Detecting Symmetry in Scalar Fields Using Augmented Extremum Graphs

Dilip Mathew Thomas and Vijay Natarajan, *Member, IEEE*

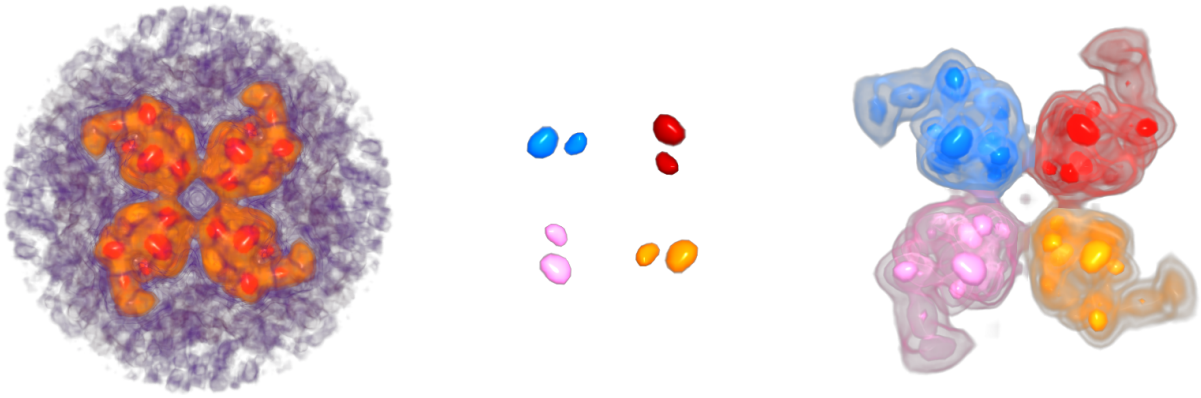


Fig. 1. Robust scalar field symmetry identification algorithm detects symmetry even in the presence of significant noise in the electron microscopy data of the Rubisco RbcL8-RbcX2-8 complex (EMDB 1654). (left) Volume rendering shows symmetry and noise in the data. (center) A set of seed cells is chosen as source vertices for traversing the augmented extremum graph of the data. During the traversal, the seed cells merge together to form four symmetric super-seeds. Seed cells that belong to a common super-seed are shown with the same color. (right) The initial estimate of symmetry is expanded in a region growing stage to identify the symmetric regions. A symmetry-aware transfer function highlights the 4-fold rotational symmetry detected in the Rubisco complex.

Abstract—Visualizing symmetric patterns in the data often helps the domain scientists make important observations and gain insights about the underlying experiment. Detecting symmetry in scalar fields is a nascent area of research and existing methods that detect symmetry are either not robust in the presence of noise or computationally costly. We propose a data structure called the augmented extremum graph and use it to design a novel symmetry detection method based on robust estimation of distances. The augmented extremum graph captures both topological and geometric information of the scalar field and enables robust and computationally efficient detection of symmetry. We apply the proposed method to detect symmetries in cryo-electron microscopy datasets and the experiments demonstrate that the algorithm is capable of detecting symmetry even in the presence of significant noise. We describe novel applications that use the detected symmetry to enhance visualization of scalar field data and facilitate their exploration.

Index Terms—Scalar field visualization, extremum graph, Morse decomposition, symmetry detection, data exploration.

1 Introduction

Several natural and man-made objects exhibit symmetry in different forms – reflectional symmetry of the wings of a butterfly, translational symmetry of the bricks in a wall, and rotational symmetry of the spokes of a wheel. In many disciplines, symmetric placement of objects results in optimal configuration – symmetric arrangement of atoms leads to stable low energy configuration of molecules and symmetric placement of weight bearing structures results in optimal load distribution. Hence, study of symmetry is important in different fields to understand various physical phenomena as well as to design, synthesize, and manufacture various materials and objects.

Computer generated models and data generated from experimental and computational methods are extensively used in many scientific disciplines. Visualizing the symmetry present in these data provide important cues to domain experts in understanding and characterizing the properties of the underlying phenomena. In this paper, we study the problem of detecting symmetric or repeating patterns from such data. Symmetry detection is an active area of research in geometry processing, computer graphics, image processing, and computer vision communities [27]. The symmetries detected are used for subsequent processing like segmentation, noise removal, beautification, feature recognition, and shape matching.

Symmetry detection in a scalar field refers to identifying regions that remain invariant under geometric transformations with respect to the geometry of the domain and the scalar values. For example, 4-fold rotational symmetry in the cryo-electron microscopy (cryo-EM) data of Rubisco RbcL8-RbcX2-8 complex (EMDB 1654) is shown on the left in Fig. 1. To detect symmetry at a reasonable computational cost, it is essential to use compact representations of the scalar field. Topological methods based on Morse theory have been successful in designing computationally efficient abstract representations like contour trees, Reeb graphs, and Morse-Smale complexes that aid in the study of scalar fields. The Morse-Smale complex is a partition of the domain into cells based on the gradient of the scalar field and has been used for segmentation [8, 39], meshing [10, 24], and studying various scientific phenomena such as turbulent structures, channel structures in porous material, and for terrain representation [6, 15, 23]. The extremum graph is a simplified representation of the Morse-Smale com-

- *Dilip Mathew Thomas is with Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. E-mail: dilip@csa.iisc.ernet.in.*
- *Vijay Natarajan is with Department of Computer Science and Automation, and Supercomputer Education Research Centre, Indian Institute of Science, Bangalore, India. E-mail: vijayn@csa.iisc.ernet.in.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

plex and was introduced for designing topological spines [9].

Graph-based representations of the scalar field like the contour tree and constellation of line features have been used in the past for symmetry detection [5, 35]. These methods assume that symmetry in the data manifests as similar subgraphs in the graph representation and employ subgraph matching algorithms to identify symmetry. However, they perform poorly when there is significant noise in the data resulting in noisy subgraphs that are no longer similar. Our method also uses a graph-based representation of the scalar field, namely the augmented extremum graph, but unlike earlier methods, we do not rely on subgraph matching for symmetry detection.

Augmenting topological constructs with geometric information results in powerful abstract representations as demonstrated recently in the study of pore networks [17, 36]. We propose to augment the extremum graph with geometric information and this integration of both topological and geometric information facilitates efficient detection of geometric symmetry. Our algorithm performs Morse decomposition of the data and selects a set of Morse cells as seed cells. The augmented extremum graph is traversed to compute distances from the seed cells to the remaining Morse cells and merge the seed cells into super-seeds. Eight seeds that merge to form four super-seeds are shown at the center in Fig. 1. The four super-seeds provide an initial estimate of the 4-fold symmetry in the data which is then expanded in a region growing stage. The 4-fold symmetry in the data thus identified is shown on the right. The main contributions of our paper are the following:

- A novel symmetry detection method that can detect symmetries in scalar fields with respect to rigid body transformations. We propose a data structure called the augmented extremum graph, which integrates topological and geometric information of the scalar field and use it for computationally efficient and geometry-aware symmetry detection.
- Our algorithm detects global symmetry even in the presence of significant noise. If the amount of noise is not high, our algorithm can detect partial symmetry as well.
- Existing techniques that detect scalar field symmetry using subgraph matching have limitations in handling noisy data [5, 35]. Our method addresses this shortcoming by using the augmented extremum graph only for distance computation and not relying on subgraph matching for symmetry detection.
- We demonstrate the effectiveness of our method through extensive experiments on several cryo-EM datasets and show that our method yields better results in the presence of noise. Thus, our method is able to bridge a gap in the applicability of previous methods to noisy datasets.
- We describe two novel applications that use our method for improving visualization of scalar field data. Additionally, we use our method to demonstrate some of the applications reported earlier in the literature.

2 Related Work

Symmetry is a well studied area in geometry processing and diverse techniques exist for its detection. Symmetry detection in scalar fields has received attention only in recent years and existing techniques are not as mature as techniques for detecting and computing symmetry in geometric objects.

