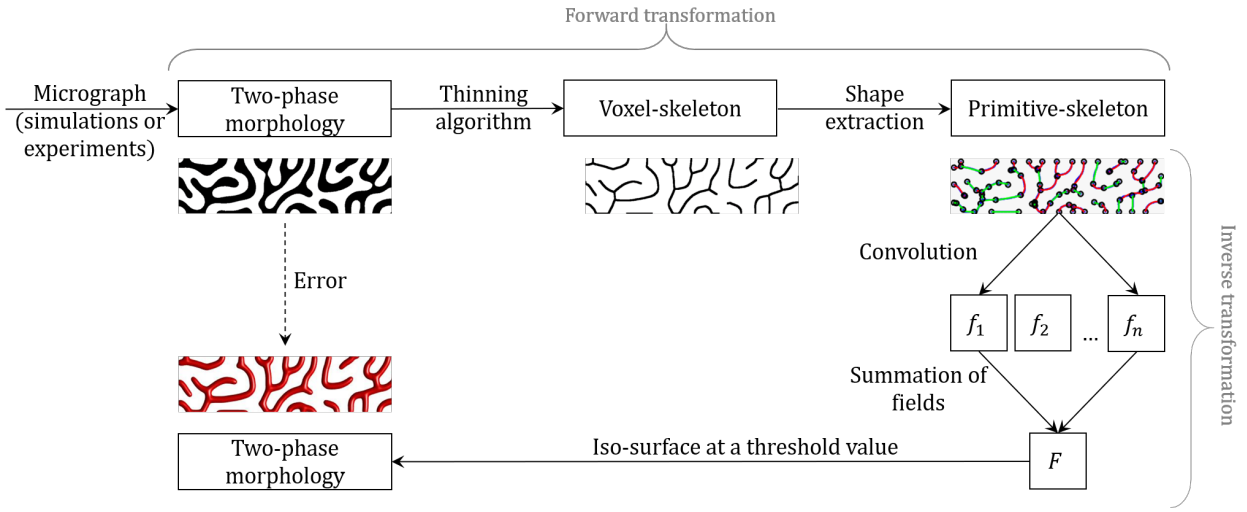

Figure 1: Two-way transformation between voxel-based microstructure representation and skeletal representation. The steps involved in the forward transformation are depicted in the top panel, while the steps involved in the inverse transformations are depicted in the bottom panel.

2.1. Skeletal representation

The skeletal representation consists of two basic elements: the skeleton, which captures topological features of the microstructure, and kernels that express the size and shape of the microstructure components (Figure 1). The skeleton can be represented in several ways. For example, the skeleton can be stored as an array of voxel positions S_v, which is the direct output of the classic skeletonization algorithm. The dimensions of this representation is smaller than the original voxel-based representation by at least one order of magnitude. The second method of representation is performed by storing the skeleton as a set of basic shapes: splines, arcs, line-segments, and points. In the simplest case, the voxels in the spatial graph will be characterized as lines. However, the set of primitives is extendable to arcs, splines, planes, and curved surfaces without the loss of generality. The decision as to the set of primitives and their complexity depends on the task at hand. For example, if the goal is to find the minimal set of basic primitives required to express the skeleton, then the increased complexity of the allowable primitives should guide the decision.

Figure 2 illustrates the procedure for primitive extraction, and the last panel shows the combination of arcs A, line-segments L, and points P. The set of A, L, and P are stored as matrices (refer to Equation 2) with o, n and m elements, respectively. The coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) represent the start and end positions of a line-segment or an arc primitive. The arcs have a radius value (r) along with the start and end positions. The position of the point primitive is stored as it's (x, y, z) coordinates.

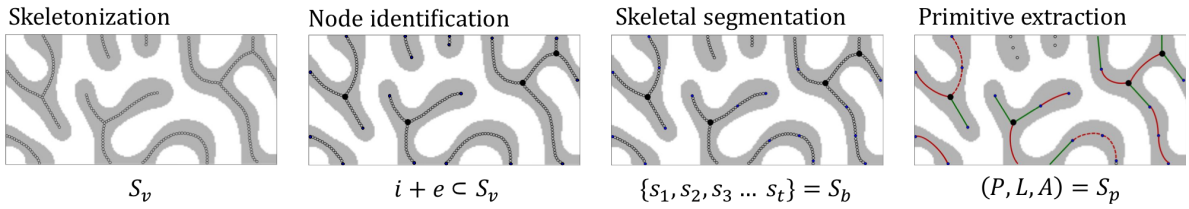

Figure 2: Steps involved in extracting primitives from skeletons with the intermediate representations. The first panel depicts the medial axis or the skeleton S_v with empty black circles, followed by the second panel with the junction nodes (larger filled circles) and end nodes (smaller filled circles) marked. The third panel shows the segmentation of branches into smaller segments of 12 pixels. The last panel depicts the skeleton built with point, line-segment and arc primitives. The dashed red lines depict an example of primitives after the merging operation (two adjacent arcs are merged and into one arc primitive).

$$
\begin{aligned}
A_{i,j} &= \begin{bmatrix} x_1^1 & y_1^1 & z_1^1 & x_2^1 & y_2^1 & z_2^1 & r^1 \\ x_1^2 & y_1^2 & z_1^2 & x_2^2 & y_2^2 & z_2^2 & r^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^o & y_1^o & z_1^o & x_2^o & y_2^o & z_2^o & r^o \end{bmatrix}_{o\times 7} \\
L_{i,j} &= \begin{bmatrix} x_1^1 & y_1^1 & z_1^1 & x_2^1 & y_2^1 & z_2^1 \\ x_1^2 & y_1^2 & z_1^2 & x_2^2 & y_2^2 & z_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^n & y_1^n & z_1^n & x_2^n & y_2^n & z_2^n \end{bmatrix}_{n\times 6} \\
P_{i,j} &= \begin{bmatrix} x^1 & y^1 & z^1 \\ x^2 & y^2 & z^2 \\ \vdots & \vdots & \vdots \\ x^m & y^m & z^m \end{bmatrix}_{m\times 3}
\end{aligned}
\tag{2}
$$

Once the set of primitives is defined, the second element of the representation consists of a kernel function used for convolution. Each primitive s_i in the skeleton S with the geometry $g_i(x)$ is convoluted with a kernel function $h(x)$ to obtain a field $f_i(x)$. The local features are obtained via integration of geometric function with a kernel (see Section 2.3). The *kernel* is a continuous, monotonic function that defines the shape of the domains around the primitive. The width of the domain is passed to the kernel function as a parameter s. For a line-segment, we added two primitive parameters r_1, and r_2 to aid in controlling the width of the domains along the segment. The contributions of convolution integral from all the primitives are added to form a convolution surface that best defines the microstructure under investigation (see the last panel in Figure 3).

2.2. Forward transformation

The forward transformation includes the following steps: skeletonization of the phase of interest to obtain the voxel skeleton, node identification by segmenting the voxel skeleton into

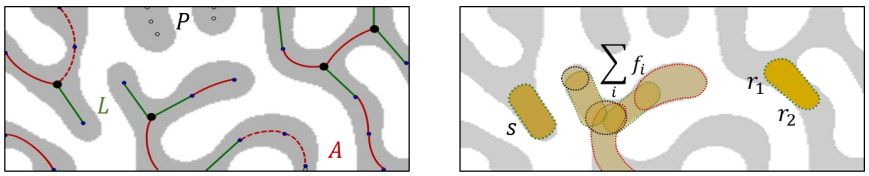

Figure 3: Elements of skeletal-based representation: (left) primitive-based skeletal representation with set of points P, line segments L and arcs A and (right) convolution surfaces using a kernel of varying complexity: with only one control variable s, or two additional radii r_1 and r_2 for end points (enabling linear change of thickness along the primitives). The convolution surface is selected after field functions are blended together using addition, and one iso-surface is selected.

skeletal segments, and skeletal segment classification into a set of basic shapes to obtain the primitive skeleton. The primitive skeleton is the key element of our skeletal representation. The panels in Figure 2 show the transition from the voxel skeleton to the primitive skeleton. We describe the steps involved in the transformation in the following sections.

