Density-based clustering: algorithms and evaluation techniques

by

Eduardo Pla Sacristán

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Multimedia and Communications

Universidad Carlos III de Madrid

Tutor and Advisor:

Iván González Díaz

September 2021

This text is dedicated to Maria Ángeles Vall Ojeda:
a magnificent archeologist,
a generous, eternal teacher,
an exemplary role model,
a wonderful grandmother.

Un beso, iaia.
Adios, adios.

You did it. You crazy son of a b___, you did it.

– Ian Malcolm, *Jurassic Park*

ACKNOWLEDGEMENTS

Gracias, en primer lugar, a mi madre, Maria Ángeles. Suena a *cliché*, pero suele ser cierto: no podría haber llegado hasta aquí sin ti. Haz sitio en el salón para otra foto con atuendo extravagante, porque allá vamos.

Quiero agradecer a mi tutor y mentor, Iván González, sus consejos y guías que me han permitido escribir este documento (y otros tantos) y, además, haber aprendido un par de cosas por el camino. Quisiera, asimismo, extender este agradecimiento a Fernando Díaz, por haber apostado por mi y haber confiado en mi trabajo estos años.

Volviendo al plano personal, agradezco a mis amigas de toda la vida de Getafe, así como a Jose, Adrián y Andrea, haber fingido que les interesaba cuando hablaba de mi tesis. También quisiera tomarme un momento para recordar a mi familia, tanto a la getafense como a la valenciana, tanto a la que se fue como a la que sigue, y en especial a mis queridos hermanos: Miguel, Marian y Elena, y a mi padre, al que tanto hubiera gustado llegar a conocer a las personas que hoy somos.

Para cerrar este asunto, quiero mentar y dar las gracias a Marta. Si estos años de tesis fueran una película, tu nombre saldría antes que el título con letras grandes y luminosas.

PUBLISHED AND SUBMITTED CONTENT

- Published contribution, wholly included in the Thesis:
  - [1] Pla-Sacristán, E., Gonzalez-Diaz, I., Martinez-Cortes, T., & Diaz-de-Maria, F. (2019). Finding landmarks within settled areas using hierarchical density-based clustering and meta-data from publicly available images. Expert Systems with Applications, 123, 315-327.
    - The material from this source included in this thesis is not singled out with typographic means and references.
- Submitted contribution, wholly included in the Thesis:
  - Pla-Sacristán, E., Gonzalez-Diaz, I., & Diaz-de-Maria, F. (submitted 2021). *SynFlex*: A generative model for benchmarking density-based clustering. Submitted to: Pattern Recognition [1].
    - The material from this source included in this thesis is not singled out with typographic means and references.

[1] Pattern Recognition: https://www.journals.elsevier.com/pattern-recognition

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

Acronym	Meaning
AMI	Adjusted Mutual Information
AP	Average Precision
API	Application Programming Interface
BoW	Bag-of-Words
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
CH	Calinski Harabasz
CHR	Cluster-to-Hypercube Ratio
DBCV	Density-Based Clustering Validation
EM	Expectation-Maximization
ENR	Element-to-Noise Ratio
FLAG	Fluctuation-Agglomeration
GPS	Global Positioning System
GT	Ground-Truth
HDBSCAN	Hierarchical-Density-Based Spatial Clustering of Applications with Noise
KDBSCAN	Kernel-Density-Based Spatial Clustering of Applications with Noise
K+VDBSCAN	Combination of KDBSCAN and VDBSCAN
KDE	Kernel Density Estimation
MI	Mutual Information
MST	Minimum Spanning Tree
NTP	Normalized Topographic Prominence
P-DBSCAN	Photo-Density-Based Spatial Clustering of Applications with Noise
PoI	Place-of-Interest
SNR	Signal-to-Noise Ratio
SV	Separation-Variance
SynFlex	Synthetic data generation with flexible parametrization
tf-idf	Term frequency - inverse document frequency
TP	Topographic Prominence
VDBSCAN	Variable-Density-Based Spatial Clustering of Applications with Noise

Table 1: *List of the most relevant acronyms used in this document in alphabetical order.*

CHAPTER 1

INTRODUCTION

Density-based clustering algorithms involve a relevant subset of all the methods developed for cluster analysis, which is one of the fundamental pillars of unsupervised learning [2]. While the origins of clustering can be traced to the early 20^{th} century [3], it is not until the 1990s that the concerns that would lead to develop density-based clustering algorithms are raised [4]. In 1996, the most popular density-based clustering algorithm to date (DBSCAN) is published [5] and, with it, many applications for density-based clustering are found within increasingly different fields over the next decades.

In this introductory chapter, we present an overview of the research that led to this dissertation, focused mainly on density-based clustering. The work presented in this document can be divided into two main blocks, which, briefly stated, are: (1) research on the development of novel density-based algorithms and (2) research on evaluation techniques and metrics for density -based clustering. The motivation that led to this approach is expressed in Section 1.1. First, the original motivation to pursue the study of density-based clustering algorithms (landmark discovery) is introduced in Section 1.1.1. After that, in Section 1.1.2, we explain the demand for an evaluation benchmark applicable to density-based clustering algorithms.

In Section 1.2, the main objectives of this thesis, which emerge from the demands and opportunities introduced in the motivation section, are presented and justified. Subsequently, we introduce the main scientific contributions of this thesis (Section 1.3). A notation guide is then included to serve as a reference for the reader (Section 1.4). Lastly, the description regarding the structure of this document is included in Section 1.5.

1.1. Motivation

Density-based clustering techniques are used in a wide range of scenarios. Since they were originally conceived for spatial databases [5], [6], the most typical usually involve this type of data. In this context, even many of the recent instances of research within the field of density-based clustering are proposed generically with spatial datasets in mind [7], [8]. Nevertheless, other lines of research have proposed alternative applications for density-based clustering. Some of them are directly related to physical, real-world spatial features, like the study of human mobility [9], trajectory segmentation [10] and landmark discovery [1], which are all mainly based on the use of GPS data. However, density-based clustering has also been applied in many other diverse contexts, such as

Figure 1.1: *Example of 9 landmarks to be discovered in the city of Madrid (Spain). Map obtained from OpenStreetMaps*[2].

audio source analysis [11], image segmentation [12] or motion pattern analysis [13].

A frequent characteristic among scenarios where density-based clustering algorithms are applicable is the scarcity of labeled data. Moreover, even when labels are available, there is often disparity of criteria in the annotations. Therefore, the development of specific evaluation techniques for density-based clustering is of paramount importance. Furthermore, there is a need to standardize these evaluation techniques to provide a sensible consensus. In this context, the acquisition of relevant data to conform an evaluation benchmark is a robust option to cover the mentioned needs.

It is worth mentioning that this thesis has been partially developed within the framework of an existing research project, which focused on obtaining touristic landmarks from publicly available data (geo-located images and some additional textual information). As a result, the research presented in this thesis is often applied to the task of landmark discovery, which becomes the standard case of study for the proposed algorithms and evaluation techniques.

1.1.1. The landmark discovery task

The main goal of automatic landmark discovery is to determine, within a certain distribution, concealed regions or points that are particularly relevant, given a specific criterion. Landmark discovery is a particularly challenging task, often due to the difficulty of determining relevant characteristics that support the identification of a landmark. This problem is illustrated in Fig. 1.1, where 9 potential landmarks to be discovered in a map of the city of Madrid, annotated by a hypothetical observer, are displayed. On this map, we can no-

[2]OpenStreetMap: https://www.openstreetmap.org/copyright

tice some of the potential problems that can be encountered in landmark discovery. The first noticeable aspect is the disparity of clusters' shapes and sizes, which, as we will discuss later on in Section 2.1.1, can be problematic for some techniques. As we reflect on the figure, other problems can arise. For instance, cluster 2 contains Spain's Royal Palace, which is undoubtedly a relevant touristic landmark. However, immediately south of it is Madrid's Cathedral which, for this (arbitrary) example, is not relevant enough to be considered a relevant landmark. And so, the first potential disparity of criteria is encountered. Similarly, cluster 9 (park *El Retiro*) is annotated as a single cluster, but a different observer could have viewed its internal monuments as singular, independent entities (dotted lines). Another potential problem is data similarity. For instance, clusters 7 (Columbus Square) and 8 (National Library) are significantly close, so assuming, for instance, that we only rely on geographical information, they could be indistinguishable to an automated system. For the rest of the clusters (1,3,4,5 and 6), we can observe another potential problem. Although no major difficulties should be expected regarding their discovery, since they are reasonably separated from the rest, their shape could vary from one annotator to another (e.g. for cluster 6, the *Atocha* train station, the landmark could include the train tracks or not).