2.1 Symmetry detection in shapes

Early work in symmetry detection explored the problem of exact symmetry detection [1, 2] and was later extended to approximate symmetry [40]. State of the art symmetry detection methods use a wide variety of techniques to detect approximate, partial, and intrinsic symmetries in shapes. Local symmetry information have been used to cast votes in a higher dimensional space where clusters of votes correspond to significant symmetries [25, 28, 31, 32, 38]. Graphs built from geometric features that are extracted from a shape have been studied to find similar subgraphs representing symmetric structures [3–5]. Shape descriptors based on spherical harmonics and Fourier methods have also been used to identify symmetry [19, 20, 26]. Readers may refer to the survey on symmetry in 3D geometry for more details [27].

2.2 Symmetry detection in scalar fields

Extensions of symmetry detection methods in shapes have been considered in the past for detecting symmetry in scalar fields. Kerber et al. [21] extend an earlier work [5] by extracting a set of line features from 3D scalar fields and model symmetry detection as a subgraph matching problem in the graph formed by these lines. However, the assumption that features in 3D scalar fields can be described using geometric line features is too restrictive in practice. Even for line features, this method cannot handle noise in the data robustly since the alignment of noisy networks of lines will be poor. Hong and Shen [18] extend an earlier work on reflective symmetry descriptors [19] and determine planes of reflection by assigning a score to each plane based on how well it acts as a plane of reflection. This method is computationally costly since the entire dataset has to be examined for each candidate plane. Our method, on the other hand, is efficient because it abstracts the scalar field using a small graph and avoids the search over the space of transformations. Moreover, their technique is restricted to identifying only reflective symmetries, whereas our method can detect symmetries under other rigid body transformations also.

The contour tree is a topological data structure that tracks changes in the topology of the level sets of a scalar field. Thomas and Natarajan [35] extract symmetry based on the assumption that regions with similar level set topology correspond to regions with similar scalar field distribution. They identify topologically similar regions by grouping together similar subtrees of the contour tree. The contour tree is stabilized in a preprocessing step to reduce the sensitivity of the method to noise. This method performs poorly when noise in the data destroys the repeating structure of the subtrees. Moreover, the method totally ignores the geometry of the underlying domain. Our method is aware of the underlying geometry and can handle noise in the data which results in wider applicability of the method. Global similarity between two given scalar fields has been studied in the context of shape matching [16, 42]. However these methods are not applicable to identifying symmetric regions within the same scalar field. Similarity between the level sets of a scalar field [7] and relationship between different scalar fields defined on the same domain [12, 29, 34] have also been studied for exploring and analysing volumetric data.

3 Background and Definitions

Consider a *scalar field* $f : \mathbb{M} \to \mathbb{R}$ defined on a simply connected domain $\mathbb{M}$. We define two simply connected regions $\mathbb{M}_1, \mathbb{M}_2 \subseteq \mathbb{M}$ to be *symmetric* with respect to f if there is a transformation T such that $\mathbb{M}_2 = T(\mathbb{M}_1)$ and $f(x) = f(T(x))$ for all $x \in \mathbb{M}_1$. Exact symmetry occurs rarely in practice. Therefore, robust symmetry detection methods address the more challenging problem of detecting symmetry in an approximate sense. We restrict our attention to rigid body transformations and consider piecewise-linear scalar fields defined on the vertices of a simplicial mesh and interpolated linearly over the elements of the mesh. For a vertex v, its *link* is defined as the mesh induced by the set of vertices adjacent to v and its *upper link* is defined as the mesh induced by the adjacent vertices having function value greater than that of v. For piecewise-linear domains, the critical points are characterized by the number of components in the upper link. A point $c \in \mathbb{M}$ is a *maximum* if its upper link is empty, a *minimum* if its upper link is same as its link and a *saddle* if the upper link consists of more than one connected component [11, 44].

A *Morse decomposition* is a segmentation of $\mathbb{M}$ where each maximum of f defines a segment, called *Morse cell*, as the union of paths of steepest gradient ascent that terminate at the maximum [13]. The Morse cells for the minima of f is defined analogously by considering $-f$ instead of f and the resulting segmentation is referred to as the Morse decomposition with respect to minima. Using the Morse decomposition, Correa et al. introduced extremum graphs for designing topological spines – a visual representation that captures both the geometry and the topology of the scalar field [9]. We compute an approximation of the Morse decomposition with respect to maxima by sorting all the vertices in the decreasing order of function values, assigning a label to each maximum, and propagating the labels to the adjacent vertices visited in the decreasing order of function values. Each

visited vertex inherits the label of the vertices in its upper link and the Morse cell associated with a maximum is the set of vertices with the label of the maximum. The Morse decomposition with respect to minima is computed analogously. If a vertex inherits multiple labels, then it belongs to the common boundary of the Morse cells associated with the labels. Maxima whose Morse cells share a common boundary are said to be adjacent. For each pair of adjacent maxima, the vertex with the highest function value on the common boundary is marked as their *shared saddle*. Note that due to degeneracies, such saddles need not necessarily exist between adjacent maxima in the true Morse decomposition. For this reason, we work with the approximate Morse decomposition described above and always introduce a shared saddle between neighboring Morse cells to capture their adjacency relationship. The nodes of a *maximum graph* are the set of maxima and their shared saddles. Edges in this graph link a pair of adjacent maxima to their shared saddle. The *minimum graph* is defined analogously. The *extremum graph* of a scalar field can refer to either the maximum graph or the minimum graph. Since symmetry identification with respect to the maximum graph and the minimum graph are analogous, henceforth we restrict our attention to the maximum graph. Fig. 2 shows a scalar field where the light gray patches correspond to regions with high function values. The maximum graph, on the left, shows adjacent maxima such as a and b connected to their shared saddle c.

Correa et al. note that the extremum graph captures the geometric proximity of the extrema much better than the contour tree. This is illustrated in Fig. 2, where the contour tree (shown partially) requires a path of five edges, *e-g-h-k-l-n*, to connect the geometrically close extrema e and n. On the other hand, the extremum graph connects them with a shorter and a more natural path *e-l-n* of length two. Also, the contour tree connects extrema a and m that are far apart with the path *a-c-h-k-m* of length four, which is shorter than the corresponding natural path *a-p-d-g-e-k-m* of length six in the extremum graph. Such unintuitive connections in contour trees fail to capture the proximity relationship between the extrema.

4 Augmented Extremum Graphs

While the abstract definition of the extremum graph captures minimal proximity information, it does not capture significant geometric information about the scalar field. The extremum graph can be viewed as a gradient flow graph, whose edges connect a shared saddle to its adjacent extrema along the path of steepest gradient ascent / descent. For a 2D scalar field, such a path in the domain can be embedded in the surface plot of the function and captures part of the geometric structure of the scalar field, see Fig. 3.

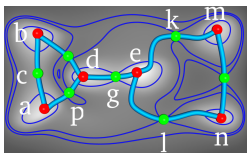
4.1 Geodesic distance between extrema

Geodesic distance between two points on a shape is an intrinsic property of the shape that remains invariant under isometric transformations and has been used for identifying symmetries of a shape [22,33]. In the rest of this paper, we use the phrase "path between two points" to refer to the path in the hypersurface plot of the scalar function. Our method requires the computation of the geodesic path between two extrema. Since computing the exact geodesic path is expensive, we propose to approximate it using the shortest path in the embedded extremum graph. However, the extremum graph consists only of edges between extrema and shared saddles and thus the shortest path between two non-adjacent extrema in the extremum graph deviates considerably from the true geodesic path. For example, consider the shortest path between the extrema c and f in the extremum graph in Fig. 3. For clarity, only a subset of the edges of the extremum graph is shown in blue. The shortest path consists of four edges - two edges that connect the extrema c and e and two edges that connect the extrema e and f through their respective shared saddles. Depending on the height of the intermediate extremum e, the length of this path can be arbitrarily different from the true geodesic path because the shortest path in the extremum graph is forced to pass through e whereas the geodesic path may bypass the extremum. To overcome this problem, we augment the extremum graph by inserting additional edges that directly connect the shared saddles of an extremum as shown by the magenta edge that bypasses the extremum e. Thus, inserting edges between the shared saddles allows extrema corresponding to noise to be bypassed. This also makes the geodesic path estimate more robust. Assume that due to noise, a shared saddle does not exist between the extrema g and l and the shortest path between g and l in the extremum graph is *g-h-i-m-l*. The path has to traverse through all the intermediate extrema, in this case the extremum i, which distorts the shortest path estimate. By directly connecting the intermediate shared saddles, shown by the magenta edge *hm*, we obtain a better estimate for the shortest path. This path is not significantly different from the true geodesic path between g and l. On the other hand, if a shared saddle did exist between g and l and the path between g and l through this shared saddle was the shortest, then our method will detect and use this path.

