VoroClust: Scalable Clustering for Remote Sensing

Nick Winovich, *Member, IEEE*, Liam Moynihan, Osama Abdelrahman, R. Derek West, *Member, IEEE*, Stephen Dauphin, J. Derek Tucker, *Senior Member, IEEE*, Gabriel Huerta, Kevin Potter, Robert Forrest, Cynthia Phillips, *Member, IEEE*, and Mohamed Ebeida

***Abstract*—Although supervised machine learning provides a powerful framework for image classification and segmentation, it requires comprehensive, consistent data sets, which are not available for many remote-sensing applications. Remote-sensing datasets are expensive to collect, and each is acquired under different environmental conditions, or with significant variations in system operating parameters. Unsupervised clustering algorithms analyze the structure of each dataset independently, rather than drawing on similarities with existing 'training' examples, and are thus well suited for practical remote-sensing applications.**

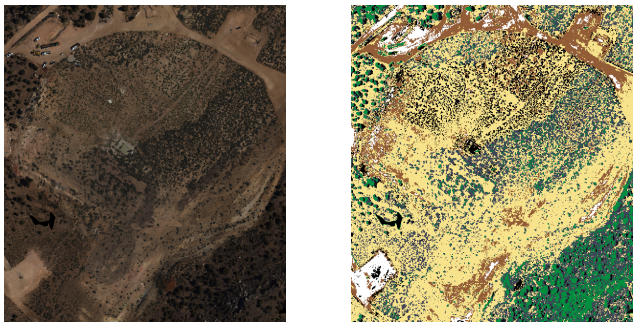
We introduce VoroClust, a fast, density-based unsupervised clustering algorithm applicable to high-resolution and high-dimensional data. VoroClust runs as fast as distance-based clustering methods, while capturing complex regional geometries at least as well as current density-based methods. It uses a data-centered sphere cover to reduce computational demands, while still capturing data topology. It then propagates clusters outward from local peaks in density. We show that VoroClust provides fast, state-of-the-art clustering for both high-resolution polarimetric synthetic aperture radar (PolSAR) and high-dimensional hyperspectral imaging (HSI) datasets.

***Index Terms*—Hyperspectral image analysis, polarimetric SAR, remote sensing, unsupervised clustering.**

I. Introduction

REMOTE-SENSING (RS) applications analyze raw sensor data. In terrain-cover identification, the goal is to identify objects such as buildings and regions of material such as water, grass, and pavement from measurements collected by sensors on aircraft and satellites. Remote-sensing technologies such as *polarimetric synthetic aperture radar* (PolSAR) and *hyperspectral imaging* (HSI) provide measurements with higher dimensionality than traditional optical images. The additional information can enable more refined distinctions between terrain-cover types, but it increases the difficulty of data analysis. This can create a critical computational bottleneck for time-sensitive remote-sensing applications that require near real-time analysis to provide actionable information (e.g., scenarios involving emergency response and management efforts related to floods, oil spills, and wildfires [1]–[3]).

While dimensionality-reduction methods such as principal component analysis [4] can speed up analysis and enable visualization, the information loss can undermine the benefits of working with PolSAR and HSI technologies. Recent research has shown the potential of supervised machine learning for remote sensing [5]–[9], but these approaches are intractable for many practical use cases due to scarcity and heterogeneity of remote-sensing data. Data collected using new and unvetted sensors, or under a variety of system operation parameters and environmental conditions (e.g., changes in weather, lighting or aircraft), can exhibit unique characteristics distinct from any existing reference data. This leads to a case-by-case analysis of sensor data, considering each dataset in isolation.



(a) (b)

Fig. 1. (a) Hyperspectral image (dimensions reduced for visualization) and (b) clusters representing different terrain covers such as soil, trees, and roads.

In this article, we restrict our attention to real-time terrain-cover identification from remote-sensing datasets with insufficient data for supervised machine learning. Thus we consider the unsupervised clustering problem: partition a d-dimensional dataset $\mathcal{D} \subset \mathbb{R}^d$, such as pixel locations representing high-dimensional feature vectors from PolSAR or HSI images, into disjoint subsets called *clusters*. Points in the same cluster are "closer" to each other than they are to points in other clusters. Closeness is typically defined relative to a distance function, such as the Euclidean distance $\rho(x, y) = (\sum_{i=1}^{d} |x_i - y_i|^2)^{1/2}$ for two d-dimensional points x and y in Euclidean space, or in terms of point density. Intuitively, a successful clustering partitions data points into interpretable groups (e.g., one cluster for *soil* and another cluster for *trees*[1]), as shown in Fig. 1.

Related work. Many clustering techniques have been proposed for remote sensing. Zhai et. al.'s review article [10] gives a high-level summary of these techniques. We restrict our attention to algorithms with asymptotic complexity that is sub-quadratic with respect to the number of data points,

Manuscript received September 19, 2025. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-NA0003525 *(Nick Winovich and Liam Moynihan contributed equally to this work.) (Corresponding author: Mohamed Ebeida.)*

Author affiliation: Sandia National Laboratories, Albuquerque, New Mexico 87123 USA (email: msebeid@sandia.gov).

[1]As is typical for image clustering (other than convolutional neural networks), we cluster only the values. Thus all pixels with "soil" values will be in a single cluster. We use 2D pixel geometry only for output visualization.

$|\mathcal{D}|$, and the data dimension, d. These are the only algorithms that are currently practical for real-time analysis. This criterion rules out methods such as Gaussian mixture models [11], [12], spectral clustering [13], sparse subspace clustering [14], and clustering by fast search and finding density peaks [15]. It also rules out Voronoi-clustering methods that explicitly compute a Voronoi diagram [16], [17], which can have $\Theta(n^{d/2})$ points for n data points in d dimensions.

Our complexity requirements also exclude most graph-based algorithms. Wang et. al.'s papers [18], [19] are two notable exceptions, which achieve sub-quadratic complexity by computing only on a subset of data points called '*anchor points*'. Despite the lower complexity, these algorithms are still too slow for real-time analysis[2]. Kokate et al. [20] survey clustering algorithms for *streaming data*, where points arrive over time. These algorithms largely focus on constraints specific to data streams, so they are not directly relevant to our remote-sensing application, where we cluster each image as a whole. However, there are some algorithmic similarities, most notably with DBSTREAM [21], which we discuss in Section II-B.

In distance-based approaches, such as k-Means [22], points in the same cluster are assumed to be positioned around a central location, and clustering is performed by identifying the center of each cluster. Density-based approaches, such as DBSCAN [23], [24] and HDBSCAN [25]–[27], model clusters more geometrically. They identify regions with high concentrations of data points and determine cluster boundaries based on drops in density [28]. k-Means and HDBSCAN are most commonly used in practice. k-Means is easy to use and is fast. However, it requires the user provide the number of clusters k and, because of its simplified assumptions on cluster geometry, it often fails to capture the true geometry of clusters, as shown in Fig. 2. k-Means does not identify noise and outlier regions, which can bias how clustering results are interpreted. HDBSCAN automatically determines the number of clusters, identifies noise and outliers, and models clusters more accurately. But it is frequently too slow for real-time applications, especially for high-dimensional data.

We compare our new algorithm to k-Means, DBSCAN, HDBSCAN, and BIRCH [29]. BIRCH enables clustering on very large datasets, but it still faces similar limitations to k-Means in terms of its simplified assumptions on cluster geometry. We describe these algorithms in more detail in Section IV-B. We also discuss the impact of applying dimensionality-reduction techniques prior to clustering in Appendix A, and we discuss clustering with more recent pre-trained models in Appendix B.

Contributions. In this work, we propose VoroClust, a novel, density-based clustering algorithm specifically designed for the real-time analysis of remote-sensing data. We define our algorithm in the full-dimensional data space to take advantage of the rich information acquired by PolSAR and HSI sensors. To reduce the computational complexity, we operate on a carefully selected subset of the data. VoroClust's complexity for a set of d-dimensional points $\mathcal{D}$ is $\mathcal{O}(d \cdot |\mathcal{D}| \cdot |\mathcal{S}|)$, where

[2]We do not have access to the code for direct comparisons, but the times reported in these papers are an order of magnitude slower than our algorithm.

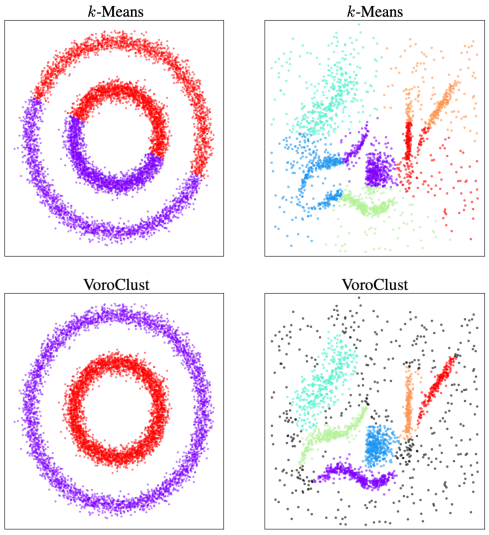


Fig. 2. The k-Means algorithm (top) works well when clusters correspond to convex, symmetric regions, but it can fail entirely for more complex data structures. k-Means also lacks noise assignment, making it difficult to identify outliers in the clustering results. The proposed density-based algorithm (bottom) provides more accurate geometric models for each cluster and detects noise (black), while maintaining minimal computational overhead.

$\mathcal{S} \subseteq \mathcal{D}$ denotes the sub-selected data and $|\mathcal{S}| \ll |\mathcal{D}|$. The algorithm parallelizes naturally and scales well empirically with respect to both the dimension and resolution of remote-sensing images. The primary contributions of this work are:

1) We introduce a fast, near real-time clustering algorithm that efficiently handles both high-dimensional and high-resolution remote-sensing datasets.
2) Our algorithm creates density-based models that accurately capture the geometry of cluster regions, as well as noise, with minimal computational overhead.
3) We define clustering rules to account for material mixing and noise common to HSI and PolSAR data [30], [31].

These contributions have been tailored to remote-sensing applications that often require high-dimensional imaging for improved discrimination, as well as high-resolution imaging for increased spatial precision or greater scene coverage. The graph-based analysis incorporates descent rules that have been specifically designed to account for mixed pixels and blurred boundaries that arise from the underlying physics of this remote-sensing imaging system application.

The remainder of this article is organized as follows. We describe the proposed VoroClust algorithm in Section II. We provide an overview of the PolSAR and HSI data formats in Section III. We describe our experimental design in Section IV, discuss the experimental results in Section V, and conclude in Section VI.

II. Methodology

A. VoroClust

In this section, we introduce VoroClust, a density-based clustering algorithm designed for remote-sensing applications. VoroClust creates clusters one at a time. It iteratively chooses

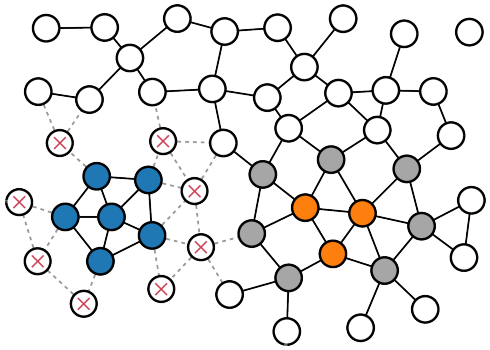


Fig. 3. Overview of the graph traversal procedure with a finalized cluster in blue and an orange cluster still in the process of expansion. The blue cluster is surrounded by boundary points, marked with red '×' symbols, which have prevented it from expanding any further. The orange cluster has a non-empty front, shown in gray, consisting of neighbors of vertices added to the cluster in previous iterations. We use this front to gather candidates for inclusion in the current cluster and to determine which vertex will be visited next.

a region with high data-point density to seed a new cluster. It then carefully extends the cluster outward into regions of decreasing density. Our algorithm controls the expansion, stopping cluster growth in a given direction when the density drops too low or upon evidence of touching a different cluster.

VoroClust runs faster than most density-based clustering algorithms by significantly down-selecting the data points using a sphere cover, as described in Section II-B. Each sphere is centered at a point from the original dataset and assigned a user-specified radius R. The sphere centers are representative of the full data set, so we perform most calculations using these points alone. Unlike grid-based approaches for down-selection, the proposed sphere cover does not introduce the curse of dimensionality since it is not axis-aligned.

Once the data has been down-selected, we construct a *graph* to model topological properties of the sphere cover. A graph has a set of *vertices*, which represent objects, and a set of *edges*, which represent relationships between objects. Graphs can be visualized using circles to represent vertices and lines to represent edges, as shown in Fig. 3. In our *density graph*, vertices represent spheres and edges connect spheres that overlap. We also assign *weights* to the vertices which store estimates for the data-point density of each sphere. Two vertices connected by an edge are referred to as *neighbors*, and we make clustering decisions for each vertex based on comparisons with its neighbors, as described in Section II-D.

Our algorithm removes some vertices and partitions the remaining vertices into connected subgraphs. Each subgraph represents a cluster, and we designate some clusters as noise. The spheres in each cluster are labeled and become the seeds of an implicit Voronoi tessellation[3] [32] in the original data space. We then assign each data point the label of its closest seed, which completes the clustering process. The high-level workflow for the algorithm is outlined in Fig. 4, and we provide the technical details involved with each step below.