Step 1: Skeletonization

In the first step, we use the skeletonization algorithm to extract the target phase's medial axis. In the classic skeletonization algorithm [21], the medial axis is represented as the subset of voxels (from input object) having more than one closest point on the object's boundary. In this work, the object corresponds to the targeted phase, and the boundary of the object corresponds to the interface between two phases.

Several approaches for performing skeletonization exist (distance transform [22], Voronoi based [23], thinning [24, 25]). In this paper, we use the thinning algorithm proposed by Pudney [25]. The object of interest is the black phase of the input microstructure, and the boundary consists of the black voxels at the phase interface. First, the distance map of the object is computed, i.e., for every voxel belonging to the object, the distance to the boundary is computed. Next, using the distance maps, the boundary is gradually eroded in the normal direction from the boundary until no further erosion is possible. As an outcome, the medial axis is identified as a set of voxels. Formally, the medial axis corresponds to the subset of points from input microstructure $S_v = p \in M$, where points p belong to the input discrete two-phase microstructure M. The first panel of Figure 2 depicts the two-phase microstructure (white and gray), and the corresponding skeleton determined for the phase is colored gray. The skeleton or the medial axis is illustrated in black points in the first panel of Figure 2.

Step 2: Node identification

In this step, we identify the junction and end nodes along with the skeleton. Junction nodes are the voxels at the intersections in the skeletal branches, and as the name suggests, end nodes are the end voxels of the skeletal branches. The nodes are identified by traversing through the skeletal voxels and inspecting their local neighborhoods. For each voxel in S_v,

we count the skeletal voxels in the Moore neighborhood. In two dimensions, the Moore neighborhood consists of eight nearest neighbors: first- and second-order neighbors. The skeletal voxels with more than three skeletal neighbors are marked as *potential junction nodes*.

Once all potential junction nodes are identified, we screen them again. When several potential junction nodes are identified in the immediate neighborhoods, only one node with the highest number of neighbors is marked as a *true junction node*. However, if more than two potential junction nodes have the highest number of potential junction nodes in their Moore neighborhood, we focus on the von Neumann neighborhood (in two dimensions, the von Neumann neighborhood consists of the four first-order neighbors). The potential junction node with the most neighbors in the von Neumann neighborhood is marked as the true junction node.

The end nodes are identified in a straightforward manner from S_v. The skeletal voxels with exactly one skeletal neighbor in the Moore neighborhood are the end nodes. The true junction nodes are illustrated as larger filled-in circles in the second panel of Figure 2; they are expressed as i, and the end nodes are smaller filled-in circles denoted by e. The branches of the voxel skeleton S_b separated by junction nodes and/or end nodes are stored separately.

Step 3: Primitive extraction

Given set of branches $S_b = \{s_1, s_2, ..., s_t\}$, we now identify the primitive (from the set of allowable primitives). For each branch s_i, using all skeletal voxels between two intersection nodes and/or end nodes, one of the primitives is selected based on the minimum least squares error of fit. The primitives can be as simple as points and line-segments, or they can be more complex shapes like splines. The shape selection of primitives depends on the goal of primitive extraction. If the goal is to extract the major building blocks of the skeleton while decreasing the dimensionality of the representation, then the set of available primitives should be diverse enough to capture the complex segments accurately. However, in this work, the goal is to represent the microstructure using two elements, primitives and kernels, while demonstrating that reconstruction is possible with low error. Hence, a more balanced approach is implemented.

In this paper, the primitives are assumed to be points, line-segments, and arcs, while limiting the kernel to one type. Moreover, we first pre-segment voxel-skeleton into short segments (of length 12 voxels) and then perform segment merging, following the work of Herold et al. [26]. This extra procedure is to ensure that the number of primitives is reflective of the local complexity; moreover, the initial pre-segmentation ensures that the reconstruction operates on fully resolved field from the convolution (see Supplemental Information for more details).

Before explaining the merging step, we first explain the initial segmentation of the skeleton S_b into segments between i and e (intersection and end nodes). We segment voxel skeleton into primitives satisfying the minimum length requirement of 12 pixels. It should be noted that it may be impossible to ensure that the skeleton segmentation satisfies the above requirement of minimal length. For example, this is the case for isolated droplets or

when longer segments are divided into multiple sub-segments, leaving shorter remainder segments. In the latter case, the merging operation typically resolves the issue. If the merging operation is impossible, e.g., for isolated droplets, two or three skeletal voxels are stored as points. Again, the decision regarding the matching primitives depends on the task at hand.

After initial division of S_v into shorter segments (satisfying the minimal length requirement), the classification into primitives is performed. For the classification, the standard regression approach is used, computing the least squares error to assign the set of available primitives to the primitive form (more details are included in the SI section). Once initial classification is completed to form S_p, merging between the nearest primitives is performed. The two adjacent primitives are combined repeatedly until the error of skeleton reconstruction is low. In Figure 2, the primitives marked with dashed lines show examples of primitives to be merged. The merging step is performed to ensure a good balance between complexity of the primitives and total number of primitives.

2.3. Inverse transformation: convolution of primitive and kernel

The complementary operation to the skeletonization of the object is convolution with a kernel function followed by iso-surface selection. The convolution operation provides a simple way to rapidly define a shape along with the skeleton that is intuitive and easy to model. Convolution surfaces have been used in computer graphics to model flexible, free-forming objects of various shapes and sizes [27]. Here, we use convolution surfaces for the purpose of inverse transformation, i.e., to convert skeletal representation back into voxel-based representation.

Formally, a convolution surface in R^3 is defined as the level set of a field function that results from the integration of a kernel function h along with a skeleton S_p, which consists of a set of geometric primitives g (i.e., points, line-segments, arcs, etc.). For each primitive, the integral of itself and a kernel h (Equation 3) is evaluated as:

$$f(\mathbf{p}) = g(\mathbf{p}) * h(\mathbf{p}) = \int_V g(\mathbf{r})h(\mathbf{p}-\mathbf{r})\, d\mathbf{r} \tag{3}$$

Given the primitives in the skeleton and the associated fields f_i, the final microstructure field $F(\overrightarrow{\mathbf{r}})$ is created by blending these features. By blending, we mean that a new, seamless field is generated by summing the fields f_i for individual primitives:

$$F(\overrightarrow{\mathbf{r}}) = \sum_i f_i(\overrightarrow{\mathbf{r}}), \quad \Gamma = \{\overrightarrow{\mathbf{r}} | T - F(\overrightarrow{\mathbf{r}}) = 0\} \tag{4}$$

Once the microstructure field is constructed, one iso-surface Γ at a threshold T can be selected from the field $F(\overrightarrow{\mathbf{r}})$. The iso-surface separates the domain of interest V into the two phases of the microstructure. For example, the gray phase in Figure 4 is an iso-surface extracted for one of the phases.

Having defined the basic terms, identifying the convolution surface that defines the interface between two phases involves three steps. In the first step, the convolution operation

is performed between the basic geometric elements $g(\mathbf{p})$ constituting the skeleton of the microstructure and the kernel function h. For each basic primitive $g(\mathbf{p})$, one field function is constructed, f_i. In the second step, the superposition principle is applied to aggregate the field functions for all the geometric elements. The outcome from this step is one field function F. In the final step, for the constructed field function, one iso-surface is chosen that segments the domain into two regions corresponding to the two phases in the microstructure.

The geometric function can be as simple as a point or a line, or as complex as a spline or a surface. The potential function *kernel* should be continuous, monotonic, and diminish to a negligible contribution beyond a certain distance from the center. The example shape is shown in Figure 4 (Cauchy kernel). The most common functions used as the kernel are Gaussian, Inverse, Polynomial, and Cauchy functions [28].

For the convolution operator to be useful in the microstructure reconstruction, the field function evaluation should be computationally efficient. Moreover, a closed-form solution of the field for a given combination of a kernel and a geometric function should be available. This requirement limits the set of kernels that can be used in combination with primitives. The kernels that yield analytical solutions decrease with the increase in complexity of the primitives. For example, for almost all kernels, closed-form solutions with point primitives have been derived. However, for complex primitives such as splines, a limited number of kernels result in a closed-form solution.