We also note that explicit computation of the gradient lines is required to embed the extremum graph, whereas an abstract representation of the extremum graph can be easily obtained once the shared saddles and their adjacent extrema are determined. Hence, instead of computing the length of the edges in the embedded extremum graph, we assign to each edge a weight equal to the approximate length of the geodesic path between the endpoints of the edge. Though we augment edges between the shared saddles for distance computation, we do not explicitly insert these edges in the extremum graph. Instead, we perform all computations on the extremum graph and infer the augmented information on-the-fly during the path computation. While all illustrations below of the extremum graph are for 2D scalar fields, the discussion holds for higher dimensions also.

4.2 Symmetry from distances

We observe that for two regions that are symmetric, the distance between a pair of extrema in one region is equal to the distance between their symmetric counterparts in the second region. Consider the paths from symmetric mountains c and l in Fig. 3 to the neighboring symmetric mountains f and g, respectively. Although the two paths differ significantly, the geodesic distance estimate is roughly the same. The noisy hill e is bypassed and the absence of the shared saddle between g and l is compensated for by directly connecting the saddles h and m. The distance between extrema is estimated robustly and similar distances is a good indicator of symmetric regions. We assume that different symmetric regions are well separated in terms of distances in the hypersurface plot of the scalar field while features within a symmetric region lie in close proximity to each other. This can be seen in



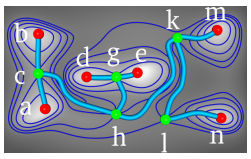


Fig. 2. Extremum graphs represents the geometric structure of scalar fields better than contour trees. (left) The extremum graph for a scalar field connects the adjacent extrema through their shared saddle and represents proximity information better than (right) the contour tree.

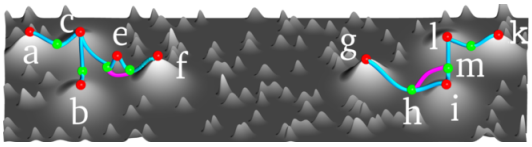


Fig. 3. The geodesic distance between an extremum and its shared saddle is approximated as the length of the corresponding edge in the embedded extremum graph. To approximate the shortest geodesic path from c to f, the intermediate extremum e is bypassed by directly connecting its shared saddles with the magenta edge. Distance between the pair of extrema c and f is similar to the distance between its symmetric pair l and g.

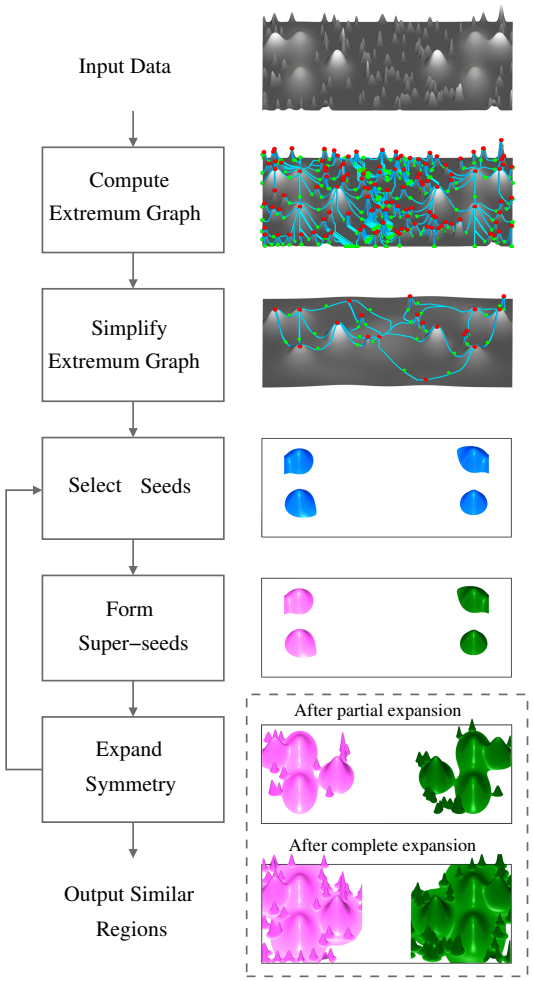


Fig. 4. Symmetry detection pipeline. The extremum graph of the input data is computed and simplified. A set of extrema are then selected as seeds and the augmented graph is traversed starting from the seeds. During the traversal, seeds merge to form symmetric super-seeds. Finally, the symmetry of the super-seeds is expanded in a region growing stage and the symmetric regions are reported.

Fig. 3 where the path between a mountain on the left and one on the right is longer than the path between a pair of adjacent mountains both of which lie on the same side. We exploit this separation between symmetric regions and remove long paths from the augmented extremum graph to obtain a set of disconnected subgraphs. These graph cuts partition the domain into different symmetric regions. In Fig. 3, the region on the left and the region on the right are identified as a pair of symmetric regions.

5 Extremum Graph and Symmetry Detection

Our symmetry detection pipeline is shown in Fig. 4. Given a scalar field as input, we first compute its Morse decomposition and the associated extremum graph. The resulting Morse decomposition may be over-segmented. We coarsen it and generate a simplified extremum graph. Next, we classify the Morse cells into different groups based on similarity of their histograms. The extrema of the cells that belong to one such group are selected as a seed set. These pre-processing steps are described in Section 5.1 and Section 5.2. Starting from each seed, the augmented extremum graph is traversed iteratively by visiting, at each iteration, the extremum that is closest to the seed. When the traversal from a seed reaches another seed, the two seeds are declared to be merged. Once all the seeds merge, we use the distance between pairs of seeds to cluster them into super-seeds. Augmented extremum graph traversal and super-seed formation is explained in Section 5.3 and Section 5.4. The Morse cells corresponding to the seeds that belong to a common super-seed are the initial estimates of the symmetric regions. The symmetry is expanded in a region growing stage by merging cells in the neighborhood of the seeds. For detecting different partial symmetries, the above procedure is repeated with a different seed set. These post-processing steps are described in Section 5.5.

5.1 Simplification of the extremum graph

We compute the extremum graph as described in Section 3 and simplify it for computationally efficient traversal. Our simplification procedure coarsens the initial Morse decomposition by merging a Morse cell into its adjacent cell based on two parameters – persistence and size. For an edge between the extremum of a cell i and the saddle shared with an adjacent cell j, if its persistence [14], which is the difference in function values between the extremum and the saddle, falls below a threshold, then i is merged into j. Similarly, if the size of a Morse cell, measured in terms of the number of vertices that lie within the cell, falls below a threshold then it is merged with an adjacent cell.

When a cell i merges into a cell j, the extremum of i, and all its shared saddles and edges are removed from the graph. The shared boundary of j also changes and the shared saddles are updated / created and edges are inserted between the shared saddles and the adjacent extrema. A saddle that exists after simplification is called a *surviving saddle* and a saddle removed during simplification is called a *cancelled saddle*. Since computing the length of geodesic paths encountered during the graph traversal is expensive, we approximate it as the sum of Euclidean distances of the edges in the path. For 3D scalar fields, we normalize the spatial coordinates and the scalar values of the input to lie within the unit hypercube in $\mathbb{R}^4$ and embed the abstract extremum graph in $\mathbb{R}^4$. The first three coordinates of a node are the spatial coordinates and the fourth coordinate is the scalar value of the corresponding critical point. For simplicity, we use uniform weights for normalization. Any normalization procedure may be used as long as it preserves the hierarchical relationship in the merging of the seeds.