[3]We avoid creating an explicit representation of the Voronoi tessellation since this is intractable in higher dimensions.

B. Sphere Cover and Density Graph

The VoroClust algorithm is defined by one principal parameter, the sphere radius $R > 0$, along with three auxiliary parameters $\alpha, \beta, \eta \in [0, 1]$ that control how noise is handled (as discussed in Section II-E). The radius parameter introduces a smoothing effect, similar to the bandwidth parameter in kernel density estimation techniques [33], and controls the level of compression during data sub-selection.

Our sub-selection procedure is based on a data-centered *sphere cover*. This is a subset of the data such that all data points are within a distance of R from at least one point in the subset. More formally, a sphere cover is defined as:

$$\mathcal{S} \subseteq \mathcal{D} \quad \text{with} \quad \mathcal{D} \subset \bigcup\nolimits_{s \in \mathcal{S}} B_R(s), \tag{1}$$

where $\mathcal{S}$ denotes the sub-selected data points, or *sphere centers*, and $B_R(s)$ is the ball of radius R centered at point s.

This allows us to reduce analysis of the full $|\mathcal{D}|$ observations to a much smaller set of $|\mathcal{S}| \ll |\mathcal{D}|$ representative points. To avoid redundancy, we construct *well-separated* sphere covers, meaning each point $s \in \mathcal{S}$ is contained in precisely one sphere:

$$s \notin B_R(s') \qquad \forall\, s' \neq s. \tag{2}$$

This is accomplished using a simple, iterative procedure: we randomly select a data point that is not contained in any of the existing spheres, include the point as the center for a new sphere in the cover, mark all points inside this sphere as *covered*, and repeat until the entire dataset has been covered.

Once the spheres have been selected, we compute the number of data points contained in each sphere and use these values as density estimates:

$$\widehat{f}(s) \;=\; \big|B_R(s) \cap \mathcal{D}\big| \;=\; \big|\{x \in \mathcal{D} \,:\, \rho(x, s) < R\}\big|. \tag{3}$$

We next construct the *density graph* $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, which is where we will perform the core clustering operations. We define this density graph with vertices, $\mathcal{S}$, given by the sphere centers (weighted by their density estimates) and edges, $\mathcal{E}$, formed between all overlapping spheres:

$$\mathcal{E} \;=\; \big\{(s, s') \;:\; s \neq s',\, B_R(s) \cap B_R(s') \neq \emptyset\big\}. \tag{4}$$

We create clusters by traversing this density graph, "visiting" each vertex a single time to determine its cluster assignment. When visiting a vertex $s \in \mathcal{S}$, we make clustering decisions based on its density estimate, $\widehat{f}(s)$, and comparisons with the density estimates of its *neighbors*, denoted by $\mathcal{N}(s)$.

This initial setup is conceptually similar to the '*microcluster*' technique introduced in DBStream [21]. In particular, DBStream uses spheres to capture incoming data and represents that data using a density graph. The authors were primarily concerned with streaming applications, so their algorithm focuses on managing and updating the graph as new data arrives. Their clustering procedure does not scale well to high-dimensional data due to its use of empirical volume estimates. Since the remote-sensing applications we are targeting often involve high-dimensional data, our graph traversal is defined using operations that extend well to higher-dimensions.

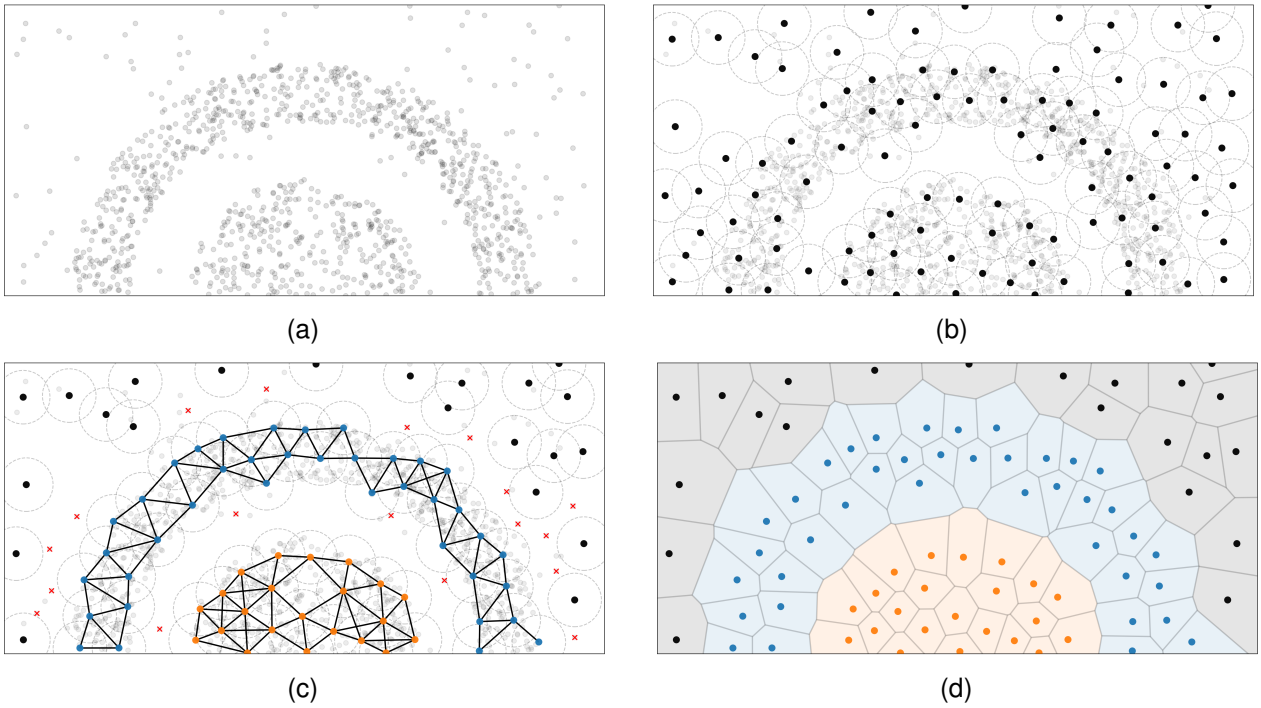


Fig. 4. Overview of VoroClust workflow. (a) The input is unlabeled data (b) a data-centered sphere cover is used to down-select points. We compute density estimates for each sphere and form a graph based on overlapping spheres. (c) Density estimates are used to cut vertices and identify subgraphs corresponding to clusters and noise. (d) Subgraphs induce an implicit Voronoi tessellation on the data space which defines the final clusters and noise regions.

C. Boundary Spheres and Cluster Subgraphs

The goal of the graph traversal is to partition the graph into smaller, disjoint subgraphs which will ultimately determine the clusters for the original dataset. While the transition from the full dataset to the representative spheres reduces the computational impact of the dataset size $|\mathcal{D}|$, we are still working in the full-dimensional space $\mathbb{R}^d$. For remote-sensing applications with high-dimensional data, graph connectivity increases sharply, and we will need to cut a significant number of vertices to form disjoint subgraphs.

We use the symbol $\mathcal{B}$ to denote the set of vertices that have been cut from the graph (along with their edges) to help separate clusters. Intuitively, these vertices correspond to boundary regions between clusters. Our goal is to find a meaningful partition of the non-boundary vertices:

$$\mathcal{S} \setminus \mathcal{B} = \bigcup_{C \in \mathscr{C}} C \quad \text{with} \quad C \cap C' = \emptyset \ \text{ for all } \ C \neq C', \quad (5)$$

where each $C \in \mathscr{C}$ corresponds to the vertices of a connected subgraph of $\mathcal{G}$. We refer to these sets as *cluster subgraphs*, or simply *clusters*, since they will determine the final labels we assign to the dataset. We have no information regarding the boundary set $\mathcal{B}$, or the cluster count $|\mathscr{C}|$, beforehand; so we will attempt to construct clusters sequentially, identifying boundary vertices along the way, until we exhaust the graph and obtain the desired partition.

D. Front Propagation and Density Descent

To ensure all clusters are connected with respect to the density graph, we expand each cluster gradually along a *front* $\mathcal{F} \subset \mathcal{S}$ of candidate vertices. Conceptually, the front models the outer limits of a cluster while it is still in the process of expansion (as depicted in Fig. 3).

At each iteration of the algorithm, we select the vertex s^* with the highest density on the front and compare it with its neighbors. We then decide whether the vertex s^* should be included in the current cluster or cut from the graph and marked as a boundary region. If the vertex is included in the cluster, we add its neighbors to the front as candidates for further expansion. If the vertex is cut from the graph, no additional vertices are added to the front. In either case, we remove s^* from the front and begin the next iteration at the highest density vertex remaining on the front.

When the front is exhausted, we finalize the current cluster and proceed to search for a new cluster (repeating until all vertices have been visited). We do this by re-initializing the front with a single element, the vertex with highest density among all remaining vertices, and propagate the new front using the same procedure described above. By construction, this front propagation technique ensures each cluster forms a connected subgraph since a vertex can only be added to a cluster if it neighbors one of the cluster's existing vertices.

While this procedure does create connected clusters, we still need to establish a set of rules for including (or cutting) vertices that results in *meaningful clusters*. We achieve the desired clustering behavior using the following, primary descent rule: a vertex is marked as a boundary region, and cut from the graph, if any of its unvisited neighbors have higher density. This criterion ensures that the front is always *descending* and that the cluster never spreads over other nearby, lower-

Algorithm 1 `on_boundary` $(s^*, \mathcal{N}(s^*) \cap \mathcal{U}, \widehat{f}_C)$

// Propagate front through noise in high-density regions
if $\widehat{f}(s^*) \geq \alpha \cdot \widehat{f}_C$ **then**
  return `False`
end if

// Stop propagation if density is too low relative to peak
if $\widehat{f}(s^*) \leq \beta \cdot \widehat{f}_C$ **then**
  return `True`
end if

// Add to boundary if the density begins to rise
if $\widehat{f}(s^*) < \widehat{f}(s')$ for any $s' \in \mathcal{N}(s^*) \cap \mathcal{U}$ **then**
  return `True`
end if

density clusters. This rule is specifically designed to handle mixed pixels that arise in remote-sensing applications, as these pixels can blur the boundaries between clusters, resulting in a separation region that still retains a non-trivial density. By tracking locations where the density transitions from descent to ascent (i.e., inflection points), the algorithm is capable of separating clusters in a manner that is robust to the presence of mixed pixels and blurred cluster boundaries. Two additional rules are incorporated to handle noise, and we describe these in the following section. We provide the precise formulation of the descent rules in Algorithm 1, and the full descent procedure is summarized in Algorithm 2.

E. Accounting for Noise

For practical applications, effectively accounting for noise is essential to an algorithm's performance. In this section, we discuss the three auxiliary parameters $\alpha, \beta, \eta \in [0, 1]$ that control how noise is handled by VoroClust.

There are two fundamentally distinct types of noise that need to be accounted for in the clustering procedure: noise intrinsic to the dataset $\mathcal{D}$ and noise resulting from the empirical density estimates. Ideally, we would like to capture the intrinsic noise in our final cluster results (since it reflects a property that is genuinely present in the data). But we want to minimize the impact of the estimation noise, since it arises from numerical limitations and does not reflect the data itself.

To reduce the impact of noisy density estimates, we modify the front propagation criteria based on two density-based thresholds $\alpha, \beta \in [0, 1]$. The *detail ceiling*, α, allows fronts to propagate more freely in high density regions (relative to the current cluster peak), and the *descent limit*, β, prevents fronts from propagating into low density regions which may comprise outliers or noise. We keep track of the current cluster's peak density $\widehat{f}_C$ by storing the density of the first vertex added to the front, as indicated in Line 11 of Algorithm 2.

For each vertex s^* on the front of cluster C, we perform two checks concerning its relative density before considering the primary descent criterion. If $\widehat{f}(s^*) \geq \alpha \cdot \widehat{f}_C$, we automatically

Algorithm 2 VoroClust Clustering Algorithm

1: // Initialize unvisited vertices $\mathcal{U}$, front $\mathcal{F}$, and boundary $\mathcal{B}$
2: $\mathcal{U} = \mathcal{S}; \quad \mathcal{F} = \emptyset; \quad \mathcal{B} = \emptyset$
3: **while** $|\mathcal{U}| > 0$ **do**
4:
5: **if** front $\mathcal{F}$ is empty **then**
6: // Initialize a new front to identify the next cluster
7: $C \leftarrow$ Next cluster index
8:
9: // Populate new front and store the peak density $\widehat{f}_C$
10: $\overline{s} \leftarrow$ Unvisited vertex with highest density
11: $\widehat{f}_C \leftarrow \widehat{f}(\overline{s})$
12: $\mathcal{F} \leftarrow \{\overline{s}\}$
13: **end if**
14:
15: // Select the highest density vertex on the front
16: $s^* \leftarrow \arg\max_{s \in \mathcal{F}} \widehat{f}(s)$
17: Remove s^* from the front and mark as visited
18:
19: // Compare with unvisited neighbors to determine if s^*
20: // is a boundary vertex or should be added to cluster
21: **if** `on_boundary` $(s^*, \mathcal{N}(s^*) \cap \mathcal{U}, \widehat{f}_C)$ **then**
22: Assign s^* to boundary $\mathcal{B}$
23: **else**
24: Assign s^* to cluster C
25: Add unvisited neighbors, $\mathcal{N}(s^*) \cap \mathcal{U}$, to the front $\mathcal{F}$
26: **end if**
27: **end while**

include the vertex s^* in the cluster since it corresponds to a high-density region. This relaxes the descent rule at the top of each cluster and reduces the impact of false peaks arising from noisy estimates, as illustrated in Fig. 5. At the other extreme, when $\widehat{f}(s^*) \leq \beta \cdot \widehat{f}_C$, we automatically cut the vertex s^*. This check prevents clusters from expanding into low-density regions and reduces the impact of outliers, as well as noise, on the final cluster geometry.

