Finding landmarks within settled areas using hierarchical density-based clustering and meta-data from publicly available images

Eduardo Pla-Sacristán*, Iván González-Díaz, Tomás Martínez-Cortés, Fernando Díaz-de-María

Signal Theory and Communications Department, Universidad Carlos III de Madrid, Spain

Abstract

The process of determining relevant landmarks within a certain region is a challenging task, mainly due to its subjective nature. Many of the current lines of work include the use of density-based clustering algorithms as the base tool for such a task, as they permit the generation of clusters of different shapes and sizes. However, there are still important challenges, such as the variability in scale and density. In this paper, we present two novel density-based clustering algorithms that can be applied to solve this: K-DBSCAN, a clustering algorithm based on Gaussian Kernels used to detect individual inhabited cores within regions; and V-DBSCAN, a hierarchical algorithm suitable for sample spaces with variable density, which is used to attempt the discovery of relevant landmarks in cities or regions. The obtained results are outstanding, since the system properly identifies most of the main touristic attractions within a certain region under analysis. A comparison with respect to the state-of-the-art show that the presented method clearly outperforms the current methods devoted to solve this problem.

Keywords: Density-based clustering, K-DBSCAN, V-DBSCAN, Hierarchical clustering, Landmark detection, Tourism

*Corresponding author

Email addresses: epla@tsc.uc3m.es (Eduardo Pla-Sacristán), igonzalez@tsc.uc3m.es (Iván González-Díaz), tmcortes@tsc.uc3m.es (Tomás Martínez-Cortés), fdiaz@tsc.uc3m.es (Fernando Díaz-de-María)

1. Introduction

The main goal of automatic landmark detection is to determine, within a certain distribution, concealed regions or points that are particularly relevant, given a specific criterion. Landmark detection is a particularly challenging task, often due to the difficulty of determining relevant characteristics that support the identification of a landmark. A usual property to identify landmarks is their *popularity*, meaning that if an event, area or attribute within a sample space is recurrent, it should be considered as a landmark.

Clustering-based approaches are often taken to address the automatic discovery of landmarks in various fields (Bernard et al., 2013; El-Feghi et al., 2004). Particularly, real-world landmark discovery applications often focus on spatial datasets (Elias, 2003). Within this context, the discovery of touristic landmarks is a field worth studying. Expert tools to support the tourist industry have been previously developed, e.g., an application to detect unexpected behavior and prevent or mitigate undesired situations within cities (Cerezo-Costas et al., 2018) or a personalized route-planner based on social-media data (Cenamor et al., 2017). Nonetheless, the discovery of relevant landmarks is a particularly recurrent topic among the state-of-the-art (Cao et al., 2010; Tang and Meng, 2006; Feng et al., 2015). The angle of the approach, however, is immensely variable.

In some works, textual contributions such as geo-located tweets have been used as part of a place-relevance detector (Frias-Martinez et al., 2012); in others, user-generated images are employed and their visual properties can be used to discriminate between landmarks (Papadopoulos et al., 2011). In this sense, the most common approach relies on geo-located images (Zheng et al., 2009; Lee et al., 2014), as they provide both location and visual information, which notably boosts the performance of the detection process. In some cases, additional meta-data attached to the source images can be also used to refine the process (Kisilevich et al., 2010).

Although the use of images and geo-located content provides powerful cues for landmark discovery, it also introduces some challenges. One example of this is that the spatial areas associated to real-world landmarks (e.g., parks, buildings) have arbitrary shapes and, therefore, are not easy to model with traditional centroid-based clustering over GPS-coordinates. To cope with this issue, spectral and density-based clustering have been employed (Ji et al., 2011; Zhou et al., 2015) and showed decent results. However, mainly due to the fact that most approaches focus on presenting solutions for large cities or individual conurbations, there is a particular problem that has not been properly solved yet: scale and density variability along the sample set. This problem is frequently encountered when the area of analysis is a region that consists of several non-connected inhabited cores (e.g., a coastal area composed of several villages). Hierarchical approaches can become a decent solution for the problem of scale (McInnes et al., 2017), whereas variations in density have been also considered and modeled in some previous works (Kisilevich et al., 2010). However, as we will demonstrate in the experimental section, neither of them provide appropriate results in a generic scenario.

In this work, we present two novel density-based clustering algorithms (*K-DBSCAN* and *V-DBSCAN*) that have a direct applicability on the task of automatic landmark detection. We will prove that the combination of these two algorithms adapts to both urban and rural areas, whether they consist of a single dense conurbation or of several non-connected inhabited cores. We take ideas from previous approaches and develop a new clustering scheme to address the problem of varying density along the sample space. Particularly, we make several assumptions regarding the radial distribution of human inhabited areas, and carry out a granularity analysis of the area, which proves to be very efficient dividing a large region into different towns or villages.

Summarizing, the main novel contributions of this research are the following:

1. A *Kernel-based* variation of DBSCAN (*K-DBSCAN*), which aims at identifying arbitrarily-shaped groups of points within a significantly sparse sample-space, without previous knowledge of the amount of resulting clusters. This algorithm will be applied to the discovery of non-connected human settled areas (towns, villages) within a region of analysis.
2. A multi-scale variation of DBSCAN that takes into account the *variations in density* when moving away from the centroid of the data (*V-DBSCAN*). This algorithm will be used for the discovery of relevant landmarks within a connected

region, considering user-provided, geodesic and text-based information.

3. A *novel public dataset*, made from user-generated contents, which contains six heterogeneous locations, from single conurbations to larger regions with non-connected cores.

The remainder of this document is structured as follows: in Section 2, the problem addressed is stated and the related state-of-the-art is briefly reviewed. In Section 3, the proposed novel algorithms are explained in detail. In Section 4, the direct application of the algorithms is defined, including the data gathering and pre-processing techniques used. In Section 5, the experiments to prove its applicability are described, and the obtained results are discussed. Finally, the conclusions and future lines of work are posed in Section 6.

2. Related work

The object of our research is the following: given an initial set of data points distributed with variable density along the sample space, group them into different clusters -representing the most relevant landmarks- allowing for arbitrary shapes and sizes. A common example of data whose density varies throughout a sample space is geographical data.

This section first describes and justifies the sources considered to analyze the problem. Then, it discusses the most employed techniques to tackle landmark discovery.

2.1. Source analysis

First, it is important to clarify the type of landmarks to be discovered (touristic attractions, trending restaurants, events, etc.) as this will heavily impact the source of the data to analyze. A relevant source to provide geo-located content is Twitter [1], which is often used as base for event exploration (Feng et al., 2015; Becker et al., 2011). It has also been used to determine specific locations, like pinpointing user's homes (Lin and Cromley, 2018). However, the research in Frias-Martinez et al. (2012) regarding the

[1]https://twitter.com/

discovery of relevant physical landmarks showed that among the resulting predicted landmarks, points unrelated to tourism (like crowded train stations or commercial areas) were also discovered. Other sources, like Foursquare [2] and Instagram [3], have similar problems, since users also tend to publish their content related to places with no touristic interest (Noulas et al., 2011; Jayarajah and Misra, 2016). Additionally, these last two have very restricted API usage. In this context, when it comes to the discovery of touristic landmarks, Flickr [4] is a much more appropriate source. Flickr is a web platform that stores user-generated multimedia content. In addition, users enrich their contents with useful meta-data (geo-location, descriptive tags, user identification, etc.). This information can be used to retrieve relevant touristic landmarks from certain regions (Kennedy et al., 2007; Ji et al., 2011).

2.2. Clustering methods for Landmark Discovery

The detection of landmarks by clustering geographical data is a recurrent research topic (Cao et al., 2010; Tang and Meng, 2006). Several clustering algorithms have been considered in the literature for this purpose, some of them being better tailored to the task.

A classic partitioning algorithm such as Mean-Shift (Comaniciu and Meer, 2002) has been previously employed in Cao et al. (2010) to attempt the discovery of relevant touristic landmarks in a worldwide dataset. However, the scenario considered in the mentioned research is far too generic and, as we will show in our experiments, Mean-Shift fails to adapt to more realistic datasets. Another classic algorithm, *k-means* (MacQueen et al., 1967), performs reasonably well working with spatial data when the number of output clusters is known (Wagstaff et al., 2001). Unfortunately, a method for determining this parameter a priori is not trivial in our scenario, as the number of landmarks strongly depends on the area of analysis. Furthermore, focusing on the use of GPS coordinates, we must take into account that the shape of a cluster representing a landmark might be irregular, which prevents the use of centroid-based clustering

[2]https://foursquare.com/
[3]https://www.instagram.com
[4]https://www.flickr.com/

techniques. Instead, spectral clustering and density-based clustering approaches have often been taken to detect touristic landmarks, as they both allow the formation of arbitrarily-shaped clusters.

The main benefit of Spectral Clustering is often dimensionality reduction (Ji et al., 2011). However, in the application addressed in this paper the spatial features are always going to be incredibly relevant among the different types of meta-data. As a result, if a dimensionality reduction is attempted, we run the risk of eventually reducing the problem to spatial clustering without any other meta-data for support. Nonetheless, in Yang et al. (2011), landmark detection is attempted using a hierarchical algorithm based on binary trees divided with spectral clustering, but the proposed system yields unacceptable recall values, and non-automatable assumptions (such as dismissing all resulting clusters that do not fall near a labeled landmark) must be made in order to achieve and assess landmark detection.