Hence, a consensus for what constitutes a landmark must be agreed upon. 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 chosen to address the automatic discovery of landmarks in various fields [14][15]. Particularly, real-world landmark discovery applications often focus on spatial datasets [16]. Within this context, the discovery of touristic landmarks is a field worth studying. Expert tools to support the tourist industry have been previously developed, like an application to detect unexpected behavior and prevent or mitigate undesired situations within cities [17] or a personalized route-planner based on social-media data [18]. Nonetheless, the discovery of relevant landmarks is a particularly recurrent topic among the state-of-the-art [19]–[21]. 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 [22]; in others, user-generated images are employed and their visual properties can be used to discriminate between landmarks [23]. In this sense, the most common approach relies on geo-located images [24][25], 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 [26].

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 [27] [28] and showed decent results. However, mainly due to the fact that most approaches focus on 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 large region that consists of several non-connected inhabited cores (e.g., a coastal area composed of several villages). Traditional density-based approaches like DBSCAN [5] fail to consider the possibility of diverse-density clusters coexisting in the same sample space. In this scenario, hierarchical approaches can become a decent solution for the problem of scale [29], whereas variations in density have been also considered and modeled in some previous works [26]. However, the former were not conceived with the specific task of landmark discovery in mind and, therefore, have some inconvenient limitations that will be later meticulously addressed in Chapters 3 and 6. These chapters also discuss the latter (P-DBSCAN) and show that it does not provide appropriate results in our scenarios.

Consequently, there was an opportunity for additional research in this context to address the aforementioned problems through the development of novel techniques (see Section 1.2).

1.1.2. Density-based clustering evaluation: a benchmark

Ideally, when a novel clustering algorithm is developed, one should provide an accurate idea of its behaviour. This is often done through empirical evidence, whether it is in generic scenarios and datasets [30] or in a particular context of application of the algorithm [31]. Furthermore, there is another option often employed when a new algorithm is implemented (or when we want to study the behaviour of existing algorithms), which consists in assessing it using a more general and rich evaluation benchmark.

Benchmark datasets are chosen to represent different characteristics in data distributions and, hence, reflect the behaviour of an algorithm in such circumstances. Such data can be either real-world data or synthetic, with each option having advantages and disadvantages. Authors in [32] and [33] both discuss these pros and cons. While the former criticizes the use of synthetic data for evaluating the actual usefulness of the algorithm, the latter argues that, in fact, employing real-world datasets makes sense if the final (and preferably unique) purpose of the algorithm is known, whereas the use of synthetic data is quite enlightening if certain specific behaviours are analyzed, as it allows the isolation of those particular characteristics while reducing the impact of others. Such an approach is obviously impossible when dealing with real-world datasets.

There are a variety of benchmark datasets that may be used to observe algorithm behaviour, from typical toy datasets (like the one shown in Fig. 1.2) to datasets explicitly gathered for specific purposes: image segmentation [35], web document [36] or audio [37] clustering. However, at the time this dissertation was written, a significant gap was found in the literature with respect to benchmark data for density-based clustering.

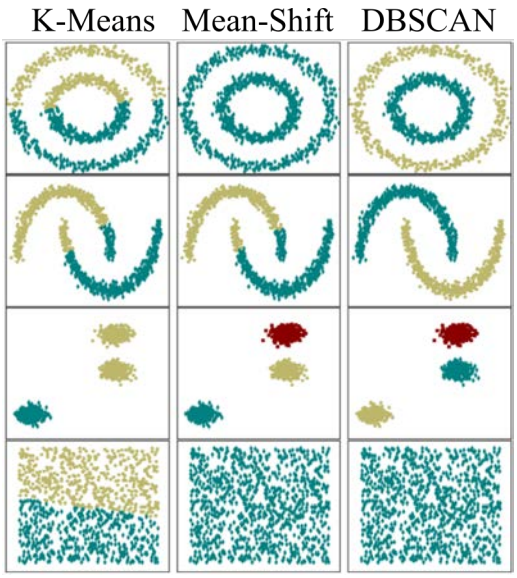


Figure 1.2: *Behaviour of several clustering algorithms (K-Means, Mean-Shift and DBSCAN) in toy datasets with different characteristics. Taken from [34].*

In other words, when density variations are paramount to correctly identify groups within the data, researchers do not have a reference benchmark to resort to, and they often have to gather specific datasets to evaluate the performance of the clustering algorithms. This is the case of automatic landmark discovery, for instance, where labeled data is not easy to find (see Section 1.1.1). Although some survey articles [33] suggest the use of synthetic, parametrizable data to study the statistical performance of the algorithms, which could potentially be adapted for density-based clustering algorithms, they emphasize on the unrealistic nature of their synthetic data, which effectively reduces the impact of the benchmark data.

Another factor that should be accounted for when considering a benchmark is the choice of the evaluation metric. There are two types of metrics to be considered when evaluating clustering partitions: (1) external metrics, which rely on some ground-truth labeled partition to be compared to the partition under analysis and provide a measure of their similarity; and (2) internal metrics, which are based on the characteristics of the data that belong to each cluster, usually providing a measure of cluster compactness and separation.

When labeled data is available, the use of external evaluation is the obvious choice, since it is based on the true nature of the data. Many comparisons between such evaluation techniques have been made in the context of clustering [38][39] and, particularly, in the context of density-based clustering [40]. However, due to the scarcity of annotated databases [41], external evaluation is not always possible.

Of course, we can also resort to internal metrics to establish the performance of an algorithm. These have also been employed when labeled databases were not available [42]. Analogously to the case described in the paragraph above, comparison studies between

such indexes have also been made [43]. However, internal metrics, as we will discuss in depth in this dissertation, do not always realistically reflect the actual behaviour of an algorithm. Therefore, their use is often restrained to certain scenarios with limited characteristics.

Overall, the current state-of-the-art evaluation possibilities, considering both the available benchmark data and the potentially applicable metrics, are insufficient to rigorously study the behaviour of density-based clustering algorithms. Naturally, these claims are further explored and properly justified in the subsequent chapters of this dissertation. In the next section, we summarize the approach taken in this research to overcome these limitations.

1.2. Thesis objectives

As it was introduced in the previous sections and will be further developed in the following chapters, there is a clear opportunity to develop the field of density-based clustering both by innovating in the context of algorithm design and by expanding the available background regarding evaluation methods in the field. This dissertation sows the seeds of these demands and attempts to address them through novel scientific contributions. To this end, the following objectives have been established:

1. To design and implement a new density-based clustering technique tailored to the task of automatic landmark discovery, i.e, suitable for scenarios where clusters may have varying shapes and densities.

2. To develop a method that automatically generates synthetic data with a flexible, parametrizable system that truthfully reflects the nature of data distributions with arbitrary characteristics such as dimensionality, noise, shape, density variations, etc.

3. To design and implement a new internal evaluation technique that resembles in the most accurately possible way the performance of external metrics within the field of density-based clustering.

4. To gather a sufficiently reflective dataset collection, both synthetic and real, to be able to build an evaluation benchmark for density-based clustering.

1.3. Scientific contributions

The pursuit of fulfilment of the aforementioned objectives led to the proposal of some novel scientific contributions discussed in this dissertation. These contributions, which are fully described and justified in the next chapters, can be grouped into three categories: (1) algorithms, (2) databases and (3) evaluation techniques.

Regarding the first category, we present two novel density-based clustering algorithms (*KDBSCAN* and *VDBSCAN*) that have a direct applicability on the task of automatic landmark discovery. 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.

With respect to the acquisition of data, geodesic content from diverse real locations was gathered to directly evaluate the behaviour of *KDBSCAN* and *VDBSCAN* in the task of landmark discovery. Furthermore, a new flexible data generation system was developed (*SynFlex*). This system's parameters can be used to imitate certain realistic data characteristics that are not often found in synthetic data, like variety of shapes, hierarchy depth or inter-cluster density variations.

Finally, regarding the third category, a new internal evaluation metric (*FLAG* index) has been designed to reflect the performance of algorithms in datasets with strong density variations and cluster shape arbitrariness. Moreover, the proposed *SynFlex* system and *FLAG* index are combined with existing external metrics from the literature (see Section 2.2.2) to build an evaluation benchmark (containing both real and synthetic data) that reflects the behaviour of a given clustering algorithm in relevant scenarios where variations in density, dimensionality, shape, noise presence and hierarchy are factors of special relevance.

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

1. A *Kernel-based* variation of DBSCAN (*KDBSCAN*), published in [1], whose goal is to 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 (*VDBSCAN*). This algorithm, published in [1], 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 partially labeled heterogeneous locations, from single conurbations to larger regions with non-connected cores. This dataset will be used to evaluate the aforementioned algorithms and compare them with other relevant algorithms.