Once the graph traversal is complete, we perform a post-processing step to handle small clusters that were identified by the algorithm but may be insignificant for practical purposes. We use a threshold parameter, η, to control how these small clusters are handled, and we provide two distinct modes for employing it based on application requirements: $\eta_{cut} \in \mathbb{N}$ and $\eta_{noise} \in [0, 1)$. The first mode removes the smaller clusters, while the second uses them to model noise and outlier regions.

For applications where noise is believed to be negligible, we use the *cut threshold* $\eta_{cut} \in \mathbb{N}$ to specify the maximum number of clusters. If the algorithm identifies excess clusters, we sort the clusters according to the total weight of their vertices. We then remove the smaller clusters from the graph, keeping only the η_{cut} largest clusters. The final labeling procedure (described in Section II-F) is performed using the

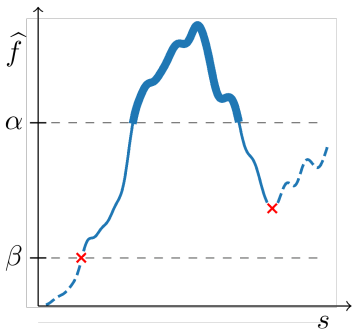


Fig. 5. Noise in the empirical density estimates often results in 'false-peaks' which can interfere with cluster expansion. The detail ceiling parameter, α, prevents clusters from artificially splitting at false-peaks by allowing fronts to propagate through density increases near the top of each cluster. The descent limit parameter, β, prevents fronts from descending too low relative to the cluster's peak density (where it may become biased by overlaps with other nearby clusters or overly influenced by noise).

remaining clusters, and all data points are assigned to one of the remaining, larger clusters.

For applications where noise identification is relevant, we use the *noise threshold* $\eta_{noise} \in [0, 1)$ to consolidate the smallest, isolated clusters into a dedicated *noise cluster*. More precisely, we collect the family of smallest clusters $\mathscr{C}_{noise} \subset \mathscr{C}$ whose cumulative size is less than $\eta_{noise} \cdot |\mathcal{D}|$ and assign these cluster regions to a single cluster label $C_{noise} = \bigcup_{C \in \mathscr{C}_{noise}} C$. This cluster provides a geometric model of noisy regions in the data, and points in these regions are explicitly marked as noise in the final cluster results.

With regard to the density graph, clustering is now complete. All that remains is to show how the clusters identified on the graph can be carried over to label the original dataset.

F. Final Label Assignment via Voronoi Partition

We label the original data space using an implicit *Voronoi tessellation* with seeds given by the non-boundary vertices of the density graph. That is, we partition $\mathbb{R}^d$ into convex regions of the form:

$$V(s) = \left\{ x \in \mathbb{R}^d : \rho(x, s) \leq \rho(x, s') \ \ \forall s' \in \mathcal{S} \setminus \mathcal{B} \right\}, \quad (6)$$

which consist of all points whose closest non-boundary vertex is $s \in \mathcal{S} \setminus \mathcal{B}$. We use these Voronoi regions to cluster the original dataset by assigning each region a label, $C(s)$, corresponding to the unique cluster $C \in \mathscr{C}$ assigned to s during the graph traversal phase. For each $s \in \mathcal{S} \setminus \mathcal{B}$, we collect the data points inside region $V(s)$ and assign them the label $C(s)$. Since the Voronoi regions partition the complete space $\mathbb{R}^d$, every data point will be assigned a label by the end of this procedure. We can also use the same procedure to efficiently label new or withheld data since the Voronoi regions implicitly cluster all points in $\mathbb{R}^d$, as illustrated in Fig. 4(d).

The final label assignment can be implemented naïvely by performing a nearest-neighbor search for each individual data point. But to improve the algorithm's computational efficiency, we want to avoid making additional distance checks whenever possible. Fortunately, we can avoid a substantial number of these checks by storing information when we compute the density estimates described in Section II-B. More precisely, when we calculate the number of interior points for a given sphere, we also track the indices of the data points contained in that sphere. Using this information, we assign all data points in $B_R(s)$ the label $C(s)$ immediately after the graph traversal (without performing any distance checks). Since the spheres included in the final clusters generally have high densities, this accounts for a significant portion of the dataset and considerably improves the speed of the algorithm. We label the remaining data points using a direct, nearest-neighbor search (often only required for 10% to 20% of the data in practice).

G. Complexity Analysis

The computational complexity of the algorithm is dominated by the initial sphere calculations. The sampling procedure used to generate sphere covers requires $\mathcal{O}(d \cdot |\mathcal{D}| \cdot |\mathcal{S}|)$ operations, corresponding to approximately $|\mathcal{D}|$ distance checks per sphere. An additional $\mathcal{O}(d \cdot |\mathcal{D}| \cdot |\mathcal{S}|)$ operations are required to estimate the density of each sphere. The complexity for constructing and traversing the density graph is $\mathcal{O}(d \cdot |\mathcal{S}|^2)$, and assigning the final labels has a worst-case complexity of $\mathcal{O}(d \cdot |\mathcal{D}| \cdot |\mathcal{S}|)$. In practice, the operations required for label assignment can be reduced significantly, as described at the end of Section II-F. The overall complexity of the VoroClust algorithm is $\mathcal{O}(d \cdot |\mathcal{D}| \cdot |\mathcal{S}|)$ with $|\mathcal{S}| \ll |\mathcal{D}|$.

For high-resolution datasets, we construct a k-d tree [34] for more efficient searches. The complexity of k-d tree construction is $\mathcal{O}(d \cdot |\mathcal{D}| \cdot \log |\mathcal{D}|)$, and computing density estimates using the k-d tree is $\mathcal{O}(d \cdot \log |\mathcal{D}| \cdot |\mathcal{S}|)$. The overall complexity is still the same due to the final labeling procedure, but we found the k-d tree implementation to be considerably faster for high-resolution datasets in practice.

The initial sphere calculations only depend on the radius parameter R. Once spheres have been assembled for a given radius, the auxiliary parameters (α, β, η) can be calibrated quickly with minimal computational overhead. Selecting parameters, especially the radius, is a key component of the algorithm. We provide a detailed discussion on parameter selection in Sections V-C through V-E.

III. Data

To assess the performance of the proposed VoroClust algorithm, we have conducted a series of experiments with applications to terrain cover identification for PolSAR and HSI remote-sensing datasets. Before discussing the results, we give a brief technical overview of each sensing modality.

A. Polarimetric Synthetic Aperture Radar

A radar system moving along a trajectory can operate in synthetic aperture radar (SAR) mode [35] to capture finely-resolved data for distant scenes. SAR systems transmit and receive a series of electromagnetic pulses during their motion along trajectories. The benefit of operating a radar system in SAR mode is that, with proper reconstruction algorithms, a much larger aperture can be synthesized than the physical aperture (antenna) of the actual radar system. This allows the

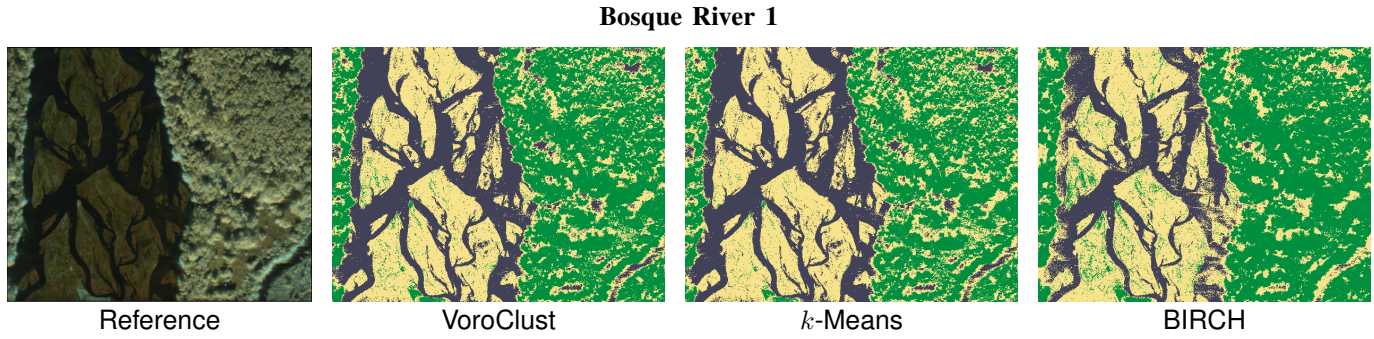


Fig. 6. *Bosque River 1* is a $3593 \times 4703 \times 6$ PolSAR dataset with terrain consisting of grass, trees, water, and sandbars. The clusters identified by VoroClust, k-Means, and BIRCH are all very similar, with only minor differences regarding vegetation on sandbars and radar shadows in the trees. VoroClust's runtime (24.5 s) is competitive with k-Means (7.7 s), and faster than BIRCH (269.3 s), while DBSCAN and HDBSCAN failed to finish clustering after several hours.

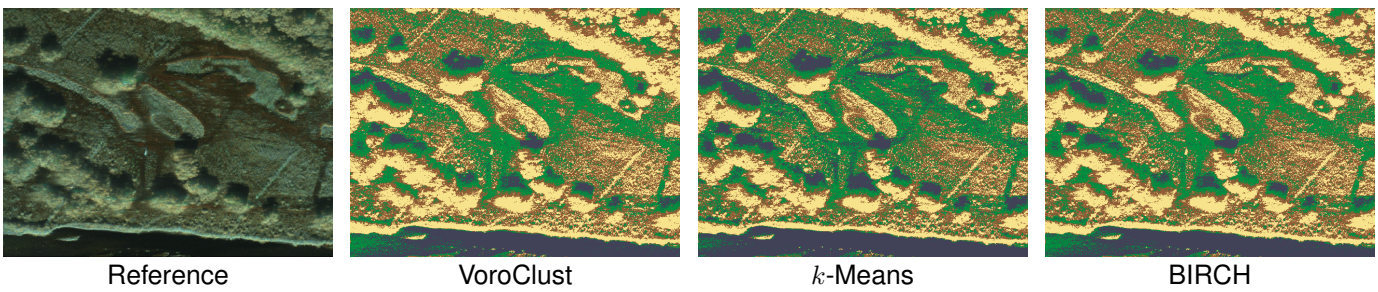


Fig. 7. *Bosque River 2* is a $4759 \times 3591 \times 6$ PolSAR dataset with terrain consisting of trees, grass, and a small section of riverbank. VoroClust, k-Means, and BIRCH identified similar clusters for this dataset and had reasonable runtimes of 44.8 s, 7.4 s, and 293.4 s, respectively. DBSCAN and HDBSCAN failed to produce results after running for several hours.

radar system to create higher-resolution images of scenes, as shown in Figs. 6–9.

A polarimetric SAR (PolSAR) system coherently measures four permutations of orthogonal electric field polarizations at each pulsing location of a trajectory. One of the most commonly used polarization bases is linear, where pulses with vertically oriented electric fields are transmitted, and responses from the scene are received on both vertically and horizontally oriented antenna receive ports. Similarly, pulses with horizontally oriented electric fields are transmitted (in a time-division multiplexed fashion), and re-radiated energy from the scene is received on vertically and horizontally oriented antenna ports. The full set of polarimetric permutations is VV, HV, VH, and HH, where the notation XY indicates polarization Y was transmitted and polarization X was received. A complex-valued SAR image is formed from each polarization channel; thus, a fully-polarimetric image set contains four different complex-valued images. The relative magnitude and phase responses between the pixels in PolSAR image sets, processed through polarimetric decompositions, gives rise to physically meaningful scattering physics of objects within a scene.

Two widely used building blocks for polarimetric decompositions are the Pauli feature vector and the corresponding coherency matrix, denoted as $\mathbf{p}$ and T, respectively [36]. For a monostatic PolSAR system, where $HV \approx VH$, $\mathbf{p}$ is a three element vector,

$$\mathbf{p} = \frac{1}{\sqrt{2}} \begin{bmatrix} HH + VV \\ HH - VV \\ HV + VH \end{bmatrix} . \tag{7}$$

T is the 3×3 matrix composed of the spatial average of $\mathbf{p}$ vectors,

$$T = \left\langle \mathbf{p}\mathbf{p}^H \right\rangle = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} , \tag{8}$$

where $\langle \cdot \rangle$ is a spatial ensemble average and the superscript H denotes a complex-conjugate transpose. By construction, T is a Hermitian matrix and is uniquely determined by its diagonal and upper off-diagonal entries.