Simplification removes many of the shared saddles and a surviving saddle may lie arbitrarily far away from the extrema that it is adjacent to after simplification. Directly using the Euclidean approximation for computing the length of a path in the simplified graph will lead to significant errors. Therefore, the cancelled saddles and the edges removed during simplification are taken into account to estimate the length of paths in the simplified graph. An edge between a surviving shared saddle and an extremum is the result of zero or more simplifications. The shared saddle of the extremum in the unsimplified graph that lies on the path to the surviving saddle is called the *originating saddle*. To keep track of edges removed during simplification, each surviving saddle stores the originating saddles of the extrema that it is adjacent to and the length of the path between the originating saddles. Consider the extremum graph shown on the left in Fig. 5. During simplification, let the cell u merge with v, cancelling the saddle n and the cell x merge with c, cancelling the saddle r as shown in the middle figure. The surviving saddle m is adjacent to a and v and stores their originating saddles, m and n. Similarly, o stores the originating saddles of v and c, namely o and r. To ensure that distances are calculated correctly, each surviving saddle also stores the distance between its two originating saddles. Before simplification, this distance stored at each saddle is zero. When a saddle is cancelled, the distance stored at

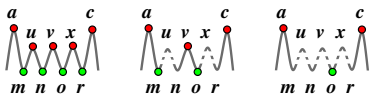


Fig. 5. Bookkeeping of distance during simplification. (left) An extremum graph where the extrema are shown in red and the shared saddles in green. (center) During simplification, u merges into v and x merges into c. The surviving saddle m stores the length of the path between m and n. Similarly, o stores the length of the path between o and r. (right) When m is cancelled, o stores the length of the path between m and r.

the surviving saddle is updated to equal the sum of the distance stored at the cancelled saddle, the distance stored at the surviving saddle, and the length of the edge connecting the two originating saddles of the cell that gets merged. Before n is cancelled, the distance stored by the surviving saddle m and the cancelled saddle n is zero. After cancelling n, the distance stored by m is the length of the edge m-n that bypasses the extremum u, calculated as the Euclidean distance between m and n. Similarly, after r is cancelled, the surviving saddle o stores the distance between o and r. Eventually, after m is cancelled, the surviving saddle o stores m and r as its originating saddles and the distance stored at o is equal to the sum of the distance stored at o (length of the path o-r), the distance stored at the cancelled saddle m (the length of the path m-n), and the distance of the edge no, which connects the originating saddles of v. Thus the evaluation of distances remains the same in the unsimplified graph and the simplified graph.

To determine the simplification threshold, we plot the drop in the number of critical points against the simplification parameter. For the dataset shown in Fig. 1, the plot for simplification with respect to persistence is shown on the left in Fig. 6. A similar plot is obtained for simplification with respect to the size of the Morse cells. In both cases, we see that initially there is a sudden drop in the number of critical points due to noise in the data. We set the thresholds to the value at which the drop in the number of critical points stabilizes, which can be identified as the beginning of the horizontal section in the plot. Later stages of the pipeline and the results are not sensitive to the exact value of the threshold chosen because the bookkeeping procedure described above ensures that the path length computed is not affected by the simplification. The main purpose of simplification is to improve efficiency of graph traversal by removing spurious critical points introduced by noise in the data. A simplification threshold of 1% of the maximum persistence or size suffices in most cases.

5.2 Seed set selection

Selection of good seed sets is crucial for meaningful symmetries to be detected by our algorithm. Ideally, seeds should be chosen such that they are symmetric and representative of the symmetry in the input. Automatic selection of meaningful features from a dataset is a challenging problem and though solutions may be designed for specific cases, a generic approach that works across datasets is not known. Hence, we do not attempt to find a generic solution to the related problem of automatic seed selection. For typical cryo-EM datasets, the histogram of the scalar values of symmetric seeds cells are very similar. We use a heuristic procedure based on histogram similarity for selecting seed sets.

We initially choose a Morse cell based on its attributes like size, persistence, and the function value of the extrema. In all our experiments, we select the Morse cell that contains the extrema with the highest function value. We then compute the histogram of each Morse cell by uniformly dividing the range of function values into thirty bins. The histogram is then normalized and the Earth Mover's Distance (EMD) between the histograms of the chosen Morse cell and all other cells is computed. Those cells whose EMD falls below a threshold together with the initially chosen cell form a seed set. To determine the threshold, we plot the number of seeds added to the seed set as the EMD threshold is increased, see the plot on the right in Fig. 6. For clarity of illustration, we show only the initial part of the plot. Initially, the number of seeds increase even for a very small increase in the EMD threshold (close to zero in the plot) as shown by the leftmost horizontal section in the plot. Further increase in the number of seeds occurs only after a significant increase in the EMD threshold as indicated by the vertical section in the plot. We can identify a meaningful seed set by setting the threshold to the value just before this jump in the EMD threshold. For the dataset shown in Fig. 1, eight seeds are identified using this procedure. For cryo-EM datasets that we use in our experiments, we have empirically determined that the threshold of 0.9 results in a good selection of seed sets in many cases.

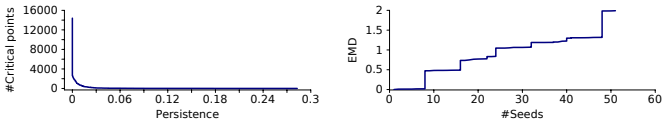


Fig. 6. (left) Plot of the number of critical points with increasing values of the simplification parameter. The value at which the drop in the number of critical points stabilizes is used as the threshold for simplification. (right) Plot of the increase in the number of seeds with increase in EMD threshold. Initially, the EMD of the seeds is nearly zero after which there is a significant increase in the EMD. The value just before the sudden increase in EMD can be used to identify seed sets.

The histogram matching procedure is a heuristic for selecting seeds and may not always give the desired result. We employ the following semi-automatic procedure for selecting seeds which performs well in practice and does not require significant effort from the user. The selection procedure uses the toporrery layout, which is in turn based on the branch decomposition representation of the contour tree, and provides a powerful user interface for exploring and selecting features from scalar field datasets [30]. A user identifies a seed (extremum) of interest either based on its attributes (such as persistence) or visual examination of the features in the data. The branches in the toporrery layout corresponding to the extrema with similar function values are then automatically highlighted. The user examines the geometry of the level sets of the highlighted branches. The user needs to examine only a few level sets in the neighborhood of the scalar value at the extrema. The user then selects a subset of the branches that exhibit symmetry in the evolution of the level sets. The extrema of the branches selected by the user are chosen as the seed set. The above procedure automatically prunes away a majority of branches in the branch decomposition layout and limits user interaction to selecting a subset of branches from the remaining small set of branches. An illustration of this procedure is shown in the supplemental material.

5.3 Augmented extremum graph traversal

Given an initial set of seeds, an efficient procedure for the augmented extremum graph traversal is employed both to form super-seeds and to expand the initial estimate of symmetry through region growing. Each seed is marked as a source vertex and traversals are initiated simultaneously and independently from each source vertex, as shown in Algorithm 1. Each traversal proceeds iteratively, where at each step, the saddle that is closest to the seed is processed. Only the source and destination extrema are contained in a path and the rest of the extrema are bypassed. To store distances from multiple seeds, each Morse cell maintains an array of distances. Similarly, each saddle maintains an array that indicates if the saddle has been visited from a seed.

Consider the surface plot of a scalar function as shown in Fig. 7. Let the mountain peaks a, b, c, and d be chosen as seeds. Initially, the edges from each seed to its adjacent shared saddles, shown on the left, are inserted into a common priority queue. The cost of each edge is the distance from the seed to the shared saddle. After this initialisation step, the edge with the shortest distance, say ci, is popped from the queue. Next, the distance from the source seed c to the extremum e adjacent to the endpoint saddle i in the path is computed as the sum of the

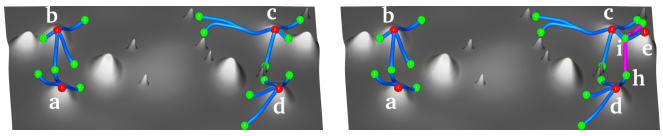


Fig. 7. Augmented extremum graph traversal. (left) Blue edges from the seeds a, b, c, and d to their shared saddles are inserted into a priority queue. (right) The shortest path c-i is popped out and extended to the adjacent saddles by adding the magenta edges. When the path c-i-h is popped out, the traversal from the seed c reaches the seed d and the two seeds merge.

```
foreach seed do                                   // initialise pq
    j = cell_id(seed);
    foreach cell_id k adjacent to j do
        path = (j,k, sad_{j,k});
        path.len = len(e_j, sad_{j,k}));
        pq.push(path);
    end
end
while not all seeds merged do                     // traverse
    p = pq.pop();
    if not visited(p.src, p.sad) then
        visited(p.src, p.sad) = true;
        foreach cell_id i adjacent to p.des do    // extend
            ext_path = (p.src, i, sad_{p.des,i});
            ext_path.len = p.len+len(p.sad, sad_{p.des,i}));
            pq.push(ext_path);
        end
        des_len = p.len + len(p.sad, e_{p.des});
        if dist(p.src, p.des) > des_len then
            dist(p.src, p.des) = des_len          // update;
            if isseed(p.des) then                 // merge seeds
                merge(p.src, p.des, des_len);
            end
        end
    end
end
```