4. A novel synthetic data generation system with flexible parametrization (*SynFlex*) that reflects real data characteristics. This system will be used to generate realistic, synthetic data to evaluate density-based clustering algorithms.

5. A new internal evaluation index (*FLAG*) based on intra-cluster *fluctuation* and the *agglomeration* of elements in the local environment of each cluster (inter-cluster dispersion). This metric will be used to evaluate the performance of clustering algorithms when annotated real-world data is unavailable.

6. A benchmark designed to reflect the behaviour of a clustering algorithm with respect to five crucial factors in cluster analysis: shape, density, noise presence, hierarchy depth and dimensionality. This is done through the use of synthetic data and a selected number of unlabeled real-world datasets.

1.4. Notation guide

In this section, we present a quick guide for the key notation used in this document and some additional clarifications for the sake of comprehension. The notation used and its description are shown in Table 1.1.

Note that, when sub-indices are defined as part of the elements name, the element is defined accordingly. For instance, let us define ξ_G as a variable, then G is part of the term (i.e., it does not define the G^{th} element of a given ξ). Furthermore, note that even though an element can be referred to as a *point*, if it is represented by a vector of coordinates, it is denoted as a vector. Finally, it is worth noticing that a set C can contain scalars, vectors or other sets.

Notation	Description
a	scalar (index, element of a vector, etc.)
$\boldsymbol{b}$	vector
C	set
$C^{\complement}$	complement of set C
b_i	i^{th} element of vector b; scalar
$b_{i,j}$	j^{th} position of the i^{th} element of vector b
$\|C\|$	Cardinality of set C
c_i	i^{th} element of set C; scalar
$\boldsymbol{c_i}$	i^{th} element of set C; vector
C_i	i^{th} element of set C; set

Table 1.1: *Notation guide.*

1.5. Structure of the document

The last section of this introductory chapter consists of a brief presentation and description of the contents of the subsequent chapters.

Chapter 2 provides a review of the current literature and related technologies. First, a review on density-based clustering is presented, including a brief overhaul of its historical context and its main practical applications. Then, we include an assessment of the current evaluation techniques applicable to unsupervised learning, highlighting the role of database acquisition and some of the main external and internal metrics. Finally, the particular context of landmark discovery is explored, since it is a fundamental focus of the research that led to this dissertation.

Chapter 3 introduces the proposed density-based clustering algorithms. *KDBSCAN* and *VDBSCAN* are two modifications of the classic DBSCAN algorithm that can be used in different contexts. Nonetheless, we apply them in the particular context of landmark discovery. The algorithms are rigorously defined and the application context is described in detail.

Chapter 4 describes the experiments carried out to validate and demonstrate the performance of the proposed algorithms (*KDBSCAN* and *VDBSCAN*). The details regarding the gathered datasets from real locations and their associated landmarks are explained, and the results obtained by the algorithms (as well as a comparison with other relevant algorithms) are discussed.

Chapter 5 introduces the techniques developed with the goal of establishing an adequate benchmark for density-based clustering algorithms. First, the synthetic data generation system with flexible parametrization (*SynFlex*) is explained and, after that, the developed internal metric (*FLAG*) is also presented. To conclude, the main characteristics of the benchmark are introduced.

Chapter 6 presents the validation experiments for the techniques presented in Chapter 5. First, the *SynFlex* system and the *FLAG* index are verified using a real-world, fully annotated dataset. Then, the particular specifications of the benchmark experiments are broken down and the results yielded by several existing clustering algorithms in this benchmark are thoroughly examined and discussed.

Chapter 7 offers closure on this dissertation by presenting the principal conclusions drawn from this thesis and offering some potential lines of work associated with this research.

CHAPTER 2

STATE OF THE ART

Clustering, as expressed by [44], is a type of classification imposed on a finite set of objects. The most important particularity of clustering among classification techniques is its unsupervised, intrinsic nature [45]. In other words, only the nature of the data itself, and not its labels, class or category (if known) is used for cluster analysis. Cluster analysis is a complex field of research that led to the design and implementation of many algorithms over the years [46]. Depending on the specific problem to be solved, different clustering criteria and, hence, different clustering algorithms, should be employed.

For instance, centroid-based algorithms, like the classic K-Means algorithm (which will be further discussed and explored in this document), use the distance from each sample to certain position vectors (centroids) to establish the belonging to each cluster [47]. Hierarchical models, on the other hand, usually rely on a dendrogram (either using a bottom-up or top-down approach) to determine the similarities between samples and group them into clusters [48]. Distribution-based algorithms, like the well-known EM algorithm [49], attempt to define density distributions for each cluster and assign the samples to the one with the highest likelihood. Lastly, density-based clustering algorithms differentiate clusters that are separated from each other by contiguous regions of low density of samples [50]. This document focuses mainly on this last type of clustering.

A worthy issue to discuss with respect to any type of unsupervised learning is evaluation. Since algorithms do not require of labels or classes to sort data into groups, there is no need to use labeled datasets to employ clustering algorithms. However, this yields the necessity of employing internal metrics for evaluation, which is not always convenient. This downside could be tackled by employing labeled databases even if there is no need for them in the learning process. However, the unavailability of labeled datasets is already a major bottleneck for the development of supervised learning algorithms [51], so this might not always be a realistic evaluation option.

In this chapter, we will first further explore the concept of density-based clustering, establishing its historical context and discussing some of its applications (Section 2.1). Additionally, we discuss the evaluation of unsupervised learning algorithms (Section 2.2), first by exploring the possibility of building labeled databases and then by discussing available evaluation metrics for the cases where labels can be obtained (external) and also for those cases in which this is not an option (internal). Finally, in Section 2.3, we review and discuss techniques that tackle real-world, practical applications that will later be the main focus of Chapters 3 and 4: landmark discovery.

2.1. Density-based clustering

Among the different ways to tackle the clustering task, density-based methods are characterized by establishing a sample's affiliation to each cluster through the analysis of its surrounding density. Consequently, clusters are considered to be high density regions of the data distribution [50]. Basic density-based clustering approaches have two convenient characteristics that make them suitable to work in certain environments: (1) the number of resulting clusters does not need to be known *a priori*, and (2) no assumptions regarding the nature of the underlying data distributions are made. This allows density-based clustering algorithms to generate clusters with arbitrary shapes, unlike most alternative approaches, which usually lead to convex clusters with globular shapes.

In this section, we explore the historical context and motivations that led to the development of density-based clustering algorithms, along with a review of their applications.

2.1.1. Historical context

The most well-known density-based clustering algorithm is probably *a density-based algorithm for discovering clusters in large spatial databases with noise (DBSCAN)* [5]. Many modifications and variations of this algorithm have been proposed for different tasks since it was published in 1996 [26], [52], [53]. In the original DBSCAN, authors proposed a method based on packing together samples based on their vicinity. If a sample has enough neighbours in its vicinity, it becomes a *core point* and can be expanded to become a more populated cluster. The minimum number of neighbours considered to become a *core point* ($MinPts$) and the extension of the vicinity (ε) are the main parameters to be adjusted by the user in DBSCAN. Take the example shown in Fig. 2.1: at the end of the expansion, a cluster is formed by core points that are connected by a high density chain of points (red samples in Fig. 2.1) and their reachable neighbours (yellow samples in Fig. 2.1). Those samples that are not reachable from any core point are considered outliers or *noise* (blue sample in Fig. 2.1).

In the original research [5], authors mention among the main motivations for their research the inability for previous, non-density-based clustering algorithms to form clusters with arbitrary shapes (spherical, linear, elongated, curved, etc.) and their requirement of prior knowledge regarding the nature of the data (type of distribution, number of clusters, etc.). Therefore, they proposed that by taking into account the variations in density directly, without making further assumptions of the underlying distribution, these limitations were removed. Although this was the original motivation, authors in [50] discuss some other natural inspirations (species divergence in zoology, topographic distributions) behind the motivation to use density-based clustering techniques.