The coherency matrices are computed for every pixel location in PolSAR image sets. For the feature vectors used in this paper, the diagonal and upper off-diagonal values are converted to decibel (dB),

$$\widetilde{T}_{ij} = 10 \cdot \log_{10} \left(|T_{ij}| \right) , \text{ for } j \geq i , \tag{9}$$

where the tilde indicates the conversion to dB. The $\widetilde{T}_{ij}$ values are also normalized to the interval $[0, 1]$ by limiting the global dynamic range across all values of $\widetilde{T}_{ij}$ then normalizing each by the dynamic range. The resulting six-dimensional PolSAR feature vectors that we use in this work are defined as:

$$\mathbf{v}_{SAR} = \begin{bmatrix} \widetilde{T}_{11} & \widetilde{T}_{22} & \widetilde{T}_{33} & \widetilde{T}_{12} & \widetilde{T}_{13} & \widetilde{T}_{23} \end{bmatrix}^T . \tag{10}$$

B. Hyperspectral Imaging

Hyperspectral imaging (HSI) is a passive optical remote sensing technique that captures measurements across broad ranges of wavelengths by filtering the incoming electromag-

Fig. 8. *Golf Course* is a $4713 \times 3593 \times 6$ PolSAR dataset with features including roads, trees, radar shadows, and varying types of grass. These features are clustered similarly by the algorithms, although there are some noticeable differences in how the types of grass are clustered; the VoroClust results include slightly less detail than k-Means and BIRCH for this scene.

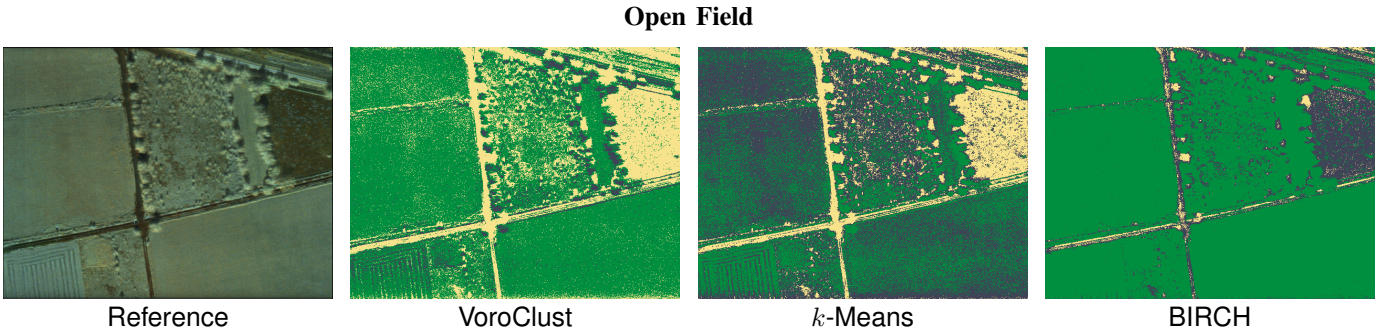


Fig. 9. *Open Field* is a $4713 \times 3593 \times 6$ PolSAR dataset. The VoroClust and k-Means results both capture the agricultural pattern in the lower-left corner of the scene, while BIRTCH misses this detail. VoroClust also provides clear, near-uniform clustering for the rectangular plots of grass, while the k-Means results are much more pixelated and noisy in these regions.

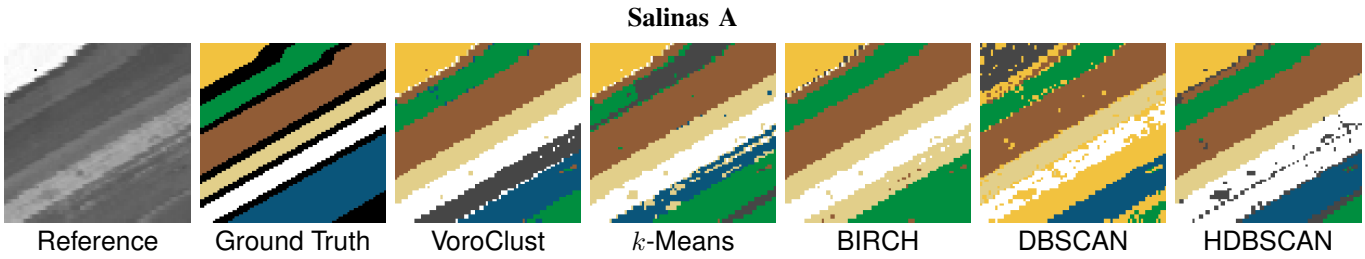


Fig. 10. *Salinas A* is an $83 \times 86 \times 204$ HSI dataset. The scene consists of several agricultural fields containing distinct types of crops. A partial ground truth is available for this dataset, and we use this to compute additional metrics when evaluating cluster quality (locations without ground-truth labels are shown in black). VoroClust receives the highest scores for both the Adjusted Rand and Fowlkes-Mallows metrics, which measure how closely the proposed clusters match the ground truth. HDBSCAN receives similar scores but has a runtime of $5.41\,\mathrm{s}$, while VoroClust takes only $0.03\,\mathrm{s}$. The scores for the remaining algorithms are much lower, with k-Means and DBSCAN producing visible clustering artifacts and BIRCH missing an entire cluster.

netic energy into narrow spectral bands. HSI collects both spatial and spectral measurements that provide discriminative information for material and target detection and characterization applications [37]. Common HSI spectral resolutions are often sufficient for differentiating materials with similar spectra such as healthy and diseased crops [38] and geologic materials [39]. HSI measurements often consist of hundreds of spectral bands at each pixel location, making the observations very high dimensional. The HSI datasets considered in this article contain between 103 to 274 spectral measurements and are shown in Figs. 10–12.

Let N denote the number of spectral measurements from an HSI sensor and h_i denote the i^{th} spectral measurement. The HSI feature vector that we use in this work is defined as:

$$\mathbf{v}_{HSI} = \begin{bmatrix} h_1 & h_2 & \cdots & h_N \end{bmatrix}^T. \tag{11}$$

IV. Experiment Design

A. Datasets

We selected four PolSAR and three HSI datasets, summarized in Table I. These datasets vary widely in the number of pixels and features. Two of the HSI datasets have ground-truth labels, which allows us to compute additional metrics when evaluating their cluster results. We now give more details on each type of dataset.

1) PolSAR Datasets: Sandia National Laboratories (SNL) collected fully-polarimetric SAR data using the SNL-developed Facility for Advanced RF and Algorithm Development (FARAD) X-band (9.6 GHz center frequency) radar system, flown on a DHC-6 airplane. The image sets were autofocused; within each image set, the same focus correction was applied to each polarization channel. The images were formed using effectively a -35 decibel (dB), $\bar{n} = 4$ Taylor

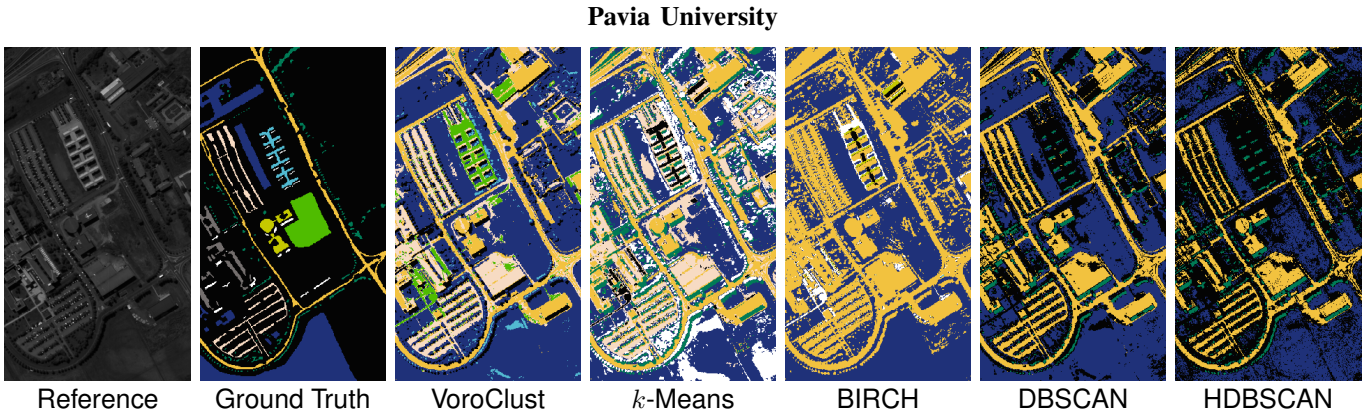


Fig. 11. *Pavia University* is a $610 \times 340 \times 103$ HSI dataset with terrain cover including grass, roads, parking lots, and a variety of buildings. DBSCAN and HDBSCAN identify similar clusters, with one cluster capturing the grass terrain (in purple) and another that combines buildings and roads into a single cluster (in yellow). BIRCH identifies these features as well, but its clusters are less sharp (e.g., blurred boundaries between roads and grass). The VoroClust and k-Means results provide more detail; for example, both separate the parking lots (in beige) from the main roads (in yellow). A partial ground truth is available for this dataset, and VoroClust receives the highest cluster-quality scores for both the Adjusted Rand and Fowlkes-Mallows metrics.

window for sidelobe level control. Calibration consists of compensating for the effective antenna patterns (magnitude and phase) of the different polarization channels, accounting for the differential phase that occurs from multiplexing the pulses, and radiometric calibration which balances the polarimetric channels and converts the image data to units of radar cross-section (RCS).

See [40] for a public version of the PolSAR data sets with images from each polarization channel. The images in [40] have a smaller geographic spatial extent than the images we used. Bosque River 1, Bosque River 2, Golf Course, and Open Field correspond to public images polSAR_02, polSAR_33, polSAR_23, and polSAR_18 respectively.

2) Nevada Road: This dataset is one of many HSI products that SNL collected with a linear pushbroom visible and near infrared (VNIR) Headwall Photonics Nano-Hyperspec hyperspectral imager with an onboard global positioning system (GPS) and inertial measurement unit (IMU), deployed on an unmanned aerial system (UAS). This system offers over 270 spectral bands over the wavelength range $\Lambda \approx 400-1000$ nm. The VNIR image sets were processed from raw to radiance products in the Headwall Photonics, Inc. SpectralView data processing and analysis software.

3) Salinas A: This is a small piece of a larger measurement of the Salinas Valley in California. It was taken by the NASA/JPL AVIRIS sensor [41], and made publicly available by the Grupo de Inteligencia Computacional (GIC)[4], of the Universidad del País Vasco. GIC provided a ground truth for this dataset.

4) Pavia University: This dataset was captured by the OpAiRS ROSIS sensor [42] over Pavia, Italy. GIC made it publicly available and provided a ground truth.

B. Comparison Algorithms

We compared the performance of VoroClust to that of four well-established unsupervised clustering algorithms: k-Means, BIRCH, DBSCAN, and HDBSCAN. We used the implementations for the first three algorithms in the SciKit Learn [43] Python package. HDBSCAN was recently added to SciKit Learn, but we used the implementation provided by the algorithm's creators [44]. We evaluated the output clusters in Python 3.11 using the scoring metrics in SciKit Learn. We now provide brief descriptions of the four reference algorithms.

1) k-Means partitions the data points into k disjoint clusters, where k is user specified, trying to minimize the average intra-cluster variance. The center of each cluster, referred to as the *centroid*, is the numerical average of the points in the cluster. The k-Means algorithm seeks to minimize the sum of the squares of the distances from each point to its cluster centroid. Lloyd's algorithm [45] is the most popular k-Means implementation. It requires an initial set of k centroids, which can be arbitrary. We use the k-Means++ [46] initialization, which selects a random data point as the first centroid, and then iteratively selects another data point with probability proportional to its squared distance from the closest, existing centroid, until it has k points. After initialization, Lloyd's algorithm assigns each point to the closest centroid, recomputes the

TABLE I
SUMMARY OF DATASETS USED FOR THE EXPERIMENTAL RESULTS. '(R)' INDICATES A REDUCED-RESOLUTION VERSION. DIM MEANS DIMENSION.

