Graph-based Strategy for Establishing Morphology Similarity

Namit Juneja, Jaroslaw Zola, Varun Chandola
{namitjun,jzola,chandola}@buffalo.edu
Computer Science and Engineering
University at Buffalo
USA

Olga Wodo
olgawodo@buffalo.edu
Materials Design and Innovation
University at Buffalo
USA

ABSTRACT

Analysis of morphological data is central to a broad class of scientific problems in materials science, astronomy, bio-medicine, and many others. Understanding relationships between morphologies is a core analytical task in such settings. In this paper, we propose a graph-based framework for measuring similarity between morphologies. Our framework delivers a novel representation of a morphology as an augmented graph that encodes application-specific knowledge through the use of configurable signature functions. It provides also an algorithm to compute the similarity between a pair of morphology graphs. We present experimental results in which the framework is applied to morphology data from high-fidelity numerical simulations that emerge in materials science. The results demonstrate that our proposed measure is superior in capturing the semantic similarity between morphologies, compared to the state-of-the-art methods such as FFT-based measures.

CCS CONCEPTS

• **Computing methodologies** → *Data assimilation*; • **Information systems** → **Similarity measures**.

KEYWORDS

morphology similarity, graph similarity, morphology comparison, semantic similarity

ACM Reference Format:
Namit Juneja, Jaroslaw Zola, Varun Chandola and Olga Wodo. 2021. Graph-based Strategy for Establishing Morphology Similarity. In *33rd International Conference on Scientific and Statistical Database Management (SSDBM 2021), July 6–7, 2021, Tampa, FL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3468791.3468819

1 INTRODUCTION

The term *morphology*, in science and engineering, refers to shapes and structures of objects. The objects of interests may be nebulae or galaxies in astronomy [14], vesicles, and tissues in biology and medicine [10], or phases in materials science [19], to name just a few. In this work, our focus is on morphologies emerging in materials science (although the techniques we propose are generic and can be applied in other contexts). Here morphology refers to the distribution of components (or phases) within a material (see, e.g., Figure 1). Morphology reflects a complex internal organization of a material, which is a result of the manufacturing process applied to obtain it. It also directly controls material's physical properties (e.g., stiffness, conductivity, etc.). Consequently, by performing data analytics on materials' morphologies we can gain insights into how materials' manufacturing processes relate to materials' properties. This question is critical as it is the key to smart materials design in which manufacturing process is tailored on demand to obtain a material with a specific set of desired characteristics.

In order to perform meaningful data analytics we first need a notion of similarity between morphologies. However, the currently available approaches, which we review in Section 5, either take highly simplified view of the morphology, e.g., focusing on pixels in the morphology images, or apply transformations that average out structural properties of morphologies or relay on sometimes difficult to satisfy assumptions. To address these shortcomings, we propose a new computationally efficient and configurable similarity measure that is based on graph abstraction. Our main idea is to simplify complex morphologies by abstracting them as graphs, that are weighted with domain specific information, and then express similarity as a distance between morphology graphs. Because both morphology graph structure and its weights have clear interpretation, our similarity can be easily tailored to the specific applications. To address computational complexity of graphs comparison, we take advantage of inherent properties of the morphology graphs to devise a linear time solution. Our experimental results demonstrate superior performance of our approach in capturing morphologies similarity on both synthetic as well as real world data, including in real-world applications (like morphologies clustering).

2 PRELIMINARIES

Consider a set $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ of N morphologies, where the morphology $\mathbf{X}_i$ is represented by a $(n \times m)$ bitmap, i.e., $\mathbf{X}_i \in \{0,1\}^{n \times m}$, and $\mathbf{X}_i(x, y)$ is a bitmap pixel at position (x, y). All morphologies in the set $\mathcal{X}$ represent some state of the same physical phenomena or process we want to study. For instance, in Figures 1 and 2 we show a sample of images and their corresponding morphologies that capture the evolution of the separation of two materials within a fixed volume. The study of such phase separation process turns out to be one of the fundamental problems in materials science [6]. The process can be modeled via a set of complex differential equations, in which case the resulting morphologies are represented via a composition field, $\phi : \Omega \rightarrow \mathbb{R}$, where $\Omega \in \mathbb{R}^D$ is the D-dimensional domain over which ϕ is defined. On the other hand, the same process can be physically executed and observed using microscopy. In such case, the resulting morphologies are directly represented as multi-channel images. In our example, the

SSDBM 2021, July 6–7, 2021, Tampa, FL, USA

ACM ISBN 978-1-4503-8413-1/21/07…$15.00
https://doi.org/10.1145/3468791.3468819

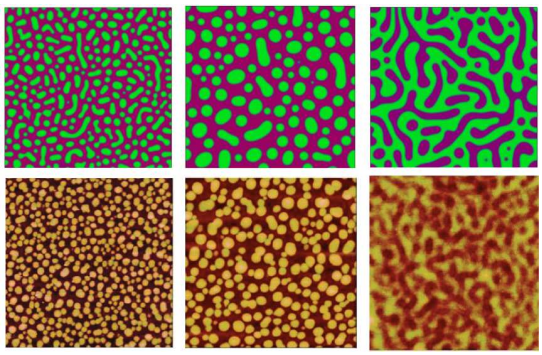


Figure 1: Example micrographs representing different phases of organic materials blending process. The top row shows predictions from a computational model. The bottom row shows related observation from atomic force microscope [16]. Corresponding morphologies are presented in Figure 2.

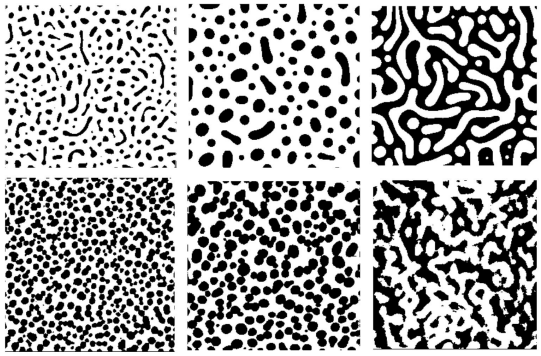


Figure 2: Morphologies corresponding to the micrographs in Figure 1.

three morphologies in the top row are the visualizations of the composition field gathered for the final state of three different numerical simulations. The three morphologies in the bottom show the actual state of the same materials as captured by microscopy. The key point here is that irrespective of the original representation, morphology can be ultimately captured as a fixed-sized bitmap. We note that this is true not only for morphologies appearing in materials science, but also in cosmology, biometry, tomography, etc.

Given the set $\mathcal{X}$, our objective is to capture the similarity between any two morphologies $\mathbf{X}_i, \mathbf{X}_j \in \mathcal{X}$. Let $f : \mathcal{X} \times \mathcal{X} \rightarrow [0, \infty)$ denote such a similarity function. As discussed earlier, f is critical for a variety of analyses one may want to perform on morphologies (e.g., searching, clustering, manifold learning, etc. [17]). In practice, the function must capture the *semantics* of the morphology, i.e., its geometric features that reflect the dynamics of its underlying process. We note that this question is different from a traditional image comparison. Specifically, the focus is on interpretability and efficiency, going beyond basic pixel, texture or color intensity matching (we provide more discussion in Section 5).

In order to be practical, the function f must satisfy two basic requirements. First, it must be *configurable*, such that it is able to capture the semantic properties of a morphology whose definition may vary from application to application. For example, when analyzing morphologies in the material separation process (Figure 2) in one application we may be interested only in shape of the features (i.e., two morphologies are similar only if they have similar droplets/grains). In another application, we may be looking in how complex the features are, which may be captured by their fractal dimension. The second requirement is that the function is *spatially invariant*, meaning that it is flexible in how it uses information about position of the features of interest within morphology. For example, consider again Figure 2 and suppose that we are interested in how well two materials separate in the bulk. If we focus only on the shape of droplets and disregard their spatial distribution, the first two morphologies are more similar to each other compared to the third one. In this case, we want to disregard spatial information since the process that generated morphologies is highly stochastic, and hence the actual position of droplets carries little physical information. However, in the presence of additional physical constraints (say morphology sandwiched between two electrodes in a device [20]) droplets closer to the constraint may be of much high importance and their location may have to be taken into account when computing the similarity. Hence the function should have some flexibility in handling such scenarios.