Algorithm 1: Augmented extremum graph traversal algorithm. For a Morse cell i, its extremum is denoted by e_i. The shared saddle between i and j is denoted by $sad_{i,j}$. A path is a 3-tuple (*src*, *des*, *sad*), where *src* is the source seed and *des* is the cell adjacent to *sad*, the saddle at the end of the path.

length of the edge *ei* and the length of the edge *ci* which was popped out, see the figure on the right. The popped path *c*-*i* is extended to the unvisited shared saddles of the adjacent extremum *e*. When the path is extended to *h*, the extremum *e* is bypassed and the shared saddles are directly connected. The length of the extended path is calculated as the sum of the length of the saddle-saddle direct edge and the length of the path popped from the queue. The extended path is inserted into the queue and the traversal proceeds to visit other saddles in the order of increasing distance from the seeds. When a saddle is visited from a seed, the distance from the seed to the adjacent extremum is updated and is later used during the region growing stage. When the path *c*-*i*-*h* is eventually popped out of the queue, the path from the source seed *c* reaches the seed *d* and the two seeds merge. The traversal continues until all the seeds merge.

5.4 Super-seed formation

Studies on the way humans perceive patterns, like Gestalt theory [41], have shown that items that are located spatially near each other are perceived to be part of a group. Repeating structures in close proximity are grouped together and perceived to be part of bigger patterns. Thus for symmetry detection, it is important to identify the formation and repetition of such bigger patterns formed from smaller patterns. In our context, seeds that are closer merge first and form a hierarchy of bigger patterns. A super-seed is the set of seeds that merge to form the biggest such repeating pattern. The above description of the way humans perceive patterns also explains our assumption on distances between symmetric regions as stated in Section 4.2. When the path from a source seed reaches another seed, the two seeds merge and the distance between the two seeds is the length of the shortest path between the two seeds. At the leaf level of the hierarchy, all the seeds, *a*, *b*, *c*, and *d*, in Fig. 7, are separate repeating units. During the traversal, *a* merges with *b* and similarly *c* merges with *d*. These merged components form the next level in the hierarchy and they further merge into a single component at the root level. A graph can be constructed by

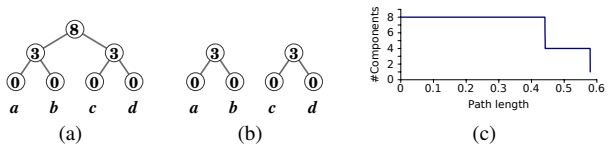


Fig. 8. Seed-merge tree and super-seed identification. (a) Labels on the interior nodes of the seed-merge tree is equal to the distance at which its children nodes merge. (b) Nodes with values above a threshold are removed from the tree and the seeds that remain connected belong to a common super-seed. (c) Plot of the number of components against the path length. The threshold for disconnecting the seed-merge tree is chosen from the rightmost horizontal interval, just before the number of components drop to one.

inserting a node for each seed and a weighted edge between two seeds that have merged. The edge weight is equal to the distance between the two seeds. After all the seeds merge, a graph cut that removes edges with high weights is used to identify the super-seeds.

Our graph cut algorithm uses a binary tree, called *seed-merge tree*, that represents the merging of seeds. Each of the seeds *a*, *b*, *c*, and *d* is a leaf node of the seed-merge tree and represents a component. When the components corresponding to two nodes merge, an interior parent node is created to represent merging of the components. Each interior node is assigned a weight equal to the distance at which the corresponding components merge in the augmented extremum graph traversal, see Fig. 8(a). All leaf nodes are assigned zero weight. Once all the seeds merge and the complete tree is constructed, we disconnect it into different subtrees by removing nodes with weight above a threshold. Thus we disconnect the seeds that merge during the augmented extremum graph traversal by means of paths whose length is above the threshold and the seeds that remain connected are identified as super-seeds. Fig. 8(b) shows two disconnected subtrees formed when the threshold is set to eight. One of the subtrees corresponds to the super-seed comprising of *a* and *b* and the other subtree corresponds to the super-seed comprising of *c* and *d*.

To determine a suitable value for this threshold, we plot the decrease in the number of components with respect to increase in the path length. Fig. 8(c) shows this plot for the dataset in Fig. 1. For this dataset, eight seeds were chosen and each of the vertical sections in the step-like plot show the sudden drop in the number of components due to the merging of the components. The merging happens at different scales and reflects the hierarchical relationship in the merging of the seeds. The number of components at the start of the horizontal sections in the plot remains unchanged till the end of the interval and shows the stability of the components in this section. The super-seeds are the largest stable components formed just before all the seeds merge into a single component. Hence, the threshold for identifying the super-seeds is chosen as any value that lies in the last horizontal section in the plot just before the number of components drop to one. We identify four super-seeds for this dataset and these are shown using four distinct colors as shown in the middle figure in Fig. 1. As

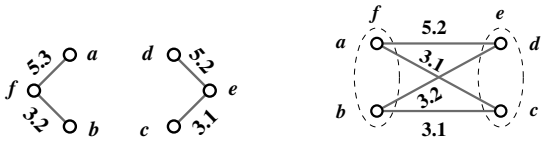


Fig. 9. (left) The distance from the candidate cells f and e to the seeds a, b and c, d are shown by the labels on the edges. (right) An edge connecting two seeds in the bipartite graph is weighted with the minimum of the distances to the seeds. The sum of weights of matching edges ad and bc, 8.3, is close to the average of the distances to the seeds, (5.3+3.2+5.2+3.1)/2 = 8.4 and hence f and e are considered to be symmetric.