On the other hand, density-based algorithms have been applied to this particular task. Methods like the seminal *DBSCAN* algorithm (Ester et al., 1996), which packs together points with respect to their vicinity, fit well with our assumptions. The original DBSCAN has been used on the particular task of landmark detection (Tang and Meng, 2006) and, additionally, several extensions of the algorithm have been proposed to deal with particular aspects of the problem: P-DBSCAN proposes the use of the number of users in a vicinity of a potential core point instead of simply considering the number of adjacent points (Kisilevich et al., 2010), thus limiting the influence of users uploading many pictures into the same location. The aforementioned work also includes a modification of the DBSCAN expansion function that handles the change of density within the sample space (Adaptive Density). Although these extensions adapt well to the problem of landmark detection, we will prove in this paper that they are not robust enough to perform landmark discovery on diverse geographical areas. Another recent algorithm is H-DBSCAN (McInnes et al., 2017), which addresses the varying density of a sample space using different values for the scale parameter, seeking a stable solution. However, the approach of the algorithm is quite generic, as it was not conceived to solve this particular problem, but to perform clustering within an n-dimensional sample space. In our context, H-DBSCAN can be applied over the GPS coordinates of photos

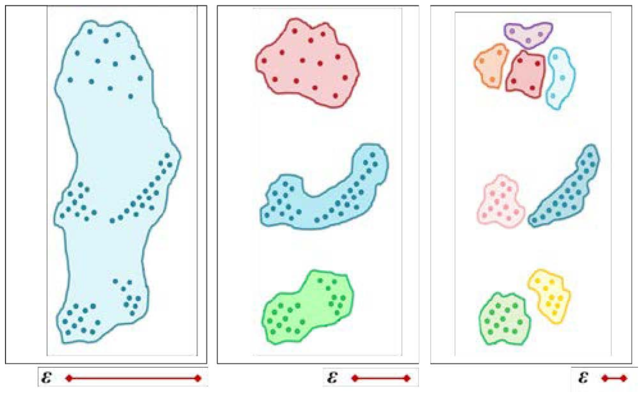

Figure 1: Results produced by DBSCAN for different values of ε (decreasing from left to right) applied to an artificial set of points.

to discover landmarks of interest. However, GPS coordinates are not sufficient to solve the problem by themselves. Our proposal here is to consider both the spatial features and the user-provided information in the samples, along with a crucial assumption of density variation, thereby solving this issue.

In conclusion, we have seen that there have been many attempts to solve this task using only geodesic coordinates, but without the support of additional sample meta-data (user information, descriptive tags) the systems fall short. Additionally, even for the cases where additional meta-data is considered, we will prove that the algorithms discussed in this paper are better suited for the problem at hand, outperforming all the methods currently being used for landmark detection.

3. Proposed algorithms

In this section, two novel algorithms for density-based clustering are presented: *K-DBSCAN* and *V-DBSCAN*. They are both variations of the well-known DBSCAN algorithm, particularly exploiting a crucial parameter: the ε distance. Conceptually, this parameter is used to determine at which scale the clustering is performed (see Fig. 1).

The first algorithm, *K-DBSCAN*, estimates the underlying density distribution of a given sample space in order to identify relevant peaks to which all the points will be

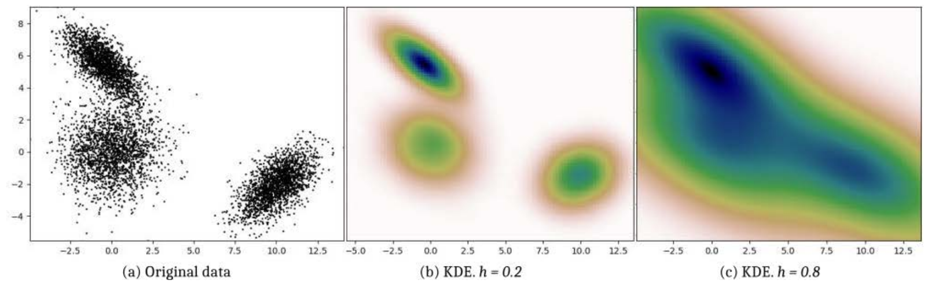

Figure 2: Kernel Density Estimation for an artificial set of points from 3 Normal distributions (a) with two bandwidth factors $h = 0.2$ (b) and $h = 0.8$ (c).

assigned based on density-reachability. On the other hand, *V-DBSCAN* takes advantage on the assumption of a gradual density drop when moving away from the centroid of the dataset, following a divisive approach to find arbitrarily-shaped clusters.

3.1. K-DBSCAN: Kernel-Density-Based Spatial Clustering of Applications with Noise

This algorithm groups sets of data in different clusters according to the density in the sample space. However, unlike many density-based clustering approaches, *K-DBSCAN* attempts to determine the number of resulting clusters prior to the clustering step. This is done by estimating the underlying distribution in the sample space and selecting its peaks.

The first step of *K-DBSCAN* is a Kernel Density Estimation (KDE) with Gaussian kernels (Scott, 2015) to estimate the underlying distribution $f_{KDE}(\mathbf{p})$ for the input vector $\mathbf{p} = \{p_1, p_2, ..., p_N\}$. At this point, the first of two specific parameters of this algorithm is specified: the bandwidth factor of the Gaussian kernels h, which is used to scale the covariance matrix of the data. This parameter impacts the scale of the analysis, since increasing the bandwidth factor will result in larger, less numerous convex areas of the estimated distribution (see Fig. 2).

The next step is to obtain the number of output clusters. To this end, all local maxima (stored in vector $\mathbf{p}^* = \{p_1^*, p_2^*, ..., p_{M^*}^*)$ are analyzed to check their validity as a cluster centroid. This is done through a measure of *topographic prominence* (Llobera, 2001). The topographic prominence tp of a peak is given by the vertical distance between the peak and its *key col*. The *key col* of a peak is the highest possible point to which we have to descend in order to climb again from it to a higher point of the distribution. This feature is useful to determine the independence of a peak.

We therefore analyze local maxima in the KDE distribution, and use prominence to decide which of them must be considered independent cluster centroids. In order to provide a generic criterion for this decision and given that the range of potential values of prominence strongly depends on the data distribution, we compute a *normalized topographic prominence* as follows: given a local maximum found in location $p_i^* \in \mathbf{p}^*$, the normalized topographic prominence $ntp(p_i^*)$ is:

$$ntp(p_i^*) = \frac{tp(p_i^*)}{f_{KDE}(p_i^*)} \tag{1}$$

At this point, we have a notion of the relevance (given by its prominence) of each local maxima, and we now need to decide which are kept as cluster centroids. For this, the value $ntp(p_i^*)$ is compared to a lower bound t, and all $p_i^* \in \mathbf{p}^*$ that do not satisfy this threshold are discarded (for being too dependent on another maxima), yielding cluster centroid vector $\mathbf{c} = \{c_1, c_2, ..., c_M\}$. This threshold t is the second of the two specific parameters of the algorithm, and it will impact the algorithm's sensitivity to crowded areas with variable internal density. In other words, it determines how close should two intertwined groups of points be to be considered as part of the same group. This can be appreciated in Fig. 2, where one could consider 3 groups of points determined by each of the 3 normal distributions (low value of t) or 2 groups determined by joining the two distributions on the left (high value of t).

Once the cluster centroids $\mathbf{c}$ are identified, each data sample $p_i \in \mathbf{p}$ must be assigned to a centroid. This assignment is not trivial, and cannot be done by simple proximity since this would alter the nature of the problem, where we are dealing with arbitrary cluster shapes. In fact, in order to preserve the essence of the data distribution, assignments must be done according to a density-based clustering algorithm. To this end, a modified version of DBSCAN is used to iteratively assign all samples to the correct centroid, i.e., the one that lies at a given local maximum $c_i \in \mathbf{c}$ for which the path from p to c_i is a monotonously increasing function.

In order to provide a more precise and formal description of *K-DBSCAN*, a pseudo-

Algorithm 1 Assignment phase of algorithm K-DBSCAN.

Input: A set of points **p**, a sub-set of M cluster centroids **c**, initial parameters ($\varepsilon_0, MinPts$), neighborhood radius update factor (η_ε).
Output: Set of clusters C

```
1:  function ASSIGN(p, c, $\varepsilon_0$, $MinPts$, $\eta_\varepsilon$)
2:      for $i$ from 0 to $M$-1 do
3:          $C_i$ = create-empty-cluster(i)
4:          $C_i$ = combine-unique($C_i$,$c_i$)
5:      end for
6:      $i = 0$
7:      $\varepsilon = \varepsilon_0$
8:      $k = 0$
9:      while points unclassified $> MinPts - 1$ do
10:         $C_i$ = expand-cluster($C_i$, p, $\varepsilon$, $MinPts$)
11:         $i = i + 1$
12:         if $i \geq M$ then
13:             $i = 0$
14:             $k = k + 1$
15:             $\varepsilon$=update($\varepsilon$,$\eta_\varepsilon$,$k$) // see eq. (2)
16:         end if
17:     end while
18:     if points unclassified $> 0$ then
19:         C = assign(p, c, $\varepsilon_0$, $MinPts - 1$, $\eta_\varepsilon$)
20:     end if
21:     return C
22: end function
```

code containing its assignment phase is provided in Alg. 1.