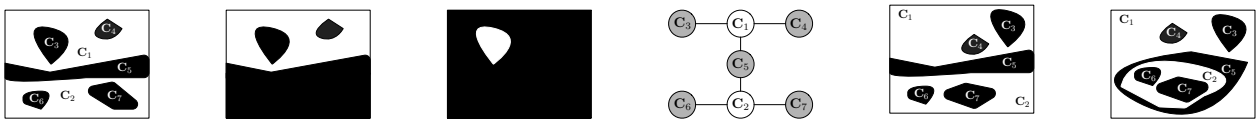
As noted above, it is inevitable that different applications will require a specific function, tailored to their particular needs. Thus, it is more appropriate to call f a family of functions. However, for simplicity of presentation we will refer to f as a function, and discuss its specific instances, whenever appropriate.

3 PROPOSED APPROACH

To satisfy the two requirements outlined in the previous section, we express the similarity between two morphologies, using concepts from graph theory. Specifically, we first show how a morphology can be represented as a *morphology graph* – an undirected graph capturing basic structural properties of the morphology, i.e., pairwise relationship between its features of interest. Next, we introduce the concept of a *signature function* to augment the morphology graphs with application-specific knowledge. Finally, we propose a graph comparison method that exploits properties of morphology graphs to represent them as vectors and compare them in a computationally and memory efficient way.

3.1 Representing Morphology as Graph

Recall that initially morphology $\mathbf{X}_i \in \mathcal{X}$ is a $(n \times m)$ bitmap, derived from an image representing some phenomena of interest. To gain more flexibility in morphology description, we propose an alternative representation that abstracts a morphology via an undirected graph. To achieve that, we first introduce the concept of *connected components*. Intuitively, a connected component in a morphology represents a local region with the same properties of interest (e.g., droplets in Figure 3). The collection of all connected components, together with their pairwise relationships, will provide us with the description from which we are going to build the target function f.



(a) Morphology X. (b) Component C_1. (c) Component C_3. (d) Morphology graph for X (also X′ and X″). (e) Morphology X′. (f) Morphology X″.

Figure 3: Example morphology with its connected components and the resulting morphology graph.

To define connected components, we begin by formalizing the notion of pixel neighborhood via the following definition.

Definition 3.1 (Pixel Neighborhood). *For any $(n \times m)$ bitmap, the neighborhood of a pixel (i, j), denoted as $nbd(i, j)$, is the set of its all adjacent pixels in the bitmap, i.e., $nbd(i, j) = \{(i', j') : |i-i'|+|j-j'| = 1, \forall i' \in \{1, \ldots, n\}, j' \in \{1, \ldots, m\}\}$.*

We note that by definition pixel neighborhood is commutative, which means that the following assertion always holds: $(i', j') \in nbd(i, j) \Leftrightarrow (i, j) \in nbd(i', j')$. This allows us to define connected component as:

Definition 3.2 (Morphology Connected Component). *For a given morphology, $\mathbf{X}$, a connected component, $\mathbf{C}$, is a $(n \times m)$ bitmap such that for every pair (i, j) and (i', j'), where $(i', j') \in nbd(i, j)$, if $\mathbf{C}(i, j) = 1$ and $\mathbf{X}(i, j) = \mathbf{X}(i', j')$ then $\mathbf{C}(i', j') = 1$. We will denote by $value(\mathbf{C})$ the pixel value $\mathbf{X}(i, j)$ of any pixel (i, j) such that $\mathbf{C}(i, j) = 1$, and will refer to it as component value or just value.*

Since the above definition ensures that any connected component is *maximal*, i.e., no additional pixels can be added to it, we can define the complete set of connected components for a morphology $\mathbf{X}$ as $C = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_{|C|}\}$.

To illustrate the above definitions consider the example in Figure 3. Here, morphology $\mathbf{X}$ has seven connected components, i.e., $|C| = 7$, that correspond to neighborhoods of pixels with the same value (for simplicity marked directly on the morphology image). The value of components $\mathbf{C}_1$ and $\mathbf{C}_2$ is *white* (or 1), i.e., $value(\mathbf{C_1}) = value(\mathbf{C}_2) = 1$, and the value of components $\mathbf{C}_3$ to $\mathbf{C}_7$ is *black* (or 0).

We are now ready to introduce our key idea, which is the *morphology graph*:

Definition 3.3 (Morphology Graph). *Given a morphology $\mathbf{X}$ and its morphology set, C, the morphology graph, $\mathcal{G}_{\mathbf{X}} = (C, E)$, is an undirected graph in which two vertices, $\mathbf{C}_u \in C$ and $\mathbf{C}_v \in C$, are connected by an edge, if and only if they are adjacent. Here adjacent means that there exists at least one pair of pixels, (i, j) and (i', j'), such that $(i, j) \in nbd(i', j')$ and $\mathbf{C}_u(i, j) = 1$ and $\mathbf{C}_v(i', j') = 1$, or equivalently $\mathbf{C}_u(i, j) \neq \mathbf{C}_u(i', j')$ or $\mathbf{C}_v(i, j) \neq \mathbf{C}_v(i', j')$. We will write $\mathcal{G}$ when morphology for which the graph is constructed is clear from the context.*

From the above definitions, we can infer several practical observations. First, the number of components in the morphology set can range from 1, if all pixels in the morphology have the same value, to $n \cdot m$, if the pixel values alternate forming a checkered pattern. Two connected components will be connected by an edge in the corresponding morphology graph only if they have different values, which means that the morphology graph is always *bipartite*. Furthermore, the graph must be *acyclic* (i.e., it is a tree), which will enable us to design efficient comparison algorithms. The acyclic property can be proven by noting that, if two nodes are connected by an edge, then either one of the corresponding components is contained inside the other component or the two components span the full length or breadth of the entire morphology area. This means that it is impossible to have another path between any pair of nodes that share an edge. In Figure 3d, we show example morphology graph where the color of each vertex corresponds to the value of its connected component.

The graph abstraction directly and compactly describes relationships between components of the morphology (e.g., which components are neighboring) disregarding components location within the morphology. Hence it addresses the *spatial invariant* requirement that we outlined in Section 2. Every morphology will have a unique graph representation, but a single graph may represent multiple morphologies. For example, consider morphologies $\mathbf{X}, \mathbf{X}'$ and $\mathbf{X}''$ in Figure 3. All three are represented by the same graph in Figure 3d. The fact that morphologies $\mathbf{X}$ and $\mathbf{X}'$ are represented by the same graph is highly desired (from our perspective, these are essentially the same morphologies – they have identical components though at varying positions). However, it is not so with morphology $\mathbf{X}''$. Here, we cannot argue that components are the same as in $\mathbf{X}$ (or $\mathbf{X}'$), especially if we look at components $\mathbf{C}_2$ and $\mathbf{C}_5$. In fact, this morphology is sufficiently different to assume that it has been generated by different process than the other two, or represents different time-step of the same generating process. To address this potential problem, we will extend our graph representation such that it carries information about properties of the individual components. Because it is very unlikely that two dissimilar morphologies will have the same graph structure and the same properties assigned to their components, the resulting representation will be more robust.

Let function $g : \{0, 1\}^{n \times m} \rightarrow \mathbb{R}$ be a user-specified function that quantifies some characteristic of interest for each component of the morphology. We will call such function a *signature function*. The signature function will be typically defined by a domain expert, taking into account specifics of the considered physical phenomena for which we obtained morphologies. It will allow us to characterize each component independently of the remaining components in the morphology. By weighting each node $\mathbf{C}$ in the morphology graph by its corresponding value $g(\mathbf{C})$ we will add critical information to the graph, fulfilling the configurability requirement described in Section 2.

