Received: 13 May 2017 | Revised: 29 August 2017 | Accepted: 14 October 2017

DOI: 10.1111/tgis.12313

RESEARCH ARTICLE

WILEY Transactions in GIS

ADCN: An anisotropic density-based clustering algorithm for discovering spatial point patterns with noise

Gengchen Mai[1] | Krzysztof Janowicz[1] | Yingjie Hu[2] | Song Gao[1]

[1] Department of Geography, University of California, Santa Barbara, California, USA

[2] Department of Geography, University of Tennessee, Knoxville, Tennessee, USA

Correspondence
Gengchen Mai, STKO Lab, Department of Geography, University of California Santa Barbara, Santa Barbara, CA 93106, USA.
Emails: gengchen_mai@geog.ucsb.edu

Abstract

Density-based clustering algorithms such as DBSCAN have been widely used for spatial knowledge discovery as they offer several key advantages compared with other clustering algorithms. They can discover clusters with arbitrary shapes, are robust to noise, and do not require prior knowledge (or estimation) of the number of clusters. The idea of using a scan circle centered at each point with a search radius *Eps* to find at least *MinPts* points as a criterion for deriving local density is easily understandable and sufficient for exploring isotropic spatial point patterns. However, there are many cases that cannot be adequately captured this way, particularly if they involve linear features or shapes with a continuously changing density, such as a spiral. In such cases, DBSCAN tends to either create an increasing number of small clusters or add noise points into large clusters. Therefore, in this article, we propose a novel anisotropic density-based clustering algorithm (ADCN). To motivate our work, we introduce synthetic and real-world cases that cannot be handled sufficiently by DBSCAN (or OPTICS). We then present our clustering algorithm and test it with a wide range of cases. We demonstrate that our algorithm can perform equally as well as DBSCAN in cases that do not benefit explicitly from an anisotropic perspective, and that it outperforms DBSCAN in cases that do. Finally, we show that our approach has the same time complexity as DBSCAN and OPTICS, namely O(n log n) when using a spatial index and O(n^2) otherwise. We provide an implementation and test the runtime over multiple cases.

1 | INTRODUCTION AND MOTIVATION

Cluster analysis is a key component of modern knowledge discovery, be it as a technique for reducing dimensionality, identifying prototypes, cleansing noise, determining core regions, or segmentation. A wide range of clustering

algorithms, such as DBSCAN (Ester, Kriegel, Sander, & Xu, 1996), OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999), K-means (MacQueen, 1967), and mean shift (Comaniciu & Meer, 2002), have been proposed and implemented over the last decades. Many clustering algorithms depend on *distance* as their main criterion (Davies & Bouldin, 1979). They assume *isotropic* second-order effects (i.e., spatial dependence) among spatial objects, thereby implying that the magnitude of similarity and interaction between two objects mostly depends on their distance. However, the genesis of many geographic phenomena demonstrates clear *anisotropic* spatial processes. As for ecological and geological features, such as the spatial distribution of rocks (Hoek, 1964), soil (Barden, 1963), and airborne pollution (Isaaks & Srivastava, 1989), their spatial patterns vary in direction (Fortin, Dale, & Ver Hoef, 2016). Similarly, data about urban dynamics from social media, the census, transportation studies, and so forth are highly restricted and defined by the layout of urban spaces, and thus show clear variance along directions. To give a concrete example, geotagged images—be it in the city or the great outdoors—show clear directional patterns due to roads, hiking trails, or simply for the fact that they originate from human, goal-directed trajectories. Isotropic clustering algorithms such as DBSCAN have difficulties dealing with the resulting point patterns and either fail to eliminate noise or do so at the expense of introducing many small clusters. One such example is depicted in Figure 1. Owing to the changing density, algorithms such as DBSCAN will classify some noise (i.e., points between the spiral arms) as being part of the cluster. To address this problem, we propose an anisotropic density-based clustering algorithm.

More specifically, the research contributions of this article are as follows:

- We introduce an anisotropic density-based clustering algorithm (ADCN).[1] While the algorithm differs in the underlying assumptions, it uses the same two parameters as DBSCAN, namely *Eps* and *MinPts*, thereby providing an intuitive explanation and integration into existing workflows.

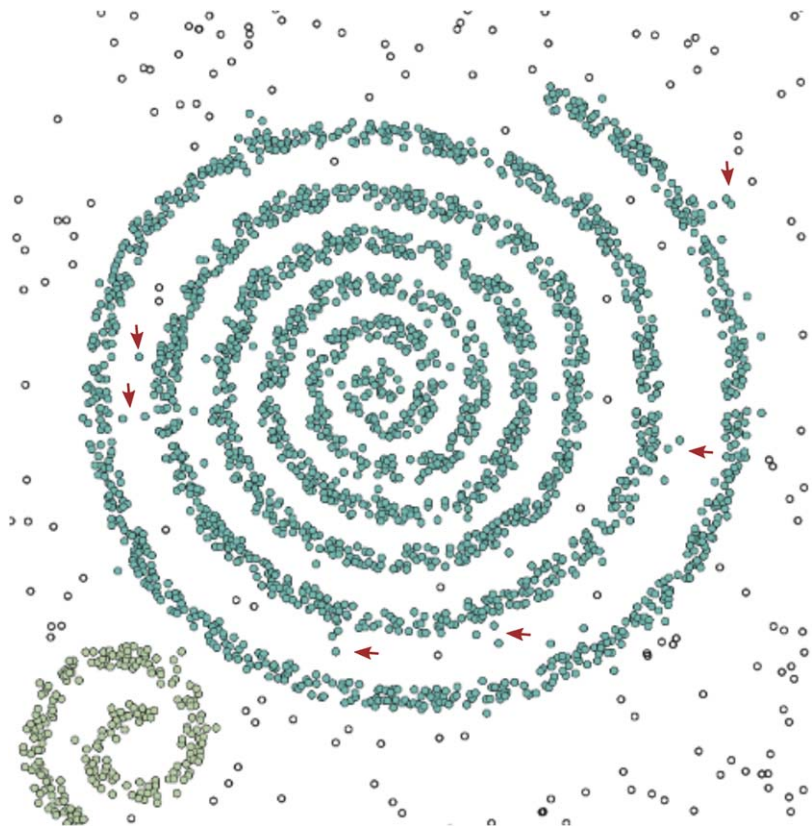

FIGURE 1 A spiral pattern clustered using DBSCAN. Some noise points are indicated by red arrows

- We motivate the need for such an algorithm by showing 12 synthetic and 8 real-world use cases, each with three different noise definitions, modeled as buffers that generate a total of 60 test cases.
- We demonstrate that ADCN performs as well as DBSCAN (and OPTICS) for isotropic cases, but outperforms both algorithms in cases that benefit from an anisotropic perspective.
- We argue that ADCN has the same time complexity as DBSCAN and OPTICS, namely O(n log n) when using a spatial index and O(n^2) otherwise.
- We provide an implementation for ADCN and apply it to the use cases to demonstrate the runtime behavior of our algorithm. As ADCN has to compute whether a point is within an ellipse instead of merely relying on the radius of the scan circle, its runtime is slower than that of DBSCAN while remaining comparable with OPTICS. We discuss how the runtime difference can be reduced by using a spatial index and by testing the radius case first.

The remainder of this article is structured as follows. First, in Section 2, we discuss related work such as variants of DBSCAN. Next, we introduce ADCN and discuss two potential realizations of measuring anisotropicity in Section 3. Use cases, the development of a test environment, and a performance evaluation of ADCN are presented in Section 4. Finally, in Section 5, we conclude our work and point to directions for future work.

2 | RELATED WORK

Clustering algorithms can be classified into several categories, including but not limited to partitioning, hierarchical, density-based, graph-based, and grid-based approaches (Deng, Liu, Cheng, & Shi, 2011; Han, Kamber, & Pei, 2011). Each of these categories contains several well-known clustering algorithms, with their specific pros and cons. Here we focus on the density-based approaches.

Density-based clustering algorithms are widely used in big geodata mining and analysis tasks, like generating polygons from a set of points (Duckham, Kulik, Worboys, & Galton, 2008; Moreira & Santos, 2007; Zhong & Duckham, 2016), discovering urban areas of interest (Hu et al., 2015), revealing vague cognitive regions (Gao et al., 2017), detecting human mobility patterns (Huang, 2017; Huang & Wong, 2015, 2016; Jurdak et al., 2015), and identifying animal mobility patterns (Damiani, Issa, Fotino, Heurich, & Cagnacci, 2016).

Density-based clustering has many advantages over other approaches. These advantages include: (a) the ability to discover clusters with arbitrary shapes; (b) robustness to data noise; and (c) no requirement to predefine the number of clusters. While DBSCAN remains the most popular density-based clustering method, many related algorithms have been proposed to compensate for some of its limitations. Most of them, such as OPTICS (Ankerst et al., 1999) and VDBSCAN (Liu, Zhou, & Wu, 2007), address problems arising from density variations within clusters. Others, such as ST-DBSCAN (Birant & Kut, 2007), add a temporal dimension. GDBSCAN (Sander, Ester, Kriegel, & Xu, 1998) extends DBSCAN to include non-spatial attributes in clustering and enables the clustering of high-dimensional data. NET-DBSCAN (Stefanakis, 2007) revises DBSCAN for network data. To improve the computational efficiency, algorithms such as IDBSCAN (Borah & Bhattacharyya, 2004) and KIDBSCAN (Tsai & Liu, 2006) have been proposed.

All of these algorithms use distance as the major clustering criterion. They assume that the observed spatial patterns are isotropic (i.e., that intensity does not vary by direction). For example, DBSCAN uses a scan circle with an *Eps* radius centered at each point to evaluate the local density around the corresponding point. A cluster is created and expanded as long as the number of points inside this circle (*Eps*-neighborhood) is larger than *MinPts*. Consequently, DBSCAN does not consider the spatial distribution of the *Eps*-neighborhood, which poses problems for linear patterns.