Name / Short Name	Type	Size	Dim	Ground Truth
Bosque River 1 / BR1	PolSAR	16,897,879	6	No
Bosque River 1 (R) / BR1(R)	PolSAR	266,208	6	No
Bosque River 2 / BR2	PolSAR	17,089,569	6	No
Bosque River 2 (R) / BR2(R)	PolSAR	151,512	6	No
Golf Course / GC	PolSAR	16,933,809	6	No
Golf Course (R) / GC(R)	PolSAR	264,012	6	No
Open Field / OF	PolSAR	16,933,809	6	No
Open Field (R) / OF(R)	PolSAR	264,461	6	No
Nevada Road / NR	HSI	349,796	274	No
Salinas A / SA	HSI	7138	204	Yes
Pavia University / PU	HSI	207,400	103	Yes

[4]Salinas and Pavia University data available at https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes

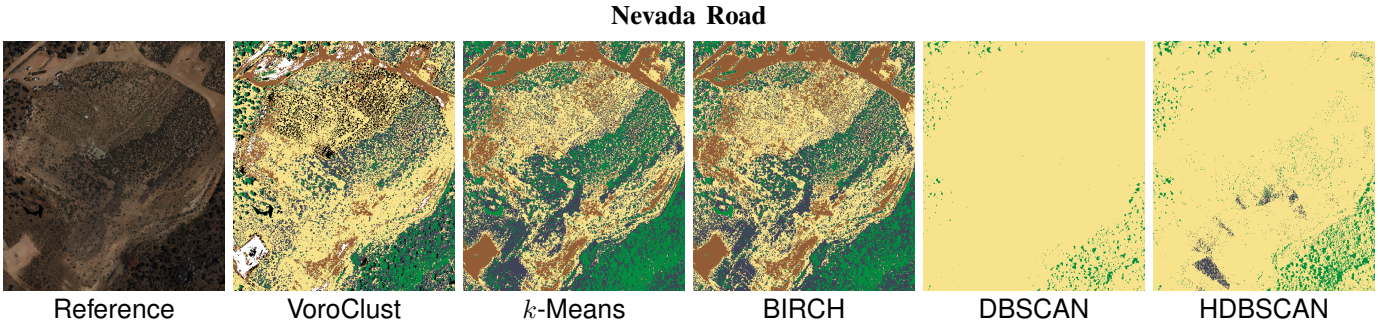


Fig. 12. *Nevada Road* is a $628 \times 557 \times 274$ HSI dataset with features including vegetation, packed dirt roads, and natural dirt terrain. The VoroClust, k-Means, and BIRCH algorithms each identify similar clusters, with a few noticeable differences. Both k-Means and BIRCH separate the natural dirt terrain into two clusters (beige and blue) while VoroClust does not differentiate between these dirt types (and blue corresponds to radar shadows). VoroClust includes more detail for the packed dirt roads, however, with an additional cluster in white that identifies areas that are driven on more frequently (e.g., the lot in the lower-left corner). VoroClust also provides outlier information for this scene (shown in black) which flags the concrete pad in the center of the image (missed by both k-Means and BIRCH) along with some of the vehicles in the upper-left corner. DBSCAN and HDBSCAN required significantly more time to cluster this dataset and ultimately identified most pixels as noise.

centroid for each cluster, and continues until the centroids have sufficiently converged. This method gives a local optimum of the objective function but does not guarantee a global optimum. Using k-means++ for initialization guarantees a clustering with average variance that is within $O(\log k)$ of optimal value [46].

k-Means is fast and gives good results for many applications, but it tends to struggle when the data has complex cluster geometries. The algorithm requires users to select an appropriate value of k, and the best automated method for determining k is still under debate [47].

2) Balanced Iterative Reducing and Clustering using Hierarchies (**BIRCH** [29]) first creates a clustering-feature (CF) tree by considering data points one at a time. The CF-tree has two parameters: the branching factor B and a diameter threshold T. The leaves of the tree contain up to a fixed number of initial clusters, each represented by a data tuple: (number of data points, sum of data points, sum of squares of data points). These values allow computation of the centroid and diameter of the leaf cluster, and also allow internal nodes to compute the values associated with the combined cluster they represent. Interior nodes have up to B children and store the CF values for the subcluster of each child. To insert a point, BIRCH starts at the root, and follows the child with the closest centroid repeatedly until it arrives at the leaf cluster with the centroid closest to the new point. It inserts the new point into that cluster if the diameter would still be within the threshold. If not, it creates a new cluster. This can involve adding new leaves and new levels to the CF-tree. If the tree gets too large, BIRCH starts over with a larger diameter threshold. BIRCH then applies a user-specified clustering algorithm on the leaf clusters, followed by an iteration of Lloyd's algorithm. It labels points too far from a cluster as outliers.

BIRCH's compact representation is well-suited for large datasets. It can also be used as a preprocessing step for other algorithms, such as k-Means.

3) Density Based Spatial Clustering of Applications with Noise (**DBSCAN**) [23] is intended for large spatial data sets with noise. It requires two parameters, a sphere diameter ϵ, and an integer MinPts. Points within a distance ϵ of a point p are p's *neighbors*, and a datapoint is *core* if is has at least MinPts neighbors. DBSCAN grows clusters point by point. All points are initially unlabeled. DBSCAN starts with an arbitrary point p, and determines if it is core. If not, it is labeled noise. Otherwise, it starts a new cluster with point p, and all its neighbors that are not already part of another cluster. This can add points originally labeled as noise. DBSCAN then expands the search from any newly added core points. When that cluster is complete (all core points in the cluster are expanded), it picks a new, unlabeled point to potentially start a new cluster. It ends when there are no unlabeled points.

DBSCAN can model clusters of arbitrary shape (e.g., spherical, linear, and non-convex clusters), and, as described above, it identifies noise and outliers.

4) Hierarchical Density Based Spatial Clustering of Applications with Noise (**HDBSCAN**) is conceptually similar to DBSCAN and aims to improve performance on variable density clusters while simplifying the parameter selection process [48]. The HDBSCAN algorithm performs DBSCAN clustering on data points using an array of ϵ values, as opposed to the single value used in DBSCAN [25]. HDBSCAN then calculates a distance matrix, extracts a hierarchy of connected components in the form of a dendrogram tree, and assigns data points to clusters by working up from the roots of the tree [48]. While the addition of the hierarchy step helps improve clustering quality, it can also increase the runtime considerably when working with high-dimensional data [44].

C. Clustering Evaluation Metrics

To assess the performance of each algorithm, we selected three metrics that provide quantitative measures of clustering quality. We computed all metric scores using the implementations from Python's SciKit Learn package [43]. When ground-truth data is available, we use the **Adjusted Rand Index** [49] and the **Fowlkes–Mallows Index** [50] to assess clustering quality. For these metrics, a higher value indicates greater similarity between clustering results and the ground truth. Since these metrics assess quality through comparisons with the ground truth, they are generally reliable indicators of clustering performance. However, ground-truth labels are

TABLE II
SUMMARY OF THE RUNTIMES FOR EACH CLUSTERING ALGORITHM WITH '×' INDICATING RUNTIMES THAT EXCEEDED 5 HOURS. THE FASTEST RUNTIMES FOR EACH DATASET ARE UNDERLINED AND SHOWN IN BOLD, AND THE SECOND-FASTEST RUNTIMES ARE ITALICIZED IN BOLD.

Runtime, seconds											
Algorithm	BR1	BR1 (R)	BR2	BR2 (R)	GC	GC (R)	OF	OF (R)	SA	PU	NR
k-Means	**7.67**	**0.16**	**7.36**	**0.18**	**16.69**	**0.30**	**10.58**	**0.21**	**0.11**	**0.54**	**3.90**
BIRCH	269.33	4.30	293.39	***2.66***	278.56	4.61	285.29	10.81	0.17	11.57	8.26
DBSCAN	×	4.40	×	3.18	×	86.95	×	13.82	0.24	240.26	365.34
HDBSCAN	×	27.70	×	15.61	×	68.29	×	43.28	5.41	2450.42	21630.34
VoroClust	***24.54***	***0.31***	***44.83***	***2.66***	***61.01***	***0.64***	***44.59***	***0.58***	**0.03**	***5.96***	***6.09***

TABLE III
SUMMARY OF DAVIES–BOULDIN CLUSTER QUALITY FOR DATASETS WHERE GROUND-TRUTH LABELS ARE NOT AVAILABLE. VOROCLUST RESULTS ARE GENERATED USING THE η_{cut} MODALITY TO PROVIDE HEAD-TO-HEAD COMPARISONS WITH ALGORITHMS THAT DO NOT ASSIGN NOISE. LOWER SCORES INDICATE BETTER CLUSTERING PERFORMANCE; BEST SCORES ARE UNDERLINED IN BOLD, AND SECOND-BEST SCORES ARE ITALICIZED IN BOLD. THE RESULT FOR DBSCAN FOR NEVADA ROAD (NR) IS CROSSED OUT BECAUSE DBSCAN LABELED ALMOST ALL POINTS NOISE. (SEE FIG. 12)

Davies–Bouldin Index (# Clusters)									
Algorithm	BR1	BR1 (R)	BR2	BR2 (R)	GC	GC (R)	OF	OF (R)	NR
k-Means	**0.6313** (3)	***0.6391*** (3)	0.8289 (4)	***0.9067*** (4)	**1.1237** (4)	1.0241 (3)	1.0316 (3)	1.0062 (3)	0.7481 (4)
BIRCH	0.6461 (3)	0.6468 (3)	**0.8050** (4)	0.9825 (4)	***1.1397*** (4)	***0.8969*** (3)	**0.8176** (3)	**0.8108** (3)	***0.7357*** (4)
DBSCAN	N/A	2.6152 (4)	N/A	3.2278 (2)	N/A	6.6158 (2)	N/A	1.9362 (3)	~~0.4117~~ (2)
HDBSCAN	N/A	2.8233 (3)	N/A	3.0556 (3)	N/A	5.2718 (3)	N/A	2.2349 (3)	2.1105 (3)
VoroClust	***0.6336*** (3)	**0.6331** (3)	***0.8139*** (4)	**0.8506** (4)	1.1959 (4)	**0.8375** (3)	***0.9052*** (3)	***0.8218*** (3)	**0.6634** (4)

TABLE IV
SUMMARY OF CLUSTERING PERFORMANCE FOR DATASETS WITH GROUND-TRUTH LABELS USING THE ADJUSTED RAND (AR), FOWLKES–MALLOWS (FM), AND DAVIES–BOULDIN (DB) METRICS. BEST SCORES ARE UNDERLINED IN BOLD, AND THE SECOND-BEST SCORES ARE ITALICIZED IN BOLD.

	Salinas A				Pavia University			
Algorithm	AR	FM	DB	# Clusters	AR	FM	DB	# Clusters
k-Means	0.7746	0.8187	***0.5869***	8	0.4978	0.6301	***0.8062***	6
BIRCH	0.7020	0.7681	**0.4776**	6	0.4212	0.6494	1.1777	7
DBSCAN	0.7857	0.8279	1.8491	7	***0.5161***	***0.6715***	2.7101	4
HDBSCAN	***0.8359***	***0.8704***	1.4702	8	0.3680	0.5509	2.9921	4
VoroClust	**0.8494**	**0.8802**	0.7790	8	**0.5513**	**0.7053**	**0.7536**	6
Ground Truth	1.0000	1.0000	0.9198	7	1.0000	1.0000	1.8603	10

difficult to generate and are not readily available for most real-world applications [51].

When datasets do not have ground-truth labels, we use the **Davies–Bouldin Index** [52] to assess results. Conceptually, a lower score indicates good inter-cluster separation and less scattered points within clusters. But the Davies–Bouldin metric assumes clusters are well-characterized by their centroids, and all calculations are defined relative to these locations. When the true clusters are non-convex, especially when two clusters have similar centoid locations (e.g., the concentric circles in Fig. 2), the scores become biased and do not provide a reliable indicator of clustering quality. In light of this limitation, we also use qualitative assessments of clustering quality (e.g., visual inspection and human explanations of clusters). These assessments show cases where metric scores, particularly Davies–Bouldin, are misleading.

V. RESULTS

In this section, we present the clustering results of the proposed VoroClust algorithm, along with four comparison algorithms, evaluated on a collection of PolSAR and HSI datasets. For the high-resolution PolSAR datasets, the runtimes of the density-based comparison algorithms exceeded several hours. To compare VoroClust's cluster quality with these algorithms, we created reduced-resolution versions of the PolSAR datasets[5]. Figs. 6–12 show visualizations of the clustering results produced by each algorithm. Tables II–IV give quantitative results, which we discuss in detail below.

A. HSI Results

Every algorithm successfully clustered Salinas A, the smallest HSI dataset. VoroClust and HDBSCAN performed best according to the Adjusted Rand and Fowlkes–Mallows metrics,

[5]We created reduced-resolution images using the Python Scikit-Image package [53] to rescale images down by a factor of eight along each axis.

indicating a strong agreement with the ground-truth labels. BIRCH and k-Means performed best according to the Davies–Bouldin index, despite receiving the worst scores for both metrics based on the ground truth. This illustrates the issues[6] with Davies–Bouldin's assumptions discussed in Section IV-C.

For the Pavia University dataset, VoroClust achieved the top scores in all three metrics. DBSCAN received the second-highest scores for Adjusted Rand and Fowlkes–Mallows, identifying clusters that align well with the ground truth but provide less detail than VoroClust (e.g., VoroClust separates parking lots from streets, as shown in Fig. 11). k-Means had the fastest runtime for this dataset, followed by VoroClust and BIRCH. VoroClust was considerably faster than the other density-based algorithms, producing clusters 40 times faster than DBSCAN and over 400 times faster than HDBSCAN.

Nevada Road is the largest HSI dataset, with 274 spectral measurements per pixel location. DBSCAN had the best Davies–Bouldin score, but, from a practical standpoint, it performed much worse than the other algorithms[7] (see Fig. 12). Omitting the DBSCAN results, VoroClust received the best Davies–Bouldin score, followed closely by BIRCH and k-Means. The k-Means algorithm had the fastest runtime for this dataset. VoroClust and BIRCH had comparable, but slightly slower runtimes than k-Means. HDBSCAN was significantly slower, taking 6 hours whereas k-Means and VoroClust took only a few seconds.