To provide concrete examples of function g, we list a few functions that emerge in materials science [1]. These include *surface-to-volume* ratio $g_{SA:V}(\mathbf{C}) = \frac{SA(\mathbf{C})}{V(\mathbf{C})}$, where SA denotes surface area and V is a volume, *fractal dimension* where $g_D(\mathbf{C})$ is computed as the Hausdorff dimension [15], or *anisotropy* defined as $g_{Ani}(\mathbf{C}) = \frac{d_x(\mathbf{C})}{d_y(\mathbf{C})}$, where $(d_x(\mathbf{C}), d_y(\mathbf{C}))$ are signed dimensions of the bounding box around $\mathbf{C}$. In real-world applications, $g_{SA:V}$ captures porosity or compactness of a component that directly affects its chemical reactivity, g_D provides information about how given component interpenetrates with its neighbors, which is important for the physical properties such as conductivity, and g_{Ani} is a proxy to characterize directional stiffness of the material that the morphology represents. We note that this list is by no means exhaustive, and the advantage of our approach is that any function g can be plugged into our general framework.

3.2 Vectorization of Morphology Graph

Given two morphologies described by their morphology graphs with the same function g, the question of finding a similarity function f becomes now the problem of graph comparison. This problem, often referred to as graph alignment, is well studied especially in the context of computational biology [9]. However, the existing approaches are usually extremely computationally demanding, and relay on some additional function to establish isomorphism between nodes of the graphs under consideration. In this work, we address both challenges by exploiting properties of the morphology graphs. The general idea derives from the observation that if two morphologies are highly similar then their morphology graphs should have roughly the same node degree and function g distributions. To test for that, we will first decide how each morphology graph should be rooted (recall that each morphology graph is a tree). Then, we will establish which nodes in one graph should be compared to which nodes in the other graph by ordering them via a Breadth-First-Search (BFS) traversal. By rooting and traversing morphology graphs in the same consistent way, we will narrow down the comparison space. Finally, to ensure that these tasks are executed in a computationally efficient way, we will leverage the fact that morphology graphs are bipartite. We note that the end result of the proposed transformations is equivalent to representing a morphology graph as a high dimensional vector, and hence will refer to it as vectorization.

Choosing the Root Node. As we mentioned, the first step in our approach is to select the root node for each morphology we wish to compare. This step is critical as it directly affects the order in which we are going to traverse our morphology graphs and hence the order in which we will compare components between the morphologies.

In our approach, the root node will represent the most significant component in the morphology. In cases where importance of the components will be clearly defined by the morphology generating process (recall our example in Section 2 where physical constraints dictate importance) we will select the root node following that criteria. In cases when such constraints will not be available, we will use the following heuristic: we will root each morphology graph by the node that is central in the graph to which it belongs. Because we are dealing with trees, this means the node with the highest number of adjacent nodes. This heuristic is based on the observation that since the central node captures the highest information about pairwise interactions between the components, it will capture the most spatial constraints (again, we expect that highly similar morphologies will have similar constraints).

It is possible that given morphology graph will have multiple nodes with the same maximal connectivity. In such cases, we will break the tie by always choosing the node with *a-priori* agreed component value (e.g., if two nodes are tied then pick the *white* one) or the smallest signature (i.e., value of g).

Constructing the Feature Vector. Given the root node r of the morphology graph $\mathcal{G}$, the second step is to traverse the graph to establish the ordering of its nodes, and thus build its vector representation. The traversing algorithm is summarized in Algorithm 1. In the essence, we perform BFS traversal over $\mathcal{G}$ starting at node r. Whenever we visit a new vertex (i.e., morphology component), we add a tuple with its corresponding signature and value to the initially empty output vector (lines 5-6). The order in which nodes are visited at given BFS level is decided by the value of their function g. Specifically, after extracting unexplored neighborhood of a vertex (denoted by function $N_{\mathcal{G}}$ in line 7) we sort it in line 8, and use the resulting order to continue exploration in lines 9-10. Observe that by ordering the nodes we further constrain which nodes will be compared with which.

To illustrate how our vectorization routine works, consider morphology graph $\mathcal{G}$ in Figure 3d, and suppose that we have function g with the following property: $g(\mathbf{C}_1)$ is the smallest value, $g(\mathbf{C}_3) < g(\mathbf{C}_4) < g(\mathbf{C}_5)$ and $g(\mathbf{C}_7) < g(\mathbf{C}_6)$. In that case, we would select $r = \mathbf{C}_1$ to be the root, since it is tied with $\mathbf{C}_2$ but has smaller signature, and by calling Vectorize$(\mathcal{G}, r)$ we would generate the following vector:

$$v = [(g(\mathbf{C}_1), 1), (g(\mathbf{C}_3), 0), (g(\mathbf{C}_4), 0), (g(\mathbf{C}_5), 0), (g(\mathbf{C}_2), 1), (g(\mathbf{C}_7), 0), (g(\mathbf{C}_6), 0)].$$

In the vector representation of a morphology, the lower dimensions of the vector, specifically the dimensions representing root and its immediate neighbors, uniquely describe the graph for which the vector has been derived (i.e., given the vector we can reconstruct its graph). But as we consider higher dimensions, the representation is no longer unique. For example, the vector v presented above could equivalently describe morphology graph in which component $\mathbf{C}_2$ is neighboring vertex $\mathbf{C}_3$ or $\mathbf{C}_4$. However, this is actually advantageous as it reflects our assumption that the actual location of the components within morphology should not matter. The dimensionality of the vector representing a morphology will be the same as the number of components in that morphology. Consequently, when comparing two vectors we will have to find a reliable way to align them such that similar components between the two morphologies are matched. Here similar means not only components with the same value, but also potentially similar signature.

3.3 Comparing Morphologies Using Vectors

We are now ready to bring all ingredients together to define function f. Given two morphologies $\mathbf{X}_i$ and $\mathbf{X}_j$ and a signature function g, to compute $f(\mathbf{X}_i, \mathbf{X}_j)$ we will first vectorize graphs $\mathcal{G}_{\mathbf{X}_i}$ and

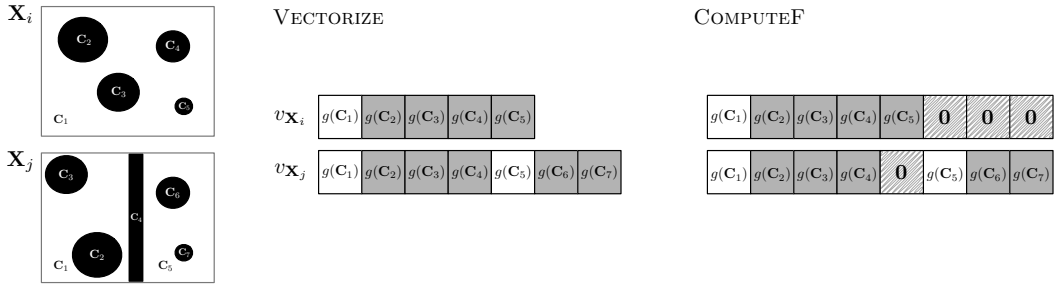


Figure 4: Two example morphologies together with their corresponding vector representation, and the alignment of the vectors induced by the COMPUTEF function.

Algorithm 1 VECTORIZE($\mathcal{G}, r$)

1: $Q \leftarrow \emptyset$
2: Q.ENQUEUE(r)
3: $v \leftarrow \emptyset$ ◁ output vector
4: **while** $Q \neq \emptyset$ **do**
5: $\mathbf{C}_u \leftarrow Q$.DEQUEUE()
6: v.APPEND($(g(\mathbf{C}_u), value(\mathbf{C}_u))$)
7: $L \leftarrow N_{\mathcal{G}}(\mathbf{C}_u)$
8: SORT($L, \mathbf{C}_i < \mathbf{C}_j$ iff $g(\mathbf{C}_i) < g(\mathbf{C}_j)$)
9: **for each** $\mathbf{C}_v \in L$ **do**
10: Q.ENQUEUE($\mathbf{C}_v$)
11: **return** v

$\mathcal{G}_{\mathbf{X}_j}$, and then we will compute the final distance between the vectors taking into account the need for alignment. This is summarized in Algorithm 2.