Some clustering algorithms do consider local directions. However, most of these so-called direction-based clustering techniques use spatial data which have a predefined local direction (e.g., trajectory data). The *local direction* of one point is predefined as the direction of the vector which is part of the trajectories with the corresponding point as its origin or destination. DEN (Zhou et al., 2010) is one direction-based clustering method which uses a grid data structure to group trajectories by moving directions. PDC+ (Wang & Wang, 2012) is another trajectory-specific DBSCAN variant that includes the direction per point. DB-SMoT (Rocha, Times, Oliveira, Alvares, & Bogorny, 2010) includes both

the direction and temporal information of GPS trajectories from a fishing vessel in the clustering process. Although all of these three direction-based clustering algorithms incorporate local direction as one of the clustering criteria, they can only be applied to trajectory data.

Anisotropicity (Fortin et al., 2016) describes the variation of directions in spatial point processes, in contrast with *isotropicity*. It is another way to describe the intensity variation in a spatial point process, other than first- and second-order effects. *Anisotropicity* has been studied in the context of interpolation, where a spatially continuous phenomenon is measured, such as directional variograms (Isaaks & Srivastava, 1989) and different modifications of kriging methods based on local anisotropicity (Boisvert, Manchuk, & Deutsch, 2009; Machuca-Mory & Deutsch, 2013; Stroet & Snepvangers, 2005). In this article, we focus on *anisotropicity* of spatial point processes. Researchers have studied *anisotropicity* of spatial point processes from a theoretical perspective by analyzing their realizations, such as detecting anisotropy in spatial point patterns (D'Ercole & Mateu, 2013) and estimating geometric anisotropic spatial point patterns (Møller & Toftaker, 2014; Rajala, Särkkä, Redenbach, & Sormani, 2016). Here, we study *anisotropicity* in the context of density-based clustering algorithms.

A few clustering algorithms take anisotropic processes into account. For instance, in order to obtain good results for crack detection, an anisotropic clustering algorithm (Zhao, Wang, & Ye, 2015) has been proposed to revise DBSCAN by changing the distance metric to geodesic distance. QUAC (Hanwell & Mirmehdi, 2014) demonstrates another anisotropic clustering algorithm, which does not make an isotropic assumption. It takes the advantages of anisotropic Gaussian kernels to adapt to local data shapes and scales and prevents singularities from occurring by fitting the Gaussian mixture model (GMM). QUAC emphasizes the limitation of an isotropic assumption and highlights the power of anisotropic clustering. However, due to the use of anisotropic Gaussian kernels, QUAC can only detect clusters which have ellipsoid shapes. Each cluster derived from QUAC will have a major direction. In real-world cases, spatial patterns will show arbitrary shapes. Furthermore, the local direction is not necessarily the same between and even within clusters. Instead, it is reasonable to assume that the local direction can change continuously in different parts of the same cluster.

3 | INTRODUCING ADCN

In this section we introduce the proposed Anisotropic Density-based Clustering with Noise (ADCN).

3.1 | Anisotropic perspective on local density

Without predefined direction information from spatial datasets, one has to compute the *local direction* for each point based on the spatial distribution of points around it. The standard deviation ellipse (SDE) (Yuill, 1971) is a suitable method to get the major direction of a point set. In addition to the major direction (long axis), the flattening of the SDE implies how much the points are distributed strictly along the long axis. The flattening of an ellipse is calculated from its long axis a and its short axis b:

$$f = \frac{a-b}{a} \tag{1}$$

Given n points, the SDE constructs an ellipse to represent the orientation and arrangement of these points. The center of this ellipse, $O(\bar{X}, \bar{Y})$, is defined as the geometric center of these n points and is calculated as:

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \bar{Y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{2}$$

The coordinates (x_i, y_i) of each point are normalized to the deviation from the mean areal center point:

$$\tilde{x}_i = x_i - \bar{X}, \quad \tilde{y}_i = y_i - \bar{Y} \tag{3}$$

Equation 3 can be seen as a coordinate translation to the new origin $(\bar{X}, \bar{Y})$. If we rotate the new coordinate system counterclockwise about O by angle θ ($0<\theta \leq 2\pi$) and get the new coordinate system X_o,Y_o, the standard deviation along the X_o axis σ_x and the Y_o axis σ_y is calculated, respectively, as:

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} (\tilde{y}_i \sin\theta + \tilde{x}_i \cos\theta)^2}{n}} \tag{4}$$

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{n} (\tilde{y}_i \cos\theta - \tilde{x}_i \sin\theta)^2}{n}} \tag{5}$$

The long/short axis of the SDE is along the direction that has the maximum/minimum standard deviation. Let σ_{max} and σ_{min} be the length of the semi-long axis and the semi-short axis of the SDE. The angle of rotation θ_m of the long/short axis is given by Equation 6 (Yuill, 1971):

$$\tan\theta_m = -\frac{A \pm B}{C} \tag{6}$$

$$A = \sum_{i=1}^{n} \tilde{x}_i^{\,2} - \sum_{i=1}^{n} \tilde{y}_i^{\,2} \tag{7}$$

$$C = 2\sum_{i=1}^{n} \tilde{x}_i \tilde{y}_i \tag{8}$$

$$B = \sqrt{A^2 + C^2} \tag{9}$$

Here, $\pm$ indicates two rotation angles θ_{max}, θ_{min} corresponding to the long and short axes.

3.2 | Anisotropic density-based clusters

In order to introduce an anisotropic perspective to density-based clustering algorithms such as DBSCAN, we have to revise the definition of an *Eps*-neighborhood of a point. First, the original *Eps*-neighborhood of a point in a dataset D is defined by DBSCAN, as given in Definition 1.

> **Definition 1** (Eps-neighborhood of a point): The Eps-neighborhood $N_{Eps}(p_i)$ of point p_i is defined as all the points within the scan circle centered at p_i with a radius Eps, which can be expressed as:
>
> $$N_{Eps}(p_i) = \{p_j(x_j, y_j) \in D | dist(p_i, p_j) \leq Eps\} \tag{10}$$

Such a scan circle results in an isotropic perspective on clustering. However, as we discuss above, an anisotropic assumption will be more appropriate for some geographic phenomena. Intuitively, in order to introduce anisotropicity into DBSCAN, we can employ a scan ellipse instead of a circle to define the *Eps*-neighborhood of each point. Before we give a definition of the *Eps*-ellipse-neighborhood of a point, it is necessary to define a set of points around a point (the search-neighborhood of a point), which is used to derive the scan ellipse; see Definition 2.

> **Definition 2** (Search-neighborhood of a point): A set of points $S(p_i)$ around point p_i is called a search-neighborhood of point p_i and can be defined in two ways:
>
> 1. The Eps-neighborhood $N_{Eps}(p_i)$ of point p_i.
> 2. The kth nearest neighbor $KNN(p_i)$ of point p_i. Here, $k = MinPts$ and $KNN(p_i)$ does not include p_i itself.

After determining the search-neighborhood of a point, it is possible to define the *Eps*-ellipse-neighborhood region (see Definition 3) and the *Eps*-ellipse-neighborhood (see Definition 4) of each point.

> **Definition 3** (Eps-ellipse-neighborhood region of a point): An ellipse ER_i is called an Eps-ellipse-neighborhood region of a point p_i *iff*:

1. Ellipse ER_i is centered at point p_i.
2. Ellipse ER_i is scaled from the standard deviation ellipse SDE_i computed from the search-neighborhood $S(p_i)$ of point p_i.
3. $\frac{\sigma'_{max}}{\sigma'_{min}} = \frac{\sigma_{max}}{\sigma_{min}}$ where σ'_{max}, σ'_{min} and σ_{max}, σ_{min} are the length of the semi-long and semi-short axes of ellipse ER_i and ellipse SDE_i.
4. $Area(ER_i) = \pi ab = \pi Eps^2$.

According to Definition 3, the *Eps*-ellipse-neighborhood region of a point is computed based on the search-neighborhood of a point. Since there are two definitions of the search-neighborhood of a point (see Definition 2), each point should have a unique *Eps*-ellipse-neighborhood region given *Eps* (using the first definition in Definition 2) or *MinPts* (using the second definition in Definition 2), as long as the search-neighborhood of the current point has at least two points for the computation of the standard deviation ellipse.

Definition 4 (Eps-ellipse-neighborhood of a point) An Eps-ellipse-neighborhood $EN_{Eps}(p_i)$ of point p_i is defined as all the points inside the ellipse ER_i, which can be expressed as:

$$EN_{Eps}(p_i) = \left\{ p_j(x_j, y_j) \in D \middle| \frac{((y_j - y_i)\sin\theta_{max} + (x_j - x_i)\cos\theta_{max})^2}{a^2} + \frac{((y_j - y_i)\cos\theta_{max} - (x_j - x_i)\sin\theta_{max})^2}{b^2} \leq 1 \right\} \quad (11)$$

There are two kinds of point in a cluster obtained from DBSCAN: *core point* and *border point*. Core points have at least *MinPts* points in their *Eps*-neighborhood, while border points have less than *MinPts* points in their *Eps*-neighborhood but are *density reachable* from at least one core point. Our anisotropic clustering algorithm has a similar definition of core point and border point. The notions of directly anisotropic density reachable and core point are illustrated bellow; see Definition 5.

Definition 5 (Directly anisotropic density reachable): A point p_j is directly anisotropic density reachable from point p_i with respect to *Eps* and *MinPts iff*:

1. $p_j \in EN_{Eps}(p_i)$.
2. $|EN_{Eps}(p_i)| \geq MinPts$ (core point condition).

If point p is directly anisotropic reachable from point q, then point q must be a core point which has no less than *MinPts* points in its *Eps*-ellipse-neighborhood. Similar to the notion of density reachable in DBSCAN, the notion of anisotropic density reachable is given in Definition 6.

Definition 6 (Anisotropic density reachable): A point p is anisotropic density reachable from point q with respect to *Eps* and *MinPts* if there exists a chain of points $p_1, p_2, \ldots, p_n$ ($p_1 = q$, $p_n = p$) such that point p_{i+1} is directly anisotropic density reachable from p_i.