In this work, we choose a small set of primitives, i.e., points and line-segments, in combination with the kernel function, referred to as the Cauchy kernel proposed by McCormack and Sherstyuk [29]. The function is defined in Equation 5:

$$h(r) = \frac{1}{(1 + s^2\mathbf{r}^2)^2}, \quad \mathbf{r} > 0 \tag{5}$$

where $\mathbf{r}$ is the distance from an arbitrary point in the domain of interest to the primitive, and s is the parameter that controls the width of the kernel.

The first two panels in Figure 4 illustrate the convolution operation for the point and line-segment primitives with the Cauchy kernel. Intuitively, as shown in the second panel of Figure 4, the kernel passes over the entire skeleton like a filter, calculating the convolution solution in our domain of interest V for every voxel on the skeleton. As a result the field function f is constructed. From the field function the one iso-contour with the same value T in the field is selected to recover the phase in the microstructure. The field functions for each primitive are aggregated to for a microstructure field used to reconstruct the input microstructure.

The end goal of the inverse transformation is to recover the microstructure. Given the skeleton represented as a set of primitives, the phase distribution can be reconstructed by adjusting the kernel parameter s of each primitive. This parameter can be tuned to control the overall thickness of the shape around given primitive. To provide additional control, for segment primitive, two radii parameters are introduced: r_1 and r_2. These radii affect the field function and enables the linear change of the domain width along the line segment, effectively changing the shape of iso-surfaces. This is demonstrated in the last panel of Figure 4,

where the gray surfaces are the reconstructed domains for three line-segment primitives. These parameters provide an additional degree of freedom that help in reconstructing the microstructure more efficiently by allowing for an explicit representation of local changes in the domains.

An explicit integral equation for varying the radius is provided for a line-segment of length l is as follows:

$$\mathbf{p}(k) = \mathbf{b} + k(\mathbf{a}),\ 0 \leq k \leq l \tag{6}$$

where $\mathbf{b}$ is the base vector and $\mathbf{a}$ is the normalised axis. The squared distance between an arbitrary point $\mathbf{r}$ and a point on the line-segment is

$$\mathbf{r^2}(k) = d^2 + k^2 - 2kda \tag{7}$$

where $d = ||\mathbf{d}||$ is the magnitude of a vector from segment base to $\mathbf{r} : \mathbf{d} = \mathbf{r} - \mathbf{b}$ [1]. The field function for line-segment is:

$$\begin{aligned} F_{line}(\mathbf{r}) &= \int_0^1 \frac{(gk + r_1)dk}{(1 + s^2\mathbf{r}^2(k))^2} \\ &= \int_0^1 \frac{(g((2s^2k - 2s^2x)/2s^2 + x) + r_1)dk}{(1 + s^2\mathbf{r}^2(k))^2} \\ &= \int_0^1 \frac{g/2s^2d(1 + s^2(d^2 + k^2 - 2kx))dk}{(1 + s^2(d^2 + k^2 - 2kx))^2} + \int_0^1 \frac{(gx + r_1)dk}{(1 + s^2(d^2 + k^2 - 2kx))^2} \\ &= \frac{g}{2s^2(p^2 + s^2s^2)} - \frac{g}{2s^2q^2} + (gx + r_1)(\frac{x}{2p^2(p^2 + s^2x^2)} + \frac{l - x}{2p^2q^2} + \frac{\theta}{2sp^3}) \end{aligned} \tag{8}$$

where r_1 and r_2 are the radius of the start point and end point of a line, respectively. The line-segment primitive of length l is defined as

$$\begin{aligned} g &= \frac{(r_2 - r_1)}{l} \\ x &= d \cdot a = (\mathbf{r} - b) \cdot a \\ \theta &= arctan[\frac{sx}{p}] + arctan[\frac{s(l - x)}{p}] \end{aligned} \tag{9}$$

p and q are distance terms defined as

$$\begin{aligned} p^2 &= 1 + s^2(d^2 - x^2) \\ q^2 &= 1 + s^2(d^2 + l^2 - 2lx) \end{aligned} \tag{10}$$

[1] Equation 5, 6 and 7 are directly adapted from the original paper and an interested reader can refer [27] for a detailed description

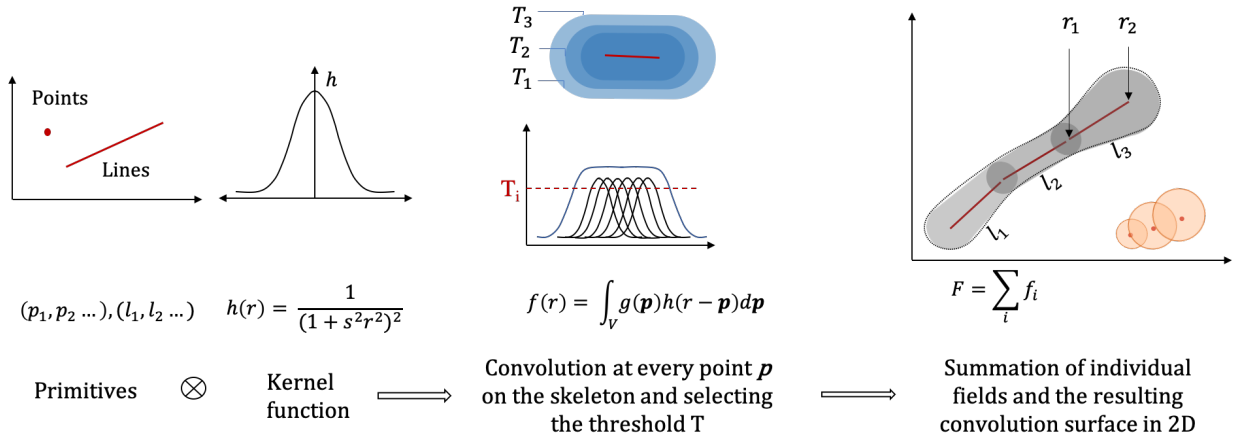

Figure 4: Inverse transformation of the primitive-based skeleton using convolution surfaces. The convolution operation for line-segment and point primitives with a kernel function is shown in the left two panels, followed by iso-surface selection in the center panel. The summation of fields of individual primitives resulting in a blended surface is shown in the right panel.

2.4. Microstructure reconstruction

The convolution operation is used to reconstruct the input microstructure. The reconstruction aims to determine the voxel-based representation of the two-phase microstructure from the skeletal-based representation. Inverse transformation or reconstruction is formalized as an optimization problem such that given the skeleton, the error of reconstruction is minimized for the kernel parameters: s, r_1 and r_2, for each primitive in S_b. In this paper, a defined domain is the size of the micrograph under consideration. We consider microstructure samples of size $(n_x \times n_y)$ pixels, which are described in detail in Section 3. The error of the reconstruction is computed as the pixel-wise difference between the microstructures. One phase is selected as the target phase for the reconstruction, with the associated skeleton used to compute the microstructure field. The error of reconstruction is computed as

$$E_r = f_n + (w \cdot f_p) \tag{11}$$

where, f_n is the fraction of voxels with negative distance between reconstruction and reference microstructure, and f_p is the fraction of voxels with positive distance between reconstruction and reference microstructure. The parameter w is the weight coefficient found effective for faster convergence of the optimization. The value of w for this paper is set to 5.

The reconstruction can be performed for various set of primitives. In this paper, we perform the reconstruction for the simplest set of primitives that includes line-segments and points. Each line segment is represented as a pair of two points (x_1, y_1, z_1) and (x_2, y_2, z_2) with two radii associated with each point (r_1, r_2). The corresponding design matrix is defined

below:

$$L_{i,j} = \begin{bmatrix} x_1^1 & y_1^1 & x_2^1 & y_2^1 & r_1^1 & r_2^1 \\ x_1^2 & y_1^2 & x_2^2 & y_2^2 & r_1^2 & r_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^n & y_1^n & x_2^n & y_2^n & r_1^n & r_2^n \end{bmatrix}_{n\times 6}$$

and consists of n line-segment primitives that define the skeleton. In some instances, the reconstruction requires considering points. Similarly, the design matrix for the points is defined as:

$$P_{i,j} = \begin{bmatrix} x^1 & y^1 \\ x^2 & y^2 \\ \vdots & \vdots \\ x^m & y^m \end{bmatrix}_{m\times 3}$$

where, (x^1, y^1) are the coordinates of the point, with m points in the skeleton.