Algorithm 2 COMPUTEF($\mathbf{X}_i, \mathbf{X}_j$)

1: $v_{\mathbf{X}_i} \leftarrow$ VECTORIZE($\mathcal{G}_{\mathbf{X}_i}$, ROOT($\mathcal{G}_{\mathbf{X}_i}$))
2: $v_{\mathbf{X}_j} \leftarrow$ VECTORIZE($\mathcal{G}_{\mathbf{X}_j}$, ROOT($\mathcal{G}_{\mathbf{X}_j}$))
3: $p_i \leftarrow 1$
4: $p_j \leftarrow 1$
5: $v_f \leftarrow \emptyset$
6: **while** $p_i < |v_{\mathbf{X}_i}| \ \wedge \ p_j < |v_{\mathbf{X}_j}|$ **do**
7: **if** $v_{\mathbf{X}_i}[p_i].value = v_{\mathbf{X}_j}[p_j].value$ **then**
8: v_f.APPEND($v_{\mathbf{X}_i}[p_i].g - v_{\mathbf{X}_j}[p_j].g$)
9: $p_i \leftarrow p_i + 1$
10: $p_j \leftarrow p_j + 1$
11: **else**
12: **if** $v_{\mathbf{X}_i}[p_i].value \neq v_{\mathbf{X}_i}[p_i - 1].value$ **then**
13: v_f.APPEND($v_{\mathbf{X}_i}[p_i].g - 0$)
14: $p_i \leftarrow p_i + 1$
15: **else**
16: v_f.APPEND($0 - v_{\mathbf{X}_j}[p_j].g$)
17: $p_j \leftarrow p_j + 1$
18: **for** $p = p_i \ldots |v_{\mathbf{X}_i}|$ **do**
19: v_f.APPEND($v_{\mathbf{X}_i}[p].g - 0$)
20: **for** $p = p_j \ldots |v_{\mathbf{X}_j}|$ **do**
21: v_f.APPEND($0 - v_{\mathbf{X}_j}[p].g$)
22: **return** $f \leftarrow \|v_f\|$

Given the vector representation of the input morphologies (lines 1-2), where function ROOT is as described in Section 3.2, the algorithm iteratively constructs vector v_f to capture similarity between signatures of the matching components. Here, the matching is established by exploiting the fact that the graphs $\mathcal{G}_{\mathbf{X}_i}$ and $\mathcal{G}_{\mathbf{X}_j}$ are bipartite, and components within given BFS level are ordered by their signature. The example of this process is visually presented in Figure 4. The algorithm first checks whether components (dimensions) to compare correspond to the same BFS level (line 7). Since the graph is bipartite, this can be established by checking if the corresponding component values are the same. If that is the case, the algorithm compares signatures of the components, one-by-one in the order in which they appear (lines 8-10). Because within a BFS level components are ordered by their signature, this process will be minimizing differences between similar morphologies and amplifying differences between diverging morphologies. In case when the number of components at given BFS level differs between morphologies (else statement in line 11), the algorithm executes the alignment step (lines 13 and 14) by penalizing the missing components (loops in lines 18 and 20 are to handle corner case of that idea). Finally, the algorithm computes and returns a norm of the vector v_f as the final similarity measure (line 22). Here, potentially any norm can be used, however, in our applications we commonly depend on L_2.

The computational complexity of Algorithm 2 is linear and bounded by the number of components in the input morphologies, i.e., it is $O(|v_{\mathbf{X}_i}| + |v_{\mathbf{X}_j}|)$. This is because the main iteration requires that each component in the morphology will be visited exactly once. The same holds true for functions VECTORIZE and ROOT. The memory complexity is $O(1)$ as we observe that the vector v_f does not have to be explicitly instantiated, and instead the computation of the norm of v_f can be integrated into the components comparison in lines 8, 13 and 16. Considering that the typical graph comparison algorithms have complexities above $O(|v_{\mathbf{X}_i}| \cdot |v_{\mathbf{X}_j}|)$ we dare to call our algorithm efficient.

4 EXPERIMENTAL VALIDATION

In this section we demonstrate the efficacy of the proposed method in capturing the semantic similarity between a pair of morphologies on both synthetic and real-world data sets. To put our results in context, we compare the proposed approach to two state-of-the-art methods, one based on direct pixel-by-pixel comparison and the

other based on FFT transformation. The prototype implementation of our algorithms and our test data sets are freely available from https://github.com/ubdsgroup/meads/.

4.1 Experimental Data

We show results on two data sets. The first set consists of synthetic morphologies in which we control the complexity of the morphologies. This permits us to analytically reason about the distances between them. The second set consists of real-world morphologies obtained via high-fidelity numerical simulations of spinodal decomposition in polymer blends. These kind of simulations are frequently used in materials science to understand physical processes occurring in manufacturing of organic solar cells, bio-sensors, and other organic thin-film based technologies [20]. They are also representative of a broader class of physical phenomena in which the stochastic nature of the underlying processes leads to similar but not identical morphologies.

When analyzing both data sets, we use *surface-to-volume* ratio ($g_{SA:V}$) as our choice of signature function, since it best represents the semantic features of the morphologies in the two data sets.

Synthetic Set. To generate set $\mathcal{X}$ of synthetic morphologies we use the following process. First, we generate an initial morphology, $\mathbf{X_1}$, which consists of few randomly sized circles (although any basic shape could be used). The circles are randomly placed within a fixed area (see Figure 5a). Each such component has component value of 0 (black), while the remaining part of the morphology has component value of 1 (white). To generate subsequent morphologies, we iteratively apply transformation $\mathbf{X}_i = t(\mathbf{X}_{i-1})$, where t randomly selects one of the following operations:

(1) *Scale up one randomly selected morphology component.* The radius of the selected component is scaled by a factor $\gamma > 1$ (see Figure 5b).
(2) *Add a new component.* If the component is placed within component other than the one representing the entire morphology area, it is centered and its radius is selected randomly with the constraint that it must be smaller than the radius of the surrounding component. The value of the new component is set to be opposite to the value of the surrounding component (see Figure 5c). Otherwise, the component is just randomly placed with the component value of 1.

Figure 5 shows an example of the above generation process. The process guarantees that for any $i < j$, the morphology $\mathbf{X}_j$ will have either more components, or have larger components than morphology $\mathbf{X}_i$. This, in turn, means that the similarity measured by an ideal similarity function should be inversely proportional to the number of transformations separating two morphologies. Therefore, when applied to the synthetic data, we expect that our similarity function f should maintain the following property: $f(\mathbf{X}_i, \mathbf{X}_j) \propto \frac{1}{j-i}, \forall i < j$.

Organic Thin-film Set. This data set consists of morphologies representing a physical phenomenon called spinodal decomposition in organic blends [8]. As mentioned earlier, morphologies in this data set are generated using a computational model of morphology evolution during thermal annealing of the thin organic films [20]. For a given input parameters representing *blend ratio*, ϕ, and *strength of interaction*, χ, which reflect fabrication conditions, the model delivers a series of morphologies that capture temporal changes that the material undergoes during the fabrication process. We refer to such ordered set of morphologies as *trajectory*. In Figure 6, we show example morphologies extracted at the beginning, in the middle and at the end of one example trajectory with $(\phi, \chi) = (0.50, 2.4)$. From the figure, we can see that even within a single trajectory we may obtain topologically diverse morphologies: ranging from bubble-like to highly interpenetrated structures. This diversity is further amplified by the changes to parameters (ϕ, χ), as we can observe in Figure 10. Finally, for exactly the same parameters (ϕ, χ) different executions of the simulation deliver morphologies that are not identical, however are semantically similar. This is because the generation process is stochastic: in the underlying physics-based model different seed values are used to initiate the random field.

To perform our tests we considered sixteen trajectories spanning $\phi \in \{0.5, 0.53, 0.56, 0.59\}$ and $\chi \in \{2.2, 2.8, 3.4, 4.0\}$, with five replicas for each trajectory.