Although anisotropic density reachability is not a symmetric relation, if such a directly anisotropic density reachable chain exits, then except for point p_n, the other $n - 1$ points are all core points. If point p_n is also a core point, then symmetrically point p_1 is also density reachable from point p_n. That means that if two points p, q are anisotropic density reachable from each other, then both of them are core points and belong to the same cluster.

Equipped with the above definitions, we are able to define our anisotropic density-based notion of clustering. DBSCAN includes both core points and border points in its clusters. In our clustering algorithm, only core points will be treated as cluster points. Border points will be excluded from clusters and treated as noise points, because otherwise many noise points will be included in clusters according to the experimental results. In short, a cluster (see Definition 7) is defined as a subset of points from the whole point dataset in which each two points are anisotropic density reachable from each other. Noise points (see Definition 8) are defined as the subset of points from the entire point dataset for which each point has less than *MinPts* points in its *Eps*-ellipse-neighborhood.

Definition 7 (Cluster): Let D be a point dataset. A cluster C is a non-empty subset of D with respect to *Eps* and *MinPts iff*:

1. $\forall p \in C$, $EN_{Eps}(p) \geq MinPts$.
2. $\forall p, q \in C$, p, q are anisotropic density reachable from each other with respect to *Eps* and *MinPts*.

A cluster C has two attributes. $\forall p \in C$ and $\forall q \in D$, if p is anisotropic density reachable from q w.r.t. *Eps* and *MinPts*, then

1. $q \in C$.
2. There must be a directly anisotropic density reachable points chain $C(q, p)$: $p_1, p_2, \ldots, p_n$ ($p_1 = q$, $p_n = p$) such that p_{i+1} is directly anisotropic density reachable from p_i. Then $\forall p_i \in C(q, p)$, $p_i \in C$.

Definition 8 (Noise): Let D be a point dataset. A point p is a noise point with respect to *Eps* and *MinPts* if $p \in D$ and $EN_{Eps}(p) < MinPts$.

Let $C_1, C_2, \ldots, C_k$ be the clusters of the point dataset D with respect to *Eps* and *MinPts*. From Definition 8, if $p \in D$ and $EN_{Eps}(p) < MinPts$, then $\forall C_i \in \{C_1, C_2, \ldots, C_k\}$, $p \notin C_i$.

According to Definition 2, and in contrast to a simple scan circle, there are at least two ways to define a search-neighborhood of the center point p_i. Thus, ADCN can be divided into an ADCN-Eps variant that uses the *Eps*-neighborhood $N_{Eps}(p_i)$ as the search-neighborhood and an ADCN-KNN variant that uses kth nearest neighbors $KNN(p_i)$ as the search-neighborhood. Figures 2 and 3 illustrate the related definitions for ADCN-Eps and ADCN-KNN. The red points in both figures represent current center points. The blue points indicate the two different

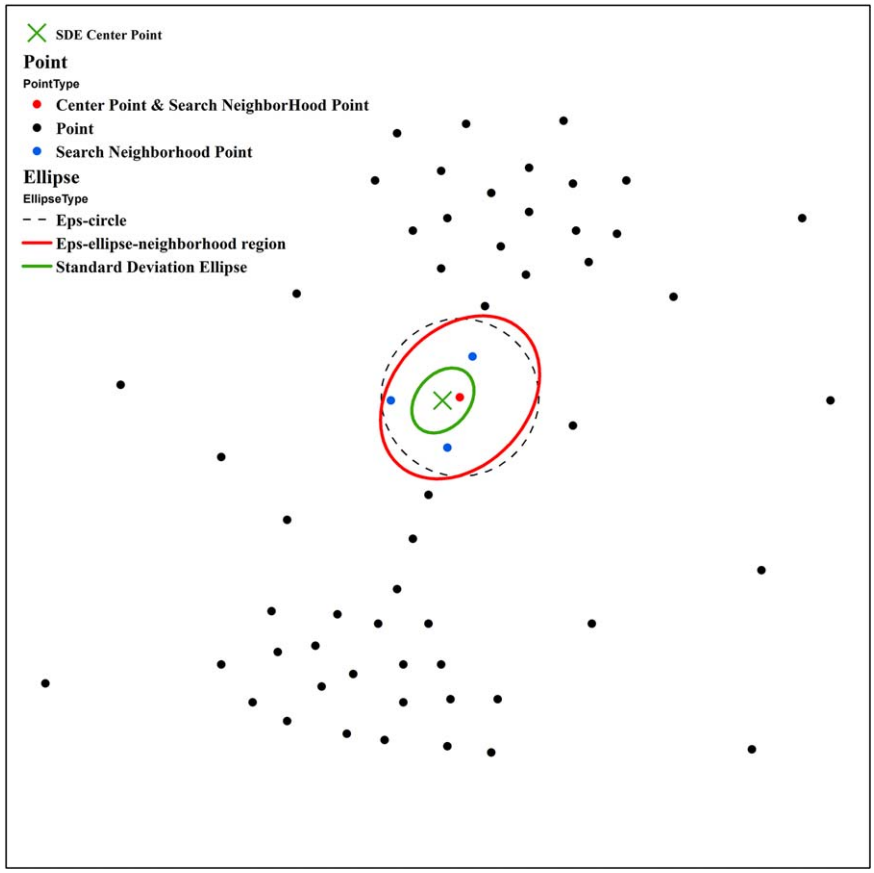

FIGURE 2 Illustration for ADCN-Eps

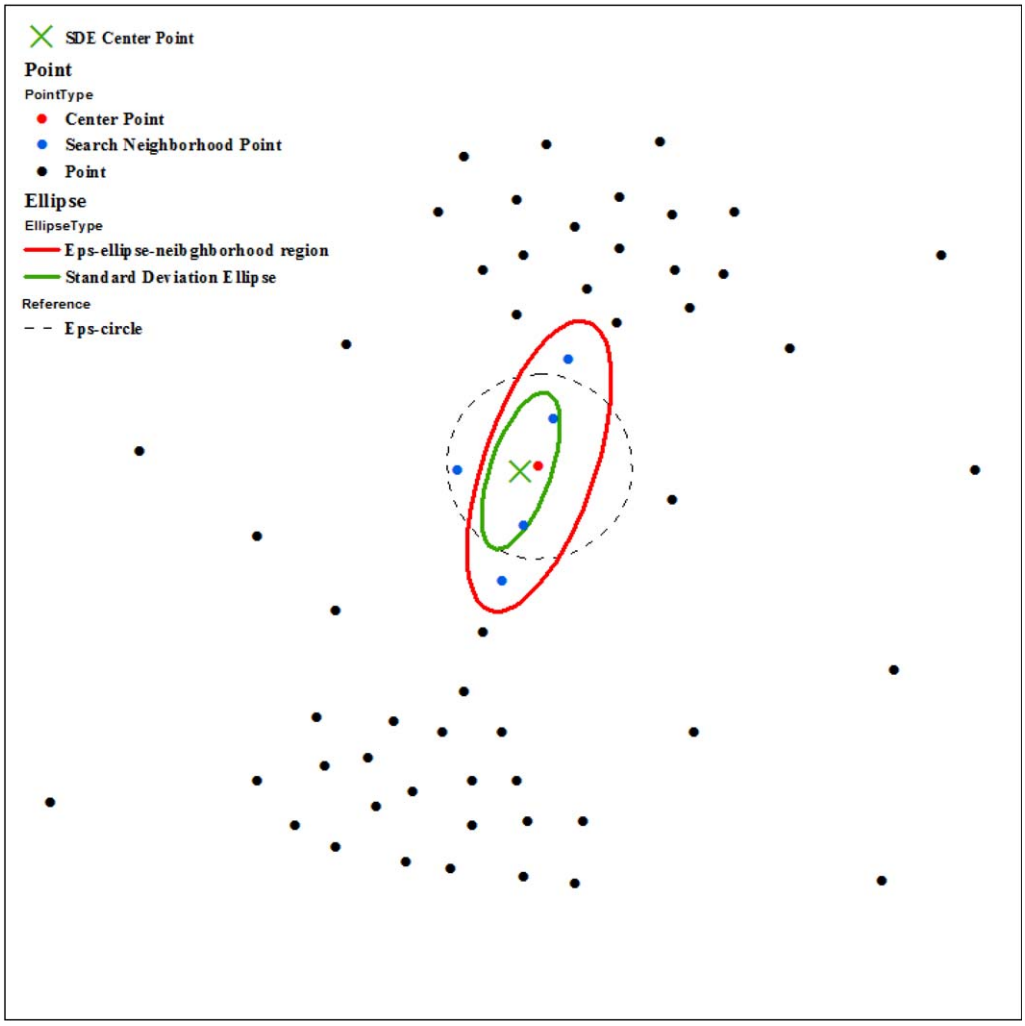

FIGURE 3 Illustration for ADCN-KNN

search-neighborhoods of the corresponding center points according to Definition 2. Note that for ADCN-Eps, the center point is also part of its search-neighborhood, which is not true for ADCN-KNN. The green ellipses and green crosses stand for the standard deviation ellipses constructed from the corresponding search-neighborhood and their center points. The red ellipses are *Eps*-ellipse-neighborhood regions, while the dashed-line circles indicate a DBSCAN-like scan circle. As can be seen, ADCN-KNN will exclude the point to the left of the linear bridge pattern, while DBSCAN includes it.

3.3 | ADCN algorithms

From the definitions provided above it follows that our *anisotropic density-based clustering with noise* algorithm takes the same parameters (*MinPts* and *Eps*) as DBSCAN, and that they have to be decided before clustering. This is for good reasons, as the proper selection of DBSCAN parameters has been well studied and ADCN can easily replace DBSCAN without any changes to established workflows.