Nevertheless, as it happened with the parameters of non-density-based algorithms, finding appropriate values for the ε parameter is not always trivial [55]. Moreover, a single value for this parameter might not suffice, since clusters with different densities

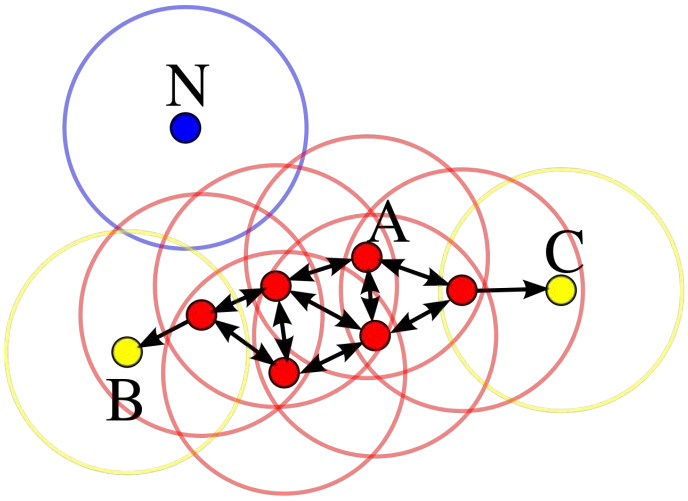


Figure 2.1: *DBSCAN cluster example with MinPts = 4 and ε equivalent to the radius of the circumferences. Figure taken from [54].*

may be present in a single data collection [1]. This problem has been mainly addressed through developing modifications of DBSCAN, as we explain in the following section.

2.1.2. Density-based clustering methods

Following the proposal of DBSCAN [5], the field of density-based clustering was greatly expanded through numerous algorithms. We can group these algorithms into two main categories based on their approach: (1) local algorithms, which start from a point and expand the clusters without taking into account the whole distribution; and (2) global algorithms, which first analyze the sample space and then use the obtained information to cluster the samples.

Local algorithms: the original DBSCAN belongs to this category, since it treats the sample space point by point. A popular group of algorithms that fall into this category are hierarchical density-based clustering algorithms. These work by building a dendrogram that can then be pruned following a given criterion. One of the first instance of hierarchical density-based algorithms in the literature is OPTICS [56], which differs from DBSCAN in that, instead of processing all points in an arbitrary order, it establishes a priority queue based on distance between points (spatially close samples are close to each other in the queue). It then stores a distance from each point to the next in the queue that can be used to build a reachability-plot (see Fig. 2.2) from which the clusters can be extracted. OPTICS' main disadvantage is that its high time complexity makes it difficult to apply for large databases. Alternatively, an improved hierarchical density-based algorithm, *HDBSCAN* [29], was proposed. *HDBSCAN* performs DBSCAN with varying ε values over the data, and then it integrates the results to achieve the best partition according to a certain cluster stability measure [57]. Additionally, the algorithm *VDBSCAN* [1], which is later explained as a part of this thesis (see Section 3.1.2) also follows a hierarchical approach.

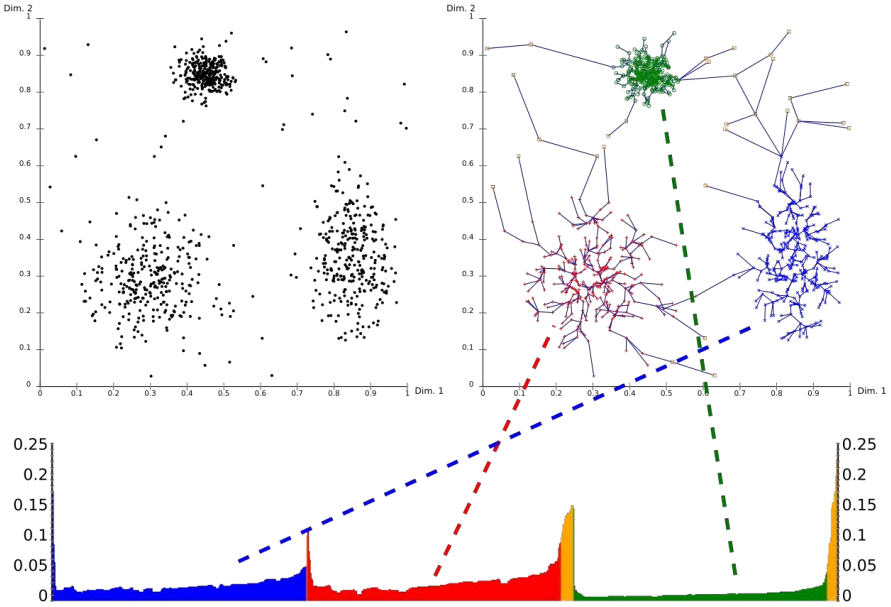


Figure 2.2: *OPTIC's reachability-plot for an example of three clusters. Figure taken from [58].*

Another type of local algorithms rely on adaptive strategies to handle density changes. This is the case of DVDBSCAN [59], which compares the density of the growing cluster to the density of the neighborhood of a *core-point* that is a candidate to be added to the cluster. If the comparison between them satisfies the joining criterion (a threshold and a similarity index) the cluster is expanded correspondingly. An additonal example of adaptive algorithm is P-DBSCAN [26], which handles the density changes through an Adaptive Density parameter that prevents the points to be added to the cluster if the density drops below a certain threshold.

Global algorithms: these category includes algorithms that analyze the whole sample space before clustering the data. This is not a common approach in density-based clustering, since one of the main motivations of developing this field was to be able to adapt to local changes in density (see Section 1.1, which is harder to do if you fix certain conditions beforehand (e.g., the number of output clusters). Nevertheless, some examples of this approach can be observed in the literature. The most popular is probably DENCLUE [60], which performs a Kernel Density Estimation (KDE) over the data to compute density attractors for each point that will determine, through a typical hill-climbing process (see Fig. 2.3), the cluster to which they are grouped. This method, however, has a large number of parameters, so it is difficult to adjust to different scenarios. A modification of this algorithm, DENCLUE-IM [61], was also proposed to improve the algorithm by eliminating the costly hill-climbing step. In this thesis, another algorithm that makes use of KDEs (*KDBSCAN* [1]) is presented as a part of this thesis, and will be explained in detail in Section 3.1.1.

As we previously discussed in Section 1.1, the research presented in this document is

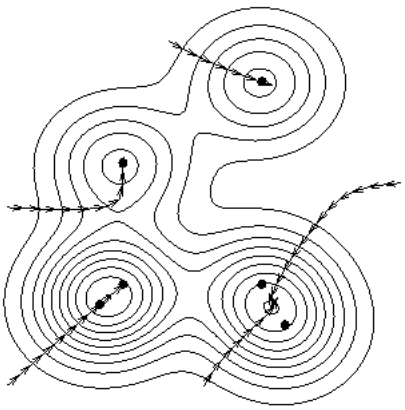


Figure 2.3: *Representation of the hill-climbing process for a given Kernel Density Estimation. Figure taken from [62].*

often tied to the particular application of landmark discovery, so most of the analysis is based on it. However, the diverse application possibilities of density-based clustering are worth reviewing. This is done in the following section.

2.1.3. Applications of density-based clustering

If we refer once more to the original publication of DBSCAN [5], we can observe that authors proposed the algorithm with spatial databases in mind. Indeed, many of the applications of density-based clustering can be associated with spatial databases. In [6], a generalization of DBSCAN is proposed for spatial databases and can be used for molecular biology, astronomical and geographical data, among others. Another density-based clustering algorithm for spatial databases is proposed in [63], with the novelty of taking into account physical obstacles that may affect the clustering process.

A particular spatial application of special interest in this thesis is landmark discovery. This application is explored in detail in Section 2.3, since, as we have already mentioned, it is one of the main goals of this research. Nevertheless, at this point it is important to specify that this task can combine the use of spatial data with other types of data, like textual [26][64] or temporal [52][65] information.

Other applications can be also found in the literature, like those that deal with image databases: image matching [66], generic color image segmentation [12] and, particularly, biomedical image segmentation [67]. Density-based clustering of sound data has also been tackled (in [68] it is applied for voice pathology detection), as well as clustering of pure textual information, like web opinions and social interactions [69]. Lastly, authors in [70] propose a method to apply density-based clustering to uncertain types of data.

Hence, we have seen that density-based clustering can have a wide range of applications aside from grouping spatial information. Later in this dissertation (Chapters 5 and 6), the effect that different types of data have on the performance of clustering algo-

rithms will be explored further.

2.2. Evaluation methods in unsupervised learning

The gathering of data to train supervised learning algorithms is often a challenge, due to the scarcity of labeled data. On the other hand, since unsupervised learning algorithms do not require labeled data to be adjusted, we do not encounter this problem at the point of developing, implementing or adjusting a system.

However, evaluation is a very different challenge. Since internal metrics are solely based on the characteristics of the clustering results (regardless of the authentic, original group of each sample) they often fail to reflect the true task-based performance of the applied algorithm or, at least, lead to scenarios where different metrics disagree on the results [71]. Therefore, if possible (i.e., if we have access to labeled data), it is convenient to evaluate clustering algorithms using external metrics.