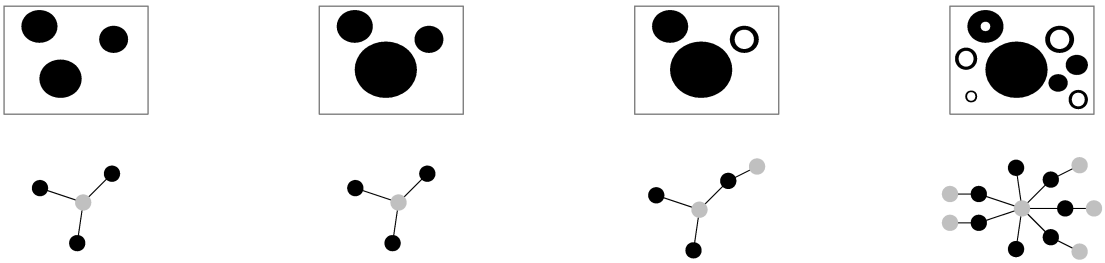
4.2 Results on Synthetic Data

In the first set of experiments, we consider $\mathcal{X}$ with 50 synthetically generated morphologies. We compute similarity between each pair of morphologies using three methods: 1) our proposed ComputeF function, 2) *pixel-by-pixel* distance in which distance between morphologies is computed as difference between their corresponding pixels, and 3) distance between the 2D Fourier transformed morphologies (2D-FFT). Here we wish to note that the FFT-based morphology distance is currently the most commonly used method in the scientific community [11, 18].

Using each of the three similarity measures we construct similarity matrix and visualize it as a heat map, as shown in Figure 7. Here, 0 means that morphologies are identical, and 1 that morphologies are highly dissimilar (in all cases, similarity is normalized by applying a linear transformation to the morphology distances). Since a morphology is most similar to itself we expect to see zeros on diagonal. This is indeed the case for all three measures. As we move away from the the diagonal, the number of transformations between the morphologies increases and, as discussed in the previous section, the distance between the morphologies should also increase proportionally. Again, as we would expect this holds true for all three measures, however, for each in slightly different way. Specifically, the pixel-by-pixel (Figure 7a) and 2D-FFT (Figure 7c) similarities show an abrupt jump as the gap between the morphology increases, whereas in the ComputeF the distance increases gradually. We consider this behaviour more in inline with the characteristics of our synthetic data discussed in the previous sub-section.

4.3 Results on Organic Thin-film Set

We further evaluate our framework using the organic thin-film data. To perform the evaluation we compute three different types of similarity matrices: 1) a *within-trajectory* matrix that captures pairwise comparison between all morphologies within a trajectory for given configuration parameters (ϕ, χ), 2) a *cross-replica* matrix that captures pairwise comparison between morphologies from two different replicas of the same trajectory, and, 3) a *late-state*



(a) Initial morphology X_1. (b) Morphology X_2 has one component scaled. (c) Morphology X_3 has one additional component. (d) Morphology evolved after several iterations.

Figure 5: Example of the synthetic morphologies generation process together with the corresponding morphology graphs.

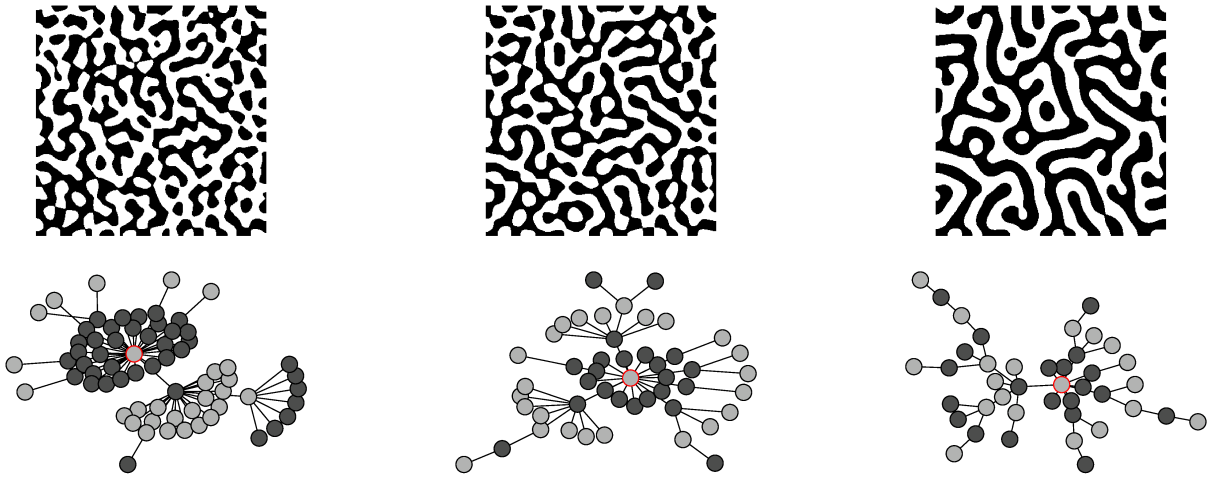
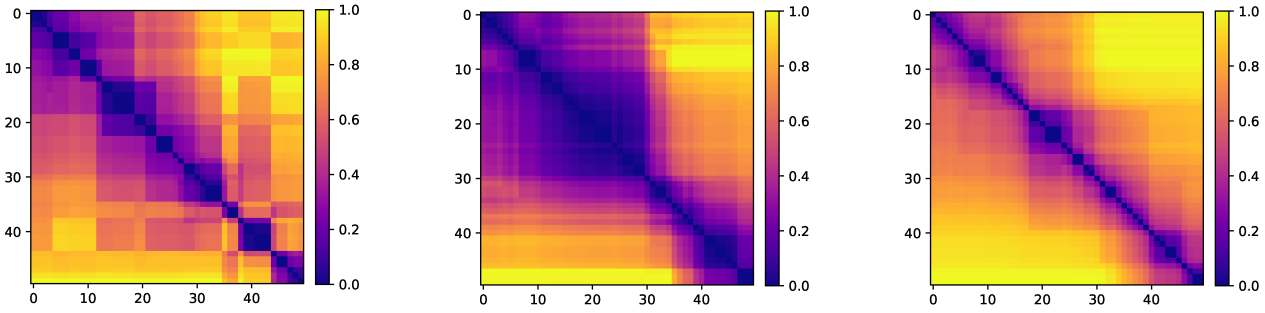


Figure 6: Example morphologies from one trajectory ($\phi = 0.50, \chi = 2.4$). For each morphology, we included the corresponding graph.



(a) ComputeF (b) Pixel-by-pixel (c) 2D-FFT

Figure 7: Comparison of morphologies from the synthetic set. Morphologies within the set are indexed starting from 0. Please view in color.

matrix that captures pairwise comparison of so called late-state morphologies across different trajectories and their replicas. Here, late-state morphology is a morphology reported at the end of trajectory, where each trajectory covers the same duration of the manufacturing process.

Within-trajectory Comparison. In this test, we perform comparison of morphologies coming from the trajectory obtained for parameters $\phi = 0.56$ and $\chi = 2.4$ (note that we observed similar results for other parameter configurations). As before, we perform all-to-all pairwise comparison using three different similarity measures. The results of this experiment are summarized in Figure 8. Two observations immediately emerge. First, both ComputeF and pixel-by-pixel exhibit similar and desired behavior: as morphologies get further apart in time (i.e., the distance between their indexes increases) their similarity decreases. Moreover, morphologies at the later stages of the trajectory (roughly with indexes above 40) tend to be more similar to each other. This is explained by the physical properties of the thin-film generating process: after the initial development, morphologies stabilize and change at much slower pace. The surprisingly good performance of pixel-by-pixel comparison is explained by the fact that within trajectory the spatial invariant requirement (which we discuss in Section 2) becomes irrelevant, and hence difference between morphologies is directly reflected by change in pixels composition. The second observation is about poor performance of the 2D-FFT similarity measure. Because even small changes to the size and shape of the components in the morphology are amplified in the frequency domain the resulting similarity measure becomes very sensitive to such changes. Consequently, except of the diagonal morphologies are deemed dissimilar. We note that this result is visually strengthened by how we normalize the heat map.