The term reconstruction of microstructures is more commonly used in materials in the context of three-dimensional reconstructions from two-dimensional micrographs [30, 31]. It is also used to describe the process of generating a set of statistically equivalent microstructures with some desired characteristics [6]. In this work, we reconstruct the input microstructure to discuss the robustness of the representation. The goal is to show that two-way transformation is possible. This is important as microstructure optimization can be performed using skeletal representation and then convoluted with the kernel to generate commonly used voxel-based representation.

3. Results and discussion

In this section, we illustrate the capabilities and efficiency of the proposed approach by performing a two-way transformation on a set of microstructures. We demonstrate the compactness of the skeletal representation in terms of the number of variables required to store the input microstructure. We compute the reconstruction error between the microstructures of forward and inverse transformation (voxel-based representation of microstructure and convolution surface representation of microstructure). We quantify the reduction in the design space and the reconstruction error for a wide range of microstructures.

3.1. Data Generation

We test our approach on two-phase microstructures of organic thin films. The microstructures are representative of spinodal decomposition between two polymers [32]. Microstructures are generated using the numerical model of the Cahn-Hilliard (CH) equation [33, 34]:

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (M_{CH} \nabla (\frac{\partial f_{FH}}{\partial \phi} - Cn^2 \nabla^2 \phi)) \tag{12}$$

where $\phi(x, t)$ is the volume fraction of polymer (linked with the blend ratio between two polymers) which evolves in time and space, M_{CH} is the mobility, Cn is the Cahn number

which characterizes the length scale of the interface between two phases, and f_{FH} is the Flory-Huggins free energy defined in Equation 13:

$$f_{FH}(\phi) = \phi \ln \phi + (1-\phi) \ln (1-\phi) + \chi\phi(1-\phi) \tag{13}$$

The first term of free energy quantifies the entropic and the second term quantifies the enthalpic energy of dispersion between two polymers with the interaction parameter χ. The Cahn-Hilliard equation captures a rich and complex collection of interacting phenomena that direct this morphology evolution - the mixture initially separates into phases very rapidly (phase-separation), followed by slow and sporadic coarsening events.

We use the above characteristics of CH equation to generate a spectrum of microstructures by varying two parameters: blend ratio (ϕ) and interaction parameter (χ). Some representative microstructures are included in Figure 5 along with the values of two parameters. By varying the blend ratio, we change the total volume fraction of the polymers. Consequently, the type of microstructure changes from interpenetrated ($\phi = 0.50$) to more droplet like morphology ($\phi = 0.56$). By varying χ parameter, we change the strength of the interactions between two polymers and consequently the initial domain size of microstructure that emerges from rapid initial phase separation. When interactions are strong (high χ), the initial domain size of microstructure that emerges from rapid initial phase separation is small. When interactions are weaker (low χ), the initial domain size of microstructure is larger at the initial phase separation state. Regardless of ϕ and χ values, due to coarsening, the domain size increases as time evolves.

We select four blend ratios $\phi = \{0.50, 0.52, 0.54, 0.56\}$, $\chi = \{2.4, 2.6, 2.8, 3.0\}$ with fixed Cahn number of $Cn = 0.032$. For each combination, we generate microstructures as a time series that we call microstructure trajectory. For each trajectory, five microstructures are taken for analysis as representative of early to the late stages of microstructure evolution. Each microstructure is stored as a matrix of dimensions 400×100. The geometry of the thin film is rectangular with an aspect ratio of four-to-one. The periodic boundary condition is applied on sides to reflect the thin film geometry, while natural boundary conditions are applied at top and bottom boundaries. Raw results from simulations are segmented into two-phase microstructures.

A set of 10 replicas of every microstructure are generated. The replicas differ in terms of random seed value used to initiate the random field of the initial conditions [2]. In summary, 16 sampling points are explored with 5 time steps per trajectory, and 10 replicas per each sampling point. Altogether, 800 microstructures are analyzed for the proposed microstructure representation approach.

3.2. Technical details

The forward transformation requires two steps: identifying the voxel skeleton and the primitive skeleton. The pixel-based skeletons of the microstructures were extracted using the Skeletonize3D plugin of FIJI, a JAVA-based image processing tool, while primitive extraction

[2]The initial field is a random uniform perturbation around ϕ, with zero mean and variance of 0.001.

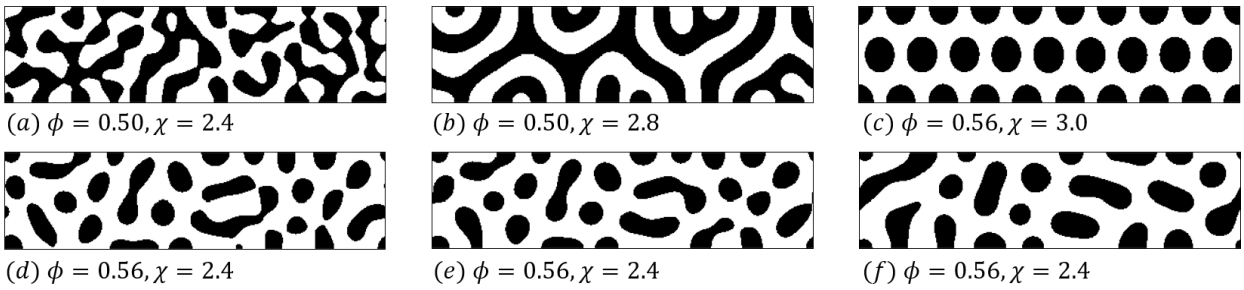

Figure 5: Example microstructures generated for testing: (a) microstructure with locally varying domains, (b) interpenetrated structure, (c) droplet structure, panels (d-f) three microstructures from the same trajectory at the first, second and fourth time index.

and reconstruction was performed using MATLAB. The identification of the voxel skeleton of each microstructure requires less than 10 seconds on an Alienware PC with 2.20GHz Intel(R) Core (TM) i7-8750H CPU and 32 GB of memory. Collectively, the forward transformation of skeletal microstructure representation requires 30 seconds on an average.

The inverse transformation is more computationally demanding. The parameters of kernels in convolution surfaces (refer Section 2.4) were optimized using the Nelder-Mead simplex algorithm in MATLAB. The evaluation of the inverse transformation for a microstructure with 60 line segments and seven point primitives takes less than three minutes to execute on the aforementioned CPU. The evaluation time includes optimization of the convolution parameters and calculating the final field F of the reconstructed microstructure.

3.3. Skeletonization and primitive extraction

In this section, we summarize the robustness of skeletal-microstructure representation in terms of reduced dimensionality of the representation space. For the microstructures under consideration, we compare the number of variables or primitives required to capture the skeleton of the microstructure by first extracting the medial axis as the voxel skeleton and then extracting the basic shapes or primitives from the medial axis as the primitive skeleton. A summary of the number of primitives required to represent the microstructure skeletons is presented in Table 1. Data in the individual rows of this table correspond to the microstructure from one trajectory, with the index marking the order in time.

The third column of Table 1 lists the total number of the voxels on the voxel-skeleton (skeleton). The number of voxels on the skeleton decreases with time (and index) from $1,006$ to 592. This trend is mirroring the dynamics of the coarsening. As domains coarsen, their complexity decreases. For comparison, the input voxel-based representation requires 400×100 variables (given the second column of the table). The observed reduction in terms of variables needed to store the microstructure is two orders of magnitude. This is under the assumption that microstructure can be reconstructed, with point being the primitive of choice and one parameters (s) per primitive.

Moving to the next level, where skeleton is reconstructed using set of primitives, Table 1 includes data for two configurations. Configuration 1 allows the skeleton to be reconstructed using: arcs, line-segments, and points; while configuration 2 allows line-segments and points.