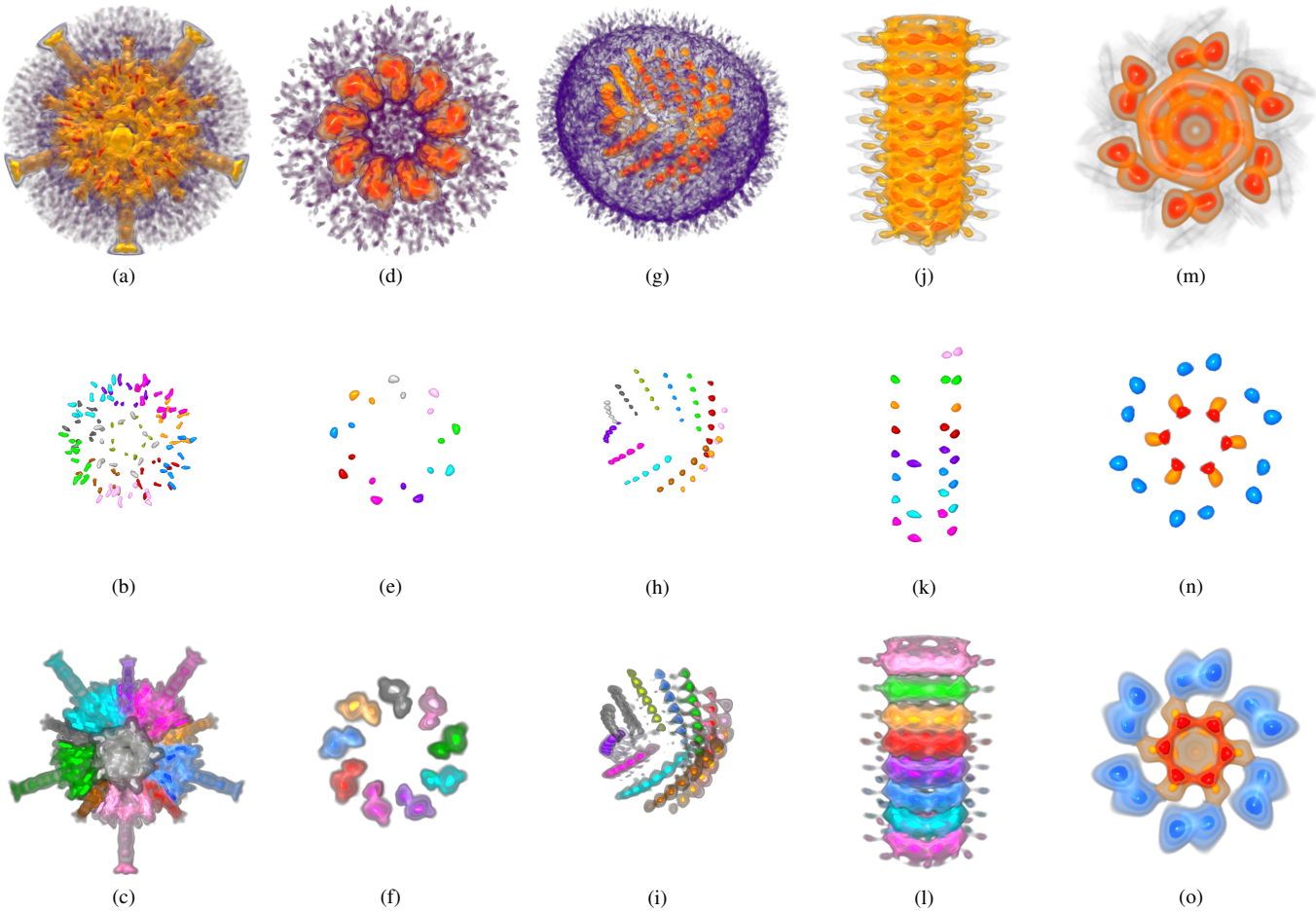


Fig. 10. Results of symmetry analysis on cryo-EM datasets. Five datasets - EMDB 1179, 1603, 5331, 2094, and 1706 are shown in the first row. EMDB 1179, 1603, and 5331 contain significant noise as shown by the violet regions in the volume rendered images. The seeds selected are shown in the second row. The extracted symmetries are shown in the third row. Symmetries are extracted even in the presence of noise. Columns 1-4 show dodecahedral, 9-fold rotational, screw, and translational symmetries and the rightmost column shows partial symmetry extraction.

described in Section 4.2, we assume that the distance at which the seeds within a super-seed merge is significantly lower compared to the distance at which seeds merge across super-seeds. This separation of distances will manifest as a distinct horizontal section in the plot used to identify the super-seeds and can be easily determined.

5.5 Symmetry expansion

Once the super-seeds are formed, its symmetry is expanded by inserting neighboring Morse cells through a region growing procedure. For detecting partial symmetry, when a Morse cell is considered as a candidate to be inserted into its closest super-seed, we ensure that a symmetric candidate exists in the remaining super-seeds. Consider candidate cells f and e and the distances to the seeds that constitute their closest super-seeds, $\{a, b\}$ and $\{c, d\}$ as shown on the left in Fig. 9. If f and e are symmetric, then the distance from f to a seed in its closest super-seed, say a, will be nearly equal to the distance from e to the corresponding symmetric seed, in this case d, and the minimum of the two distances will be nearly equal to the average of the distances. When the distances from f to each seed in its super-seed satisfy this property, we consider the distribution of distances from f and e to be similar and treat them to be symmetric. To determine this we construct a complete bipartite graph as shown on the right in Fig. 9, where the nodes in each partition are the seeds that constitute the closest super-seed and an edge between two seeds is weighted with the minimum of the distances from f and e to the respective seeds. Next, we compute a maximum weight matching and if the weight of the matching is close to the average of the distances, we consider f and e to be symmetric. Based on empirical results, we set this threshold to 80% of the average distance to the seeds. The ideal threshold depends on the noise in the data and selecting the threshold involves a trade-off between tolerance to noise and quality of the detected symmetry. If each super-seed has a symmetric candidate, then these candidates are inserted into the symmetric region of their super-seeds and the regions are expanded. The procedure is repeated till all candidate cells visited during the graph traversal are considered. For noisy data, the farther we traverse away from the seed cells, the distance estimate accumulates errors as noise causes variations in the shortest path estimate. This makes it difficult to compare distance distribution for detecting partial symmetry. However, for identifying global symmetry, the symmetry verification at each step can be avoided since, by definition, each candidate cell has a symmetric counterpart in all the super-seeds. For global symmetry, we continue the graph traversal, even after the seeds merge, till all the cells are visited by at least one of the seeds. The symmetric region corresponding to a super-seed is then reported as the set of all cells that are closest to the super-seed.

6 Results and Discussion

We now present experimental results of our symmetry detection algorithm run on different cryo-EM datasets.

Global and partial symmetry. The first row in Fig. 10 shows volume rendered images of five cryo-EM datasets. The volume rendering shows the lower function values in dark violet color and they correspond to noise in the data. The first four columns show global symmetry extracted by our algorithm and the fifth column shows an example

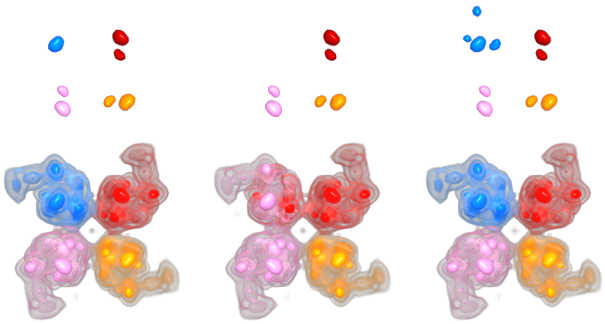


Fig. 11. Sensitivity to seed set selection. (left) When one seed is dropped, the symmetry detected is unaffected. (center) When the blue super-seed is dropped by removing both of its seeds, the Morse cells which were closest to the blue super-seed are reassigned to the remaining super-seeds. (right) When two new seeds are inserted, the symmetry detected is again unaffected.

of different partial symmetries extracted. For all datasets, except the first, seed selection was done based on histogram matching.

Fig. 10(a) shows a dataset with dodecahedral symmetry detected using 120 seeds. The Morse cells corresponding to the seeds show considerable variation in their geometry and as a result histogram comparison is not meaningful in this case. Hence, seeds were selected semi-automatically based on the procedure described in Section 5.2. We choose 120 maxima with the highest function values and the selected seeds merge to form twelve super-seeds. The seeds belonging to a common super-seed are shown with the same color in Fig. 10(b). The extracted symmetric regions after region growing are shown in Fig. 10(c). Though only ten small Morse cells are selected per symmetric region, the region growing stage correctly identifies symmetry of the remaining Morse cells even though some of these cells in the long fibre-like structures are far away from the seeds at the base. The second column shows 9-fold rotational symmetry detected by our algorithm after selecting eighteen seeds that merge to form nine super-seeds. The third column shows a dataset with twelve repeating strand-like structures arranged helically and our algorithm detects screw symmetry of the repeating strands. Fig. 1 shows another example where our algorithm is able to detect 4-fold rotational symmetry in the data using eight seeds which merge to form four super-seeds. The volume rendering of all these datasets indicate that there is significant amount of noise in the data. However, due to the robustness of the technique we are able to identify the symmetry present in the data. The fourth column shows a dataset with translational symmetry detected by the algorithm. Three different partial symmetries extracted from the dataset in Fig. 10(m) are shown in Fig. 10(o) as orange, blue, and red segments in a symmetry-aware segmentation. The seed sets used in each case is shown in Fig. 10(n) with the same color as the color of the symmetry-aware segment.