Cross-replica Comparison. In the next test, we compare morphologies belonging to two replicas of the same trajectory with $\chi = 0.54$ and $\phi = 2.4$ but generated using a different initial seed value. As stated before, the replica morphologies (i.e., morphologies belonging to different replica but representing the same stage of the generating process) are semantically similar, and we would expect that fact to be reflected by a good similarity measure. The result of the experiment is summarized in Figure 9.

From the figure we can observe that ComputeF reports high similarity along the approximate diagonal, and within blocks corresponding to early and late stages of the trajectory (similar to what we saw in Figure 8). This indicates that the morphologies maintain a degree of similarity at the corresponding steps across replicas, which is expected based on what we know about the generating process. The other two functions (pixel-by-pixel and 2D-FFT) are clearly unable to detect meaningful similarities. This supports our claim that good similarity measure must be spatially invariant.

Late-state Morphologies Comparison. In the last set of experiments, we focus on five replicas corresponding to the four different combinations of the design parameters such that $\phi \in \{0.50, 0.56\}$ and $\chi \in \{2.4, 3.0\}$. From each of the resulting 20 trajectories, we select late-state morphologies. As previously, we expect that replica morphologies will be similar to each other. At the same time, because we are using diverse configuration parameters, we expect that morphologies obtained for different design parameters will noticeably differ. In Figures 10 and 11 we show the input morphologies together with their morphology graphs.

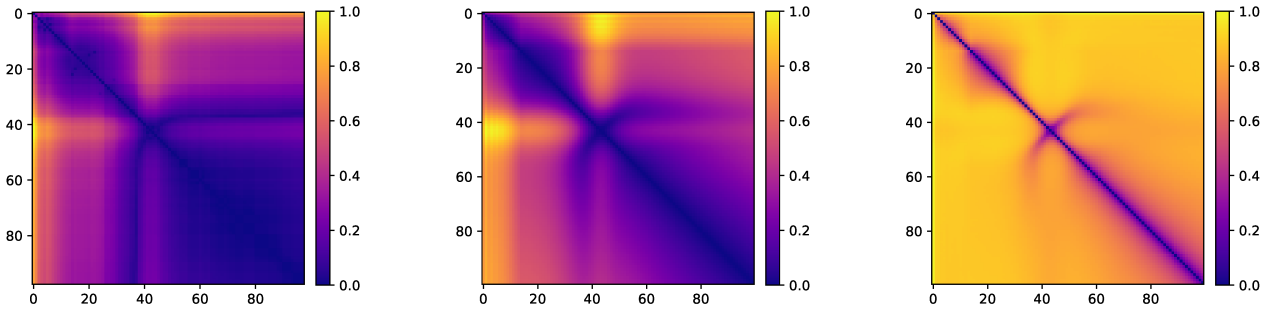
To perform the analysis, we begin with comparison of similarity matrices obtained for all 20 morphologies. From Figure 12 we can see that ComputeF function is able to capture the similarity between the morphologies within each replica, while pixel-by-pixel and 2D-FFT are clearly under-performing. If we look at the regions of the heat map that correspond to cross-replica comparison (regions marked with boxes A-D) we can see that only ComputeF correctly reports minimal values within each block. At the same time, it provides visible separation from other blocks (that correspond to different parameters ϕ and χ). However, that separation is not entirely clear for blocks A and B. To investigate this further, we provide an alternative visualization by embedding each morphology onto a two dimensional manifold by applying the *Multi-dimensional Scaling* (MDS) algorithm [13] on each similarity matrix. The resulting embeddings are shown in Figure 13a.

The results reveal that embedding obtained using ComputeF similarity matrix delivers strong clustering of morphologies corresponding to the same design parameters (depicted by the same color). One notable exception are morphologies with indexes 6 and 19, which are outliers. However, closer inspection of these morphologies in Figure 10 shows that indeed these morphologies may be considered semantically different from the remaining replicas. In fact, upon investigation we discovered that the computational model failed to converge (due to numerical instabilities) when simulating the trajectories to which outlier morphologies belong. This demonstrates the robustness of our framework as well as its capability to quantify outliers and anomalous morphology data.

5 RELATED WORK

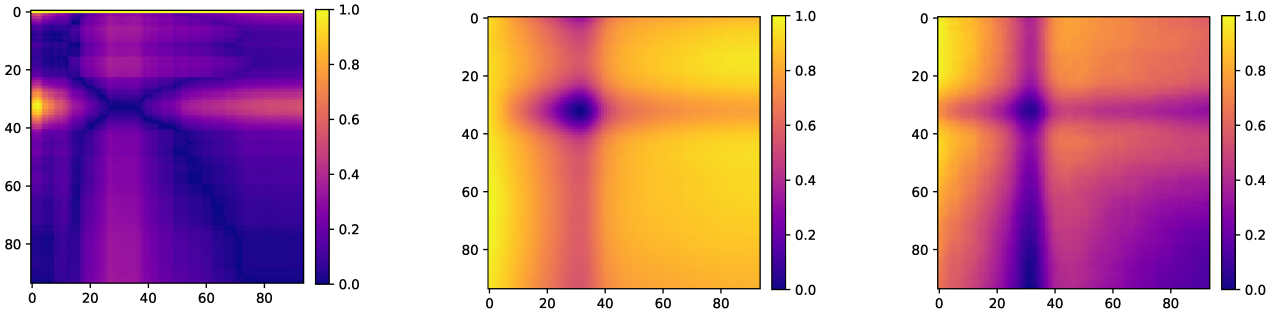
As we already discussed, the problem of morphology comparison emerges in many scientific applications. The commonly used approaches focus on pixel-level algorithms, often combined with 2D-FFT to address the problem of spatial invariance [11, 18]. As we demonstrated in our experiments, these strategies are usually insufficient to handle complexity of the real-world morphologies, and lack flexibility offered by our concept of signature function. The idea of representing morphology by a graph structure has been explored by Cecen et al. in [7], as well as Wodo et al. in [21]. However, in both works the focus was on capturing fine-grained details of the morphology, not for the purpose of comparison, but to deliver a surrogate model. Consequently, in these approaches, each pixel/voxel of a morphology is represented as a graph vertex. In our work, the morphology is converted into a graph to represent both higher-level morphology semantic and structural characteristics sufficient to reliably compare different morphologies.

The idea of determining visual similarity on the basis of semantic features has been explored in the past in the form of representing images in terms of latent space embedding using convolution neural networks. This along with several other approaches [12] often referred as deep learning metrics fall short because of two keys reasons. One, training such models usually require large amounts of data, which are expensive to obtain as morphologies are often generated as a result of complicated scientific procedures. Second,



(a) ComputeF (b) Pixel-by-pixel (c) 2D-FFT

Figure 8: Comparison of morphologies within-trajectory. Morphologies are indexed starting from 0. Please view in color.



(a) ComputeF (b) Pixel-by pixel (c) 2D-FFT

Figure 9: Comparison of morphologies across replicas. Rows correspond to morphologies in the first replica, columns in the second, with morphologies indexed from 0. Please view in color.

we need interpretable methods, that can be understood by the domain scientists. Metric learning methods are often used as "black-boxes" and do not offer any explanation into why two morphologies are considered similar or not.

In our work, we use a dedicated graph comparison method. As we explained, this is because the existing general-purpose approaches are too computationally expensive. The two most popular methods are the Graph Edit Distance [4] and Maximum Common Sub Graph [5], which inspired several other [2]. The computation of graph edit distance is NP-Complete. For graphs with more than 16 nodes, it becomes impossible to compute accurate results in a reasonable amount of time [3].

6 CONCLUSIONS

Morphologies as objects of interest emerge in a diverse range of scientific applications. We consider their significance in the domain of materials science as a motivation to develop a framework that helps us efficiently determine the similarities between different morphologies. The proposed methodology uses structural and semantic characteristics as a foundation and provides flexibility for the users to modify it according to their needs with the use of different signature functions. Furthermore, we demonstrate how the similarity framework performs significantly better when compared to the commonly used similarity measures such pixel-by-pixel comparison or 2D-FFT. The framework exposed certain outliers in the morphology data sets that otherwise were hard to identify unless observed manually. We hope that our framework will enable more systematic use of data analytics in the analysis of morphologies (especially in materials science).