Regardless of the configurations our results indicate that the skeletal representation requires 40 primitives on average to represent the skeleton. For configuration 1, initially 48 primitives are required (compared with 15 at the later stages of the process). For configuration 2, we report similar number of primitives, here 50 primitives are determined to represent the skeleton of the same microstructure. The number of points remain the same in both configurations.

Further, in Figure 6, we compare the number of primitives required to represent skeletons of the three example microstructures from different trajectories. For these microstructures, we additionally compare the error of fit per primitive (E_s in voxels) and the average primitive length (l in voxels). The average primitive length is the average length of all primitives in a given skeleton of one phase in the microstructure. Comparing configuration 1 and 2, the E_s is lower when both line-segments and arcs are used compared to when only line-segments are used. However, for line-segments, it is still considerably low, with 0.75 voxels per primitive and the average primitive length around 22 voxels. The E_s is reported to be consistently low (less than one pixel per primitive) for all the microstructures investigated.

Table 1: The number of primitives required to represent the microstructures from spinodal decomposition using a combination of arcs, line-segments, and points.

Idx	Input Space	Skeleton	Configuration 1				Configuration 2		
			Arcs	Lines	Points	Total	Lines	Points	Total
1	40,000	1,006	18	19	11	48	53	11	64
2	40,000	828	13	21	3	37	42	3	45
3	40,000	712	11	16	2	29	33	2	35
4	40,000	608	7	9	0	16	21	0	21
5	40,000	592	5	9	1	15	18	1	19

Figure 7 depicts the trend of the number of primitives required to represent the microstructure skeleton. The figure consists of nine panels; each panel depicts the trend for a selected combination of input parameters (ϕ and χ). Each panel is labeled with ϕ and χ values at the top. For example, the first panel depicts the results for $\phi = 0.50$ and $\chi = 2.4$. Each panel shows the number of primitives required to represent the skeleton as a function of the time index in the microstructure trajectory (one to five). We include results for two configurations as discussed in Table 1. The number of primitives when a combination of arcs, segments, and points are used is plotted in a dashed black line. On the same plot, the number of primitives is represented for the simpler configuration when only segments and points are used is plotted as a solid blue line. The representative microstructures for the cases in question are shown below every panel.

The total number of primitives remains comparable irrespective of the type of microstructure and configuration. The number decreases with the evolution of the microstructures. This is because the microstructure becomes less complex at later stages with reduced number of branches. The number ranges from 50-60 for the initial stages and 10-20 for the later stages. The number of primitives for both the configurations remains similar with a difference of ≈ 10 primitives. In some case, the number of primitives is almost identical between

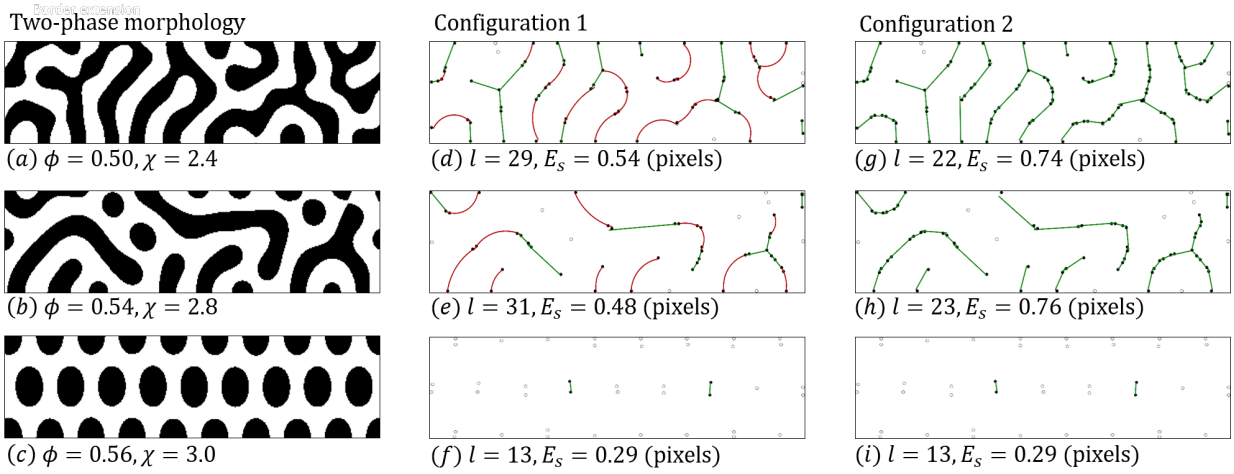

Figure 6: Primitive-skeleton for the two configurations under consideration. The left column (a, b, and c) includes the original microstructure, the middle column (d, e, and f) depicts configuration 1 (arcs, line-segments and points allowed to represent the skeleton), and the third column (g, h, and i) depicts configuration 2 (only line-segments and points allowed to represent the skeleton). For each of the primitive-based skeleton, the average length of the primitive (l in pixels) and the error of fit (E_s in pixels) are given.

the two configurations. For example, for $\phi = 0.56$ microstructures (third row in Figure 7) two curves almost overlap. This can be explained by inspecting the microstructures. For χ values 2.4 and 3.0, most of the microstructure domains are droplets with points selected as primitives. At $\chi = 2.8$, the structure comprises of straight branches, and so the counted primitives consist mostly of segments.

All together, for the analyzed thin-film microstructures, we report a reduction of two orders in the parameter space (in relation to the input 400×100 pixels). For the above analysis, it was observed that increasing the complexity in primitive type (configuration 1 vs. configuration 2) does not lead to a significantly lower number of primitives required to reconstruct the skeleton. However, the number of primitives depends on the choice of primitives. We anticipate that adding more complex primitives like splines will reduce the dimensions significantly. For configuration 1, the error of extracting primitives, E_s, is consistently low, with values remaining below one pixel per primitive for all microstructures. Therefore, for the reconstruction step, we use configuration 2 (line-segments and points). We use convolution surfaces to convey that the shape and size of the domains can be retained, and the microstructures can be reconstructed from the skeletal branches to the original structure, i.e., to demonstrate the two-way transformation.

3.4. Reconstruction

The second step of the two-way transformation involves reconstruction of the input microstructure from the skeletal-based representation. The aim is to demonstrate that inverse transformation is possible and that the reconstruction error is low. We seek to reconstruct two-phase microstructure based on a skeleton represented as a set of primitives (line-segments and points). The line-segment and point primitives extracted from the pixel-based skeletons