VoroClust also provides useful information regarding outlier regions in the Nevada Road dataset. From the results in Fig. 12, we see a concrete pad in the center of the image is determined to be an outlier by VoroClust. k-Means and BIRCH both identify this region as dirt and provide no indication that a man-made structure exists. This is a key shortcoming of the distance-based algorithms: individual objects, and regions with unique types of terrain cover, are often too small to form clusters on their own. As a result, they are assigned labels based on the more prominent features in the scene, making outlier objects indistinguishable from their surroundings.

For remote-sensing applications, identifying anomalous regions can be critical. We believe noise assignment is an important feature of VoroClust. While DBSCAN and HDBSCAN can also identify noise, they are orders of magnitude slower than VoroClust, thus generally unsuitable for real-time applications. We computed all metric scores using the η_{cut} modality of VoroClust[8], (i.e., removing small clusters), but we observed similar performance when using noise assignment.

B. PolSAR Results

For the Bosque River 1 and Bosque River 2 datasets, the VoroClust, k-Means, and BIRCH algorithms all performed similarly in terms of the Davies–Bouldin metric and the qualitative results. k-Means and BIRCH slightly outperform VoroClust on the Golf Course dataset, obtaining better metric scores and providing more detailed clusters, as shown in Fig. 8.

For the Open Field dataset, BIRCH has the best Davies–Bouldin score, but its clusters are less detailed than the VoroClust and k-Means results (see Fig. 9). VoroClust and k-Means both clearly identify the agricultural pattern in the scene, which BIRCH misses. VoroClust also provides a clean, homogeneous representation of the fields in the scene, while k-Means has much noisier performance in these regions.

k-Means was consistently the fastest algorithm for the full-resolution PolSAR datasets, with an average runtime of 10.6 s, while VoroClust averaged 43.7 s, and BIRCH averaged 281.6 s. The DBSCAN and HDBSCAN algorithms were both outliers on the PolSAR datasets, with worse Davies–Bouldin metric scores and considerably longer runtimes. We tested these algorithms on reduced-resolution versions of the datasets to produce results in a reasonable time frame; however, the clustering quality for these algorithms was inferior in both the metrics and qualitative assessments.

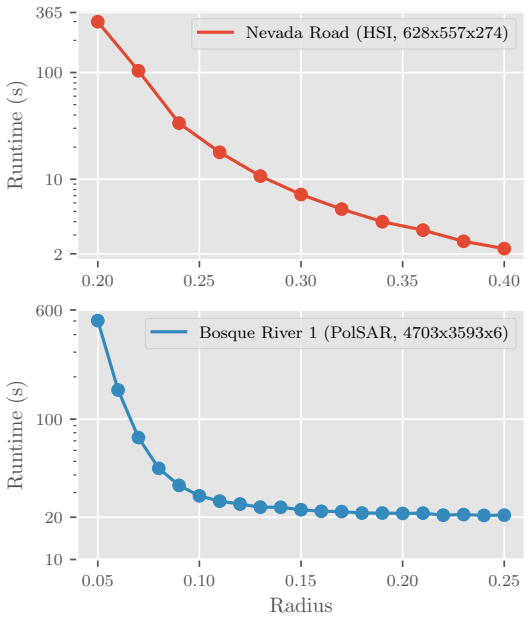


Fig. 13. The runtime for VoroClust increases significantly as the sphere radius is decreased (since the sphere count $|\mathcal{S}|$ approaches $|\mathcal{D}|$ in the limit $R \to 0$). In practice, as shown in prior results, VoroClust is fast because it can produce good labelings with a relatively large radius.

C. Performance

Table II summarizes algorithm runtimes. The k-Means algorithm is the fastest algorithm for all but the smallest dataset, Salinas A. VoroClust is the second fastest, demonstrating competitive performance across all ranges of dimensionality and dataset size. VoroClust is significantly faster than the density-based alternatives. VoroClust runs successfully on high-resolution data where DBSCAN and HDBSCAN timed-out, and VoroClust runtimes were several orders of magnitude shorter in general.

[6]The ground truth itself receives a Davies–Bouldin score of 0.9198 for the Salinas A dataset, which suggests relatively poor cluster quality.

[7]DBSCAN labels the vast majority of points as noise and identifies one, very small cluster for trees. The trees have low intra-cluster variance, and due to the formulation of the Davies–Bouldin index, the relatively large separation between the centers of the two clusters has an exaggerated impact on the score.

[8]Including noise in the metric scores biases the comparisons in favor of algorithms without noise (since noise 'clusters' have high intra-cluster variance). Omitting noise biases scores in favor of algorithms with noise assignment (since they can remove difficult, intermixed regions in the data).

The runtime of VoroClust depends strongly on the choice of radius. With a smaller radius, more spheres are required to cover the dataset. This increases the operation count for each step of the algorithm. Fig. 13 shows the effect on runtime. Some scenes required much smaller radius values than others for VoroClust to capture enough detail to produce quality clusters. Thus some similarly sized datasets had noticeably different runtimes for VoroClust.

D. Parameter Sensitivity

VoroClust results depend on multiple parameters, as well as the randomness of sphere sampling. Since parameter selection is currently manual, algorithm (non)sensitivity to each parameter and randomness is important for usability.

To test this, we needed a way to compare the quality of many results. While it is usually easy to judge qualitatively at a glance, that is not practical at large scale. The Davies-Bouldin Index does not require a ground truth, but it can be unreliable by giving very good scores to very poor results. Instead, we use the Adjusted Rand Index. Since this requires a ground truth for comparison, we chose a good VoroClust result for our sensitivity-test ground truth. This is valid because these ARI scores are only used to compare VoroClust with itself, rather than against other algorithms as in earlier sections.

Next, we generated results over a wide range of radii, keeping the secondary parameters α and β fixed to the same values as were used for the ground truth. For each radius value, we ran VoroClust 30 times to compute an average Adjusted Rand Index and to estimate the variance in the score introduced by the randomized sphere cover. Finally, to test sensitivity to α and β, we picked a good radius from the previous test and a fixed seed to control randomness. We then computed ARI for the full range of α and β. These results are shown in Fig. 14 and Fig. 15. A high ARI, close to 1.0, indicates close agreement with a reference clustering result selected by a subject matter expert. For some datasets, such as Bosque River 1 or Pavia University, we see consistent results over a wide range of R, and also over wide ranges of the secondary parameters. However, other datasests such as Nevada Road are less predictable. The ARI are lower on average, which is partly a consequence of the experiment relying on comparison to a single baseline run. Upon further visual inspection, we found many labelings with low ARI which still appear to represent the data well despite differences with the chosen baseline.

The parameter sweeps show the full allowable ranges of parameters, but the underlying theory suggests we should almost always have a detail ceiling above 0.5 and a descent limit below 0.5, so the bottom left corner of these plots is the most important region to consider. Changes to the descent limit are explicitly designed to expand/contract decision boundaries between clusters, and selecting a very low detail ceiling will merge clusters by construction, which can both lead to large drops in ARI in their extremes. Sensitivity plots for additional datasets are provided in Appendix C.

E. Parameter Selection

When searching for effective parameters, we recommend starting with large values of R, which can be evaluated quickly,

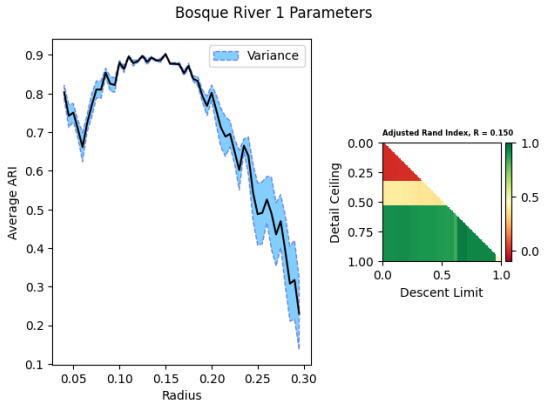


Fig. 14. (Left) The Adjusted Rand Index compares results to a ground truth. Radius is varied over a wide range, and auxiliary parameters are held constant. At each radius, we run 30 times to test sensitivity to random seed. We show the mean and variance, with the shaded blue region representing one standard deviation above and below the mean. High values of ARI over a wide range show that VoroClust results are very stable for Bosque River 1. (Right) We hold the radius and seed constant, while testing VoroClust using the full range of the auxiliary parameters: 'descent limit' and 'detail ceiling'. A high ARI indicates significant agreement with a ground truth.

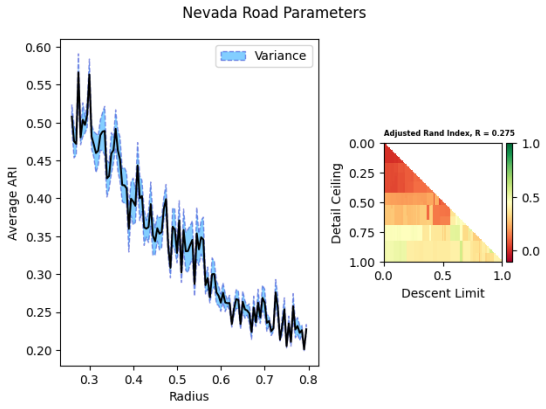


Fig. 15. In comparison to Bosque River 1 shown in Fig. 14, Nevada Road has far more variability due to randomness and changing parameters.

and gradually reducing the radius until distinct clusters begin to form. We found that the radius should generally give a ratio of spheres to data points between 1:100 and 1:1000. This number is dependent on the resultion and scale of your data. For the auxiliary parameters, we recommend starting with the default values: $\alpha = 0.8$, $\beta = 0.25$, and $\eta = 0.05$.

Most of VoroClust's runtime is spent in sphere sampling, which depends only on the radius. After choosing an effective radius, most users can quickly tune the auxiliary parameters to improve cluster quality based on application requirements without repeating sphere selection. Increasing the noise threshold η can remove smaller clusters. Decreasing the detail ceiling parameter α can expand clusters. For applications with very little noise, the descent limit β can be dropped to zero to obtain sharper cluster results. If very few clusters are appearing, users may need to raise the descent limit β, which encourages clusters to break at higher density levels. For the HSI datasets, we found that considerably higher descent limits were required. This is likely due to the material mixing

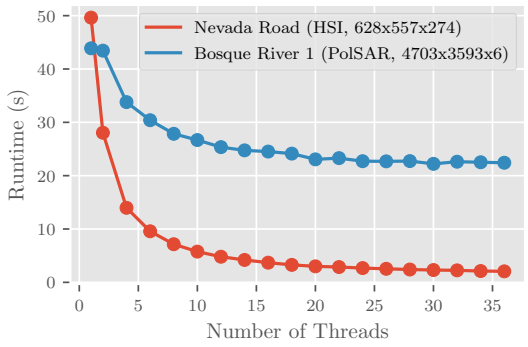


Fig. 16. VoroClust performance with multiple threads. Note, VoroClust uses a k-d tree for fast searches in lower dimensions (Bosque River 1). The construction of the k-d tree is not parallel, which creates a floor for multi-threaded runtime: approximately 22 s for the Bosque River dataset shown above.

effect common in HSI data and was the original motivation for including the descent limit parameter in the algorithm.

F. Implementation Details and Parallelization

The VoroClust algorithm admits a natural parallel implementation. Most steps of the algorithm are trivially parallelizable with OpenMP parallel loops. The only exception is sphere selection, where we use multithreaded batch processing due to data dependencies between loop iterations.

For low-dimensional data, such as PolSAR, we construct a k-d tree [34] from the input data for more efficient searches. k-d tree construction has a fixed cost that depends only on the input data. This forces a minimum (serial) cost in multi-threaded runs[9]. VoroClust does not use a k-d tree for the HSI datasets since the speed advantage becomes negligible in higher-dimensions. Fig. 16 shows VoroClust scaling for high-dimensional and low-dimensional examples.

Although we do not currently have a precised procedural method for fine tuning these parameters, there are some key intuitive rules that can be used to guide parameter calibration. If VoroClust produces a large number of small clusters, this can indicate that the radius is too small because density variation within a cluster is being captured as boundaries between clusters. In addition to adjusting the radius, this effect could also be managed by reducing the detail ceiling. This causes noise near the high density cores of clusters to be ignored, preventing breakage into smaller clusters until the overall density is smaller. Conversely, if VoroClust produces a single dominant cluster this can indicate a radius which is too large: single spheres are crossing the gaps between clusters without major drops in density. It is also possible for the highest density cluster to spread out to cover all of the low density regions, taking a greater fraction of the domain simply because of the order that clusters are defined. This can be controlled by raising the descent limit, which breaks propagation in low density regions and allows subsequent clusters to spread to their outskirts properly.

[9]Omitting the k-d tree implementation removes this runtime barrier, but we found the k-d tree improved performance when using limited cores.

G. Hardware

We performed most experiments on an Intel® Core™ i5-11600 CPU at 2.8 GHz, with 6 cores and 12 logical processors. The parallel scaling tests shown in Fig. 16 were performed on a single node of the Attaway supercomputer, at Sandia National Laboratories, which contains the 2.3 GHz Intel® Xeon® Gold 6140 CPU, with 2 sockets containing 18 cores each.