The assignment phase starts considering a set **p** of N points, in which a subset of M centroid clusters **c** has been previously identified. At the beginning, a set of M clusters is created, each one containing one centroid $c_i \in$ **c** (lines 2-4 in Alg. 1), and an initial value ε_0 is set. Let us note that the instruction *combine-unique*, which is used here to add a point to the current cluster, will be used along the algorithm to merge two sets of elements avoiding repetition. After this initialization, the algorithm operates iteratively until the stopping condition (discussed later) is met.

Algorithm 2 Expand-Cluster function.

Input: A set of points **p**, a cluster C to be expanded and parameters (ε, $MinPts$).
Output: Expanded cluster C

```
1:  function EXPAND-CLUSTER($C$, p, $\varepsilon$, $MinPts$)
2:      $Q = C$
3:      while $Q$ is not empty do
4:          $p$ = extract-from-queue($Q$)
5:          $H$ = obtain-neighborhood($p$, p, $\varepsilon$)
6:          $H$ = remove-classified-points($H$)
7:          if size($H$) $> MinPts$ then
8:              $C$ = combine-unique($C$,$H$)
9:              $Q$ = combine-unique($Q$,$H$)
10:         end if
11:     end while
12:     return $C$
13: end function
```

At each iteration k, DBSCAN's expansion function (defined in Alg. 2) is applied to each cluster (line 10 in Alg. 1) using the current value of ε and a fixed value of $MinPts$. Clusters are expanded by annexing unclassified points in the neighborhood of the *core-points* (represented by set Q in Alg. 2). Additionally, if some point p' in the neighborhood H is a *core-point* itself, it is added to the cluster as well, and its neighborhood H' is subsequently considered to be potentially included as well. This expansion continues until no new *core-points* can be reached from the current *core-points* in the cluster.

At the end of every iteration k of the assignment phase, ε is updated following the next recursive equation:

$$\varepsilon^{(k)} = \varepsilon^{(k-1)} \cdot (1 + \eta_\varepsilon) \tag{2}$$

where η_ε is the *neighborhood radius update factor*, which governs how the scale parameter ε grows at each iteration.

DBSCAN's $MinPts$ parameter establishes the number of points inside the ε distance to consider a core-point. In other words, for a fixed ε, it influences the expansion rate and the stopping condition. As the assignment phase of *K-DBSCAN* will want to iteratively traverse the database until all points are assigned, $MinPts$ needs to be just small enough so that the algorithm does not fall into an endless loop. And even in that case, the maximum amount of unclassified points left is $MinPts - 1$. Therefore, we set the default value $MinPts = 4$ (established in Ester et al. (1996)) and iterate until all samples but the last $MinPts - 1$ are assigned. This defines the stopping condition, and then we can assign all remaining samples by recursively calling our assignment function decreasing $MinPts$ (lines 18-20).

Two other initial parameters have been mentioned in this assignment phase: ε_0 and η_ε. However, these only control the scale space, and both their values should ideally be as low as possible. This is further discussed in Section 5.2.2.

The output of *K-DBSCAN* is, therefore, the set of clusters resulting after the assignment of all samples has been completed. As it will be later shown in Section 4, *K-DBSCAN* is effective to discriminate large groups of points presented in sparse sample spaces, especially when those groups have arbitrary shapes and sizes and have internal density variations, but are overall similar regarding volume of points.

3.2. V-DBSCAN: Variable-Density-Based Spatial Clustering of Applications with Noise

This algorithm is a hierarchical modification of the DBSCAN algorithm. *V-DBSCAN* is designed to take advantage of sample spaces with varying density and, more specifically, datasets where density decreases when moving away from a global maximum, which we call *density centroid*.

In order to provide a better understanding of the algorithm, a pseudo-code version of V-DBSCAN is provided in Alg. 3 (let us note that, in the provided pseudo-code, we denote $|C|$ as the cardinality of a set or collection C).

V-DBSCAN is an iterative divisive clustering algorithm that, at each new iteration, works at a finer resolution, subdividing several clusters into smaller ones. At each level, the variable ε defines the neighborhood radius of DBSCAN and therefore determines the scale or resolution of the process. For the first layer, an initial value (ε_0) is specified, which will be then updated in the following levels. Opposite to *K-DBSCAN*, we want to start the algorithm with a large enough value ε_0, that avoids splitting large clusters in the first iteration. Hence, as it happened in *K-DBSCAN*, it is known that its ideal value is as high as possible (see Section 5.2.2).

The goal of V-DBSCAN is to assign each data point p_i in a set **p** to a collection of clusters $C = \{C_0, C_1, C_2, ..., C_D\}$. For initialization purposes, a single cluster C_0 is created, which contains all samples in our set **p**. At each level of the algorithm, we operate independently over each cluster C_i: considering the subset of data samples belonging to each cluster, we perform DBSCAN (its pseudo-code is included in Alg. 4) using the current value of ε and a fixed value of $MinPts$ (the default value $MinPts = 4$ established in Ester et al. (1996) works for most databases). As a result, each cluster C_i is subdivided into a new set of sub-clusters S and the next step is to decide which of the new sub-clusters are relevant. To this end, we first sort the set of sub-clusters by their relevance (line 11 in Alg. 3), which can be simply measured by their cardinality $|C_i|$. Then, we compare each sub-cluster S_j with its complement $S_j^{\complement}$, which contains the remaining points in S, to decide whether S_j should be considered an independent cluster or not (line 14). The comparison is made according to a Subdivision Criterion that considers the internal properties of clusters S_j and $S_j^{\complement}$ (it will be described in depth in Section 3.2.1). If the sub-cluster S_j is found to be different enough from $S_j^{\complement}$, their samples are removed from the original cluster C_i, and the cluster S_j is added to the collection C.

Once this process is repeated for all the clusters present in the set C, we check if at least one sub-division has been performed or, instead, the set C has remained unaltered (line 20). In the latter case the variable that controls the number of static levels is increased. Before starting the next iteration, the neighborhood radius ε is updated (line 25) according to eq. (3), leading to a finer resolution analysis.

$$\varepsilon^{(k)} = \varepsilon^{(k-1)} \cdot (1 - \eta_\varepsilon) \tag{3}$$

Again, analogously to *K-DBSCAN*, the initial parameter η_ε would ideally be as low as possible (see Section 5.2.2).

The algorithm continues operating until the stopping condition is met. This happens when a sufficient number of iterations (scale levels) have not produced modifications in the results. In other words, convergence is attained.

3.2.1. Subdivision criterion

Similar to the criterion employed in Forsyth and Ponce (2002) for Agglomerative Clustering, in order to decide if a sub-cluster S_j within a cluster C_i is independent enough to be separated, we compare the distance between S_j and its complement sub-cluster $S_j^{\complement}$ with their respective internal distances:

$$Dist(S_j, S_j^{\complement}) \lesseqgtr MInt(S_j, S_j^{\complement}) \tag{4}$$

The distance between the two clusters $Dist(S_j, S_j^{\complement})$ is simply computed as the minimum distance between both sets of points, i.e., the distance between the two closest points from both sets:

$$Dist(S_j, S_j^{\complement}) = \min_{p \in S_j, q \in S_j^{\complement}} d_c(p, q) \tag{5}$$

On the other hand, the minimum internal distance of the sub-clusters is computed as:

$$MInt(S_j, S_j^{\complement}) = min(int(S_j) + \tau(S_j), int(S_j^{\complement}) + \tau(S_j^{\complement})) \tag{6}$$

where $int(C)$ and $\tau(C)$ are, respectively, the internal distance and the regularization term for cluster C.

First, we have computed the internal distance $int(C)$ of a cluster C as the maximum value among the average distances of the K nearest neighbors of each point in the cluster.

$$int(C) = \max_{p_i \in C} \frac{1}{K} \sum_{k=1}^{K} d(p_i, p_k^i) \tag{7}$$

where p_k^i is the k-nearest neighbor of the point p_i. In other words, we take the worst possible case representing the sparsity of the cluster. For the sake of simplicity, we set the number of nearest neighbors $K = MinPts$ used for DBSCAN.

Finally, a regularization term $\tau(C)$ is added to the internal distance to model our assumptions regarding the problem. Specifically, with this term we attempt to compensate two facts: 1) small clusters show low internal distances, which favors the generation of excessively small sub-clusters that we want to avoid; and 2) we want to favor larger and sparse clusters in those regions within the sample space that are far away from the density centroid. To cope with both requirements, we have defined the following regularization term:

$$\tau(C) = \frac{\kappa \cdot I(C)}{s_C} \tag{8}$$

where κ is the parameter that controls regularization; s_C is the ratio between the cardinality of the cluster $|C|$ and the cardinality of the sample space $|\mathbf{p}|$ (the size of the dataset), which penalizes small clusters; and $I(C)$ is a measure of the isolation of the cluster, which will be higher for clusters located far from the *density centroid*. Specifically, $I(C)$ is defined as follows:

$$I(C) = dist(m_C, m_{C^{\complement}}) \tag{9}$$

i.e., as the distance between the mass-center of the cluster and that of its complement. Conceptually, the isolation of a cluster represents how distant it is from the main agglomeration of points within a sample space (*density centroid*).

The most relevant parameter of this algorithm is, hence, the regularization parameter κ, which will determine the flexibility of the Subdivision Criterion. In other words, it will establish how prone are clusters to be separated at each level. It should also be noticed that the regularization factor is useful to enforce algorithm convergence, since, given that clusters are smaller in new iterations, it hinders subsequent divisions.