In this section, we first discuss the possibilities for database acquisition (Section 2.2.1), whether it is real data (labeled and unlabeled) or synthetically generated. After that, the current literature on external metrics (Section 2.2.2) and internal metrics (Section 2.2.3) applicable to clustering algorithms is reviewed.

2.2.1. Database acquisition for density-based clustering

Obtaining data and, particularly, labeled data, can be a challenging task. Privately owned databases are not always accessible [72] and generating real databases by gathering and annotating the data personally can be a strenuous and tedious process. This process is particularly discussed for the context of this research, along with an analysis of the available sources, in Section 2.3.

A suitable solution to the scarcity of data is resorting to synthetic data generation. However, data generation systems are quite diverse, and finding one that appropriately fills our needs is not always possible.

Some data generation algorithms are centered in categorical and relational data generation. In [73], a system is designed to generate data suitable for online transactions and streaming applications, while the work in [74] centers on producing data for statistical testing. Others are even more task specific, devoted to produce data for healthcare applications [75] or fraud detection [76].

Regardless of the application, another drawback that we can encounter is the system's requirement for already existing data as an input (which serve as an example of the data to be produced) to be able to model the data [74] or the specification of certain scenario characteristics [77].

Since this research is focused on the analysis of density-based clustering algorithms,

we would ideally need a system able to generate data with varying densities and arbitrary shapes and sizes for the groups of samples. In this context, the system developed in [78] is promising. It is able to generate clusters of different densities by using different statistical distributions to generate each cluster. Additionally, in contrast to the rest of the literature reviewed in this section, it also provides variability of shapes. However, it does so via linear equations and transformations (causing the final shape of the cluster to be based on specific shapes like rings, crosses or alphabet letters). Hence, the generation of differently shaped clusters looks rather synthetic and, thus, not realistic.

Therefore, there is an opportunity to fill this gap in the literature by developing a system that flexibly generates realistic datasets with varying densities, shapes and sizes. This will be further explored in Section 5.1.

2.2.2. External evaluation metrics

External evaluation for clustering algorithms is possible by comparing two clustering results: (1) the predicted groups for each samples and (2) the original group to which the sample belong. In this section, we review some of the most important external evaluation metrics for clustering. For this research, we focus on single group clustering (i.e., each sample has unique membership to a single group), although fuzzy versions of some of the metrics have been proposed [79].

One of the most popular evaluation metrics is the Rand index [80], which compares the similarity between the predicted clustering and the original one by considering, for all samples, when they are grouped into the same cluster and when they are not. The ratio between the number of agreeing pairs and the total number of pairs is the result of the index. This simple metric is often used in the context of external clustering evaluation [81]. However, it suffers from two main limitations: (1) it tends to its maximum value when the number of predicted clusters increases, and (2) the value of the Rand index when comparing different random clustering results is not stable. To tackle these, a modification of the Rand Index, which adjusts its behaviour for chance has been proposed: the Adjusted Rand Index [82]. The latter is mentioned as the index of choice in [83] when comparing two partitions with a different amount of clusters.

Another popular index proposed to tackle Rand's limitations is the Fowlkes-Mallows index [84], which, when comparing predicted labels to ground-truth labels, can be computed as the geometric mean of the precision and the recall. Although this index tends to perform well with noise presence, it has the drawback of returning unusually high values when the number of predicted clusters is low [85].

Also worth mentioning is the V-Measure score [86], initially conceived for natural language processing, but potentially applicable to any given clustering problem. This metric proposes two complementary concepts: homogeneity and completeness. Homogeneity measures the extent to which predicted clusters are formed by samples of a single cate-

gory (the same ground-truth cluster), whereas completeness measures how many samples from a ground-truth cluster are elements of the same predicted cluster. The V-Measure is then computed as the harmonic mean between homogeneity and completeness, and can be weighted to favour one or the other. One of the main advantages of this method is that it provides a comprehensive interpretation of the results by combining two separate scores that represent specific desirable properties of an ideal clustering result.

To conclude, we also consider a general metric in probability: Mutual Information (MI). The term *mutual information* was coined in [87], but it had already been defined by Shannon in [88], and it is one of the most popular methods to assess the dependence between two random variables. In the context of Information Theory, for which it was originally proposed, MI can be understood as a measure of the amount of information we have about one variable by observing the other. Nevertheless, MI has also been used in the context of clustering evaluation for different tasks [89], [90]. Furthermore, a modification of mutual information adjusted for chance, similar to that of the Rand index, has been proposed for the specific context of clustering evaluation [91].

The appropriateness in the choice of the metric highly depends on the task at hand. A metric useful to evaluate clustering partitions in a certain context does not have to be useful in another. For this reason, in Chapter 6 of this document, we study the suitability of these metrics for our particular scenario of application.

2.2.3. Internal evaluation metrics

External metrics are often preferable to internal metrics, since they are based on ground-truth information about the data. However, their use is not always an option, as discussed in Section 2.2.1. Therefore, in this section, we review some of the most relevant alternatives in the current literature to evaluate clustering results internally. Analogously to what was mentioned in the previous section, internal metrics for fuzzy clustering have also been proposed [92], but this research is focused on non fuzzy algorithms, so they lie out of the scope of this review.

There are many internal metrics that can be used to evaluate clustering results. These metrics usually consider a relationship between the compactness or dispersion of the individual clusters and the separation among them [93]. The differences between the metrics usually lie within how these two concepts are computed and compared.

In this context, the Dunn index [94], undoubtedly one of the most popular internal metrics, considers the relationship between the worst case of compactness (maximum intra-cluster dispersion) and separation (minimum distance between two clusters). The main disadvantage of this metric is that, since it compares worst case scenarios, it is not very stable, i.e., if an algorithm generates even a single cluster with undesirable characteristics, it can have a strong impact in the result.

The Davies-Bouldin index [95], on the other hand, performs a pairwise analysis of the

clusters and computes a *quality* value ($R_{i,j}$) for every pair:

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{2.1}$$

where S_i is the dispersion of the i^{th} cluster in the collection and $M_{i,j}$ is the distance between clusters, usually calculated as the distance between cluster centroids. Then, the *quality* of each cluster is the maximum value of its pairings. The final value of the index is the average of every cluster's *quality*. Since the value of $R_{i,j}$ is desired to be small, a lower value of the Davies-Bouldin index suggests a better partition than a higher one. Since it analyses the worst case for every cluster, in contrast to the Dunn index which does it for the whole colleciton, the effect of a single rogue cluster (or very few of them) is mitigated. Nevertheless, as it happened with Dunn, Davies-Bouldin does not consider intra-cluster and inter-cluster density variations. Furthermore, it does not explicitly penalize the excessive generation of clusters, which usually favours intra-cluster dispersion.

A metric that tackles this last problem is the Calinski-Harabasz index [96]. Like Dunn, it is calculated as a ratio between inter-cluster dispersion (computed as the distance of the cluster centroid to the center of the sample space) and intra-cluster dispersion (computed as the sum of distances from every sample in the cluster to the center of the cluster). The main novelty is, as we stated, the introduction of a penalization for excessively generating clusters. The Calinski-Harabasz index (CH) can be expressed as follows:

$$CH = \frac{N_p - N_{Cl}}{N_{Cl} - 1} \cdot \frac{D_{inter}}{D_{intra}} \tag{2.2}$$

where N_{Cl} is the number of generated clusters, N_p is the total number of data samples and D_{inter} and D_{intra} are, respectively, the inter-cluster and intra-cluster dispersion.

Another alternative metric is the Silhouette score [97], which measures, for every point in the sample space, the similarity with respect to the points grouped in its same cluster (cohesion) and to the ones belonging to different clusters (separation). Cohesion (a) is computed as the mean distance to the rest of the points inside the cluster and separation (b) as the distance to the closest point that belongs to a different cluster. Then, the Silhouette coefficient of a sample is computed as:

$$s = \frac{b - a}{max(a, b)} \tag{2.3}$$

The final value of the metric is the average of all coefficients. Thus, the value is in the range $[-1, +1]$, with higher values suggesting a better partition than lower ones.