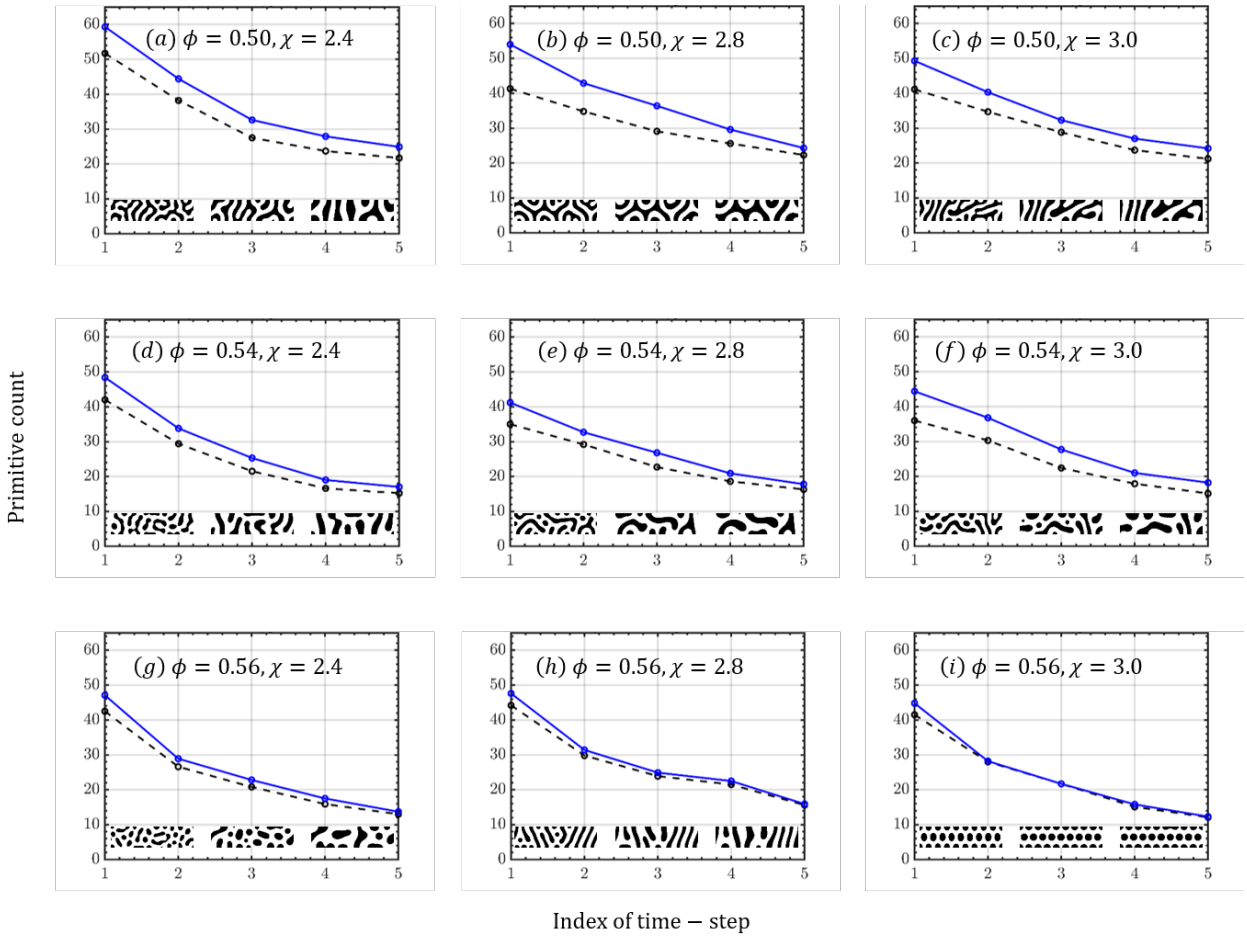

Figure 7: The total number of primitives used to represent skeletons of different microstructures. The number of primitives for configuration 1 (arcs, line-segments, and points) is plotted in black dashed lines, and configuration 2 (line-segments and points) is plotted in solid blue lines. Each row represents a particular ϕ value and column χ. For example panels (a), (b) and (c) are representative of the behavior of number of primitives along time for $\phi = 0.50$ and panels (a), (d) and (g) for $\chi = 2.4$. The example microstructures for each of the ϕ and χ combinations are included below each plot.

are stored as a matrix of dimensions $n \times 8$ and $m \times 3$ as explained in Section 2.4. The shapes around the primitives are reconstructed with Cauchy kernel and it's kernel parameter s. Additionally for segments two radii controlling the width of the domains (r_1 and r_2) are included.

Six representative microstructures are shown in Figure 8 (a-f). For each microstructure, we include two microstructures: the input microstructures (black voxels), and the reconstructed microstructures (gray voxels). For each reconstructed microstructure, we additionally add the outline of the input structures (black curves) for easier visual comparison. The six microstructures are representative of a wide range of morphology types, ranging from interpenetrated structures (top left) to droplet structures (bottom right). The first column of microstructures (a, c, e) shows continuous structures with uniform domains and smooth branching, whereas the microstructures in second column (b, d, f), especially the first two microstructures, have locally varying shapes and domain sizes. The reconstructed microstructures in the first column have smaller reconstruction error than the first two microstructures in the second column. However, overall, the reconstructions visually match the original microstructures.

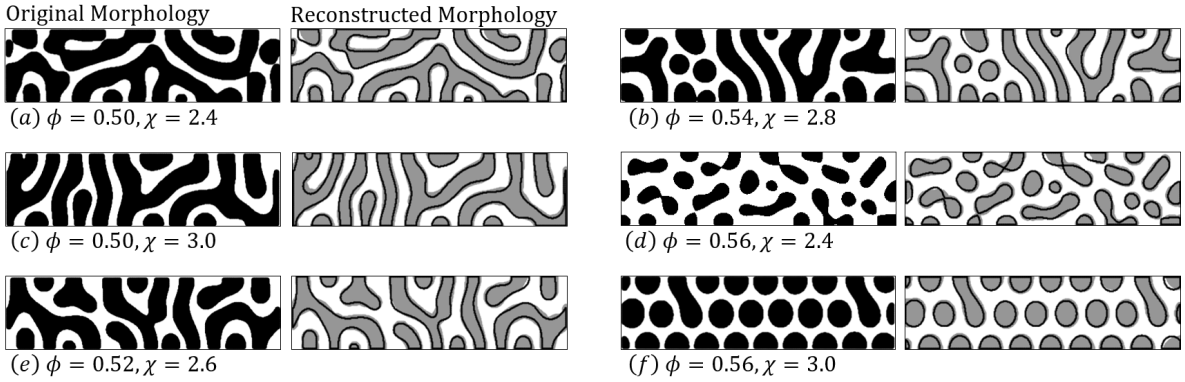

Figure 8: Convolution surfaces (gray) reconstructed from initial microstructures (black). The outline of microstructures are drawn in black for comparison purposes. The left column ((a), (b) and (c)) includes the performance of the proposed algorithm for smooth and long microstructures, and the right column ((d), (e) and (f)) shows the performance for locally varying domains, and droplets. All the structures included are from the initial stages of microstructure evolution.

To quantify the reconstruction for all considered microstructures, we calculate the error of reconstruction. The errors in reconstruction for the microstructures at three ϕ and χ are plotted in Figure 9. The error is defined as the pixel-wise difference between the reconstructed and the original microstructures, reported as the fraction of all pixels in the morphology. This error is calculated as an average of the 10 replicas that are analyzed. As a reminder, the replicas correspond to the same process (the same ϕ and χ) but are different realizations of the same process.

Results in Figure 9 confirm low reconstruction error between 6-11%. Reconstruction error decreases with microstructures evolving over time (index of time steps) as the structure becomes less complex. The error rate is around 9% for the initial time steps, and it decreases

to 6% for the final stages. The values range from 8% to 6% for interpenetrated structures and from 6.5% to 5% for the droplet-like structure. This is because the skeletons of the droplets are stored as points, and the convolution surfaces can be built more efficiently. The error for interpenetrated structures ($\phi = 0.50$, $\chi = 2.4$) and droplet-like structures ($\phi = 0.56$, $\chi = 3.0$) are very similar, demonstrating the efficacy of skeleton-based microstructure representation.

This error can be further decreased by using shorter segments in the skeletons. In the extreme case, the skeleton can be represented as a set of points with the reconstruction error converging to one percent. However, if the aim is to reduce the dimensionality of the microstructure, the representation proposed in this paper is more preferred. If only points are used to represent skeleton, the number of primitives and the corresponding design space increases. Hence, the set of allowed primitives should be chosen according to the application. A decision has to be made as to the dimensionality reduction of the number of primitives vs. the efficient representation of the shape and domain sizes of the microstructure.

4. Conclusions

In this paper, we introduced the skeletal microstructure representation to aid the characterization, inverse design, and acceleration of quantitative structure-property relationships in materials systems sensitive to microstructure. In our representation, the microstructures are mapped to a lower-dimensional representation using skeletonization and primitive extraction. We reported the reduction in the representation size by three orders of magnitude. We demonstrated the robustness of the representation with low reconstruction error for both skeletonization and phase distribution. Although the methodology was demonstrated for two-dimensional thin-films, it can be effectively generalized for other microstructure types, including multi-phase systems and three-dimensional anisotropic structures.

Data availability

The datasets generated and analyzed during the current study are available from the corresponding authors on reasonable request.

Acknowledgements

This work was supported by National Science Foundation (1906344). Authors also acknowledge the support provided by the Center for Computational Research at the University at Buffalo.

References

[1] K. Rajan, Phase transformations in a wrought Co-Cr-Mo-C alloy, Metallurgical Transactions A 13 (7) (1982) 1161–1166.