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation (Office of Advanced Cyberinfrastructure – OAC-1910539, Civil, Mechanical, & Manufacturing Innovation – CMMI-1906344). Computing facilities were provided by the University at Buffalo's Center for Computational Research.

REFERENCES

[1] Viviana Avila and Rishi Raj. 2019. Flash sintering of ceramic films: The influence of surface to volume ratio. *Journal of the American Ceramic Society* 102, 6 (2019),

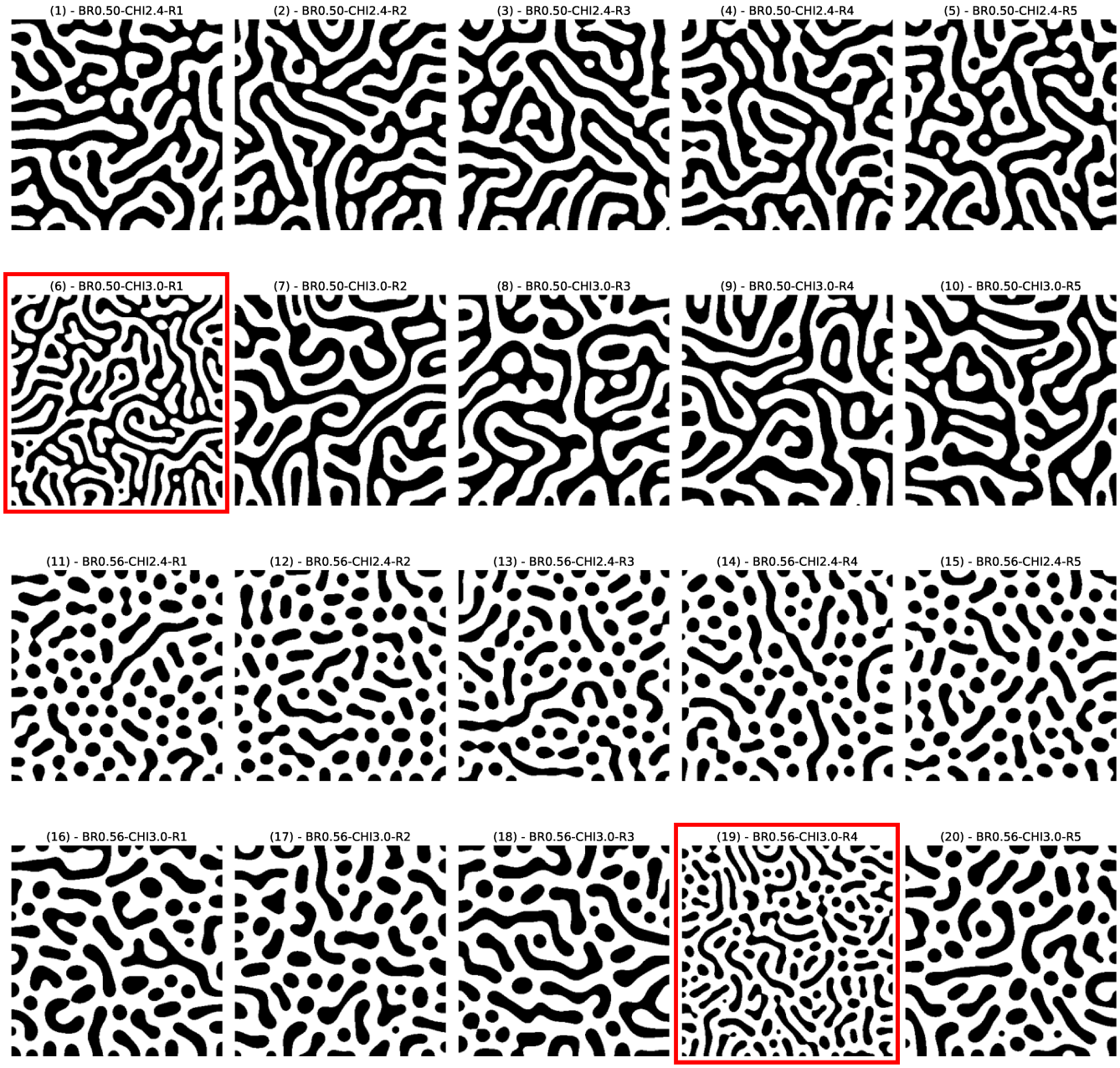


Figure 10: Late state morphologies for four different parameter combinations. Each row corresponds to five replicas for one parameter combination. The highlighted morphologies correspond to stochastic "outliers" that are semantically distinct from other replicas. BR denotes parameter ϕ and CHI denotes parameter χ. Index of each morphology is given in parenthesis above the image.

3063–3069.

[2] David B Blumenthal, Nicolas Boria, Johann Gamper, Sébastien Bougleux, and Luc Brun. 2020. Comparing heuristics for graph edit distance computation. *The VLDB Journal* 29, 1 (2020), 419–458.

[3] David B Blumenthal and Johann Gamper. 2018. On the exact computation of the graph edit distance. (2018).

[4] Horst Bunke. 1983. What is the distance between graphs. *Bulletin of the EATCS* 20 (1983), 35–39.