To conclude, the output of *V-DBSCAN* is a set of D clusters that effectively describe a sample space with variable density. The key concepts to its novelty are (a) its hierarchical structure based on divisive clustering, by which we are able to capture different scales within a variable-density sample space; and (b) the introduction of the isolation concept, which makes *V-DBSCAN* effective against distributions with a radially-decreasing density.

4. Application: Touristic Landmark Detection

In this section, we propose touristic landmark detection as a direct application for the novel algorithms explained in Section 3. Our proposal for landmark discovery is enclosed in a large-scale project, and will become a processing block of a system devoted to automatically generate travel guides. Once the landmarks are identified, multimedia content (text, images, video) must be retrieved and included in the guide. To this end, we can use publicly available user-generated contents from social networks and multimedia platforms.

The work presented here focuses on the automatic discovery of landmarks, defined by their GPS coordinates and other user-provided information. With this goal in mind, we can identify three levels of hierarchy in our analysis, which will be referred to throughout this document: a) *region*, which defines the geographical area under analysis, i.e., the complete sample space $\mathbf{p}$; b) *Place-of-Interest (PoI)*, which consists on each independent conurbation present within a region (for instance, each of the villages in a coastal area); and c) *landmark*, which represents each monument, park or other type of relevant element within any given *PoI*.

The system is divided into three main blocks, as displayed in Fig. 3: (1) the data gathering block, which will access Flickr to obtain the necessary data from a certain region; (2) the data pre-processing block, which will prepare the raw data for our clustering analysis; and (3) the landmark discovery block, which will make use of the novel algorithms discussed in Section 3 to discover the most relevant touristic places within the analyzed region. Each of the blocks considered will be discussed in-depth in the following subsections.

4.1. Data gathering

Given an area of interest, we aim to gather a dataset of user-generated contents, that will be used to automatically discover the main landmarks within the area. Working with user-generated content fits well with our definition of landmark, which is based on the popularity of a place. In addition, as the system is supposed to work in any area provided as a query, we need to automatically generate the corresponding dataset,

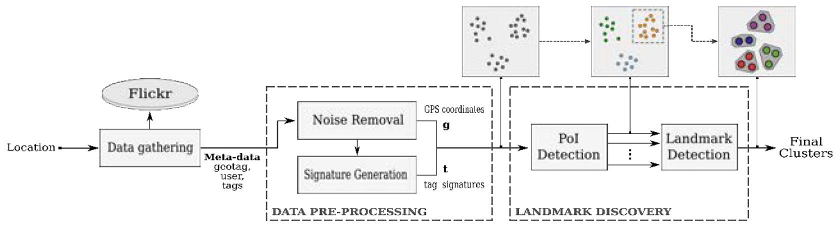

Figure 3: Pipeline of the complete system. It is divided in three main blocks: data gathering, which accesses the online platform to conform the dataset (Section 4.1); data pre-processing, which removes inherent noise (Section 4.2.1) and generates the bag-of-words model signatures (Section 4.2.2); and landmark discovery, which performs *PoI* and landmark detection through hierarchical density-based clustering (Sections 4.3.1 and 4.3.2, respectively).

associated to the area, which contains the images' geo-location and associated text (tags, descriptions, titles, etc.). As it was discussed in Section 2, we found the ideal platform for our purposes is Flickr. The Flickr API allows for downloading images while specifying constraints for their meta-data in the query, which is very useful to generate an appropriate dataset.

In the experiments shown in this paper, a circular query area is defined specifying the GPS coordinates of its center (latitude and longitude), and a radius (in km). However, our system could be easily adapted to work with arbitrarily-shaped areas defined by irregular masks. Once the query is defined, we retrieve from Flickr all the images that have been geo-located in that area since 2005 (along with the identification of the users that uploaded them). Moreover, if available, a list of user-generated tags is obtained for each image.

4.2. Data pre-processing

This module of the system is in charge of preparing the data for the subsequent clustering stage. It performs two independent tasks: a) Noise Removal and b) Generation of Textual Signatures.

4.2.1. Noise Removal

One of the main problems with user-generated images is the noise inherent to their annotations. Our analysis revealed two main causes:

- Incorrect tagging: some users upload their pictures from a trip in a single session with a unique geo-location (regardless of the exact place associated with each photo). Additionally, it is not uncommon for a user to upload numerous images of a single monument or event sharing the same descriptive tags. This causes an artificial density of points at some places. To overcome this issue, we only allow unique samples at each exact GPS location, meaning that one location can present multiple samples if and only if the user that posted them or the tags attached to them are different.

- Non-relevant content: although less often than in other platforms, such as Twitter or Instagram, retrieved pictures may show contents related to personal events (weddings, birthday parties) therefore not being relevant for touristic purposes. To tackle this, a list of stop-words was employed to avoid generic non-relevant content in the final database.

Hence, the output of this module is a final set of samples $\mathbf{p} = \{p_1, p_2, ..., p_N\}$, each of them associated with an image in Flickr, and composed of two features, namely: a) GPS coordinates of the picture; and b) a list of textual tags associated with it. This collection of samples is then fed to the next sub-module in order to generate the tag signature of each image.

4.2.2. Tag Signature Generation

Unfortunately, geodesic data is often not enough to discriminate between neighboring landmarks, specially when the distance between them is small compared to their sizes. In these cases, additional information like image meta-data is necessary to increase the dimensionality of the representation and enhance the discrimination.

Since the tags are often scarce, unstructured and noisy, we have built a *Bag-of-Words* (BoW) model, in which the set of tags for each image is transformed into a fixed-length signature vector using a *tf-idf* (Gerard and Michael, 1983) approach. It is worth mentioning that alternative techniques, such as the more advanced *word2vec* (Mikolov et al., 2013), were also tested to obtain vector representations of the tag space. The results, however, did not show any improvement over BoW for this scenario.

This data pre-processing stage ends up with a set of input features, each one composed of a GPS coordinate vector $\mathbf{g}$ and a textual (based on the descriptive tags) signature vector $\mathbf{t}$, as illustrated in Fig. 3.

4.3. Landmark Discovery

Landmark Discovery constitutes the main task of our system. This section will describe the two modules in charge of performing this operation. The first one, called *PoI Detection*, aims to identify non-connected settled areas within the region of analysis, allowing the subsequent module to operate independently in each location. To this end, the algorithm *K-DBSCAN* described in Section 3.1 will be employed. The second module, *Landmark Detection*, consists on a multi-scale analysis of each identified location using the algorithm *V-DBSCAN* (discussed in Section 3.2), which will ultimately identify each independent touristic landmark.

V-DBSCAN takes advantage of the following assumption, which is true for most urban areas, towns, and villages: gradual residential, commercial or industrial growth accumulates infrastructure around a center and, furthermore, more compact and densely-distributed clusters tend to be near that center (e.g., squares, churches, buildings), whereas larger and sparsely-distributed locations (parks, zoos, stadiums, etc.) are more often located in the outskirts. Nonetheless, we have seen that, for this assumption to be valid, we need to previously separate each *PoI* using *K-DBSCAN*.

Fig. 4 displays an example of a region that is clearly a single conurbation (Valencia) and one that contains several isolated villages (coastal area of Euskadi).

4.3.1. Place-of-Interest *Detection Module*

The input to this module is a feature vector containing geodesic coordinates for each sample, along with the tag signature computed in the previous module (see Section 4.2.2). For now, however, we ignore the tag signature and simply work with the geodesic coordinates. The *K-DBSCAN* algorithm is applied to these set of coordinates yielding N clusters as the module's output, each one representing an individual *PoI*.

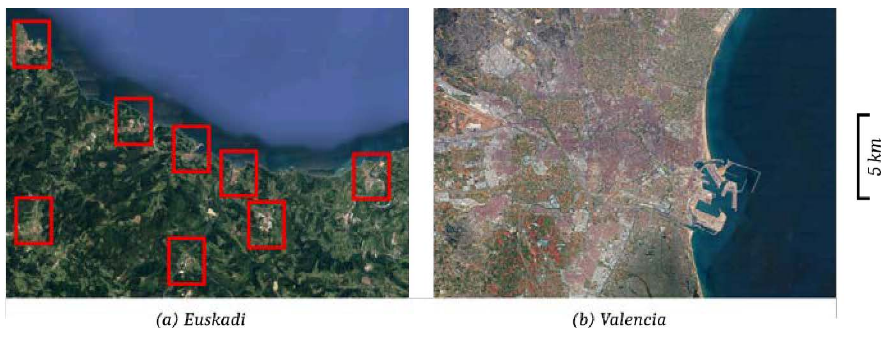

Figure 4: Example of a region with multiple PoIs (a) and a single PoI (b). Both locations are displayed at the same scale.

4.3.2. Landmark Detection Module

This second module aims to discover, within each detected *PoI*, its most prominent landmarks. To this end, it makes use of *V-DBSCAN*. Due to the aforementioned reasons, in *V-DBSCAN* we concurrently use geodesic coordinates and tag signatures to describe the data samples. Consequently, the distance between two points $d_c(p_1, p_2)$ is computed as a weighted linear combination of two independent distances (both normalized between 0 and 1), one regarding each considered feature:

$$d_c(p_1, p_2) = \gamma \cdot \hat{d}_g(p_1, p_2) + (1 - \gamma) \cdot d_t(p_1, p_2) \tag{10}$$

where $\hat{d}_g(p_1, p_2)$ stands for a normalized version of the geodesic distance; $d_t(p_1, p_2)$ is the distance between the two textual signatures, and $\gamma \in [0, 1]$ is a weighting parameter that will be discussed later.