As shown in Algorithm 1, ADCN starts with an arbitrary point p_i in a point dataset D and discovers all the *core* points which are anisotropic density reachable from point p_i. According to Definition 2, there are two ways to get the search-neighborhood of point p_i which will result in different *Eps*-ellipse-neighborhoods $EN_{Eps}(p_j)$ based on the derived

Algorithm 1. ADCN(*D*, *MinPts*, *Eps*)

Input: A set of *n* points *D*(*X*, *Y*); *MinPts*; *Eps*

Output: Clusters with different labels $C_i[\cdot]$; a set of noise points *Noi*[·]

```
1   for each point p_i(x_i, y_i) in the set of points D(X, Y) do
2       mark p_i as visited;
3       //Get Eps-ellipse-neighborhood EN_Eps(p_i) of p_i
4       ellipseRegionQuery(p_i, D, MinPts, Eps);
5       if |EN_Eps(p_i)|<MinPts then
6           add p_i to the noise set Noi[·];
7       else
8           create a new cluster C_i[·];
9           add p_i to C_i[·];
10          for each point p_j(x_j, y_j) in EN_Eps(p_i) do
11              if p_j is not visited then
12                  mark p_j as visited;
13                  //Get Eps-ellipse-neighborhood EN_Eps(p_j) of point p_j
14                  ellipseRegionQuery(p_j, D, MinPts, Eps);
15                  if |EN_Eps(p_j)| ≥ MinPts then
16                      let EN_Eps(p_i) be the merged set of EN_Eps(p_i) and EN_Eps(p_j);
17                      add p_j to the current cluster C_i[·];
18                  else
19                      add p_j to the noise set Noi[·];
20                  end
21              end
22          end
23      end
24  end
```

Eps-ellipse-neighborhood region in Algorithm 2. Hence, ADCN can be implemented by two algorithms (ADCN-Eps and ADCN-KNN). Algorithm 2 needs to take care of situations where all points of the search-neighborhood $S(p_i)$ of point p_i are strictly on the same line. In this case, the short axis of the *Eps*-ellipse-neighborhood region ER_i becomes *zero* and its long axis becomes *infinity*. This means that $EN_{Eps}(p_i)$ is reduced to a straight line. The process of constructing the *Eps*-ellipse-neighborhood $EN_{Eps}(p_i)$ of point p_i becomes a point-on-line query.

According to Algorithm 3, ADCN-Eps uses the *Eps*-neighborhood $N_{Eps}(p_i)$ of point p_i as the search-neighborhood, which will be used later to construct the standard deviation ellipse. In contrast, ADCN-KNN (Algorithm 4) uses a *k*th nearest neighborhood of point p_i as the search-neighborhood. Here, point p_i will not be included in its *k*th nearest neighborhood. As can be seen, the runtimes of ADCN-Eps and ADCN-KNN are heavily dominated by the search-neighborhood query which is executed on each point. Hence, the time complexities of ADCN, DBSCAN, and OPTICS are $O(n^2)$ without a spatial index and $O(n \log n)$ otherwise.

Algorithm 2. ellipseRegionQuery(p_i, D, $MinPts$, Eps)

Input: p_i, D, $MinPts$, Eps

Output: Eps-ellipse-neighborhood $EN_{Eps}(p_i)$ of point p_i

```
//Get the search-neighborhood S(p_i) of point p_i. ADCN-Eps and ADCN-KNN use different functions
ADCN-Eps: searchNeighborhoodEps(p_i, D, Eps); ADCN-KNN: searchNeighborhoodKNN(p_i, D, MinPts);
compute the standard deviation ellipse SDE_i based on the search-neighborhood S(p_i) of point p_i;
scale ellipse SDE_i to get the Eps-ellipse-neighborhood region ER_i of point p_i to make sure Area(ER_i)=π×Eps^2;
if the length of the short axis of ER_i==0 then
    //The Eps-ellipse-neighborhood region ER_i of point p_i is diminished to a straight line. Get Eps-ellipse-neighborhood EN_Eps(p_i) of point p_i by finding all points on this straight line ER_i
else
    //The Eps-ellipse-neighborhood region ER_i of point p_i is an ellipse. Get Eps-ellipse-neighborhood EN_Eps(p_i) of point p_i by finding all the points inside ellipse ER_i
end
return EN_Eps(p_i);
```

Algorithm 3. searchNeighborhoodEps(p_i, D, Eps)

Input: p_i, D, Eps

Output: The search-neighborhood $S(p_i)$ of point p_i

```
//This function is used in ADCN-Eps//Get all the points whose distance from point p_i is less than Eps
for each point p_j(x_j, x_j) in the set of points D(X, Y) do
    if sqrt((x_i−x_j)^2+(y_i−y_j)^2) ≤ Eps then
        add point p_j to S(p_i);
end
return S(p_i);
```

4 | EXPERIMENTS AND PERFORMANCE EVALUATION

In this section, we will evaluate the performance of ADCN from two perspectives: clustering quality and clustering efficiency. In contrast to the scan circle of DBSCAN, there are at least two ways to determine an anisotropic neighborhood. This leads to two realizations of ADCN, namely ADCN-KNN and ADCN-Eps. We will evaluate their performance using DBSCAN and OPTICS as baselines. We selected OPTICS as an additional baseline, as it is commonly used to address some of DBSCAN's shortcomings with respect to varying densities.

According to the research contributions outlined in Section 1, we intend to establish: (a) that at least one of the ADCN variants performs as well as DBSCAN (and OPTICS) for cases that do not explicitly benefit from an anisotropic perspective; (b) that the aforementioned variant performs better than the baselines for cases that *do* benefit from an anisotropic perspective; and finally (c) that the test cases include point patterns typically used to test density-based clustering algorithms as well as *real-world* cases that highlight the need for developing ADCN in the first place. In addition, we will show runtime results for all four algorithms.

Algorithm 4. searchNeighborhoodKNN(p_i, D, $MinPts$)

Input: p_i, D, $MinPts$

Output: The search-neighborhood $S(p_i)$ of point p_i

```
1   //This function is used in ADCN-KNN//Get the kth nearest neighbor of point p_i, excluding p_i itself
2   KNNArray = new Array(MinPts);
3   distanceArray = new Array(|D|);
4   KNNLabelArray = new Array(|D|);
5   for each point p_j(x_j, y_j) in the set of points D(X, Y) do
6       KNNLabelArray[j] = 0;
7       distanceArray[j] = sqrt((x_i - x_j)^2 + (y_i - y_j)^2);
8       if j==i then
9           KNNLabelArray[j] = 1;
10  end
11  for each k in 0: (MinPts-1) do
12      minDist = ∞;
13      minDistID = 0;
14      for each j in 0: |D| do
15          if KNNLabelArray[j]! = 1 then
16              if minDist > distanceArray[j] then
17                  minDist = distanceArray[j];
18                  minDistID = j;
19      end
20      KNNLabelArray[minDistID] = 1;
21      KNNArray[k] = minDistID;
22      add the point with minDistID as ID to S(p_i);
23  end
24  return S(p_i);
```

4.1 | Experiment designs

We have designed several spatial point patterns as test cases for our experiments. More specifically, we generated 20 test cases with three different noise settings for each of them. These consist of 12 synthetic and 8 real-world use cases, resulting in a total of 60 case studies. Note that our test cases do not only contain linear features such as road networks, but also cases that are typically used to evaluate algorithms such as DBSCAN (e.g., clusters of ellipsoid and rectangular shapes).

In order to simulate a "ground truth" for the synthetic cases, we created polygons to indicate different clusters and randomly generated points within these polygons and outside them. We took a similar approach for the eight real-world cases. The only difference is that the polygons for real-world cases have been generated from buffer zones with a 3 m radius of the real-world features (e.g., existing road networks). This allows us to simulate patterns that typically occur in geotagged social media data.

Although we use this approach to simulate the corresponding spatial point process, the distinction between clustered and noise points in the resulting spatial point patterns may not be so obvious, even from a human's perspective.

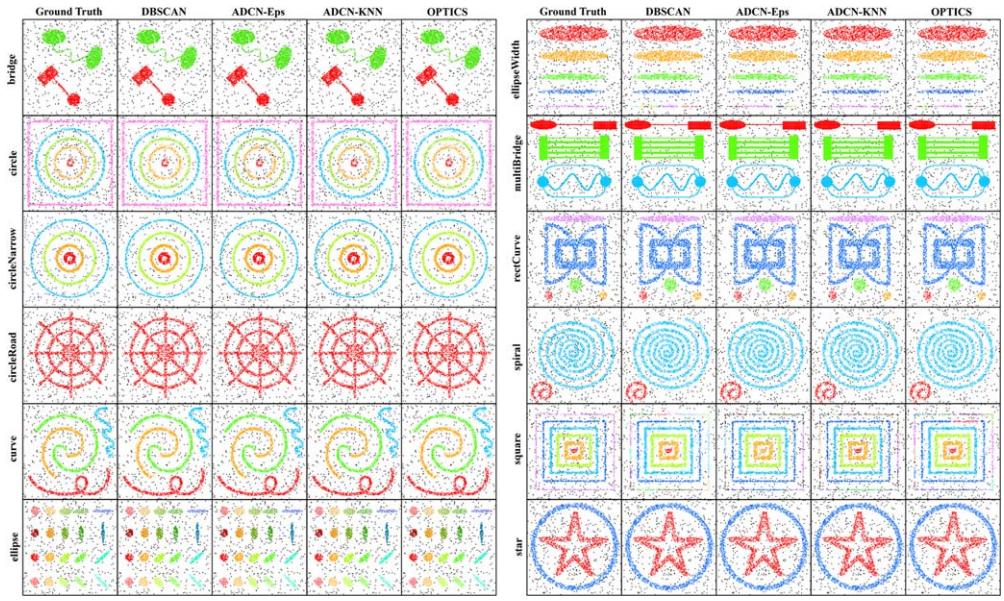

FIGURE 4 Ground truth and best clustering result comparison for 12 synthesis cases

To avoid cases in which it is unreasonable to expect algorithms and humans to differentiate between noise and pattern, we introduced a clipping buffer of 0, 5, and 10 m. For comparison, the typical position accuracy of GPS sensors on smartphones and GPS collars for wildlife tracking is about 3–15 m (Wing, Eklund, & Kellogg, 2005) (and can decline rapidly in urban canyons).