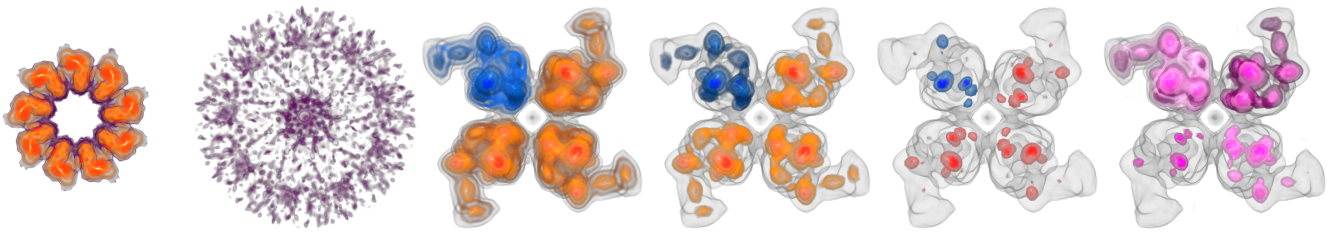
Sensitivity to seed selection. Ideally, seeds should be chosen such that they are symmetric and representative of the symmetry in the input. However, our method can tolerate asymmetric distribution of seeds among the symmetric regions. Consider an ideal set of seeds and the assignment of each Morse cell to the seed that is closest to it. Erroneous omission of a seed from the ideal set affects the assignment of only those cells that were originally assigned to the omitted seed. Cells in the neighborhood of the omitted seed are reassigned to the remaining seeds while other cells are unaffected. Similarly, insertion of a new seed into the ideal set affects the assignment of only those Morse cells that belong to the neighborhood of the new seed. Also note that the distance between a pair of seeds is calculated independent of the remaining seeds. Thus, deletion or insertion of seeds causes changes only in the local neighborhood of the seeds. We illustrate this for the dataset shown in Fig. 1 by modifying the ideal set of two seeds per super-seed. When one seed is dropped, the blue super-seed is formed with only one seed. However, the symmetric regions detected are unaffected as shown in the left column of Fig. 11 since many of the Morse cells that were closest to the dropped seed are now closest to the remaining blue seed. In the extreme case, when two seeds are dropped resulting in only three super-seeds, the Morse cells that were assigned to the blue seeds are reassigned to the remaining closest seeds - pink and red as shown in the middle column. Note that the symmetric region corresponding to the orange super-seed is unaffected since dropping of seeds only causes local changes to the detected symmetry. When two new seeds are inserted, the blue super-seed consists of four seeds as shown in the right column. The Morse cells assigned to the other super-seeds are unaffected and thus the symmetric regions detected are again not affected. For each of these cases, the plot used to identify the super-seeds is similar to the plot shown in Fig. 8(c) and is included in the supplemental material. The fourth column in Fig. 10 shows another example where each symmetric region is a ring with 9-fold rotational symmetry. Ideally, nine seeds from each ring should have been selected as the seed set. However, due to variations in the histogram, the algorithm selects two, three, or four seeds from the rings as shown in Fig. 10(k). Though the seed distribution is inconsistent, symmetry can be detected as long as the seeds chosen are such that the distance between seeds belonging to different symmetric regions is higher compared to the distance between seeds within the same symmetric region. This ensures that seeds within a symmetric region merge first and form super-seeds that represent the symmetry in the data. Eight super-seeds are formed in this case corresponding to the translational symmetry and the symmetric regions are correctly identified as shown in Fig. 10(l). Results on more datasets are shown in the supplemental material.

Comparison with the contour tree method. Our symmetry detection method uses Morse decomposition, which is based on gradient flow topology as compared to the approach that uses the contour tree [35], which is based on level set topology. The symmetric segments identified by the two methods may be different and it is not meaningful to compare the results of the two methods. So, we qualitatively evaluate the two methods and list the pros and cons of both approaches.

The contour tree based method detects symmetry by identifying similar subtrees from the branch decomposition representation of the contour tree. Since branches corresponding to noise in the data can destroy the similarity of the subtrees, this method requires the removal of such branches. For this purpose, the branch decomposition is simplified by removing low persistence branches under the assumption that only low amplitude noise exists in the data. Though our method also simplifies the extremum graph for computational efficiency, in contrast to the assumption made by the contour tree based method, we do not necessarily require that the noise is removed through the simplification step. This is because the presence of noise does not affect the distance calculation and hence our method can handle noise of larger amplitude in the data. Moreover, our method incorporates geometric information for more effective symmetry detection and also uses a region growing procedure to identify the largest symmetric region. In comparison, the contour tree based method ignores geometric information and does not necessarily identify the largest symmetric region. We illustrate these advantages in the supplemental material with a real-world dataset.

One of the major limitations of our method is that the symmetry detected depends on the choice of seed sets used and selection of seed sets may require user interaction unlike the contour tree method. The contour tree method is well suited for identifying partial symmetry as opposed to our method which requires different seed sets to be identified for each partial symmetry in the domain. The contour tree method also has the advantage that it can detect symmetry at multiple scales since the branch decomposition representation induces a natural hierarchy on the branches. The supplemental material shows additional results of symmetry detected by our results on datasets used in earlier work [35]. It can be seen that our method is limited to detecting symmetries at the largest scale whereas the contour tree method can detect different partial symmetries as well as symmetries at different scales.

Performance. Table 1 reports the running time of our algorithm for the datasets shown in Fig. 10. The time taken for building the graph includes computation of the Morse decomposition as well as the simplification of the initial extremum graph constructed. Earth Mover's



(a) (b) (c) (d) (e) (f)

Fig. 12. (a) The seed cells for the dataset in Fig. 10(d) together with the Morse cells that lie in its proximity are extracted. The volume can also be cropped automatically to include only the selected features and this reduces the size of the volume from $160 \times 160 \times 160$ to $92 \times 93 \times 50$. (b) The remaining Morse cells which correspond to noise in the data. The transfer function used in (a) and (b) is identical to the one in Fig. 10(d). (c)-(e) Two internal layers are peeled off from the segment shown in blue to reveal the features in the interior and the corresponding actions are automatically performed on the other three symmetric segments. (f) The symmetric regions are rendered differently by peeling different internal layers and shows complementary information in a single view.

Table 1. Running time, measured in seconds, for various steps in the symmetry detection pipeline. All experiments were performed on a 2 GHz Intel Xeon processor with 8GB RAM.

EMDB dataset#	#vertices	build graph	EMD computation	traverse graph
2094	100^3	5.4	1.5	3.2
1654	112^3	7.6	3.3	1.1
1706	130^3	8.5	0.04	0.5
1603	160^3	34.2	8.9	45.4
5331	240^3	106.5	19.8	77.6
1179	255^3	106.9	-	409.1

Distance computation is fairly fast in practice. When the number of seeds is large, graph traversal dominates the computational cost of the algorithm since the saddles are visited multiple times by the graph traversal initiated from each seed. In the current implementation, the weights of the edges are computed on the fly during the graph traversal. The augmented edges that directly connect the shared saddles are not present in the extremum graph and their weights have to be computed during the traversal. This additional computation increases the time spent for the graph traversal. We believe a more efficient implementation of the algorithm can significantly improve the running time. The running time does not include time for I/O.

7 Applications

We describe two novel applications of our method for enhancing visualization of scalar field data. We also demonstrate some of the existing applications that use symmetry information for better visualization.

Proximity-aware volume visualization. Distances from the seed cells to the remaining Morse cells computed during the augmented extremum graph traversal can be used for proximity-aware selection and visualization of features. A user can specify the proximity he is interested in and only those Morse cells that satisfy the proximity criteria, measured in terms of the distance from the seed cells will be selected and extracted. Thus a given set of features that the user deems as important (seed cells) can be visualized together with only those features that lie in its proximity. We use this technique for separating the features from noise as shown in Fig. 12(a) and Fig. 12(b).

Linked volume editing. Exploration of features in 3D scalar field data often involves significant effort from the user to interact with the volume and focus on the features of interest. Symmetry-aware segmentation obtained by our method can offer considerable assistance to users in performing such time consuming interactions. User can interact with one of the segments by applying different volume editing operations and the same operations can be automatically applied to the remaining segments. We show an illustration of this technique in the context of peeling features in a volume as shown in Fig. 12(c)-(e).

Symmetry-aware transfer function design. Features in a volume that lie within the same range of function values can be highlighted differently using spatially-aware transfer functions [37,43]. Such transfer functions can be made symmetry-aware and enables selective visualization and hiding of features in the volume [35]. Symmetry-aware transfer functions have been used in the bottom row of Fig. 10 to highlight the symmetric segments extracted by our method.

Multi-mode volume rendering. Symmetry information identified by our method helps in presenting complementary information from different symmetric segments in a single view by rendering each segment differently [18] as shown in Fig. 12(f).