The concept of separation introduced by the Silhouette score (distance to closest point in a different cluster) makes more sense when analyzing sample spaces with clusters of different shapes and sizes. This is depicted in Fig. 2.4, where one can observe that cluster A is visibly closer to B than to C, but other metrics comparing distance between cen-

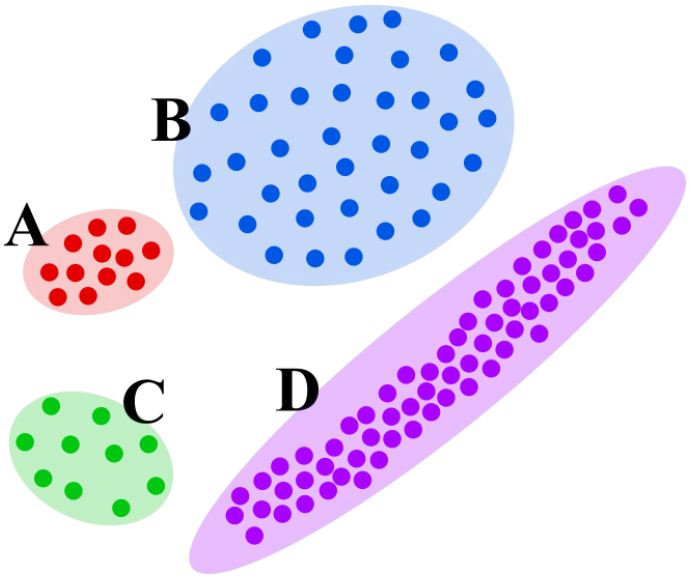


Figure 2.4: *Toy example of a clustering partition with 4 clusters with different shapes and sizes.*

troids would disagree. However, the cohesion concept does not cope well with arbitrarily shaped clusters. For instance, elongated groupings would be penalized (even if they have a significantly high density throughout the region of the cluster) with respect to globular ones. This can be appreciated with cluster D in Fig. 2.4, for which even if its clear that its densely packed throughout its extension and, therefore, fairly compact, the cohesion values for points in its extremes would cause it to provide a worse cohesion measure than cluster C, for instance, which is visibly less compact but its globular shape provides a more stable distance to the cluster center. This is not ideal when analyzing scenarios where density variations are not caused by underlying data distributions that produce globular-shaped clusters (e.g., Normal distributions).

Incidentally, as it is posed in Section 2.1, this is precisely the type of context in which density-based clustering are usually applied. Therefore, even if the discussed generic metrics can be useful to provide an idea of the performance of clustering algorithms in certain environments, it is clear that, in the context of density-based clustering, task-specific metrics should be considered.

Task-specific metrics have already been explored in the literature. The Density-Based Clustering Validation (DBCV) index [98] was designed to favour algorithms that assume arbitrary changes in density throughout the sample space or, to express it alternatively, those that do not make assumptions regarding the underlying distribution of the space. It measures the *validity* of every cluster as (once again) a relationship between intra-cluster dispersion and inter-cluster separation. Similarly to the Silhouette score, the separation between clusters is measured as the minimum distance between points that belong to different clusters. Then, the separation measure (D_C) of an individual cluster is the smallest among them (the distance to the closest neighbouring cluster). However, the main nov-

elty resides in that intra-cluster dispersion (S_C) is computed as the heaviest branch of the Minimum Spanning Tree (MST) of the cluster (i.e., the farthest distance between adjacent points). The value of a cluster's validity (V) is then similar to the Silhouette coefficient:

$$V = \frac{D_C - S_C}{\max(S_C, D_C)} \tag{2.4}$$

Again, analogously to Silhouette, the final value of the metric is the average of every cluster. However, this time it is the weighted average along the clusters, where weights are proportional to cluster cardinality. Thus, the range and desirable values of this metric are the same as the Silhouette score. The main disadvantage of this metric is that, by considering the heaviest branch of the MST as a measure of dispersion, it inevitably forces the metric to assume all clusters in a good partition will have similar densities. In other words, if we take a look once again at Fig. 2.4, clusters like B and D (with visibly different densities) being identified simultaneously would be heavily penalized by DBCV.

Another denisty oriented internal metric is the Separation-Variance (SV) index [99]. This index considers intra-cluster variance (V_k) and inter-cluster separation (S_k). Separation is computed similarly to DBCV and Silhouette, although authors do not specify a concrete method to calculate distance between clusters. The variance is defined as the distance from the centroid of the cluster to the farthest point in the cluster. The final value of the metric is the computed as the ratio between the sum of the separation measures for all clusters and the sum of the clusters' variances.

The main potential weakness of this metric is that, once again, it penalizes coexistence of clusters with notably different densities, since it directly sums all values of variance without taking into account the actual size or relative position of each cluster.

Even though the last two described metrics (DBCV and SV index) are task specific and could potentially be useful for our problem, we have already discussed some of their limitations and, later in this document, we will objectively show that they do not always correlate well with external metrics. Hence, they lack the capability to truthfully reflect the quality of the results in our application scenario. As a result, a novel internal metric to tackle this problem is proposed in this thesis (see Section 5.2).

2.3. End-user application: *touristic landmark discovery*

We have seen that one of the main applications of density-based clustering is automatic landmark discovery (Section 2.1.3) and novel methods based on density-based clustering are presented in this document to tackle this problem (discussed in Chapter 3). However, this task has also been approached through other techniques and, therefore, the decision to pursuit the research line that led to this document (density-based clustering) needs proper justification.

The landmark discovery task was introduced in Section 1.1 as the determination of

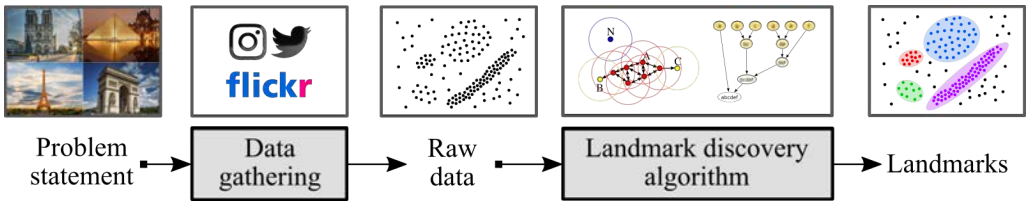


Figure 2.5: *Pipeline representing the process of landmark discovery.*

relevant elements within a distribution by following a specific criterion. For this research, we focus on geographical landmarks, which can be directly represented by raw geographical data (e.g, GPS coordinates) but also by associated images or text, as we introduced in Section 1.1.1. The pipeline of the landmark discovery process is illustrated in Fig. 2.5.

The first step of the displayed pipeline is the problem statement, i.e., the type of landmarks to be discovered (touristic attractions, trending restaurants, events, etc.). It is important to clarify this as it will heavily impact the source of the data to analyze. As we will further explore in Chapter 3, the main targets of this research are touristic landmarks. After this is decided, we can acquire the corresponding data to fit our needs (see Fig. 2.5). Section 2.3.1 provides an insight on the available data sources for our application context.

Once the data has been gathered, the next step in the pipeline is to apply a relevant algorithm that is able to locate the main relevant landmarks. In the context of touristic landmark discovery, this would imply the identification of the regions containing the most relevant monuments, parks, squares, streets or buildings of interest, etc. The detection of landmarks by clustering geographical data is a recurrent research topic [19][20] which is followed in this thesis. Section 2.3.2 further explores which clustering algorithms are better suited to the task at hand.

2.3.1. Source analysis

The most straightforward method to find geographical landmarks is to analyze geo-located content. In this sense, a relevant source to provide this type of content is Twitter [3], which is often used as base for event exploration [21][100]. It has also been used to determine specific locations, like pinpointing user's homes [101]. However, the research in [22] regarding the 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 [4] and Instagram [5], have similar problems, since users also tend to publish their content related to places with no touristic interest [102][103]. Additionally, these last two have very restricted API usage. Alternatively, Flickr [6] is a much more appropriate source for the detection of touristic

[3]Twitter: https://twitter.com/
[4]Foursquare: https://foursquare.com/
[5]Instagram: https://www.instagram.com
[6]Flickr: https://www.flickr.com/

landmarks. 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 [104][27].

In this thesis, we consider the geographical information and the additional metadata of the photos (particularly, textual descriptive tags) as the basis for landmark discovery through density-based clustering. Nevertheless, alternative approaches can be found in the literature, and these will be further explored in the following section.

2.3.2. Relevant clustering methods for landmark discovery

Several clustering algorithms have been considered in the literature for the purpose of landmark discovery, some of them being better tailored to the task.

A classic partitioning algorithm such as Mean-Shift [105] has been previously employed in [19] 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 in Chapter 4, Mean-Shift fails to adapt to more realistic datasets. Another classic algorithm, *k-means* [106], performs reasonably well working with spatial data when the number of output clusters is known [107]. The same happens with the classic approach of Agglomerative Hierarchical clustering [108], for which selecting the optimal number of final clusters often yields the clustering results that best satisfy the relationship between intra-cluster and inter-cluster dispersion. Unfortunately, a method for determining this parameter *a priori* is not trivial in the majority of scenarios. This is particularly relevant for the case of landmark discovery, 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 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 [27]. However, in the application addressed in this thesis 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 [109], landmark discovery is attempted using a hierarchical algorithm based on binary trees divided with spectral clustering, but non-automatable assumptions must be made in order to achieve and assess landmark discovery (e.g., dismissing all resulting clusters that do not fall near a labeled landmark).