The generated spatial point patterns of 12 synthetic and 8 real-world use cases with 0 m buffer distance are shown in the first column of Figures 4 and 5. Note that in all test cases, points generated from different polygons are pre-labeled with different cluster IDs, which are indicated by different colors in the first column of Figures 4 and 5. Points generated outside polygons are pre-labeled as *noise*, which are shown in *black*. These generated spatial point patterns serve as the *ground truth*, used in our clustering quality-evaluation experiments.

In order to demonstrate the strength of ADCN, we need to compare the performance of ADCN with that of DBSCAN and OPTICS from two perspectives: clustering quality and clustering efficiency. The experiment designs are as follows:

- As for clustering quality evaluation, we use several clustering quality indices to quantify how good the clustering results are. In this work, we use normalized mutual information (NMI) and the Rand index. We will explain these two indices in detail in Section 4.3. We stepwise tested every possible parameter combination of *Eps*, *MinPts* computationally on each test case. For each clustering algorithm, we select the parameter combination which has the highest NMI or Rand index. By comparing the maximum of NMI and Rand index across different clustering algorithms in each test case, we can find the best clustering technique.
- As for clustering efficiency evaluation, we generate spatial point patterns with different numbers of points by using the polygons of each test case mentioned earlier. For each clustering algorithm and each number of points setting, we computed the average runtime. By constructing a runtime curve of each clustering algorithm, we were able to compare their runtime efficiency.

4.2 | Test environment

In order to compare the performance of ADCN with that of DBSCAN and OPTICS, we developed a JavaScript test environment to generate patterns and compare results. This allows us to generate use cases in a Web browser, such as

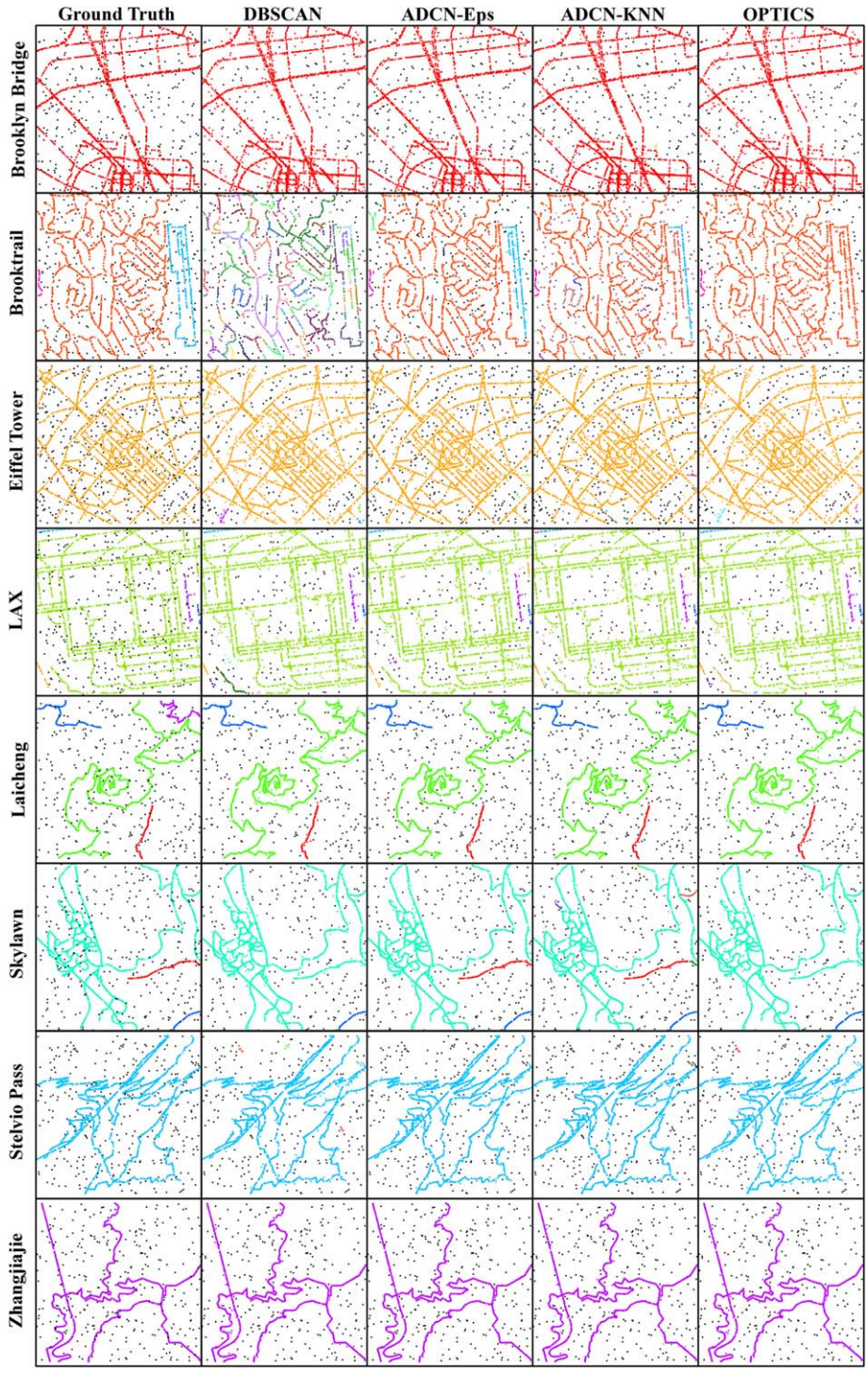

FIGURE 5 Ground truth and best clustering result comparison for eight real-world cases

Firefox or Chrome, or load them from a GIS, change noise settings, determine DBSCAN's Eps via a KNN distance plot, perform different evaluations, compute runtimes, index the data via an R-tree, and save and load the data. Consequently, what matters is the *runtime behavior*, not the exact performance (for which JavaScript would not be a suitable

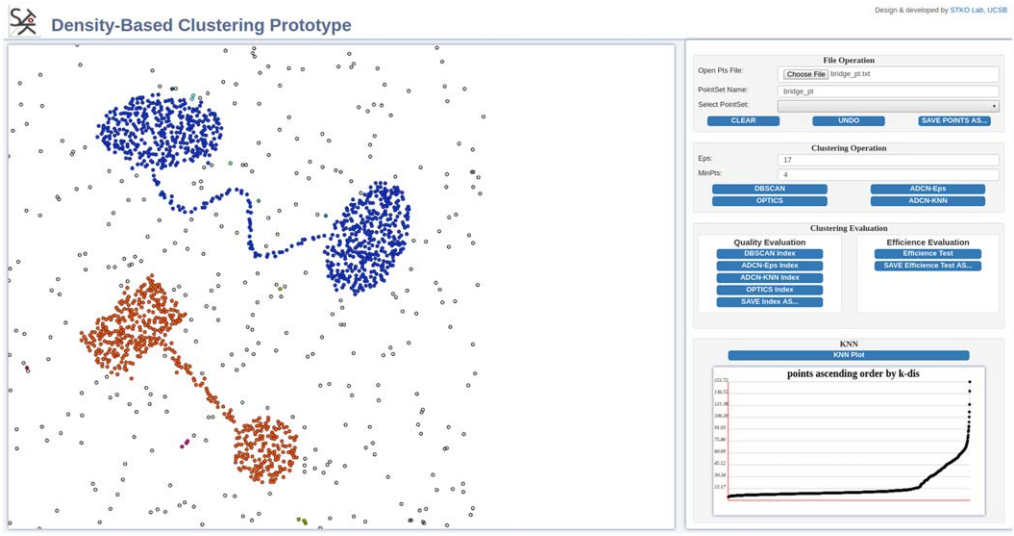

FIGURE 6 The density-based clustering test environment

choice). All cases have been performed on a *cold* setting (i.e., without any caching using an Intel i5–5300U CPU with 8-GB RAM on an Ubuntu 16.04 system). This JavaScript test environment, as well as all the test cases, can be downloaded from http://stko.geog.ucsb.edu/adcn/.

Figure 6 shows a snapshot of this test environment. The system has two main panels. The map panel on the left side is an interactive canvas in which the user can click and create data points. The tool bar on the right side is composed of input boxes, selection boxes, and buttons which are divided into different groups. Each group is used for a specific purpose, which will be discussed below.

The "File Operation" tool group is used for point dataset manipulation. For simplicity, our environment defines a simple format for point datasets. Conceptually, a point dataset is a table containing the coordinates of points, their ground truth memberships, and the memberships produced during the experiments. The ground truth and experimental memberships are then compared to evaluate the cluster algorithms. The "Open Pts File" box is used for loading point datasets produced by other GIS. The data points can also be abstract points which represent objects, such as documents (Fabrikant & Montello, 2008), in a feature space. The prototype takes the coordinates of points and maps out these points after rescaling their coordinates based on the size of the map panel. During the clustering process, it uses Euclidean distance as the distance measure.

The "Clustering Operation" tool group is used to operate clustering tasks. The "Eps" and "MinPts" input boxes let users enter the clustering parameters for all clustering algorithms. The "DBSCAN," "OPTICS," "ADCN-Eps," and "ADCN-KNN" buttons are for running the algorithms. As for the implementation of DBSCAN and OPTICS, we used a JavaScript clustering library from GitHub (https://github.com/uhho/density-clustering). This library has basic implementations of DBSCAN, OPTICS, K-MEANS, and some other clustering algorithms without any spatial indexes. Our ADCN-KNN and ADCN-Eps algorithms were implemented using the same data structures as used in the library. Such an implementation ensures that the evaluation result will reflect the differences of the algorithms rather than be affected by the specific data structures used in the implementations. Finally, we implemented an R-tree spatial index to accelerate the neighborhood search. We have used the R-tree JavaScript library from GitHub (https://github.com/imbcmdth/RTree).

The "Clustering Evaluation" tool group is composed of "Quality Evaluation" and "Efficiency Evaluation" subgroups. As for the clustering quality evaluation, we implemented two metrics—NMI and Rand index—to quantify the goodness of the clustering results. The first four buttons in this subgroup will run the corresponding clustering algorithm on the current dataset, based on all possible parameter combinations. They will

compute two clustering evaluation indexes for each clustering result. The "SAVE Index As..." button will save these results to a text file.