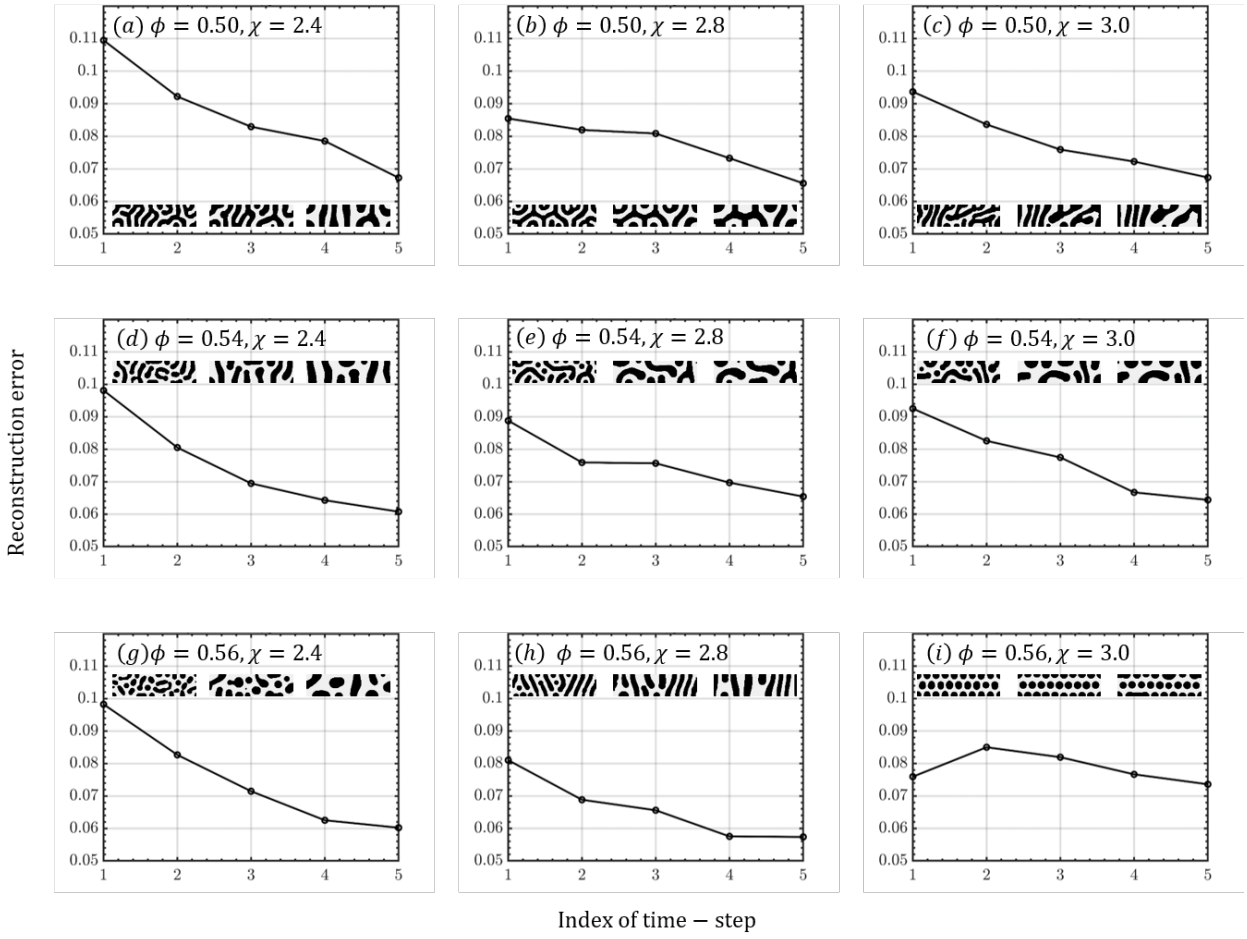

Figure 9: Average error of reconstructed microstructures using configuration 2 (line-segments and points) and convolution surfaces for a wide range of microstructures representing spinodal decomposition. Each row represents a particular ϕ value and column χ. For example panels (a), (b) and (c) are representative of the behavior of number of primitives along time for $\phi = 0.50$ and panels (a), (d) and (g) for $\chi = 2.4$. The reconstructed microstructures for each of the ϕ and χ combinations are included in each plot.

[2] O. Wodo, J. Roehling, A. Moule, B. Ganapathsubramanian, Quantifying organic solar cell morphology: A computational study of three-dimensional maps, Energy and Environmental Science 6 (2013) 3060–3070.

[3] L.-Q. Chen, Phase-field models for microstructure evolution, Annual review of materials research 32 (1) (2002) 113–140.

[4] C.-K. Lee, O. Wodo, B. Ganapathysubramanian, C.-W. Pao, Electrode materials, thermal annealing sequences, and lateral/vertical phase separation of polymer solar cells from multiscale molecular simulations, ACS applied materials & interfaces 6 (23) (2014) 20612–20624.

[5] S. Samudrala, O. Wodo, S. Suram, S. Broderick, K. Rajan, B. Ganapathysubramanian, A graph-theoretic approach for characterization of precipitates from atom probe tomography data, Computational materials science 77 (2013) 335–342.

[6] R. Bostanabad, Y. Zhang, X. Li, T. Kearney, L. C. Brinson, D. W. Apley, W. K. Liu, W. Chen, Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques, Progress in Materials Science 95 (2018) 1 – 41.

[7] H. Xu, Y. Li, C. Brinson, W. Chen, A Descriptor-Based Design Methodology for Developing Heterogeneous Microstructural Materials System, Journal of Mechanical Design 136 (5), 051007.

[8] S. Torquato, Statistical description of microstructures, Annual Review of Materials Research 32 (1) (2002) 77–111.

[9] Y. Jiao, F. H. Stillinger, S. Torquato, A superior descriptor of random textures and its predictive capacity, Proceedings of the National Academy of Sciences 106 (42) (2009) 17634–17639.

[10] S. Yu, C. Wang, Y. Zhang, B. Dong, Z. Jiang, X. Chen, W. Chen, C. Sun, Design of non-deterministic quasi-random nanophotonic structures using fourier space representations., Sci Rep 7 (3752).

[11] M. Teubner, Level surfaces of gaussian random fields and microemulsions, Europhysics Letters (EPL) 14 (5) (1991) 403–408.

[12] Z. Yang, X. Li, L. Catherine Brinson, A. N. Choudhary, W. Chen, A. Agrawal, Microstructural materials design via deep adversarial learning methodology, Journal of Mechanical Design 140 (11).

[13] B. L. DeCost, E. A. Holm, A computer vision approach for automated analysis and classification of microstructural image data, Computational materials science 110 (2015) 126–133.

[14] S. Torquato, J. D. Beasley, Y. C. Chiew, Twopoint cluster function for continuum percolation, The Journal of Chemical Physics 88 (10) (1988) 6540–6547.

[15] D. M. Turner, S. R. Niezgoda, S. R. Kalidindi, Efficient computation of the angularly resolved chord length distributions and lineal path functions in large microstructure datasets, Modelling and Simulation in Materials Science and Engineering 24 (7) (2016) 075002.

[16] D. M. Dimiduk, E. A. Holm, S. R. Niezgoda, Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering, Integrating Materials and Manufacturing Innovation 7 (3) (2018) 157–172.

[17] S. Yang, A. Tewari, A. M. Gokhale, Modeling of non-uniform spatial arrangement of fibers in a ceramic matrix composite, Acta Materialia 45 (7) (1997) 3059 – 3069.

[18] S. Yu, Y. Zhang, C. Wang, W.-k. Lee, B. Dong, T. W. Odom, C. Sun, W. Chen, Characterization and design of functional quasi-random nanostructured materials using spectral density function, Journal of Mechanical Design 139 (7).

[19] D. T. Fullwood, S. R. Niezgoda, S. R. Kalidindi, Microstructure reconstructions from 2-point statistics using phase-recovery algorithms, Acta Materialia 56 (5) (2008) 942–948.