On the other hand, density-based algorithms have been applied to this particular task.

Methods like the seminal *DBSCAN* algorithm [5], which, as we discussed in Section 2.1, 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 discovery [20] and, additionally, several extensions of the algorithm have been proposed to deal with particular aspects of the problem: P-DBSCAN [26]. This algorithm, aside from including the Adaptive Density modification discussed in Section 2.1.2, 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, thus limiting the influence of users uploading many pictures into the same location. Although these extensions adapt well to the problem of landmark discovery, we will prove in this dissertation that they are not robust enough to perform landmark discovery on diverse geographical areas. Similarly, DVDBSCAN [59] proposes an adaptive criterion to expand the clusters (see Section 2.1.2). However, the experiments presented in the original paper [59] are not very conclusive, since they provide a single synthetic database to draw conclusions.

Another fairly recent algorithm is *HDBSCAN* [29], 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, *HDBSCAN* can be applied over the GPS coordinates of photos to discover landmarks of interest. However, GPS coordinates are often 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 (e.g., descriptive textual tags), along with a crucial assumption of density variation (explained in the next chapter), 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 dissertation are better suited for the problem at hand, outperforming all the methods currently being used for landmark discovery.

CHAPTER 3

A NOVEL APPROACH TO LANDMARK DISCOVERY: *KDBSCAN & VDBSCAN*

This chapter presents a novel approach to a well-known clustering problem: automatic landmark discovery. As Section 2.3 mentions, landmark discovery tasks are often tackled through the use of clustering algorithms and, specifically, density-based clustering algorithms. The research conducted for this Thesis yielded a novel clustering alternative that can be used to address this problem. Particularly, two new density-based clustering algorithms (*KDBSCAN* and *VDBSCAN*) are proposed.

At this point, it is important to mention that landmark discovery is a particular problem with limited (albeit extensive) applicability. Furthermore, the nature of the exploited databases (i.e., two-dimensional geographical data) further directs the applicability of the algorithms, even if, as it is explained later on in Section 3.2.2, we can partially extend the dimensionality of geographical data with textual features. Nevertheless, the proposed algorithms are designed so that their applicability transcends the characteristics of this type of data. This particular aspect will be explored in Chapter 5, where different clustering algorithms are tested against various databases of variable, changing characteristics, with the goal of analyzing the behaviour of these algorithms with respect to changes in the data's nature.

Therefore, it is imperative that the definition of the clustering algorithms is independent from the final application task. In consequence, both algorithms are first described from a generic mathematical perspective (Section 3.1) and later applied to the task at hand: automatic landmark discovery (Section 3.2). In that section, the complete discovery system is described, consisting on three modules: (1) the data gathering module, (2) the data pre-processing module, and (3) the landmark detection module. This last module makes use of the proposed algorithms (*KDBSCAN* and *VDBSCAN*) to tackle the final discovery task.

Each algorithm has been design to adapt to a particular scenario of interest, as we will further explain in Section 3.2.3. Briefly stated, *KDBSCAN* was designed to identify places of interest in scarcely distributed regions (see Fig. 3.1.a) while *VDBSCAN* was designed to identify individual landmarks within those places of interest (see Fig. 3.1.b).

Figure 3.1: *Representation of the objectives of each algorithm in the context of touristic landmark discovery: (a) KDBSCAN identifies the places of interest within a certain region, and (b) VDBSCAN identifies the touristic landmarks inside each region.*

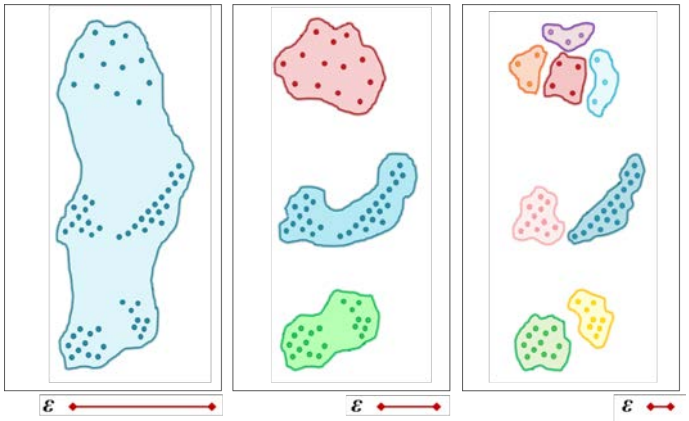


Figure 3.2: *Results produced by DBSCAN for different values of ε (decreasing from left to right) applied to an artificial set of points.*

3.1. Proposed density-based clustering algorithms

In this section, two novel algorithms for density-based clustering are presented: *KDBSCAN* and *VDBSCAN*. 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. 3.2).

The first algorithm, *KDBSCAN*, estimates the underlying density distribution of a given sample space in order to identify relevant peaks to which all the points will be assigned based on density-reachability. On the other hand, *VDBSCAN* 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.

3.1.1. *KDBSCAN*: 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, *KDBSCAN* attempts to determine the number of resulting clusters prior to the clustering step. This approach, as it was discussed in Section 2.1.2, is not common for density-based clustering. However, some data distributions can benefit from it, like, for instance, distributions with large areas where density is extremely low but have high inter-cluster density variations. Furthermore, even for the cases where these variations are not severe and other parametric approaches like k-means could be applied, the problem of determining the number of output clusters remains to exist. To tackle this, *KDBSCAN* first estimates the underlying distribution in the sample space and then it selects its peaks.

The first step of *KDBSCAN* is to obtain a Kernel Density Estimation (KDE) with Gaussian kernels [110] that estimates the underlying distribution $f_{KDE}(\boldsymbol{p})$ for the input set of samples $P = \{\boldsymbol{p_1}, \boldsymbol{p_2}, ..., \boldsymbol{p_N}\}$. At this point, the first of two algorithm-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. 3.4).

The next step consists in obtaining the number of output clusters. To this end, the M^* local maxima of the distribution are identified and stored in vector $P^* = \{\boldsymbol{p}^*_{\boldsymbol{1}}, \boldsymbol{p}^*_{\boldsymbol{2}}, ..., \boldsymbol{p}^*_{\boldsymbol{M^*}}\}$. These local maxima are analyzed to check their validity as a cluster centroid. This is done through a measure of *topographic prominence* [111]. 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 an equal or higher point of the distribution. This concept is illustrated in Fig. 3.3. As it can be observed, even though the displayed $\boldsymbol{p_1}$ is lower than $\boldsymbol{p_1}$, its topographic prominence is greater, since the difference with its *key col* is larger. It is worth noting that for a unique absolute maximum in a distribution ($\boldsymbol{p_4}$) there are no equal or higher points, so the *key col* is considered to be the minimum value of the distribution. The feature of topographic prominence 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 $\boldsymbol{p}^*_i$, the normalized topographic prominence $NTP(\boldsymbol{p}^*_i)$ is:

$$NTP(\boldsymbol{p}^*_i) = \frac{TP(\boldsymbol{p}^*_i)}{f_{KDE}(\boldsymbol{p}^*_i)} \tag{3.1}$$

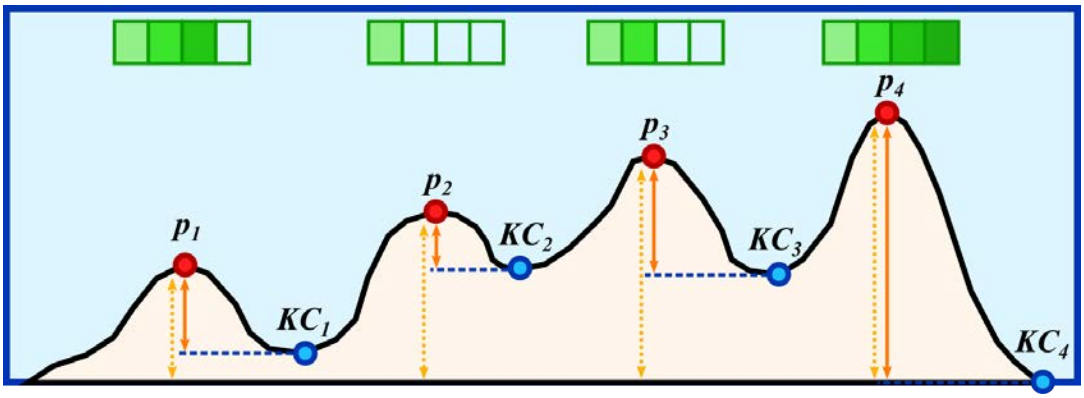


Figure 3.3: *Example illustrating the concept of topographic prominence. In red, the local maxima of the distribution ($\boldsymbol{p_i}$); in blue, their respective key cols (KC_i); in orange, the topographic prominence; in yellow (dotted) the value of the maxima in the distribution ($f_{KDE}(\boldsymbol{p_i^*})$); and in green, over each maxima, a representation of its normalized topographic prominence.*