Symmetry-aware query selection. Segmentation of the volume into its symmetric parts facilitates query-driven selection of features. User can query for features similar to a selected feature and the symmetric features identified can then be presented to the user.

8 Conclusions

In this paper, we present an integrated geometric and topological approach for detecting symmetric regions in a scalar field. We believe that our method is a significant improvement over existing methods since it can robustly detect symmetry even in the presence of significant noise. The proposed method is computationally efficient. We show through experiments that our algorithm can detect symmetry under different types of transformations in real-world datasets.

Perhaps the biggest limitation of our method is that the symmetry detection critically depends on the selection of a meaningful set of seeds. A bad selection of seeds may lead to incorrect formation of super-seeds, which in turn affects the quality of the detected symmetries. Though we describe the method used for seed selection, a robust and widely applicable method for automatic selection of seeds remains an open problem. Another limitation is that the simplification procedure is not symmetry-aware. An interesting open problem is to design a simplification method that ensures that the simplified Morse cells are appropriately distributed among the symmetric regions.

The limitations of our technique primarily arise from using only local information about symmetry. We believe that similar to methods that detect symmetry in geometric shapes [25, 28, 31, 32, 38], scalar field symmetry detection methods will also benefit from a clustering based analysis. Such an approach will help obtain more global information about the symmetry and may also lead to symmetry detection methods that are insensitive to missing regions and imperfections in the symmetry. We also believe that extraction of symmetry information will lead to new methods and tools that aid users in visualization and data analysis in future.

Acknowledgments

This work was supported by a grant from Department of Science and Technology, India (SR/S3/EECE/0086/2012) and by the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science. Travel was supported by IBM travel grant. Volume rendered images used in the paper were generated using Voreen (www.voreen.org).

REFERENCES

[1] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete Comput. Geom.*, 3(3):237–256, Jan. 1988.

[2] M. J. Atallah. On symmetry detection. *IEEE Trans. Computers*, 34(7):663–666, 1985.

[3] A. Berner, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel. A graph-based approach to symmetry detection. In *Proc. Symposium on Volume and Point-Based Graphics*, pages 1–8, 2008.

[4] A. Berner, M. Wand, N. Mitra, D. Mewes, and H. Seidel. Shape analysis with subspace symmetries. *Computer Graphics Forum*, 30(2):277–286, 2011.

[5] M. Bokeloh, A. Berner, M. Wand, H. Seidel, and A. Schilling. Symmetry detection using feature lines. *Computer Graphics Forum*, 28(2):697–706, 2009.

[6] P. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Trans. Vis. Comput. Graph.*, 10(4):385–396, 2004.

[7] S. Bruckner and T. Möller. Isosurface similarity maps. *Comput. Graph. Forum*, 29(3):773–782, 2010.

[8] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the Morse-Smale complex and the Connolly function. In *Proc. Symposium on Computational Geometry*, pages 351–360. ACM, 2003.

[9] C. D. Correa, P. Lindstrom, and P.-T. Bremer. Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Trans. Vis. Comput. Graph.*, 17(12):1842–1851, 2011.

[10] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066, 2006.

[11] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amer Mathematical Society, 2010.

[12] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Local and global comparison of continuous functions. In *Visualization, 2004. IEEE*, pages 275–280. IEEE, 2004.

[13] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete & Computational Geometry*, 30(1):87–107, 2003.

[14] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[15] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1432–1439, 2007.

[16] M. Hilaga, Y. Shinagawa, T. Komura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH*, pages 203–212, 2001.

[17] U. Homberg, D. Baum, S. Prohaska, U. Kalbe, and K. J. Witt. Automatic extraction and analysis of realistic pore structures from μCT data for pore space characterization of graded soil. In *Proc. 6th Int. Conference on Scour and Erosion (ICSE-6)*, pages 66–73, 2012.

[18] Y. Hong and H.-W. Shen. Parallel reflective symmetry transformation for volume data. *Computers & Graphics*, 32(1):41–54, 2008.

[19] M. M. Kazhdan, B. Chazelle, D. P. Dobkin, T. A. Funkhouser, and S. Rusinkiewicz. A reflective symmetry descriptor for 3D models. *Algorithmica*, 38(1):201–225, 2003.

[20] M. M. Kazhdan, T. A. Funkhouser, and S. Rusinkiewicz. Symmetry descriptors and 3D shape matching. In *Symposium on Geometry Processing*, pages 117–126, 2004.

[21] J. Kerber, M. Wand, J. Krüger, and H. Seidel. Partial symmetry detection in volume data. In *Vision, Modeling, and Visualization*, pages 41–48. The Eurographics Association, 2011.

[22] V. G. Kim, Y. Lipman, X. Chen, and T. A. Funkhouser. Möbius transformations for global intrinsic symmetry analysis. *Comput. Graph. Forum*, 29(5):1689–1700, 2010.

[23] D. Laney, P. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Trans. Vis. Comput. Graph.*, 12(5):1053–1060, 2006.

[24] T. Lewiner, H. Lopes, and G. Tavares. Applications of Forman's discrete Morse theory to topology visualization and mesh compression. *IEEE Trans. Vis. Comput. Graph.*, 10(5):499–508, 2004.

[25] Y. Lipman, X. Chen, I. Daubechies, and T. Funkhouser. Symmetry factored embedding and distance. *ACM Transactions on Graphics (TOG)*, 29(4):103, 2010.

[26] A. Martinet, C. Soler, N. Holzschuch, and F. X. Sillion. Accurate detection of symmetries in 3D shapes. *ACM Trans. Graph.*, 25(2):439–464, 2006.

[27] N. Mitra, M. Pauly, M. Wand, and D. Ceylan. Symmetry in 3D geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, pages 29–51, 2012.

[28] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM Trans. Graph.*, 25:560–568, 2006.

[29] S. Nagaraj, V. Natarajan, and R. S. Nanjundiah. A gradient-based comparison measure for visual analysis of multifield data. *Comput. Graph. Forum*, 30(3):1101–1110, 2011.

[30] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. The toporrery: computation and presentation of multi-resolution topology. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pages 19–40, 2009.

[31] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas. Discovering structural regularity in 3D geometry. *ACM Trans. Graph.*, 27(3), 2008.

[32] J. Podolak, A. Golovinskiy, and S. Rusinkiewicz. Symmetry-enhanced remeshing of surfaces. In *Proc. Symposium on Geometry Processing*, pages 235–242, 2007.

[33] D. Raviv, A. Bronstein, M. Bronstein, and R. Kimmel. Full and partial symmetries of non-rigid shapes. *International journal of computer vision*, 89(1):18–39, 2010.

[34] D. Schneider, A. Wiebel, H. Carr, M. Hlawitschka, and G. Scheuermann. Interactive comparison of scalar fields based on largest contours with applications to flow visualization. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1475–1482, 2008.

[35] D. Thomas and V. Natarajan. Symmetry in scalar field topology. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2035–2044, 2011.

[36] D. M. Ushizima, D. Morozov, G. H. Weber, A. G. C. Bianchi, J. A. Sethian, and E. W. Bethel. Augmented topological descriptors of pore networks for material science. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2041–2050, 2012.

[37] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 13(2):330–341, 2007.

[38] K. Xu, H. Zhang, A. Tagliasacchi, L. Liu, G. Li, M. Meng, and Y. Xiong. Partial intrinsic reflectional symmetry of 3D shapes. *ACM Trans. Graph.*, 28(5), 2009.

[39] I. Yamazaki, V. Natarajan, Z. Bai, and B. Hamann. Segmenting point-sampled surfaces. *The Visual Computer*, 26(12):1421–1433, 2010.

[40] H. Zabrodsky, S. Peleg, and D. Avnir. Symmetry as a continuous feature. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(12):1154–1166, 1995.

[41] R. Zakia. *Perception and imaging*. Focal Press, 2001.

[42] X. Zhang, C. L. Bajaj, and N. A. Baker. Affine invariant comparison of molecular shapes with properties. In *ICES Tech Report*, 2005.

[43] J. Zhou and M. Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1481–1488, 2009.

[44] A. J. Zomorodian. *Topology for Computing*. Cambridge University Press, 2005.