The normalized geodesic distance is computed as follows:

$$\hat{d}_g(p_1, p_2) = min\left(\frac{d_g(p_1, p_2)}{f_d},\ 1\right) \tag{11}$$

where $\hat{d}_g(p_1, p_2)$ is the geodesic distance separating p_1 and p_2, f_d is a scaling factor, and the distance is clipped to a maximum of 1. The normalized distance is not sensitive to f_d, since it has been set to a large enough value (1 km in our experiments) to assume that two samples at a distance of f_d are not adjacent neighbors within the same cluster.

Furthermore, the distance between textual signatures d_t is computed using cosine similarity:

$$d_t(p_1, p_2) = 1 - \frac{\mathbf{t}_1^T \mathbf{t}_2}{||\mathbf{t}_1||_2||\mathbf{t}_2||_2} \tag{12}$$

where vectors $\mathbf{t}_1, \mathbf{t}_2$ are the tf-idf *textual signatures* of points p_1 and p_2.

The rationale behind this combined distance is the following: in general, geodesic coordinates successfully discriminate most of the landmarks within a *PoI*. However, it was observed that they are insufficient when analyzing high density areas, where landmarks are located extremely close to each other (buildings within the same square, neighboring monuments, etc.). This problem is particularly relevant when the size of a particular landmark is big compared to its distance to neighboring monuments (e.g., the size of a museum or a park located in the center of a city might be quite larger than their distance to neighboring landmarks). Hence, in these scenarios it is useful to increase the dimensionality of the descriptors in order to attain a better discrimination. In order to weight the relative influence of each feature (geodesic or textual), we have set the value of $\gamma = 0.999$. Although it might seem that this value neglects the influence of the textual distance, let us note that geodesic distances, despite being bounded in [0,1] interval, are in general small, specially in the last iterations. In those cases, the textual distance becomes relevant.

Another aspect of this module that is worth mentioning is that the relevance concept used to sort the sub-clusters in *V-DBSCAN* (see Section 3.2) is here defined as the number of users that take part in the cluster, rather than just their cardinalities. This introduces a more appropriate concept of cluster relevance.

The output of this last module is a set of the most prominent landmarks within each *PoI*, each of them represented by a single cluster. Fig. 5 shows a result of the complete system. As desired, the algorithm is able to properly detect compact clusters within crowded areas of the center of the city, and keep larger and less dense clusters in the outskirts. The performance of the system will be thoroughly assessed in Section 5.

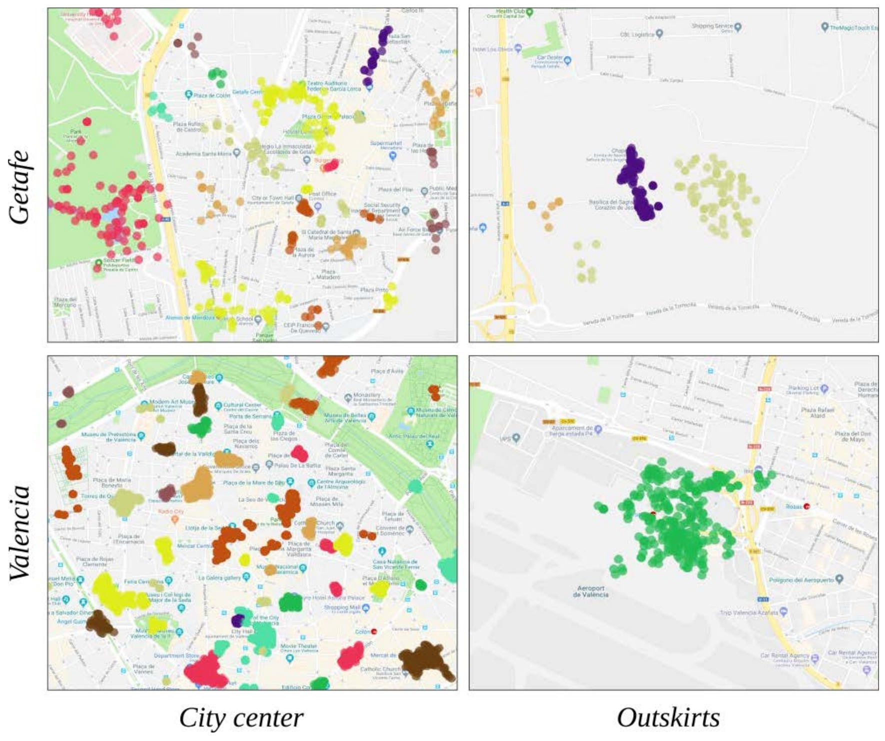

Figure 5: *V-DBSCAN* results in urban regions. At the bottom, Valencia; at the top, Getafe. The left side represents centric zones, while the right hand side represents places in the outskirts. Data belonging to different clusters is represented by different colors.

5. Experiments and results

5.1. Evaluation Metrics

Clustering techniques are usually non-supervised and, thus, their evaluation becomes a challenging task. For this work, intrinsic statistical metrics and indexes were discarded as they did not correlate well with our goals. On the other hand, extrinsic metrics require a set of *ground-truth* (GT) annotations to establish comparisons with the system output. Getting accurate labels for each data sample is impractical in our scenario, since we are dealing with thousands of images for each location under analysis. Nevertheless, as the goal of our work is to detect landmarks, we can assess our approach from an Information Retrieval perspective, comparing the discovered locations with a GT list of GPS coordinates associated with the main landmarks of the

City	(lat,lon)	#GT	N	D
$Tapia$	$(43.567, \text{-}6.951)$	22	2003	223
$Getafe$	$(40.301, \text{-}3.722)$	24	3884	806
$Jerez$	$(36.684, \text{-}6.137)$	53	11467	832
$Valencia$	$(39.469, \text{-}0.377)$	57	112587	744
$Guadalaj.$	$(41.079, \text{-}3.202)$	24	2208	474
$Euskadi$	$(43.284, \text{-}2.309)$	56	21077	1011

Table 1: Characteristics of the databases after preprocessing. Respectively: city; latitude and longitude of the approximate center in decimal degrees; length of GT-location list; number of samples; length of dictionary (number of words).

region. This list is much simpler and faster to obtain, and avoids labeling every data sample. However, it requires to perform an *alignment process* between an automatically generated set of clusters C (groups of data samples), and a ground-truth GPS location vector GT. In our case, we consider that $C_i \in C$ and $GT_j \in GT$ are aligned if the minimum geodesic distance between GT_j and the points in C_i is lower than a very restrictive threshold $TH_d = 50m$.

Once the alignment between the automatically generated clusters C and the ground-truth vector GT is defined, we can evaluate the performance of the system using *Average Precision* (AP), a well-established metric in Information Retrieval (Manning et al., 2008).

5.2. Dataset and Experimental Setup

5.2.1. Dataset

We made our dataset publicly available [5]. It consists of six different regions of interest, all located in Spain. For the sake of generality, we have considered four diverse single-*PoI* regions, including a small village, Tapia de Casariego (Asturias), two medium-sized cities, Jerez de la Frontera (Andalucia) and Getafe (Madrid), and a large city, Valencia (Comunitat Valenciana). In addition, we also included two multi-*PoI* regions corresponding to famous touristic areas: Guadalajara (region including various villages with Black Architecture), and Euskadi (region between rivers Lea and Oria,

[5] https://github.com/plasavall/LanDete

with several coastal touristic villages). For each location, the data gathering module (see Section 4.1) was used to retrieve images and their corresponding meta-data (GPS coordinates, tags), and we asked local tourist information centers to generate a list of landmarks and places of interest in the area. The details are displayed in Table 1.

5.2.2. Parameter validation

We have compared our approach with other solutions found in the literature tackling the automatic discovery of landmarks or the clustering of data (see Section 5.3). As all the approaches have several parameters with an important impact on the results, we have followed a process that ensures a fair comparison between algorithms. When possible, we have set their values to those proposed by the authors in the corresponding papers. In the case of clustering approaches not used for landmark discovery, we have followed a cross-validation strategy on a subset of the data.

With respect to the proposed algorithms (*K-DBSCAN* and *V-DBSCAN*), throughout this document we have distinguished between two different types of parameter: *scale-space* and *algorithm-specific* parameters.

Scale-space parameters. Parameters ε_0, η_ε and (just for *V-DBSCAN*) L define together the scale space. ε_0 represents the initial scale, L is the number of levels the algorithm results' are allowed to remain unchanged, and η_ε determines the scale relation between consecutive levels. Hence, they define both the limits and the degree of discretization of the scale space. It is clear that considering large ranges and fine discretization will provide a better performance at the expense of an increase on the complexity. Nevertheless, the algorithms' behavior with respect to these parameters is quite stable, and a set of optimal values was found, providing a good trade-off between performance and complexity.