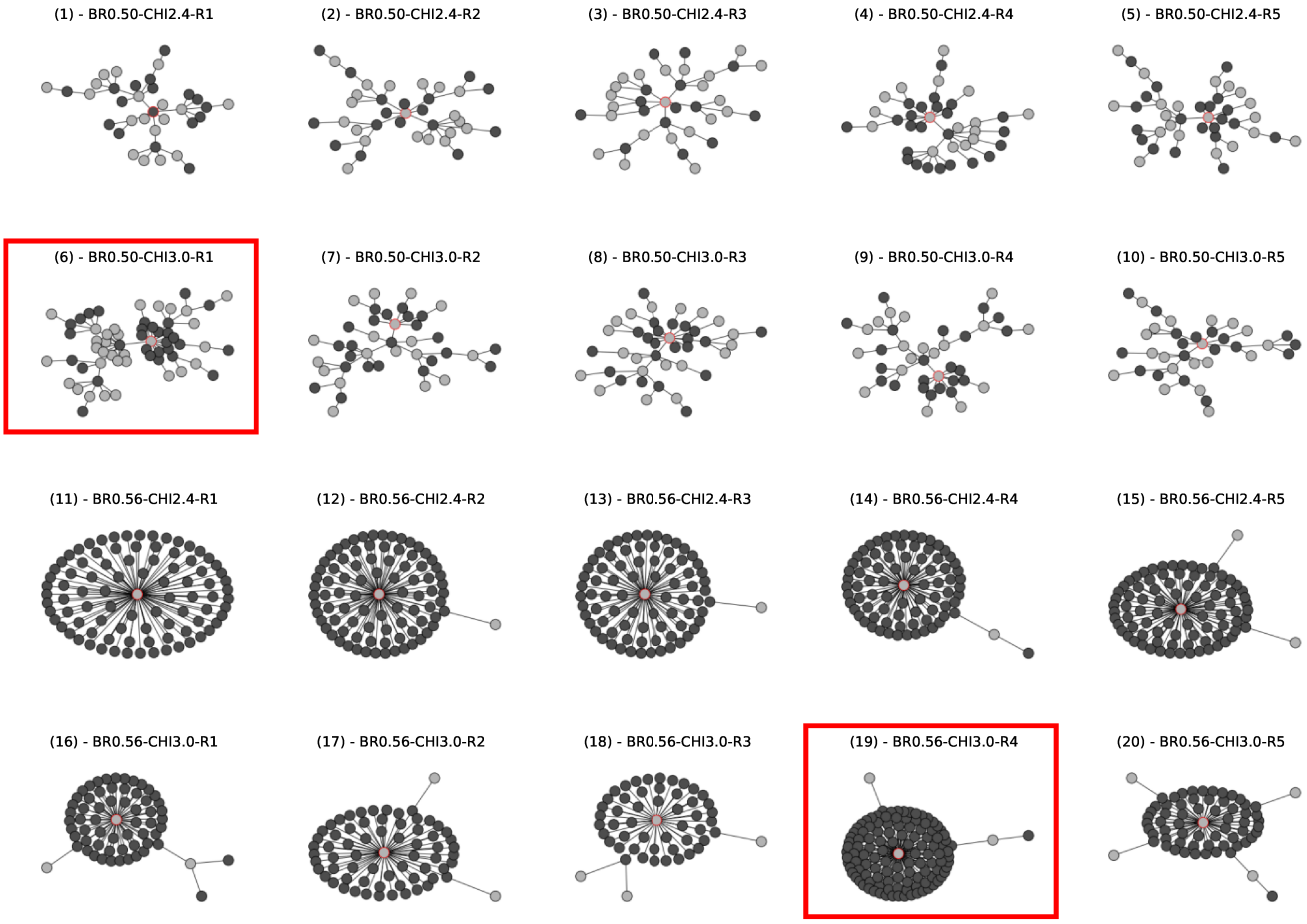
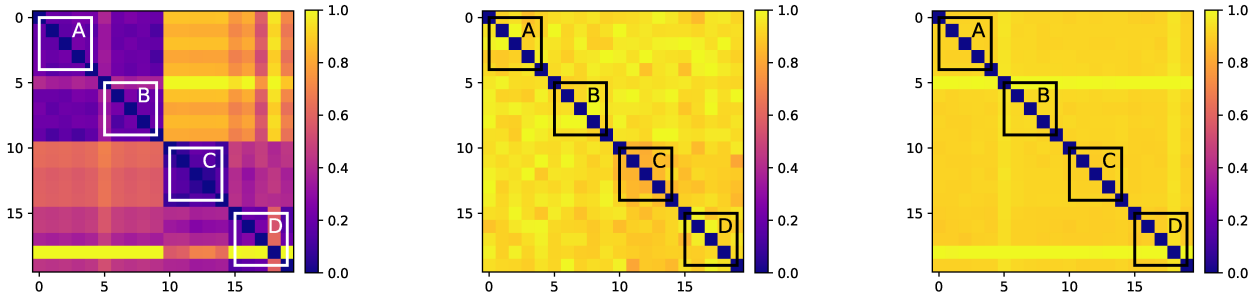
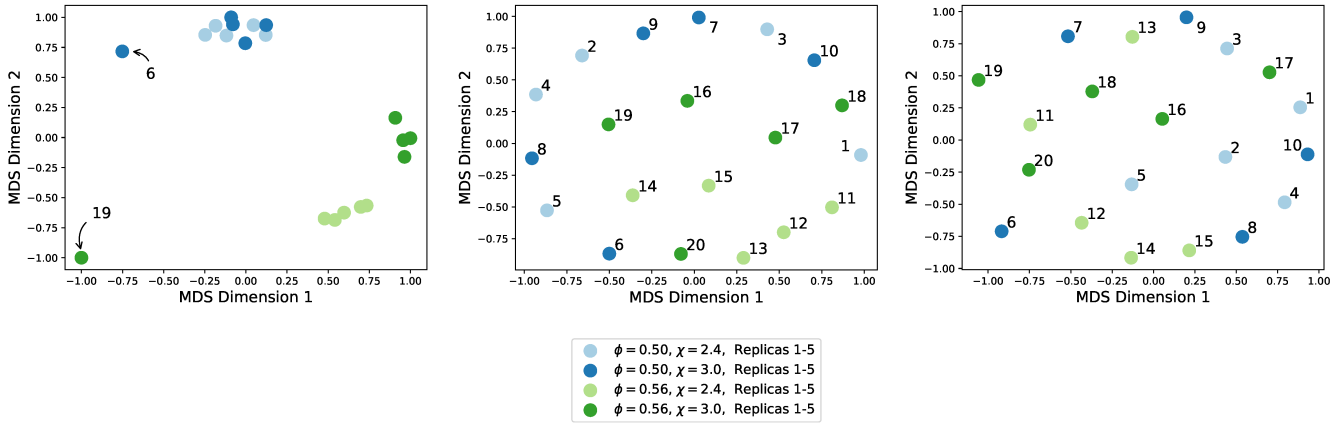


Figure 11: Graphs corresponding to the late stage morphologies shown in 10.



(a) ComputeF (b) Pixel-by-pixel (c) 2D-FFT

Figure 12: Comparison of late-state morphologies.



(a) ComputeF **(b) Pixel-by-pixel** **(c) 2D-FFT**

Figure 13: 2D embedding of late-state morphologies obtained via MDS over similarity matrices from Figure 12 .

[5] Horst Bunke and Kim Shearer. 1998. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters* 19, 3 (1998), 255–259. https://doi.org/10.1016/S0167-8655(97)00179-7

[6] John W Cahn. 1961. On spinodal decomposition. *Acta metallurgica* 9, 9 (1961), 795–801.

[7] Ahmet Cecen, Eric A Wargo, Anne C Hanna, David M Turner, Surya R Kalidindi, and Emin C Kumbur. 2011. Microstructure analysis tools for quantification of key structural properties of fuel cell materials. *ECS Transactions* 41, 1 (2011), 679.

[8] Pierre-Gilles de Gennes. 1980. Dynamics of fluctuations and spinodal decomposition in polymer blends. *The Journal of Chemical Physics* 72, 9 (1980), 4756–4763.

[9] Ahed Elmsallati, Connor Clark, and Jugal Kalita. 2016. Global Alignment of Protein-Protein Interaction Networks: A Survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 13, 4 (July 2016), 689–705. https://doi.org/10.1109/TCBB.2015.2474391

[10] Frank Jülicher. 1996. The morphology of vesicles of higher topological genus: conformal degeneracy and conformal modes. *Journal de Physique II* 6, 12 (1996), 1797–1824.

[11] Surya R Kalidindi, Stephen R Niezgoda, and Ayman A Salem. 2011. Microstructure informatics using higher-order statistics and efficient data-mining protocols. *Jom* 63, 4 (2011), 34–41.

[12] Mahmut Kaya and H. Bilge. 2019. Deep Metric Learning: A Survey. *Symmetry* 11 (08 2019), 1066. https://doi.org/10.3390/sym11091066

[13] J.B. Kruskal and M. Wish. 1978. *Multidimensional Scaling*. Sage Publications.

[14] Chris J Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, et al. 2008. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society* 389, 3 (2008), 1179–1189.

[15] Benoit B Mandelbrot, Dann E Passoja, and Alvin J Paullay. 1984. Fractal character of fracture surfaces of metals. *Nature* 308, 5961 (1984), 721–722.

[16] Svante Nilsson, Andrzej Bernasik, Andrzej Budkowski, and Ellen Moons. 2007. Morphology and phase segregation of spin-casted films of polyfluorene/PCBM blends. *Macromolecules* 40, 23 (2007), 8291–8301.

[17] Frank Schoeneman, Varun Chandola, Nils Napp, Olga Wodo, and Jaroslaw Zola. 2020. Learning Manifolds from Dynamic Process Data. *Algorithms* 13, 2 (Feb. 2020), 30. https://doi.org/10.3390/a13020030

[18] Yue Sun, Ahmet Cecen, John W Gibbs, Surya R Kalidindi, and Peter W Voorhees. 2017. Analytics on large microstructure datasets using two-point spatial correlations: Coarsening of dendritic structures. *Acta Materialia* 132 (2017), 374–388.

[19] Salvatore Torquato and HW Haslach Jr. 2002. Random heterogeneous materials: microstructure and macroscopic properties. *Appl. Mech. Rev.* 55, 4 (2002), B62–B63.

[20] Olga Wodo and Baskar Ganapathysubramanian. 2012. Modeling morphology evolution during solvent-based fabrication of organic solar cells. *Computational Materials Science* 55 (2012), 113–126.

[21] Olga Wodo, Srikanta Tirthapura, Sumit Chaudhary, and Baskar Ganapathysubramanian. 2012. A graph-based formulation for computational characterization of bulk heterojunction morphology. *Organic Electronics* 13, 6 (2012), 1105–1113.