VI. Conclusion

We introduced a new unsupervised clustering algorithm, VoroClust, designed for terrain-cover identification in remote-sensing datasets. The algorithm begins by generating a set of spheres that covers the full dataset. This reduces complexity, and allows the algorithm to scale well to high-resolution data, such as PolSAR images. We compute density estimates for each sphere and form a density graph to model topological properties of the data distribution. VoroClust traverses this graph by descending from local peaks in density, breaking the graph into disjoint components that represent clusters. The spheres from each cluster induce an implicit Voronoi tessellation of the data space, which provides a geometric model for each cluster, as well as noise and outlier regions.

We tested the algorithm using an extensive collection of remote-sensing datasets, including PolSAR and HSI images with a wide range of resolutions and feature dimensions. We provide a detailed assessment for each dataset. Overall, VoroClust matches, or exceeds, the cluster quality of several established algorithms including k-Means++, BIRCH, DBSCAN, and HDBSCAN. VoroClust's speed is far more competitive with that of distance-based algorithms, such as k-Means++, than any of the existing density-based alternatives. Unlike distance-based approaches, however, our algorithm does not make simplifying assumptions on cluster geometry, and it is capable of providing noise and outlier information which k-Means++ cannot.

The current VoroClust algorithm has several key opportunities for improvement. With multiple tunable parameters, it can take some effort to achieve the desired clustering results for a given dataset. In future work, we hope to improve the algorithm's ease-of-use by automating the selection of the auxiliary parameters. Secondly, the reliance on a single global radius for sphere coverage can limit the ability to capture certain geometries. A method of automated parameter selection could enable adaptive sphere radii to more accurately cluster regions that have distinct intrinsic scales. With additional research targeting these improvements, we believe VoroClust can be extended to provide a flexible clustering framework that adapts to the natural scales of the data. This regional scale information could also help simplify, and potentially automate, the parameter selection process in the future. Additionally, while the algorithm was designed explicitly with remote-sensing applications in mind, the proposed clustering algorithm can also be applied to more general datasets, and we plan on directing future research efforts to assess how well VoroClust generalizes to data collected from applications outside of the remote-sensing domain.

VII. Acknowledgments

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Appendix A
Results with Dimensionality-Reduced Data

A standard way to accelerate clustering on high-dimensional data is to perform dimensionality reduction, e.g., using principal component analysis (PCA), before clustering. This approach reduces the computational costs of clustering by mapping the original data to a lower-dimensional space, while attempting to retain as much information about the original dataset as possible. The dimension for HSI images is the number of spectral bands. PCA can map the original image with shape $[H, W, d]$ to a reduced representation of shape $[H, W, d_R]$, with $d_R \ll d$.

To assess how dimensionality reduction impacts the performance and cluster quality of VoroClust, we conducted additional tests where applied PCA to reduce the dimensionality of each image to $d_R = 3$. We then applied either VoroClust or k-Means for comparison. Table V summarizes the results.

For k-Means, applying PCA before clustering causes minimal change in the Davies-Bouldin scores. This seems to indicate that the k-Means algorithm is not using the higher-dimensional features in the original datasets. This is perhaps unsurprising since the algorithm is designed around centroids and averages of points, and these values are likely less sensitive to the effects of dimensionality reduction.

For VoroClust, applying PCA before clustering consistently produces lower-quality clusters (with the Bosque River 1 dataset being the only exception). This suggests that VoroClust can take advantage of information in the higher-dimensional representation that is lost during dimensionality reduction. Since the difference in scores is relatively modest, there is likely not a significant amount of data lost for these particular examples. However, for more complex datasets, it is possible that more PCA modes would be required to achieve comparable cluster results. Algorithms that can cluster in the original data space need not select the number of PCA modes and they ensure that data loss is not impacting the clustering results.

While collecting data on the runtimes of the PCA and non-PCA approaches, we discovered that the choice of hardware had a significant impact on the timing results. We found the Intel® Core™ i5-11600 CPU that we used for the results in Section V performs quite poorly when running the large-matrix operations required to apply PCA to the remote-sensing datasets. Table VI summarizes the runtimes with the Intel chip. On average 56% of the runtime was spent performing PCA. To provide a more fair comparison of the PCA runtimes, we conducted additional tests using a more modern Apple M3 Max chip. Table VII summarizes these results.

Overall, we do not observe any significant change in runtime for k-Means when using PCA, although the PCA approach can be significantly slower when running on older hardware. For VoroClust, there are some instances where PCA yields a noticeable speed-up in the algorithm; however, there is also a drop in cluster quality for these examples. This suggests that the decision of whether or not to use PCA should be made based on the trade-off between cluster quality and runtime constraints.

TABLE V
Summary of the Davies-Buildin scores for VoroClust and k-Means with and without dimensionality reduction.

Davies-Bouldin Index							
Algorithm	BR1	BR2	GC	OF	SA	PU	NR
VoroClust	***0.6270***	**0.7427**	**1.0892**	**0.8075**	**0.6166**	**0.7867**	**0.5827**
+ PCA	**0.5859**	***0.8057***	***1.0960***	***0.8211***	0.6786	0.8452	***0.6066***
k-Means	0.6310	0.8295	1.1232	1.0326	0.7209	0.8065	0.7476
+ PCA	0.6315	0.8312	1.1254	1.0316	***0.6513***	***0.8063***	0.7483

TABLE VI
Summary of the runtimes for VoroClust and k-Means with and without dimensionality reduction. [Intel® Core™ i5-11600]

Runtime, seconds							
Algorithm	BR1	BR2	GC	OF	SA	PU	NR
VoroClust	29.15	47.93	68.81	52.93	***0.04***	6.24	***6.16***
+ PCA	28.06	35.46	27.38	28.39	0.17	***3.20***	13.75
k-Means	**7.75**	**12.36**	**10.93**	**7.08**	**0.04**	**0.45**	**3.50**
+ PCA	***14.15***	***16.59***	***15.66***	***13.71***	0.18	3.29	13.26

TABLE VII
Summary of the runtimes for VoroClust and k-Means with and without dimensionality reduction. [Apple M3 Max]

Runtime, seconds							
Algorithm	BR1	BR2	GC	OF	SA	PU	NR
VoroClust	11.07	17.18	20.43	17.26	**0.01**	1.80	**2.09**
+ PCA	10.41	14.18	11.02	10.92	***0.05***	0.73	***3.27***
k-Means	**2.86**	**3.67**	***3.56***	**2.60**	0.06	**0.66**	3.59
+ PCA	***3.00***	***3.73***	**3.52**	***2.82***	0.06	***0.70***	3.56

Appendix B
Pre-trained Models for Remote-Sensing Data

Another emerging strategy for clustering and image segmentation is the use of large, pre-trained transformer models, such as the Segment Anything Model (SAM 3) [54], [55]. These models are trained on a diverse corpus of image data and have been shown to perform well at zero/few-shot segmentation tasks for a variety of applications [56], [57]. Since the models are trained on standard RGB images with three channels (i.e., three dimensions per pixel), a dimensionality-reduction technique such as PCA must first be applied to reduce the dimensionality of PolSAR and HSI datasets to three dimensions before using a pre-trained model. We conducted a short series of numerical tests to determine whether a pre-trained transformer model could potentially be used to cluster remote-sensing datasets. Due to the large number of parameters

required by these transformer models, running these models typically requires a hardware accelerator such as a GPU.

For our experiments with pre-trained models, we used the SAM 3 model running on an NVIDIA L40S GPU. We initially used channel-wise PCA for dimensionality reduction. However, we found that even after shifting and rescaling values to match the values of standard RGB images, the model struggled to produce any meaningful clustering results with these inputs. The model performed considerably better on RGB reference images that were created for visual inspection (i.e., those shown at the far left of the figures in the main text).

The SAM 3 model also accepts text-based 'prompts' to direct the model to perform specific tasks. For example, if an application requires identifying all of the buildings in a scene, the model can be provided this information through the prompt and will (if successful) return pixel-wise segmentation masks for buildings, while omitting other features in the scene. We experimented with a series of different prompting strategies, but ultimately found limited success when working with the PolSAR and HSI remote-sensing datasets.

On Nevada Road, SAM 3 fails to identify any objects at all for the majority of prompts. The prompt '*Identify the dirt in this overhead image*' yielded the output closest to clustering results, which is shown in Figure 17. SAM 3 can identify a few (though not all) of the vehicles when prompted to do so, but we found no prompts that could produce any meaningful full clustering of the terrain. This is consistent with Zhang et al.'s findings [58]. They note that pre-trained transformer models generally have trouble with 'amorphous regions' such as roads and other types of land cover. They propose an extended architecture specifically to address this problem. This approach would still require GPU hardware and a dedicated server instance of the model in order to provide anywhere near real-time predictions.

On the Pavia University dataset, the SAM 3 model can identify buildings with straight-forward prompting, as shown in Figure 18. However, we counld not find any prompts that produced meaningful clustering of any of the roads or terrain types in the scene. Also, most of the segmentation results exhibited imprecise boundaries, often blurring regions between buildings and the surrounding terrain.

In practice, we observed that the resulting three-channel representations for remote-sensing images often have less visual contrast and much less diverse color ranges than one is accustomed to seeing in standard RGB images. It is possible that this perceptible difference in dimensionality-reduced images has a negative impact on the performance of pre-trained models, as we found the results on the PolSAR and HSI remote-sensing datasets were far less accurate than the results on standard RGB overhead images.

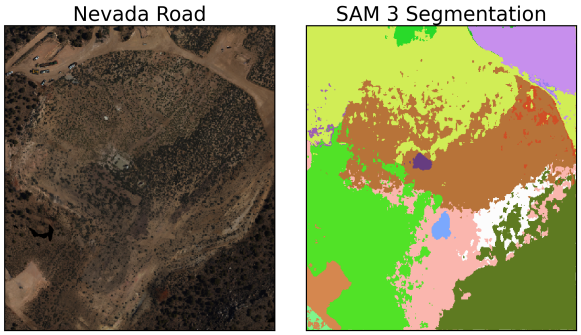


Fig. 17. SAM 3 segmentation results on the RGB reference image for the Nevada Road dataset when provided prompt '*Identify the dirt in this overhead image*'.

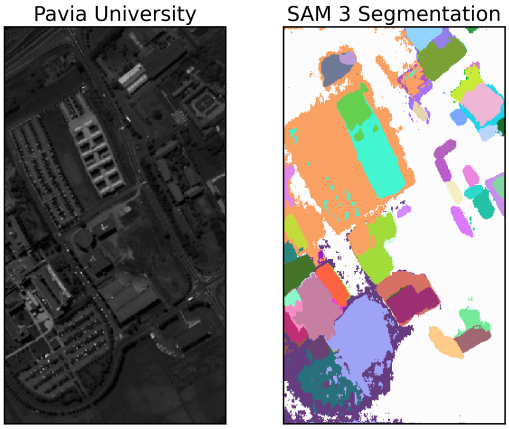


Fig. 18. SAM 3 segmentation results on the RGB reference image for the Pavia University dataset when provided prompt '*Identify the buildings in this overhead image*'.

APPENDIX C
PARAMETER SENSITIVITY ANALYSIS

To assess the sensitivity of the VoroClust algorithm to the selection of hyperparameters, we conducted a series of numerical studies by sweeping the radius and auxiliary parameters and computing the change in cluster quality relative to the baseline results presented in the main text using the Adjusted Rand Index. The results of the sensitivity studies are summarized in Figures 19–23.

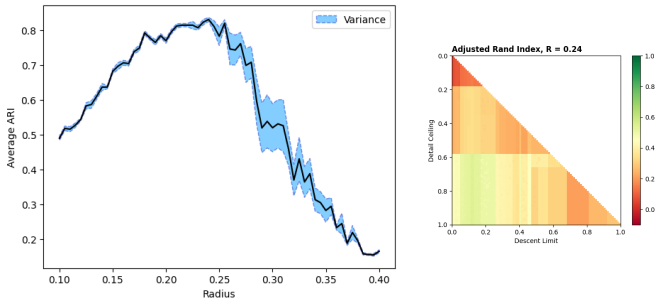


Fig. 19. Parameter sensitivity results for Pavia University dataset.

REFERENCES

[1] P. Boccardo and F. Giulio Tonolo, "Remote sensing role in emergency mapping for disaster response," in *Engineering Geology for Society and Territory-Volume 5: Urban Geology, Sustainable Planning and Landscape Exploitation*. Springer, 2014, pp. 17–24.

[2] S. Liu and M. E. Hodgson, "Satellite image collection modeling for large area hazard emergency response," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 118, pp. 13–21, 2016.

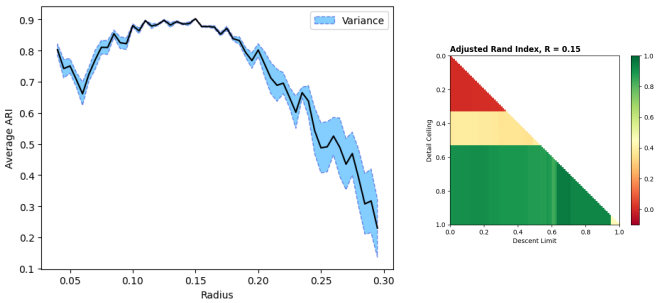


Fig. 20. Parameter sensitivity results for Bosque River 1 (Reduced).