For *K-DBSCAN*, values of $\varepsilon_0 = 200\ m$ and $\eta_\varepsilon = 0.1$ were set. Regarding *V-DBSCAN*, it is worth mentioning that, unlike the case of *K-DBSCAN*, ε is no longer a spatial magnitude, as we are using the aforementioned combined distance metric $d_c(p_1, p_2)$. The value of this parameter was set to $\varepsilon_0 = 0.2$. Additionally, we set $\eta_\varepsilon = 0.1$ and $L = 8$.

Algorithm-specific parameters. There are two specific parameters in *K-DBSCAN*: the bandwidth factor of the kernels h, and the prominence lower bound t. Regarding the former, it is obvious that we need to adjust it large enough so that the resulting distribution presents isolated maxima in the center of each *PoI*. On the other hand, the latter controls which maxima are discarded and, hence, not considered as valid density centroids. This second parameter is harder to adjust, since we have no prior information on about the distribution. A cross-validation strategy was followed to adjust these parameters, yielding values of $h = 0.35$ and $t = 0.4$, which provided the more stable results in terms of AP.

The only specific parameter for *V-DBSCAN* is the regularization parameter κ. As it was explained, the regularization factor avoids the proliferation of excessively small clusters and helps the algorithm to attain convergence. After validation, we set $\kappa = 1.7 \cdot 10^{-3}$. This low value makes sense, since s_C (which represents the size of the cluster) takes low values due to its normalization.

5.3. Results

In this section, we present the results of the complete system, compared with several methods in the literature for the task of automatic landmark detection. Furthermore, a comparison between the application of our approach with and without the *PoI* Detection Module has also been made. To this end, each region will be analyzed as if it contained a single *PoI*. This will show the influence of *K-DBSCAN* in those regions with more than one *PoI*.

Finally, we provide an assessment of the usability of our method to generate automatic travel guides.

5.3.1. Results for Landmark Discovery

In this section, we compare the performance of our approach with various alternatives found in the literature: some of them, despite being generic solutions for clustering data, have been previously used in our scenario to cluster geo-spatial data: k-means is used in Wagstaff et al. (2001) to detect road lanes from GPS data, while in Tang and Meng (2006) and Cao et al. (2010), DBSCAN and Mean-Shift are used, respectively, to find meaningful locations based on GPS data (see Section 2). Other algorithms have been particularly designed to deal with variable-density sample populations (H-DBSCAN (McInnes et al., 2017), Hierarch. Agglom. (Rasmussen, 1992)). Finally, some algorithms were specifically proposed to tackle the task of landmark discovery (P-DBSCAN and P-DBSCAN with adaptive density (Kisilevich et al., 2010)).

The results obtained for the six considered locations are displayed in Table 2, which shows the performance in terms of AP. It is worth mentioning that, for k-means, which is non-deterministic, the performance was obtained as the average of 40 executions of the algorithm (the standard deviation is also displayed in this case). Table 2.a shows the results obtained in the regions with a single *PoI*, whereas Table 2.b shows those containing multiple *PoI*. Two versions of our approach are included, with and without the *PoI* Detection Module, i.e., using both of the proposed algorithms (*K-DBSCAN* and *V-DBSCAN*) and using just *V-DBSCAN*.

In addition, Fig. 6 shows the output of the different algorithms for the center of Jerez de la Frontera, providing supplementary visual support to our analysis of the results.

At first glance, one can clearly notice that our proposal outperforms the algorithms found in the literature.

With respect to generic solutions previously used to cluster geo-spatial data, it is clear that, since they do not consider the concepts of density and scale variations, a common set of valid parameters for every case does not exist. Therefore, they fail to address the task of landmark discovery. However, Mean-Shift and k-means perform significantly better than DBSCAN. This is due to the fact that they do not consider the concept of noise, and therefore assign every data sample to a cluster, allowing more potential alignments. One could argue that this is not the objective we have in mind, as the output of this procedure will be closer to district separation than to landmark discovery. In other words, they divide the sample space into Voronoi-cells, rather than detecting relevant independent locations within an enclosed space. This can be appreciated in Fig. 6.

In contrast, Hierarchical Agglomerative clustering and H-DBSCAN take density variations into consideration. The former is particularly tricky to adjust, as the pre-

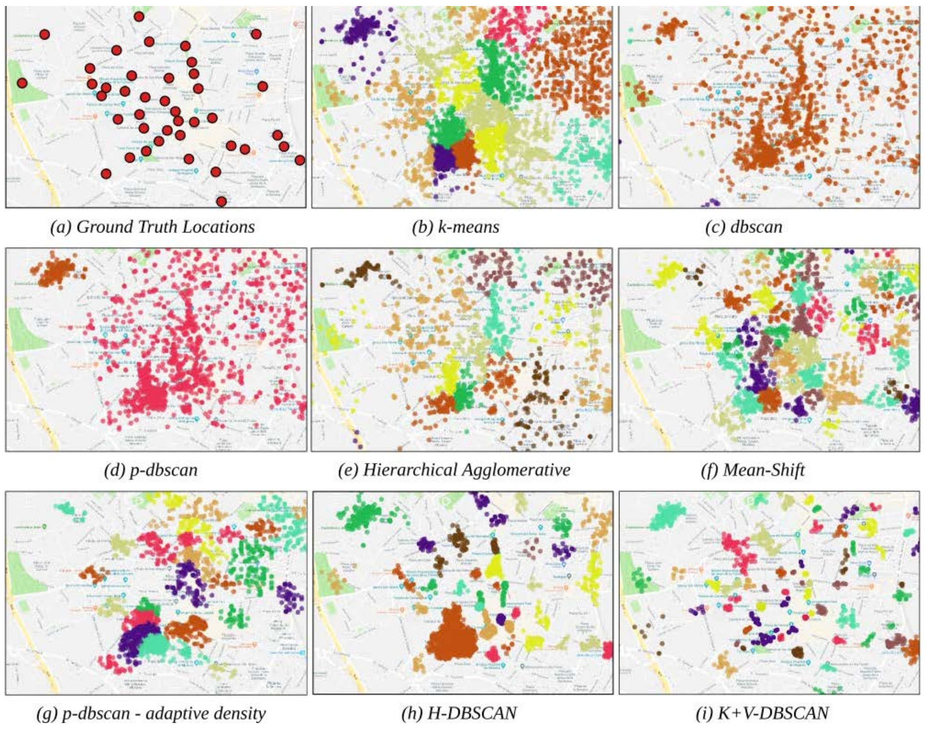
(a) Ground Truth Locations (b) k-means (c) dbscan (d) p-dbscan (e) Hierarchical Agglomerative (f) Mean-Shift (g) p-dbscan - adaptive density (h) H-DBSCAN (i) K+V-DBSCAN

Figure 6: Visual comparison between clustering algorithms in a central area of Jerez de la Frontera. Note that, for each cluster, a maximum of 1000 points are displayed.

sented problem does not provide a general way to set the number of output clusters a priori. In this context, a similar issue occurs with k-means and, in fact, the results obtained by these two algorithms are quite similar for the six scenarios. H-DBSCAN, on the other hand, produces a set of clearly defined, heavily discriminated clusters (see Fig. 6.h), which results in a conceptual improvement with respect to the previous algorithms. However, approaches addressing this particular problem were not found in the literature at the time this research was performed. Hence, the algorithm works exclusively with spatial coordinates, yielding average numerical performances.

Finally, there are two algorithms specifically proposed for landmark detection. Although P-DBSCAN takes into account additional information other than the spatial coordinates, i.e., the user who posted the picture (see Section 2), it does not consider variations of density, so its performance is generally poor. Indeed, its Adaptive Den-

Algorithm	Tapia	Getafe	Jerez	Valencia	Average
DBSCAN (1996)	0.25	0.30	0.29	0.38	0.30
k-means (1967)	**0.55** (0.05)	0.70 (0.04)	0.35 (0.02)	0.58 (0.03)	0.54
Mean-Shift (2002)	0.50	0.72	0.40	0.53	0.53
Hierarch. Agglom. (1992)	0.44	0.72	0.37	0.54	0.51
H-DBSCAN (2017)	0.40	0.71	0.42	0.53	0.51
P-DBSCAN (2010)	0.23	0.52	0.16	0.15	0.26
P-DBSCAN Addt (2010)	0.31	0.65	0.37	0.56	0.47
V-DBSCAN	0.53	**0.73**	**0.54**	**0.61**	**0.60**
K+V-DBSCAN	0.53	**0.73**	**0.54**	**0.61**	**0.60**

(a) Single-PoI regions.

Algorithm	Guadalaj.	Euskadi	Average
DBSCAN (1996)	0.09	0.15	0.12
k-means (1967)	0.67 (0.03)	0.54 (0.02)	0.60
Mean-Shift (2002)	0.62	0.43	0.52
Hierarch. Agglom. (1992)	0.67	0.55	0.61
H-DBSCAN (2017)	0.63	0.51	0.57
P-DBSCAN (2010)	0.55	0.50	0.52
P-DBSCAN Addt (2010)	0.45	0.37	0.41
V-DBSCAN	0.67	0.59	0.63
K+V-DBSCAN	**0.75**	**0.77**	**0.76**

(b) Multi-PoI regions.

Table 2: Performance, expressed as Average Precision (standard deviation, when non-deterministic), of the different algorithms in the proposed locations. The 7 methods in the literature are divided according to their approach for landmark discovery. Under the double separation line, the results for our proposed system with (K+V-DBSCAN) and without (V-DBSCAN) the *PoI* Detection Module are displayed.

sity modification provides better results in single-*PoI* regions, but fails to adapt to areas with more complex density distributions (multi-*PoI* regions).