Efficiency evaluation is another important part for comparing clustering algorithms. The "Efficiency Evaluation" button will run these four clustering algorithms on datasets with different sizes. The "SAVE Efficiency Test As..." button can be used further to save the result into a text file.

Finally, the "KNN" tool group is used to draw the KNN plot of the current dataset based on the *MinPts* parameter specified by the user. For each point, the KNN plot obtains the distance between the current point and its *k*th nearest point (here *K* is *MinPts*). Then, it ranks the *k*th nearest distance of each point in ascending order. The KNN plot can be used to estimate the appropriate *Eps* for the current point dataset given *MinPts*. More details on this estimation can be found in the original DBSCAN paper (Ester et al., 1996).

Note that we provide the test environment to make our results reproducible and to offer a reusable implementation of ADCN, without implying that JavaScript would be the language of choice for future, large-scale applications of ADCN.

4.3 | Evaluation of clustering quality

We use two clustering quality indices—NMI and Rand index—to measure the quality of clustering results of all algorithms. NMI originates from information theory and has been revised as an objective function for clustering ensembles (Strehl & Ghosh, 2002). NMI evaluates the accumulated mutual information shared by the clusters from different clustering algorithms. Let n be the number of points in a point dataset D, with $X=(X_1, X_2, \ldots, X_r)$ and $Y=(Y_1, Y_2, \ldots, Y_s)$ two clustering results from the same or different clustering algorithms. Note that noise points will be treated as their own cluster. Let $n_h^{(x)}$ be the number of points in cluster X_h and $n_l^{(y)}$ be the number of points in cluster Y_l. Let $n_{h,l}^{(x,y)}$ be the number of points in the intersect of cluster X_h and cluster Y_l. Then the normalized mutual information $\Phi^{(NMI)}(X,Y)$ is defined in Equation 12 as the similarity between two clustering results X and Y:

$$\Phi^{(NMI)}(X,Y)=\frac{\sum_{h=1}^{r}\sum_{l=1}^{s} n_{h,l}^{(x,y)}\log\frac{n\cdot n_{h,l}^{(x,y)}}{n_h^{(x)}\cdot n_l^{(y)}}}{\sqrt{\left(\sum_{h=1}^{r} n_h^{(x)}\log\frac{n_h^{(x)}}{n}\right)\left(\sum_{l=1}^{s} n_l^{(y)}\log\frac{n_l^{(y)}}{n}\right)}} \tag{12}$$

The Rand index (Rand, 1971) is another objective function for clustering ensembles from a different perspective. It evaluates to what degree two clustering algorithms share the same relationships between points. Let a be the number of pairs of points in D that are in the same cluster in X and in the same cluster in Y. b is the number of pairs of points in D that are in different clusters in X and Y. c is the number of pairs of points in D that are in the same cluster in X and in different clusters in Y. Finally, d is the number of pairs of points in D that are in different clusters in X and in the same cluster in Y. The Rand index $\Phi^{(Rand)}(X,Y)$ is then defined as:

$$\Phi^{(Rand)}(X,Y)=\frac{a+b}{a+b+c+d} \tag{13}$$

For both NMI and the Rand index, larger values indicate higher similarity between two clustering results. If a ground truth is available, both NMI and the Rand index can be used to compute the similarity between the results of an algorithm and the corresponding ground truth. This is called the *extrinsic method* (Han et al., 2011).

We use the aforementioned 20 test cases to evaluate the clustering quality of DBSCAN, ADCN-Eps, ADCN-KNN, and OPTICS. All these four algorithms take the same parameters (*Eps*, *MinPts*). As there are no established methods to determine the best overall parameter combination (we use KNN distance plots to estimate *Eps*) with respect to NMI and the Rand index, we stepwise tested every possible parameter combination of *Eps*, *MinPts* computationally. An interactive 3D visualization of the NMI and the Rand index results with changing *Eps* and *MinPts* for the *spiral* case with 0 m buffer distance can be accessed online (http://stko.geog.ucsb.edu/adcn/). Table 1 shows the maximum NMI and the Rand index results for the four algorithms over all test cases. Note that for each case,

TABLE 1 Clustering quality comparisons

		NMI				Rand index			
Case	Buffer	DBSCAN	ADCN-Eps	ADCN-KNN	OPTICS	DBSCAN	ADCN-Eps	ADCN-KNN	OPTICS
bridge	0 m	0.937	0.957	0.957	0.937	0.985	0.991	0.992	0.985
	5 m	0.948	0.966	0.967	0.949	0.989	0.993	0.994	0.989
	10 m	0.938	0.973	0.968	0.944	0.988	0.995	0.995	0.989
circle	0 m	0.864	0.865	0.912	0.864	0.955	0.964	0.978	0.955
	5 m	0.859	0.897	0.916	0.859	0.955	0.974	0.978	0.955
	10 m	0.864	0.911	0.923	0.864	0.960	0.979	0.982	0.960
circleNarrow	0 m	0.914	0.951	0.958	0.914	0.974	0.988	0.991	0.974
	5 m	0.939	0.946	0.965	0.939	0.983	0.987	0.993	0.983
	10 m	0.923	0.962	0.962	0.923	0.976	0.991	0.992	0.976
circleRoad	0 m	0.689	0.704	0.725	0.689	0.934	0.945	0.952	0.934
	5 m	0.737	0.758	0.779	0.737	0.950	0.963	0.962	0.951
	10 m	0.730	0.778	0.821	0.730	0.946	0.963	0.971	0.946
curve	0 m	0.918	0.946	0.955	0.918	0.978	0.989	0.991	0.978
	5 m	0.924	0.947	0.956	0.924	0.980	0.990	0.992	0.980
	10 m	0.916	0.943	0.947	0.916	0.978	0.988	0.989	0.978
ellipse	0 m	0.978	0.982	0.976	0.978	0.996	0.997	0.995	0.996
	5 m	0.979	0.982	0.980	0.979	0.996	0.997	0.996	0.996
	10 m	0.975	0.980	0.978	0.974	0.996	0.997	0.996	0.996
ellipseWidth	0 m	0.917	0.935	0.935	0.917	0.985	0.989	0.988	0.985
	5 m	0.919	0.933	0.939	0.919	0.988	0.989	0.989	0.988
	10 m	0.931	0.938	0.941	0.931	0.990	0.991	0.991	0.989
multiBridge	0 m	0.935	0.790	0.957	0.938	0.983	0.935	0.992	0.984
	5 m	0.958	0.883	0.977	0.958	0.992	0.968	0.996	0.992
	10 m	0.964	0.830	0.985	0.964	0.994	0.947	0.998	0.994
rectCurve	0 m	0.886	0.893	0.907	0.886	0.963	0.969	0.973	0.963
	5 m	0.909	0.910	0.908	0.915	0.974	0.977	0.974	0.974
	10 m	0.921	0.923	0.911	0.922	0.975	0.977	0.977	0.975
spiral	0 m	0.740	0.756	0.774	0.740	0.913	0.930	0.938	0.913
	5 m	0.776	0.812	0.809	0.776	0.927	0.946	0.948	0.927
	10 m	0.745	0.788	0.795	0.745	0.918	0.950	0.952	0.918
square	0 m	0.745	0.751	0.794	0.745	0.934	0.920	0.944	0.934
	5 m	0.751	0.778	0.830	0.752	0.932	0.928	0.959	0.932
	10 m	0.744	0.716	0.801	0.743	0.935	0.893	0.944	0.935
star	0 m	0.887	0.901	0.914	0.887	0.968	0.977	0.980	0.968
	5 m	0.903	0.899	0.916	0.900	0.974	0.977	0.982	0.974
	10 m	0.902	0.778	0.909	0.902	0.974	0.924	0.981	0.974
Brooklyn Bridge	0 m	0.378	0.542	0.490	0.378	0.888	0.930	0.925	0.888
	5 m	0.442	0.604	0.579	0.440	0.900	0.943	0.941	0.900
	10 m	0.504	0.639	0.581	0.507	0.915	0.950	0.944	0.915
Brooktrail	0 m	0.441	0.431	0.421	0.440	0.742	0.765	0.756	0.742
	5 m	0.476	0.512	0.489	0.475	0.750	0.825	0.800	0.750
	10 m	0.387	0.555	0.498	0.387	0.712	0.852	0.799	0.711
Eiffel Tower	0 m	0.397	0.481	0.492	0.397	0.851	0.882	0.898	0.851

(Continues)

TABLE 1 (Continued)

Case	Buffer	NMI				Rand index			
		DBSCAN	ADCN-Eps	ADCN-KNN	OPTICS	DBSCAN	ADCN-Eps	ADCN-KNN	OPTICS
	5 m	0.459	0.566	0.571	0.459	0.868	0.906	0.921	0.868
	10 m	0.411	0.553	0.553	0.411	0.861	0.907	0.923	0.861
LAX	0 m	0.557	0.607	0.593	0.557	0.867	0.898	0.905	0.867
	5 m	0.591	0.667	0.584	0.591	0.883	0.921	0.903	0.883
	10 m	0.485	0.590	0.637	0.479	0.857	0.903	0.925	0.857
Laicheng	0 m	0.768	0.807	0.804	0.768	0.857	0.874	0.874	0.857
	5 m	0.761	0.815	0.808	0.761	0.856	0.878	0.905	0.856
	10 m	0.773	0.823	0.809	0.773	0.861	0.880	0.911	0.861
Skylawn	0 m	0.618	0.822	0.733	0.618	0.871	0.956	0.927	0.871
	5 m	0.642	0.690	0.807	0.642	0.877	0.899	0.955	0.877
	10 m	0.729	0.703	0.822	0.729	0.927	0.905	0.957	0.927
Stelvio Pass	0 m	0.640	0.715	0.717	0.656	0.945	0.962	0.963	0.946
	5 m	0.739	0.791	0.768	0.739	0.962	0.974	0.975	0.962
	10 m	0.686	0.798	0.766	0.686	0.953	0.975	0.978	0.953
Zhangjiajie	0 m	0.760	0.832	0.799	0.760	0.964	0.978	0.976	0.964
	5 m	0.772	0.868	0.839	0.772	0.967	0.987	0.982	0.967
	10 m	0.835	0.911	0.873	0.835	0.978	0.991	0.990	0.978

the best parameter combination with the maximum NMI does not necessarily yield the maximum Rand index. However, among all these 60 cases, there are 39, 35, 27, and 39 cases for DBSCAN, ADCN-Eps, ADCN-KNN, and OPTICS, respectively, in which the best parameter combination for the maximum NMI is also the maximum Rand index. For those cases where parameter combinations of maximum NMI and maximum Rand index do not match, their parameters tend to be close to each other because NMI and the Rand index values are changing continuously, while *Eps* and *MinPts* increase. This indicates that the NMI and Rand index have a medium to high similarity in terms of measuring the clustering quality.