[20] K. Mangipudi, E. Epler, C. Volkert, Topology-dependent scaling laws for the stiffness and strength of nanoporous gold, Acta Materialia 119 (2016) 115 – 122.

[21] T. Y. Zhang, C. Y. Suen, A fast parallel algorithm for thinning digital patterns, Commun. ACM 27 (3) (1984) 236239. doi:10.1145/357994.358023.
URL https://doi.org/10.1145/357994.358023

[22] G. Borgefors, Distance transformations in digital images, Computer vision, graphics, and image processing 34 (3) (1986) 344–371.

[23] J. W. Brandt, V. R. Algazi, Continuous skeleton computation by voronoi diagram, CVGIP: Image understanding 55 (3) (1992) 329–338.

[24] J. Mukherjee, B. Chatterji, P. P. Das, Thinning of 3-d images using the safe point thinning algorithm (spta), Pattern Recognition Letters 10 (3) (1989) 167–173.

[25] C. Pudney, Distance-ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images, Computer Vision and Image Understanding 72 (3) (1998) 404–413.

[26] J. Herold, T. F. Stahovich, Speedseg: A technique for segmenting pen strokes using pen speed, Computers & Graphics 35 (2) (2011) 250 – 264, virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage.

[27] J. McCormack, A. Sherstyuk, Creating and rendering convolution surfaces, Computer Graphics Forum 17 (2) (1998) 113–120.

[28] A. Sherstyuk, Kernel functions in convolution surfaces: A comparative analysis, The Visual Computer 15 (4) (1999) 171–182. doi:10.1007/s003710050170.
URL https://doi.org/10.1145/357994.358023

[29] A. Sherstyuk, Convolution surfaces in computer graphics.

[30] H. Xu, D. A. Dikin, C. Burkhart, W. Chen, Descriptor-based methodology for statistical characterization and 3d reconstruction of microstructural materials, Computational Materials Science 85 (2014) 206 – 216.

[31] Y. Jiao, F. H. Stillinger, S. Torquato, Modeling heterogeneous materials via two-point correlation functions. ii. algorithmic details and applications, Phys. Rev. E 77 (2008) 031135.

[32] O. Wodo, B. Ganapathysubramanian, Modeling morphology evolution during solvent-based fabrication of organic solar cells, Computational Materials Science 55 (2012) 113–126.

[33] O. Wodo, B. Ganapathysubramanian, Computationally efficient solution to the Cahn-Hilliard equation: adaptive implicit time schemes, mesh sensitivity analysis and the 3D isoperimetric problem, Journal of Computational Physics 230 (2011) 6037–6060.

[34] J. W. Cahn, J. E. Hilliard, Free energy of a nonuniform system. i. interfacial free energy, The Journal of chemical physics 28 (2) (1958) 258–267.

Supplementary Information

4.1. Error of Fit

The E_s reported in Figure 6 is the average error of fit over all p primitives extracted from the skeleton S_p:

$$E_s = \frac{1}{p}\sum_{i=1}^{p} e_i \tag{14}$$

where e_i is the absolute difference between the skeleton and the primitive (line-segment or arc) - defined in Equation 15 and 16. For example, in the first microstructure (panel (a) of Figure 6), the primitive skeleton of configuration 1 (arcs and line-segments) has an error of $E_s = 0.54$ pixels. This is the average error over 37 primitives (18 arcs and 19 line-segments).

Formally, for each primitive, we calculate the error as follows:

$$e = \frac{1}{n}\sum_{i=1}^{n} \mid A_l x_i + B_l - y_i \mid \tag{15}$$

$$e = \frac{1}{n}\sum_{i=1}^{n} \mid \sqrt{(x_i + a_c)^2 + (y_i + b_c)^2} - r \mid \tag{16}$$

where n is the total number of pixels with coordinates (x_i, y_i) in the skeletal branch under consideration for fitting (with coefficient of fit A_l and B_l and a_c, b_c, c_c for line segment and arc, respectively). [3]

In summary, we compute the error of fit for two candidate primitives, the error is in unit of voxels. In the paper, we report the error of fit per primitive. This is an average error over all primitives in a given microstructure.

4.2. Minimum line-segment length for fully resolved field

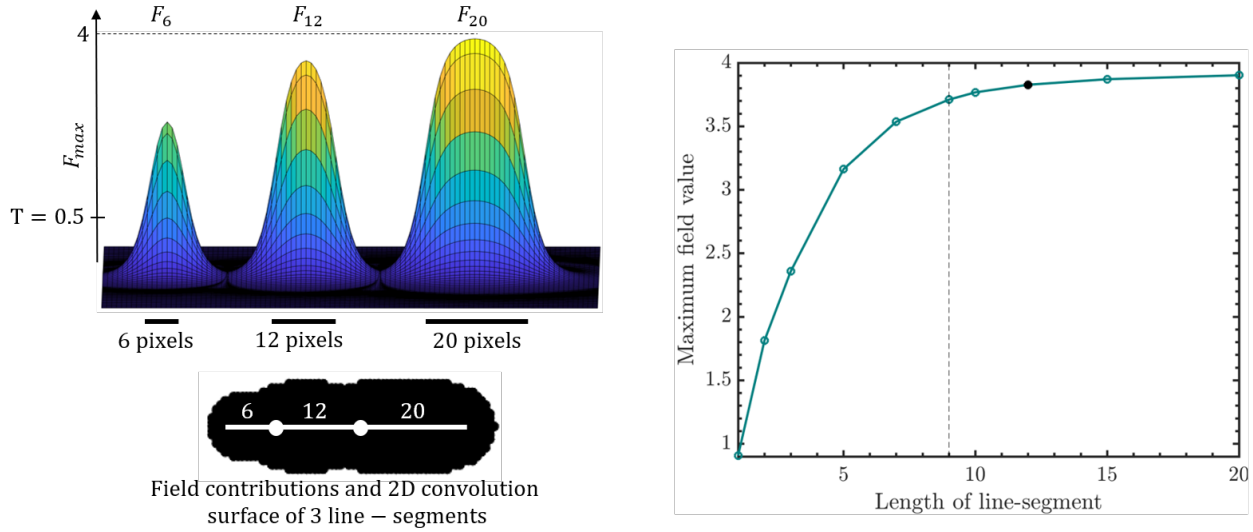

Figure 10: Field contributions by line-segments of length 6 and 12 pixels and the middle slice of their resulting convolution surface (left). Fully resolved field of segments longer than 12 pixels vs. under-resolved field for segments smaller than 12 pixels (right).

The minimum length of 12 pixels is chosen based on the criteria related to field function F from the convolution operation. For short line-segments, the field is not fully resolved (the maximum possible value of the function is not reached). Figure 10 depicts the field contributions from three line segments of length 6, 12 and 20 pixels. The maximum field for the segment of length 6 (pixels) is ≈ 3.2 and for segment of length 12 and 20 it is ≈ 4.

The problem becomes evident when two fields are blended together to form a convolution surface. The convolution surface is obtained by keeping the values of iso-contour (threshold), s (kernel parameter) and radii of the segments fixed. This bottom left of Figure 10 illustrates this. The thickness of the convolution surface formed by the segment of length 6 pixels is smaller than that of 12 and 20 pixels, despite the thickness parameters being constant for both cases. This results in an uneven surface. As a consequence, at the reconstruction step

[3] where primitives are defined as: $y = A_l x + B_l$ and $x^2 + y^2 + 2a_c x + 2b_c y + c_c = 0$.

we observe the low convergence and high error of reconstruction. To some extent this issue can be mitigated by the choice of s parameter for a shorter segment. However, to avoid this additional layer of complexity, we put the limitation on the segment length.

To find the minimum length of the segment (the minimum length criteria), the maximum value of the field function is the quantity of interest. The right panel of Figure 10 depicts the increasing maximum value of the field for increasing segment length with saturation at ≈ 4 for *fully resolved field*. The results show that for segments longer than 12 pixels (marked with black filled point on the curve), the maximum value remains the same (here, ≈ 4). Hence, the minimum allowed segment length is set to 12 pixels. And since in this work, we work with discrete data, we choose to provide the rule of thumb in terms of number of pixels per segment.