Regarding our proposal, the developed system was tested in all locations with and without the *PoI* Detection Module. We can clearly see the impact that *K-DBSCAN* has on the results looking at the single-*PoI* cases, with an average improvement of 6% with respect to the second best approach. In addition, when considering multi-*PoI* regions, the results improve by an additional 13% when we combine *K-DBSCAN* and *V-DBSCAN* (*K+V-DBSCAN*).

The visual comparison in Fig. 6 shows that our system produces an outcome that is closer to our idea of landmark: a well defined, limited area associated with a monument, area or building of interest. In this sense, the algorithm in the literature that comes closer to achieve this is H-DBSCAN, but it performs notably worse than our proposal. From our point of view, the main reasons that support this particular result are: (a) the introduction of the isolation concept in *V-DBSCAN*, which takes advantage of the assumption made in Section 4.3 regarding the radial distribution of settled areas; and (b) the use of additional meta-data, which improves the discriminant capability in the central areas of the cities, where GPS coordinates are not enough to separate close landmarks. Hence, including descriptive tags leads to results with better-shaped, more accurate clusters.

Finally, it is also relevant to discuss why, for every algorithm, performance is so location-dependent. From our point of view, this is likely due to the nature of the different GT lists. Even though the criteria for developing the GT was the same for all regions, in practice it is impossible to achieve analogous GT lists, since they had been generated by different experts, each one being a touristic expert of the region under analysis. Additionally, the six locations are very diverse in their nature, not only in terms of population, but also shape, total area, population density, touristic appeal, etc. On the one hand, we have four single-*PoI* regions of increasing population and density (Tapia, Getafe, Jerez and Valencia). On the other, we have two multi-*PoI* regions (Guadalajara and Euskadi) that are also very different population-wise. However, the fact that *K+V-DBSCAN* outperforms the rest of the algorithms in every location gives a pretty good idea of the good scalability and flexibility of our system.

To summarize, we have seen that the developed algorithms (*K-DBSCAN* and *V-DBSCAN*) not only constitute novel contributions to the field of density-based clustering, but they also have a proven direct applicability in scenarios with variable density data distributions, such as Touristic Landmark Detection. *V-DBSCAN* exploits data distributions where density gradually decreases from the data center of mass. Hence, it is very useful when applied to geodesic data points belonging to a connected region (cities, villages, etc.). In addition, the inclusion of a bag-of-words model that influences point-to-point-distance helps discriminating different entities within the crowded areas. Lastly, when combined with *K-DBSCAN*, *V-DBSCAN* can still be used to analyze data distributions with several high-density groups of points of similar sizes, separated by potentially large low-density areas. In our application scenario, this would be the case for regions that contain several towns or villages separated by long distances. By including a specific step for *PoI*-Detection (*K-DBSCAN*), we are able to separate large groups of points, therefore preventing the centroid of the data to be shifted to a non-relevant place (e.g., an isolated sample between two villages). Consequently, *V-DBSCAN* can be used independently over each detected group.

5.3.2. Assessment for automatic generation of travel guides

A very direct application of our landmark detection system is the automatic generation of travel guides. To this end, we need to produce the greatest possible number of relevant landmarks, keeping a large enough accuracy and avoiding false detections related to non-touristic places. In Fig. 7 we analyze the precision of our system when we increase the number of detected landmarks. In other words, it displays how the accuracy of the system behaves when we consider just the top K relevant clusters. It is worth mentioning that the predicted clusters are sorted by relevance, i.e., by popularity. For this experiment, our goal would be providing a precision as close to 1 as possible for the largest possible K. Indeed, we can see that, for the majority of the scenarios, *K+V-DBSCAN* provided the most robust result. In the case of Valencia, for instance, the precision did not drop until the 25^{th} cluster was analyzed, while the precision for the second best algorithm (for this scenario: DBSCAN) dropped at the 11^{th} predicted cluster. The only exception to this behavior is Getafe, where Mean-Shift and H-DBSCAN held perfect precision for two more clusters than our system. Nonetheless, their overall performance was still slightly lower than ours considering this particular scenario and, what is more, this difference is heightened when we consider the other five regions.

This proves that our system provides more relevant touristic recommendations than the compared approaches. For most of the scenarios presented, our system is able to present at least 10 relevant landmarks per location, with no false alarms. The only exception to this is Tapia de Casariego (where only the top-6 predicted clusters had perfect precision), and even in this case the accuracy at the 10^{th} cluster is quite decent (0.8).

In order to further support the argument of this system's direct applicability as a travel guide generator, we established a comparison with the well-known touristic web

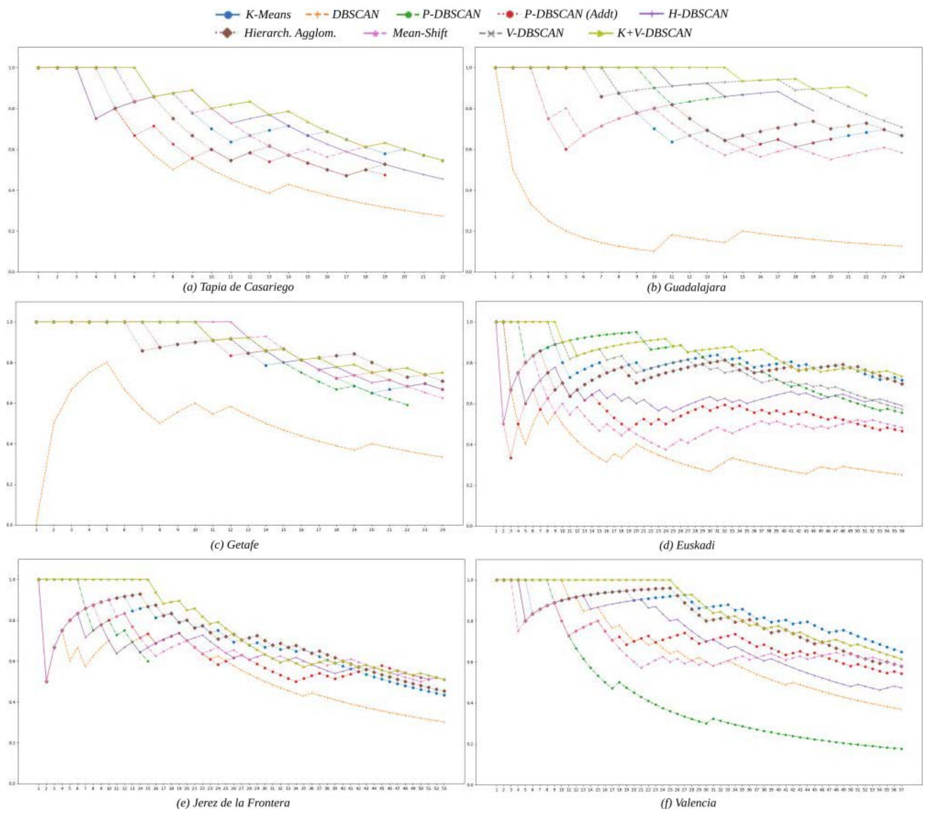

Figure 7: Variation of the precision when providing up to $K = 1, 2, ..., G$ clusters as the output of the system, where G is the length of the GT location list.

platform *TripAdvisor* [6]. Table 3 shows the number of relevant landmarks that were returned by TripAdvisor when a certain region was analyzed. For the sake of simplicity, only single-*PoI* regions where considered for this comparison. Additionally, it is worth mentioning that shops, restaurants and bars were not considered as landmarks when developing the GT lists (unless the building itself had some architectonic or cultural relevance). Note that we obtain more results from TripAdvisor for mainstream touristic destinations (Jerez, Valencia), but even in those cases the number of relevant landmarks is less than half the amount of ground-truth locations considered for this research (see Table 1). Therefore, we can infer that TripAdvisor is often insufficient to provide a

[6]https://www.tripadvisor.es

Region	N_t	Prec(N_t)	Prec($1.1N_t$)	Prec($1.2N_t$)	Prec($1.5N_t$)
Tapia	4	1.0	1.0	1.0	1.0
Getafe	9	1.0	1.0	1.0	0.923
Jerez	18	0.888	0.895	0.857	0.741
Valencia	25	1.0	0.926	0.866	0.784

Table 3: Number of landmarks provided by TripAdvisor (N_t) for each region with a single urban core and the precision of our system when providing the same N_t clusters, as well as when providing 10%, 20% and 50% more than TripAdvisor.

significant landmark list (this is particularly remarkable for less mainstream places). In the remaining columns of Table 3, we can observe the precision of our system when we attempt to provide the same amount of clusters than TripAdvisor, and also when we produce 10%, 20% and 50% more than them. One can observe that we are able to generate up to 20% more landmarks than TripAdvisor with total certainty of their relevance for the case of Tapia and Getafe and maintaining the accuracy over 0.85 for the largest cities. This proves that our system can be a very powerful tool when it is used as a travel guide generator, especially for less visited or less developed areas, where travel guides might not be available.

6. Conclusions and further work

The main goal of this work was to find clustering algorithms able to adapt to data distributions with complex density variations. In this paper, we proposed two novel density-based clustering algorithms: 1) *K-DBSCAN* is designed to identify, from the underlying distributions of the data, clusters of points with high intra-cluster density variations that may be separated by sparse areas; and 2) *V-DBSCAN* is designed to exploit data distributions where density decreases radially from a center of mass.