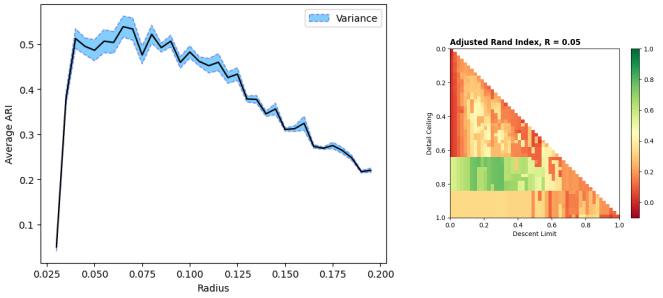


Fig. 21. Parameter sensitivity results for Golf Course (Reduced).

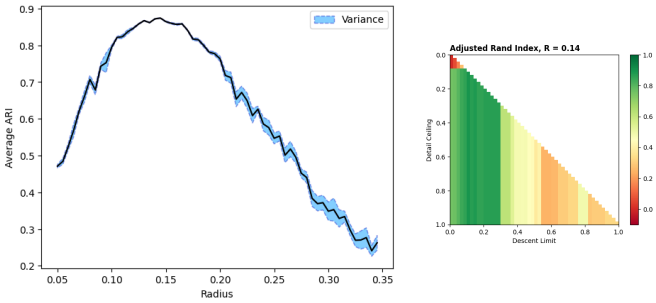


Fig. 22. Parameter sensitivity results for Open Field (Reduced).

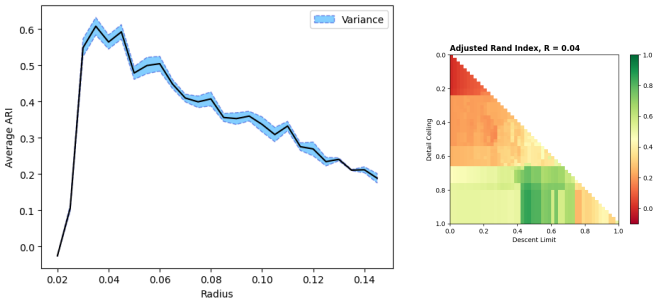


Fig. 23. Parameter sensitivity results for Bosque River 2 (Reduced).

[3] K. E. Joyce, K. C. Wright, S. V. Samsonov, and V. G. Ambrosia, "Remote sensing and the disaster management cycle," *Advances in geoscience and remote sensing*, vol. 48, no. 7, pp. 317–346, 2009.

[4] C. Rodarmel and J. Shan, "Principal component analysis for hyperspectral image classification," *Surveying and Land Information Science*, vol. 62, no. 2, pp. 115–122, 2002.

[5] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017.

[6] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 5966–5978, 2020.

[7] D. Hong, Z. Han, J. Yao, L. Gao, B. Zhang, A. Plaza, and J. Chanussot, "Spectralformer: Rethinking hyperspectral image classification with transformers," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2021.

[8] H. Chen, Z. Qi, and Z. Shi, "Remote sensing image change detection with transformers," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.

[9] S. K. Roy, A. Deria, D. Hong, B. Rasti, A. Plaza, and J. Chanussot, "Multimodal fusion transformer for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–20, 2023.

[10] H. Zhai, H. Zhang, P. Li, and L. Zhang, "Hyperspectral image clustering: Current achievements and future lines," *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 4, pp. 35–67, 2021.

[11] R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.

[12] C. M. Bishop, "Pattern recognition and machine learning," *Springer google schola*, vol. 2, pp. 1122–1128, 2006.

[13] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[14] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[15] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[16] H. Koivistoinen, M. Ruuska, and T. Elomaa, "A voronoi diagram approach to autonomous clustering," in *International Conference on Discovery Science*. Springer, 2006, pp. 149–160.

[17] D. Reddy and P. K. Jana, "A new clustering algorithm based on voronoi diagram," *International Journal of data mining, modelling and management*, vol. 6, no. 1, pp. 49–64, 2014.

[18] R. Wang, F. Nie, and W. Yu, "Fast spectral clustering with anchor graph for large hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 11, pp. 2003–2007, 2017.

[19] R. Wang, F. Nie, Z. Wang, F. He, and X. Li, "Scalable graph-based clustering with nonnegative relaxation for large hyperspectral image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 7352–7364, 2019.

[20] U. Kokate, A. Deshpande, P. Mahalle, and P. Patil, "Data stream clustering techniques, applications, and models: comparative analysis and discussion," *Big Data and Cognitive Computing*, vol. 2, no. 4, p. 32, 2018.

[21] M. Hahsler and M. Bolaños, "Clustering data streams based on shared density between micro-clusters," *IEEE transactions on knowledge and data engineering*, vol. 28, no. 6, pp. 1449–1461, 2016.

[22] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[24] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, jul 2017. [Online]. Available: https://doi.org/10.1145/3068335

[25] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. [Online]. Available: https://doi.org/10.21105/joss.00205

[26] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, nov 2017. [Online]. Available: https://doi.org/10.1109%2Ficdmw.2017.12

[27] ——, "Accelerated hierarchical density based clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 33–42.

[28] P. Bhattacharjee and P. Mitra, "A survey of density based clustering algorithms," *Frontiers of Computer Science*, vol. 15, pp. 1–27, 2021.

[29] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *SIGMOD Record*, vol. 25, no. 2, p. 103–114, June 1996. [Online]. Available: https://doi.org/10.1145/235968.233324

[30] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE journal of*

selected topics in applied earth observations and remote sensing, vol. 5, no. 2, pp. 354–379, 2012.

[31] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1923–1938, 2018.

[32] A. Rushdi, L. P. Swiler, E. T. Phipps, M. D'Elia, and M. S. Ebeida, "Vps: Voronoi piecewise surrogate models for high-dimensional data fitting," *International Journal for Uncertainty Quantification*, vol. 7, no. 1, 2017.

[33] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.

[34] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[35] C. V. Jakowatz, Jr., D. E. Wahl, P. E. Eichel, D. C. Ghiglia, and P. A. Thompson, *Spotlight-mode Synthetic Aperture Radar: A Signal Processing Approach*. Springer, 1996.

[36] J. Lee and E. Pottier, *Polarimetric Radar Imaging: From Basics to Applications*. CRC Press, 2002.

[37] N. M. Nasrabadi, "Hyperspectral target detection: An overview of current and future challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 34–44, 2014.

[38] A. K. Mahlein, E. C. Oerke, U. Steiner, and H. W. Dehne, "Recent advances in sensing plant diseases for precision crop protection," *European Journal of Plant Pathology 133*, May 2012.

[39] F. D. van der Meer, H. M. van der Werff, F. J. van Ruitenbeek, C. A. Hecker, W. H. Bakker, M. F. Noomen, M. van der Meijde, E. J. M. Carranza, J. B. de Smeth, and T. Woldai, "Multi- and hyperspectral geologic remote sensing: A review," *International Journal of Applied Earth Observation and Geoinformation*, vol. 14, no. 1, pp. 112 – 128, 2012.

[40] R. D. West, E. Steinbach, M. Ebeida, and A. Henriksen, "Fully Polarimetric, High-Resolution Synthetic Aperture Radar Image Set for Albuquerque, New Mexico," https://doi.org/10.5281/zenodo.11371700, 2024.

[41] G. Vane, R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, and W. M. Porter, "The airborne visible/infrared imaging spectrometer (aviris)," *Remote Sensing of Environment*, vol. 44, no. 2, pp. 127–143, 1993, airbone Imaging Spectrometry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/003442579390012M

[42] S. Weide, M. Bachmann, P. Gege, C. Schwarz, S. Holzwarth, and A. Muller, "Opairs – optical airborne remote sensing and calibration facility," 03 2009.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, mar 2017. [Online]. Available: https://doi.org/10.21105%2Fjoss.00205

[45] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[46] D. Arthur and S. Vassilvitskii, "k-Means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[47] E. Schubert, "Stop using the elbow criterion for k-Means and how to choose the number of clusters instead," *ACM SIGKDD Explorations Newsletter*, vol. 25, no. 1, pp. 36–42, 2023.

[48] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172.

[49] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, 1985.

[50] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American Statistical Association*, vol. 78, no. 383, pp. 553–569, 1983. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1983.10478008

[51] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, 2001.

[52] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[53] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[54] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.

[55] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang *et al.*, "Sam 3: Segment anything with concepts," *arXiv preprint arXiv:2511.16719*, 2025.

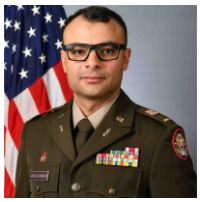
[56] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, "Segment anything in medical images," *Nature communications*, vol. 15, no. 1, p. 654, 2024.

[57] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.

[58] K. Li, S. Zhang, Y. Deng, Z. Wang, D. Meng, and X. Cao, "Segearth-ov3: Exploring sam 3 for open-vocabulary semantic segmentation in remote sensing images," *arXiv preprint arXiv:2512.08730*, 2025.

Nick Winovich (Member, IEEE) is a research scientist and Senior Member of the Technical Staff at Sandia National Laboratories. He received his BA from the University of Notre Dame in 2012, a masters degree from the University of Oregon in 2015, and a PhD in mathematics from Purdue University in 2021. Nick's research focuses on the intersection of machine learning, probability theory, and partial differential equations with an emphasis on scientific and engineering applications.

Liam Moynihan is a Senior Member of the Technical Staff at Sandia National Laboratories. He received a B.S. in Applied Physics from Rensselaer Polytechnic Institute in 2015, and an M.S. in Scientific Computing from the University of Utah in 2022. His research interests include clustering and meshing algorithms. He also supports Sandia's remote sensing mission as a Software Engineer.



Osama Abdelrahman is an Executive Officer in the United States Army National Guard. His research concentrates on utilizing and applying the principles and tools of Data Science in the field of National Security.

R. Derek West (Member, IEEE) received his BS and MS degrees concurrently in electrical engineering in 2008 and a PhD in electrical engineering in 2011, all from Utah State University. He is currently a Distinguished Member of the Technical Staff at Sandia National Laboratories. His research interests are in statistical signal processing and synthetic aperture radar.

Stephen Dauphin received his BS in Mathematics in from California Polytechnic State University, San Luis Obispo, in 2004, his MS in Computational Mathematics from Texas A&M University in 2013, and his PhD in Mathematics from Colorado State University in 2017. He is currently a Senior Member of the Technical Staff at Sandia National Laboratories. His research focuses on polarimetric synthetic aperture radar.

J. Derek Tucker (Senior Member, IEEE) is a Distinguished Member of the Technical Staff at Sandia National Laboratories and Adjunct Faculty at University of Illinois Urbana-Champaign. He received his B.S. in Electrical Engineering Cum Laude and M.S. in Electrical Engineering from Colorado State University in 2007 and 2009, respectively. In 2014 he received the Ph.D. degree in Statistics from Florida State University in Tallahassee, FL under the co-advisement of Dr. Anuj Srivastava and Dr. Wei Wu. He currently is leading research projects in the area of satellite image registration and statistical functional modeling of optical emissions. His research is focused on pattern theoretic approaches to problems in image analysis, computer vision, and signal processing.

Gabriel Huerta received the Ph.D. degree in statistics from Duke University, Durham, NC, USA, in 1998. From 2002 to 2017, he was a Professor of Statistics with the University of New Mexico, Albuquerque, NM, USA, where he taught a variety of statistics courses at both the undergraduate and graduate levels. He joined Sandia National Laboratories, Albuquerque, NM, USA, as a Distinguished Member of the Technical Staff in 2018. He has authored or co-authored many peered reviewed papers and has presented extensively at conferences. He supports a variety of projects that concern Bayesian model calibration for material sciences applications, expert elicitation approaches for reliability assessments, the application of surrogate models for UQ problems and connections between UQ and machine learning. Currently he is supporting the development of methods for extreme value analysis for evaluating the impact of compound extreme weather events. His research interests include Bayesian methods, time series, spatio-temporal analysis, statistical learning and UQ. Dr. Huerta is an Associate Editor for the journal Environmetrics.

Kevin Potter is a member of the technical staff at Sandia National Laboratories' Applied Machine Intelligence department. Kevin's research concentrates on applying and developing cutting edge research in machine learning to a wide range of science and engineering problems.

Robert Forrest is a member of the technical staff at Sandia National Laboratories where his research interests include nuclear power, cybersecurity, and nonproliferation. As a member of the systems research group, he specializes in data driven methods and analysis to inform policy for national security.

Cynthia Phillips (Member, IEEE) received a B.A. degree in Applied Mathematics from Harvard University and a PhD in Computer Science from the Massachusetts Institute of Technology. She is a laboratory fellow at Sandia National Laboratories. Her broad research interests include combinatorial optimization, parallel computing, and diverse applications.

Mohamed Ebeida is a research scientist at Sandia National Laboratories. His main areas of expertise are computational geometry and uncertainty quantification. He graduated from University of California, Davis in 2008 with a Ph.D. in mechanical and aeronautical engineering and a master's in applied mathematics. He worked for two years as a postdoc at Carnegie Mellon University. In 2010, he joined Sandia National Laboratories, where he works in exploring the potential of Voronoi decompositions for a wide range of non-traditional applications including mesh generation, data compression and Machine learning.