As for the 60 test cases, ADCN-KNN has a higher maximum NMI/Rand index than DBSCAN in 55 cases and a higher maximum NMI/Rand index than OPTICS in 55 cases; see Figures 7 and 8. Furthermore, ADCN-

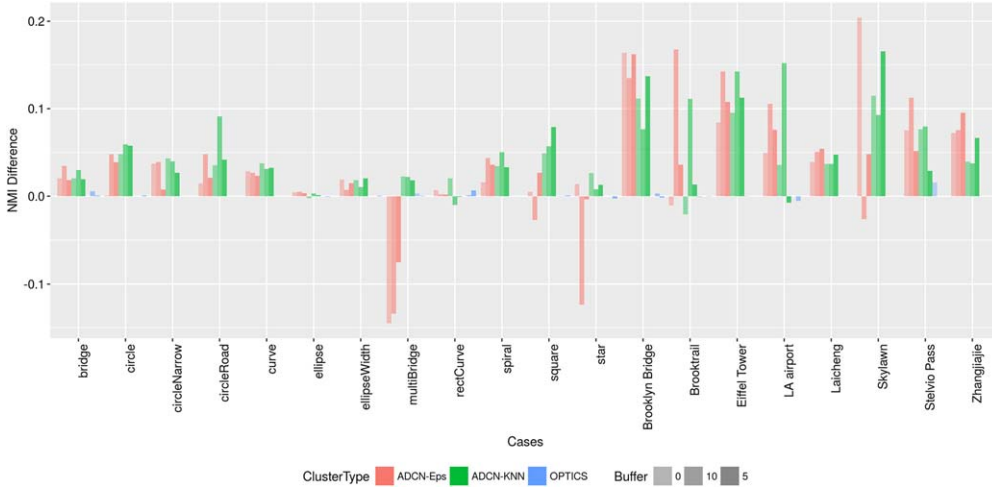

FIGURE 7 Clustering quality comparisons: NMI difference between three clustering methods and DBSCAN for each case. Synthetic cases are on the left, real-world cases on the right

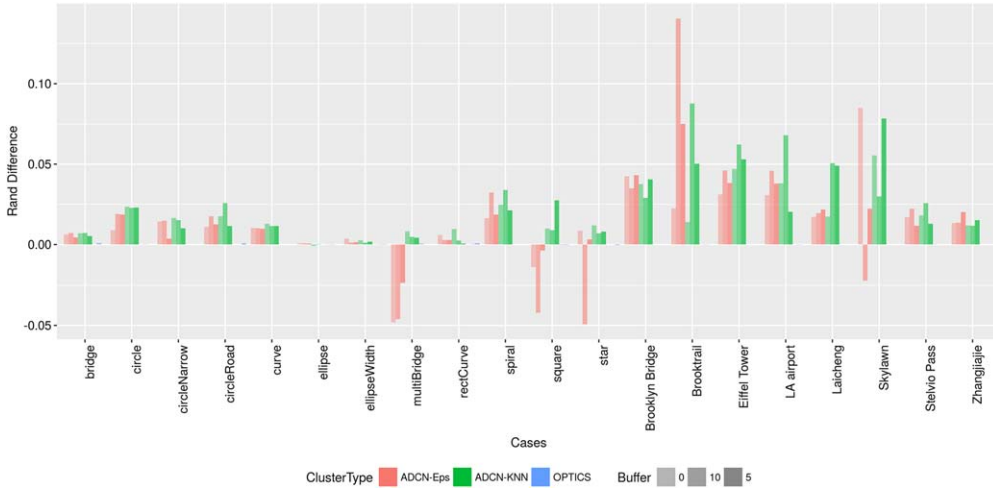

FIGURE 8 Clustering quality comparisons: Rand index difference between three clustering methods and DBSCAN for each case. Synthetic cases are on the left, real-world cases on the right

KNN has a higher maximum NMI/Rand index than ADCN-Eps in 31 cases; see Table 2. This indicates that ADCN-KNN gives the best clustering results among the tested algorithms. Our test cases not only contain linear features, but also cases that are typically used to evaluate algorithms such as DBSCAN (e.g., clusters of ellipsoid and rectangular shapes). In fact, these are the only cases where DBSCAN slightly outcompetes ADCN-KNN (i.e., the maximum NMI/Rand indexes of ADCN-KNN and DBSCAN are comparable). Summing up, ADCN-KNN performs better than all other algorithms when dealing with anisotropic cases and equally as well as DBSCAN for isotropic cases. In the following paragraphs, we will use ADCN-KNN and ADCN interchangeably.

Figures 4 and 5 show the point patterns as well as the best clustering results of all algorithms for the 12 synthesis cases and 8 real-world cases without buffering (i.e., with the 0 m buffer distance). By comparing the best clustering results of these four algorithms, we find some interesting patterns: (a) *Connecting clusters along local directions*, ADCN has a better ability to detect the local direction of spatial point patterns and connect the clusters along this direction; and (b) *Noise filtering*, ADCN does better at filtering out noise points-A good example of *connecting clusters along local directions* is the *ellipseWidth* case in Figure 4. As for the thinnest cluster at the bottom, the other three algorithms (except ADCN-KNN) extract multiple clusters from these points, while ADCN-KNN is able to "connect" these clusters to a single one. Many cases show the *noise filtering* advantage of ADCN. For example, the *bridge* case, the *multiBridge* case in Figure 4, and the *Brooklyn Bridge* case in Figure 5 reveal that ADCN is better at detecting and filtering out noise points along bridge-like features.

TABLE 2 The number of cases with maximum NMI/Rand index for each clustering algorithm

# Cases	Max NMI	Max Rand index
DBSCAN	1	0
ADCN-Eps	25	19
ADCN-KNN	33	41
OPTICS	1	0

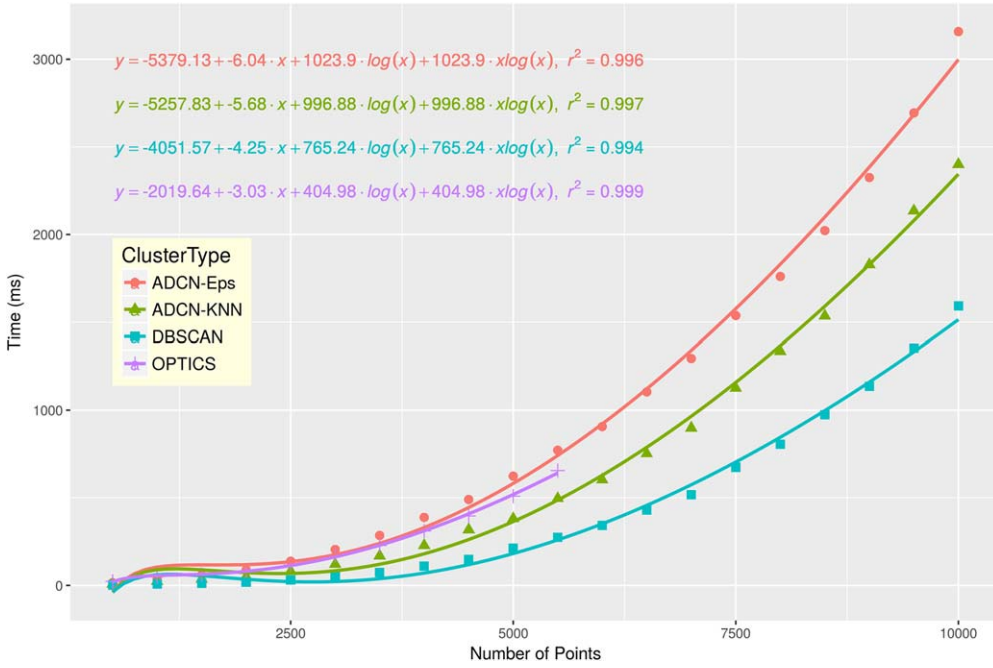

FIGURE 9 Comparison of clustering efficiency with different dataset sizes; runtimes are given in milliseconds (the used OPTICS library failed on datasets exceeding 5,500 points)

4.4 | Evaluation of clustering efficiency

Finally, this subsection discusses runtime differences of the four tested algorithms. Without a spatial index, the time complexity of all algorithms is $O(n^2)$. *Eps*-neighborhood queries consume the major part of the runtime of density-based clustering algorithms (Ankerst et al., 1999) and, therefore, also of ADCN-KNN and ADCN-Eps in terms of *Eps*-ellipse-neighborhood queries. Hence, we implemented an R-tree to accelerate the neighborhood queries for all algorithms. This changes their time complexity to $O(n \log n)$.

In order to enable a comprehensible comparison of the runtimes of all algorithms on different sizes of point datasets, we performed a batch of performance tests. The polygons from the 20 cases shown above have been used to generate point datasets of different sizes ranging from 500 to 10,000 in 500-step intervals. The ratio of noise points to cluster points is set to 0.25. *Eps*, *MinPts* are set to 15, 5 for all of these experiments. The average runtimes for the same size of point datasets is depicted in Figure 9.