We have presented automatic touristic landmark discovery as a direct application for these two methods. Finding touristic landmarks is a daring endeavor. Not only because of the subjective nature of the task, but also because of the challenge of data gathering and meta-data analysis. Nonetheless, understanding how and why people visit different places might help build more sustainable tourism models in popular destinations, as well as attract interest in less popular ones. Furthermore, manually crafting travel guides is a tedious process that hinders the availability of guides in less mainstream destinations. This particular scenario raises the need for automatic tools to generate valuable content. To tackle this problem, we have made use of Flickr, a public web platform that stores user-generated multimedia content. The development of this research resulted in the generation of a dataset containing all the studied regions, which has been made publicly available.

Taking all the above into account, we have assessed our algorithms in a real scenario (a dataset obtained from Flickr). First, *K-DBSCAN* was used to discriminate independent conurbations (*Places of Interest*) within a certain region. After that, we applied *V-DBSCAN* to each resulting conurbation to provide an estimation of the landmark distribution within the region. The obtained results prove that this approach outperforms the current methods in the literature regarding landmark detection (6% increase over second best for individual cities or towns). This improvement is particularly significant when analyzing regions with multiple conurbations (15% increase over second best method in the literature).

It is worth mentioning that the output of the system presented here is designed to be the input to another system, yet to be fully developed, which will attempt to find, from the clusters provided by our Landmark Detector, the most iconic image, i.e., the view that best represents the corresponding place. This gives us plenty of open research lines to pursue. Nonetheless, there are still a few lines of work that could be explored regarding the research performed for this paper. For example, one would be to explore the inclusion of the visual analysis in the clustering procedure, in the same way that we have included a textual analysis through the *bag-of-words* model. However, image processing techniques are often extremely power-consuming, so this should be done only if an exceptionally robust input for the next stage of the project was required. Additionally, some sub-modules of the methods could be used for alternative tasks. Particularly, *K-DBSCAN*'s assignment phase could be applied to any problem that required density-based clustering but in which there exists prior information regarding the nature of the resulting clusters, e.g., clustering refinement.

Acknowledgements

This work has been partially supported by the National Grants RTC-2016-5305-7 and TEC-2017-84395-P of the Spanish Ministry of Economy and Competitiveness.

References

Becker, H., Naaman, M., and Gravano, L. (2011). Beyond trending topics: Real-world event identification on twitter. *Icwsm*, 11(2011):438–441.

Bernard, E., Naveau, P., Vrac, M., and Mestre, O. (2013). Clustering of maxima: Spatial dependencies among heavy rainfall in france. *Journal of Climate*, 26(20):7929–7937.

Cao, L., Luo, J., Gallagher, A. C., Jin, X., Han, J., and Huang, T. S. (2010). A worldwide tourism recommendation system based on geotaggedweb photos. In *ICASSP*, pages 2274–2277. Citeseer.

Cenamor, I., de la Rosa, T., Núñez, S., and Borrajo, D. (2017). Planning for tourism routes using social networks. *Expert Systems with Applications*, 69:1–9.

Cerezo-Costas, H., Fernández-Vilas, A., Martín-Vicente, M., and Díaz-Redondo, R. P. (2018). Discovering geo-dependent stories by combining density-based clustering and thread-based aggregation techniques. *Expert Systems with Applications*, 95:32–42.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619.

El-Feghi, I., Sid-Ahmed, M. A., and Ahmadi, M. (2004). Automatic localization of craniofacial landmarks for assisted cephalometry. *Pattern Recognition*, 37(3):609–621.

Elias, B. (2003). Extracting landmarks with data mining methods. In *International Conference on Spatial Information Theory*, pages 375–389. Springer.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Feng, W., Zhang, C., Zhang, W., Han, J., Wang, J., Aggarwal, C., and Huang, J. (2015). Streamcube: hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1561–1572. IEEE.

Forsyth, D. A. and Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Frias-Martinez, V., Soto, V., Hohwald, H., and Frias-Martinez, E. (2012). Characterizing urban landscapes using geolocated tweets. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 239–248. IEEE.

Gerard, S. and Michael, J. M. (1983). Introduction to modern information retrieval.

Jayarajah, K. and Misra, A. (2016). Can instagram posts help characterize urban micro-events? In *Information Fusion (FUSION), 2016 19th International Conference on*, pages 130–137. IEEE.

Ji, R., Gao, Y., Zhong, B., Yao, H., and Tian, Q. (2011). Mining flickr landmarks by modeling reconstruction sparsity. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 7(1):31.

Kennedy, L., Naaman, M., Ahern, S., Nair, R., and Rattenbury, T. (2007). How flickr helps us make sense of the world: context and content in community-contributed media collections. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 631–640. ACM.

Kisilevich, S., Mansmann, F., and Keim, D. (2010). P-dbscan: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In *Proceedings of the 1st international conference and exhibition on computing for geospatial research & application*, page 38. ACM.

Lee, I., Cai, G., and Lee, K. (2014). Exploration of geo-tagged photos through data mining approaches. *Expert Systems with Applications*, 41(2):397–405.

Lin, J. and Cromley, R. G. (2018). Inferring the home locations of twitter users based on the spatiotemporal clustering of twitter data. *Transactions in GIS*, 22(1):82–97.

Llobera, M. (2001). Building past landscape perception with gis: Understanding topographic prominence. *Journal of Archaeological Science*, 28(9):1005–1014.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Evaluation in information retrieval*, page 139–161. Cambridge University Press.

McInnes, L., Healy, J., and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011). Exploiting semantic annotations for clustering geographic areas and users in location-based social networks. *The social mobile web*, 11(2).

Papadopoulos, S., Zigkolis, C., Kompatsiaris, Y., and Vakali, A. (2011). Cluster-based landmark and event detection for tagged photo collections. *IEEE MultiMedia*, 18(1):52–63.

Rasmussen, E. M. (1992). Clustering algorithms. *Information retrieval: data structures & algorithms*, 419:442.

Scott, D. W. (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, Inc, second edition edition.

Tang, J. and Meng, L. (2006). Learning significant locations from gps data with time window. In *Geoinformatics 2006: GNSS and Integrated Geospatial Applications*, volume 6418, page 64180J. International Society for Optics and Photonics.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584.

Yang, Y., Gong, Z., et al. (2011). Identifying points of interest by self-tuning clustering. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 883–892. ACM.

Zheng, Y.-T., Zhao, M., Song, Y., Adam, H., Buddemeier, U., Bissacco, A., Brucher, F., Chua, T.-S., and Neven, H. (2009). Tour the world: building a web-scale landmark recognition engine. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, pages 1085–1092. IEEE.

Zhou, X., Xu, C., and Kimmons, B. (2015). Detecting tourism destinations using scalable geospatial analysis based on cloud computing platform. *Computers, Environment and Urban Systems*, 54:144–153.

Algorithm 3 V-DBSCAN Algorithm, including DBSCAN clustering function.

Input: A set of points **p**, initial neighborhood radius (ε_0), neighborhood radius update factor (η_ε), correction factor (κ), unchanged levels threshold (L).

Output: Set of clusters C (predicted landmarks).

```
 1: function V-DBSCAN($\mathbf{p}, \varepsilon_0, \eta_\varepsilon$)
 2:     $C_0$ = create-empty-cluster(0)
 3:     $C_0$ = combine-unique($\mathbf{p}$,$C_0$)
 4:     $\varepsilon = \varepsilon_0$
 5:     $C$ = create-set$\{C_0\}$
 6:     $unchanged = 0$
 7:     while $unchanged < L$ do
 8:         $C_{old} = C$
 9:         for $i$ from 0 to $|C| - 1$ do
10:             $S$ = dbscan($C_i$, $\varepsilon$)
11:             $S$ = sort-by-relevance($S$)
12:             for $j$ from 0 to $|S| - 1$ do
13:                 $S_j^{\complement}$ = remove-points-from-cluster($S_j, C_i$)
14:                 if $Dist(S_j, S_j^{\complement}) > MInt(S_j, S_j^{\complement})$ then
15:                     $C$ = add-cluster-to-set($S_j$, $C$)
16:                     $S$ = remove-cluster-from-set($S_j, S$)
17:                 end if
18:             end for
19:         end for
20:         if $C_{old} == C$ then
21:             $unchanged = unchanged + 1$
22:         else
23:             $unchanged = 0$
24:         end if
25:         $\varepsilon =$ update($\varepsilon$,$\eta_\varepsilon$) // see eq. (3)
26:     end while
27:     return $C$
28: end function
```

Algorithm 4 DBSCAN clustering function.

Input: A set of points D and parameters (ε, $MinPts$).

Output: Set of clusters C

```
1:  function DBSCAN($D$, $\varepsilon$, $MinPts$)
2:      $i = 0$
3:      while points unclassified $> 0$ do
4:          $p$ = get-next-unclassified-point($D$)
5:          $C_i$ = create-empty-cluster($i$)
6:          $C_i$ = combine-unique($C_i$,$p$)
7:          $C_i$ = expand-cluster($C_i$,$D$, $\varepsilon$, $MinPts$)
8:          if $|C_i| == 1$ then
9:              classify-as-noise($C_i$)
10:         else
11:             $i = i + 1$
12:         end if
13:     end while
14:     return $C$
15: end function
```