Unsurprisingly, the runtime of all algorithms increases as the number of points increases. The runtime of ADCN-KNN is larger than that of DBSCAN and similar to that of OPTICS. As the size of the point dataset increases, the ratio of the runtimes of ADCN-KNN to DBSCAN decreases from 2.80 to 1.29. The original OPTICS paper states a 1.6 runtime factor compared with DBSCAN. The used OPTICS library failed on datasets exceeding 5,500 points. We also fit the runtime data to the $x \log(x)$ function. Figure 9 shows the fitted curves and functions of each clustering algorithm. We can see that all R^2 of these functions are larger than 0.95, which means that the $x \log(x)$ function well captures the trends of the real runtime data of these clustering algorithms. For ADCN, our implementation tests for point-in-circle for the radius of the major axis before computing point-in-ellipse to significantly reduce the runtime. Further implementation optimizations are possible, but beyond the scope of this article.

5 | SUMMARY AND OUTLOOK

In this work, we proposed an anisotropic density-based clustering algorithm (ADCN). Both synthetic and real-world cases have been used to verify the clustering quality and efficiency of our algorithm compared with DBSCAN and

OPTICS. We demonstrate that ADCN-KNN outperforms DBSCAN and OPTICS for the detection of anisotropic spatial point patterns and performs equally well in cases that do not explicitly benefit from an anisotropic perspective. ADCN has the same time complexity as DBSCAN and OPTICS, namely O(n log n) when using a spatial index and O(n^2) otherwise. With respect to the average runtime, the performance of ADCN is comparable to that of OPTICS. Our algorithm is particularly suited for linear features such as are typically encountered in urban structures. Application areas include but are not limited to cleaning and clustering geotagged social media data (e.g., from Twitter, Flickr, or Strava), analyzing trajectories collected from car sensors, wildlife tracking, and so forth. Future work will focus on improving the implementation of ADCN, as well as on studying cognitive aspects of clustering and noise detection of linear features.

ORCID

Gengchen Mai http://orcid.org/0000-0002-7818-7309
Song Gao http://orcid.org/0000-0003-4359-6302

ENDNOTE

[1] This article is a substantially extended version of the short paper by Mai, Janowicz, Hu, and Gao (2016). It also adds an open source implementation of ADCN, a test environment, as well as new evaluation results on a larger sample.

REFERENCES

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In *Proceedings of ACM SIGMOD Conference* (pp. 49–60). Philadelphia, PA: ACM.

Barden, L. (1963). Stresses and displacements in a cross-anisotropic soil. *Geotechnique*, *13*, 198–210.

Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, *60*(1), 208–221.

Boisvert, J., Manchuk, J., & Deutsch, C. (2009). Kriging in the presence of locally varying anisotropy using non-Euclidean distances. *Mathematical Geosciences*, *41*, 585–601.

Borah, B., & Bhattacharyya, D. (2004). An improved sampling-based DBSCAN for large spatial databases. In *Proceedings of the First International Conference on Intelligent Sensing and Information* (pp. 92–96). Chennai, India: IEEE.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *24*(5), 603–619.

Damiani, M. L., Issa, H., Fotino, G., Heurich, M., & Cagnacci, F. (2016). Introducing presence and stationarity index to study partial migration patterns: An application of a spatio-temporal clustering technique. *International Journal of Geographical Information Science*, *30*(5), 907–928.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *2*, 224–227.

Deng, M., Liu, Q., Cheng, T., & Shi, Y. (2011). An adaptive spatial clustering algorithm based on Delaunay triangulation. *Computers, Environment & Urban Systems*, *35*, 320–332.

Duckham, M., Kulik, L., Worboys, M., & Galton, A. (2008). Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, *41*, 3224–3236.

D'Ercole, R., & Mateu, J. (2013). On wavelet-based energy densities for spatial point processes. *Stochastic Environmental Research & Risk Assessment*, *27*, 1507–1523.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of Second International Conference on Knowledge Discovery and Data Mining* (pp. 226–231). Portland, OR: AAAI.

Fabrikant, S. I., & Montello, D. R. (2008). The effect of instructions on distance and similarity judgements in information spatializations. *International Journal of Geographical Information Science*, *22*, 463–478.

Fortin, M. J., Dale, M. R., & Ver Hoef, J. M. (2016). *Spatial analysis in ecology*. Retrieved from https://doi.org: 10.1002/9781118445112.stat07766.pub2

Gao, S., Janowicz, K., Montello, D. R., Hu, Y., Yang, J.-A., McKenzie, G., … Yan, B. (2017). A data-synthesis-driven method for detecting and extracting vague cognitive regions. *International Journal of Geographical Information Science*, *31*, 1245–1271.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques*. Oxford, UK: Elsevier.

Hanwell, D., & Mirmehdi, M. (2014). QUAC: Quick unsupervised anisotropic clustering. *Pattern Recognition*, *47*(1), 427–440.

Hoek, E. (1964). Fracture of anisotropic rock. *Journal of the South African Institute of Mining & Metallurgy*, *64*, 501–523.

Hu, Y., Gao, S., Janowicz, K., Yu, B., Li, W., & Prasad, S. (2015). Extracting and understanding urban areas of interest using geotagged photos. *Computers, Environment & Urban Systems*, *54*, 240–254.

Huang, Q. (2017). Mining online footprints to predict user's next location. *International Journal of Geographical Information Science*, *31*, 523–541.

Huang, Q., & Wong, D. W. (2015). Modeling and visualizing regular human mobility patterns with uncertainty: An example using Twitter data. *Annals of the Association of American Geographers*, *105*, 1179–1197.

Huang, Q., & Wong, D. W. (2016). Activity patterns, socioeconomic status and urban spatial structure: What can social media data tell us?. *International Journal of Geographical Information Science*, *30*, 1873–1898.

Isaaks, E. H., & Srivastava, R. M. (1989). *Applied geostatistics*. New York, NY: Oxford University Press.

Jurdak, R., Zhao, K., Liu, J., AbouJaoude, M., Cameron, M., & Newth, D. (2015). Understanding human mobility from Twitter. *PloS One*, *10*, e0131469.

Liu, P., Zhou, D., & Wu, N. (2007). VDBSCAN: Varied density based spatial clustering of applications with noise. In *Proceedings of the 2007 International Conference on Service Systems and Service Management* (pp. 1–4). Chengdu, China: IEEE.

Machuca-Mory, D. F., & Deutsch, C. V. (2013). Non-stationary geostatistical modeling based on distance weighted statistics and distributions. *Mathematical Geosciences*, *45*(1), 31–48.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. Cam, M. Le, & J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. *1*, pp. 281–297). Berkeley, CA: University of California Press.

Mai, G., Janowicz, K., Hu, Y., & Gao, S. (2016). ADCN: An anisotropic density-based clustering algorithm. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (p. 58). San Francisco, CA: ACM.

Møller, J., & Toftaker, H. (2014). Geometric anisotropic spatial point pattern analysis and Cox processes. *Scandinavian Journal of Statistics*, *41*, 414–435.

Moreira, A., & Santos, M. Y. (2007). Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. In *Proceedings of the Second International Conference on Computer Graphics Theory and Applications* (pp. 61–68). Barcelona, Spain: GRAPP.

Rajala, T. A., Särkkä, A., Redenbach, C., & Sormani, M. (2016). Estimating geometric anisotropy in spatial point patterns. *Spatial Statistics*, *15*, 100–114.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*, 846–850.

Rocha, J. A. M., Times, V. C., Oliveira, G., Alvares, L. O., & Bogorny, V. (2010). DB-SMoT: A direction-based spatio-temporal clustering method. In *Proceedings of the Fifth IEEE International Conference on Intelligent Systems* (pp. 114–119). London, UK: IEEE.

Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining & Knowledge Discovery*, *2*, 169–194.

Stefanakis, E. (2007). NET-DBSCAN: Clustering the nodes of a dynamic linear network. *International Journal of Geographical Information Science*, *21*, 427–442.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles: A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Stroet, C. B. T., & Snepvangers, J. J. (2005). Mapping curvilinear structures with local anisotropy kriging. *Mathematical Geology*, *37*, 635–649.

Tsai, C.-F., & Liu, C.-W. (2006). KIDBSCAN: A new efficient data clustering algorithm. In L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, & J. Zurada (Eds.), *International Conference on Artificial Intelligence and Soft Computing – ICAISC 2006: Proceedings of the Eighth International Conference, Zakopane, Poland, June 25–29, 2006* (pp. 702–711). Berlin, Germany: Springer Lecture Notes in Artificial Intelligence Vol. 4029.

Wang, J., & Wang, X. (2012). A spatial clustering method for points-with-directions. In T. Li, H. S. Nguyen, G. Wang, J. W. Grzymala-Busse, R. Janicki, A.-E. Hassanien, & H. Yu (Eds.), *Rough sets and knowledge technology: Proceedings of the Seventh International Conference, RSKT 2012, Chengdu, China, August 17–20, 2012* (pp. 194–199). Berlin, Germany: Springer Lecture Notes in Artificial Intelligence Vol. 7414.

Wing, M. G., Eklund, A., & Kellogg, L. D. (2005). Consumer-grade global positioning system (GPS) accuracy and reliability. *Journal of Forestry*, *103*, 169–173.

Yuill, R. S. (1971). The standard deviational ellipse: An updated tool for spatial description. *Geografiska Annaler: Series B, Human Geography*, *53*(1), 28–39.

Zhao, G., Wang, T., & Ye, J. (2015). Anisotropic clustering on surfaces for crack extraction. *Machine Vision & Applications*, *26*, 675–688.

Zhong, X., & Duckham, M. (2016). Characterizing the shapes of noisy, non-uniform, and disconnected point clusters in the plane. *Computers, Environment & Urban Systems*, *57*, 48–58.

Zhou, W., Xiong, H., Ge, Y., Yu, J., Ozdemir, H. T., & Lee, K. C. (2010). Direction clustering for characterizing movement patterns. In *Proceedings of the 11th IEEE International Conference on Information Reuse and Integration* (pp. 165–170). Las Vegas, NV: IEEE.

How to cite this article: Mai G, Janowicz K, Hu Y, Gao S. ADCN: An anisotropic density-based clustering algorithm for discovering spatial point patterns with noise. *Transactions in GIS*. 2018;00:1–22. https://doi.org/10.1111/tgis.12313