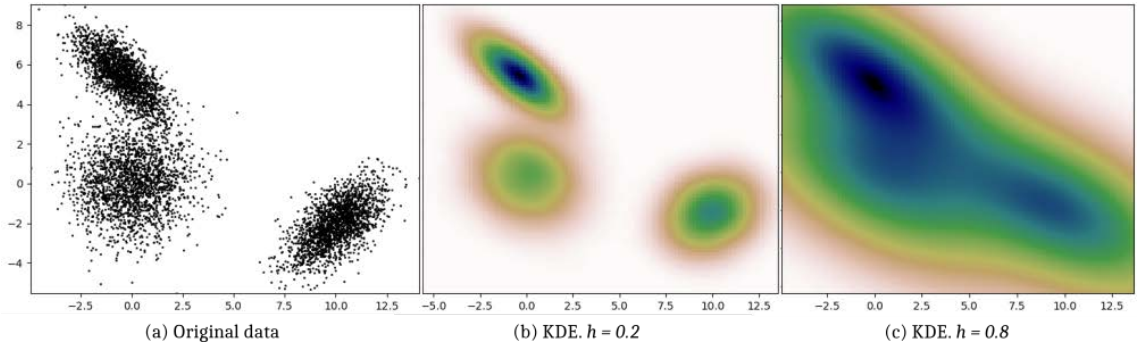


Figure 3.4: *Kernel Density Estimation for an artificial set of points from 3 Normal distributions (a) with two bandwidth factors $h = 0.2$ (b) and $h = 0.8$ (c).*

This concept is also illustrated in Fig. 3.3, where we can see that even though $TP(\boldsymbol{p_3}) > TP(\boldsymbol{p_1})$, the value of the distribution of $\boldsymbol{p_1}$ is significantly lower, so its normalized topographic prominence (the green bar in Fig. 3.3) is higher.

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 $NTP(\boldsymbol{p_i^*})$ is compared to a lower bound t, and all $\boldsymbol{p_i^*} \in P^*$ that do not satisfy this threshold are discarded (for being too dependent on another maxima), yielding cluster centroid vector $Z = \{\boldsymbol{\zeta_1}, \boldsymbol{\zeta_2}, ..., \boldsymbol{\zeta_M}\}$. This threshold t is the second of the two algorithm-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. 3.4.a, 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).

Algorithm 1 Assignment phase of algorithm *KDBSCAN*.

Input: A set of points P, a sub-set of M cluster centroids Z, initial parameters $(\varepsilon_0, MinPts)$, neighborhood radius update factor (η_ε).

Output: Set of clusters C

1: **function** ASSIGN(P, Z, ε_0, $MinPts$, η_ε)
2: **for** i **from** 0 **to** M-1 **do**
3: C_i = create-empty-cluster(i)
4: C_i = combine-unique(C_i,$\boldsymbol{\zeta_i}$)
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, ε, $MinPts$)
11: $i = i + 1$
12: **if** $i \geq M$ **then**
13: $i = 0$
14: $k = k + 1$
15: ε=update(ε,η_ε,k) // *see Eq. (3.2)*
16: **end if**
17: **end while**
18: **if** points unclassified > 0 **then**
19: C = assign(P, Z, ε_0, $MinPts - 1$, η_ε)
20: **end if**
21: **return** C
22: **end function**

Once the set of cluster centroids Z are identified, each data sample $\boldsymbol{p_i} \in P$ 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 $\boldsymbol{\zeta_i} \in Z$ for which the path from p to $\boldsymbol{\zeta_i}$ is a monotonously increasing function.

In order to provide a more precise and formal description of *KDBSCAN*, a pseudo-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 Z has been previously identified. At the beginning, a set of M clusters is created, each one containing one centroid $\boldsymbol{\zeta_i} \in Z$ (lines 2-4 in Alg. 1), and an initial

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, ε, MinPts)
2:      Q = C
3:      while Q is not empty do
4:          p = extract-from-queue(Q)
5:          H = obtain-neighborhood(p, P, ε)
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
```

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.

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:

$$\varepsilon^{(k)} = \varepsilon^{(k-1)} \cdot (1 + \eta_\varepsilon) \tag{3.2}$$

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 neighbors inside a radius ε to consider a core-point. In other words, for a given ε, it influences the expansion rate and the stopping condition. As the assignment phase of *KDBSCAN* aims 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 $MinPts - 1$, since a larger amount would become a core-point

Figure 3.5: *Example of a radially decreasing density distribution in the city of Madrid. As we move away from the city center (density centroid) the density of samples decreases.*

by itself. Therefore, we set the default value $MinPts = 4$ (established in [5]) 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 4.2.2.

The output of *KDBSCAN* is, therefore, the set of clusters resulting after the assignment of all samples has been completed. As it will be later shown in Section 3.2, *KDBSCAN* 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. An illustrative example of such scenario would be a geographic region containing multiple urban nuclei (which would contain several landmarks themselves) separated by long distances.

3.1.2. *VDBSCAN*: Variable-Density-Based Spatial Clustering of Applications with Noise

This algorithm is a hierarchical modification of the DBSCAN algorithm. *VDBSCAN* 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* (see Fig. 3.5).

In order to provide a better understanding of the algorithm, a pseudo-code version of *VDBSCAN* 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).

VDBSCAN 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 *KDBSCAN*, 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 *KDBSCAN*, it is known that its ideal value is as high as possible (see Section 4.2.2).

The goal of *VDBSCAN* is to assign each data point $\boldsymbol{p_i} \in P$ 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 [5] 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 $|C_i|$ (although alternative strategies could be followed, as we will later discuss in Section 3.2.3). 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 3.1.2). 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. (3.3), leading to a finer resolution analysis.

$$\varepsilon^{(k)} = \varepsilon^{(k-1)} \cdot (1 - \eta_\varepsilon) \tag{3.3}$$

Again, analogously to *KDBSCAN*, the initial parameter η_ε would ideally be as low as possible (see Section 4.2.2).

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.

Subdivision criterion

Similar to the criterion employed in [112] 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:

$$Dist(S_j, S_j^{\complement}) \lneqq MInt(S_j, S_j^{\complement}) \tag{3.4}$$

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:

$$Dist(S_j, S_j^{\complement}) = \min_{\boldsymbol{p} \in S_j, \boldsymbol{q} \in S_j^{\complement}} d_p(\boldsymbol{p}, \boldsymbol{q}) \tag{3.5}$$

where $d_p(\boldsymbol{p}, \boldsymbol{q})$ is the distance between points $\boldsymbol{p}$ and $\boldsymbol{q}$.

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

$$MInt(S_j, S_j^{\complement}) = min(int(S_j) + \tau(S_j), int(S_j^{\complement}) + \tau(S_j^{\complement})) \tag{3.6}$$

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.

$$int(C) = \max_{\boldsymbol{p}_i \in C} \frac{1}{K} \sum_{k=1}^{K} d_p(\boldsymbol{p}_i, \boldsymbol{p}_k^{(i)}) \tag{3.7}$$

where $p_k^{(i)}$ is the k-nearest neighbor of the point $\boldsymbol{p_i}$. 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:

$$\tau(C) = \frac{\kappa \cdot I(C)}{r_C} \tag{3.8}$$

where κ is the parameter that controls regularization; r_C is the ratio between the cardinality of the cluster $|C|$ and the cardinality of the sample space $|P|$ (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:

$$I(C) = d_p(m_C, m_{C^{\complement}}) \tag{3.9}$$

In other words, 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 *VDBSCAN* 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 *VDBSCAN* effective against distributions with a radially-decreasing density.

3.2. Practical application of *K+VDBSCAN*: *touristic landmark discovery*

In this section, we propose touristic landmark discovery as a direct application for the novel algorithms explained in Section 3.1. Our proposal for landmark discovery is enclosed in a large-scale research project, and will become a processing block of a system devoted to automatically generate travel guides. Once the landmarks are identified on a given area, multimedia content (text, images, video) is 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 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. 3.6: (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 3.1 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.

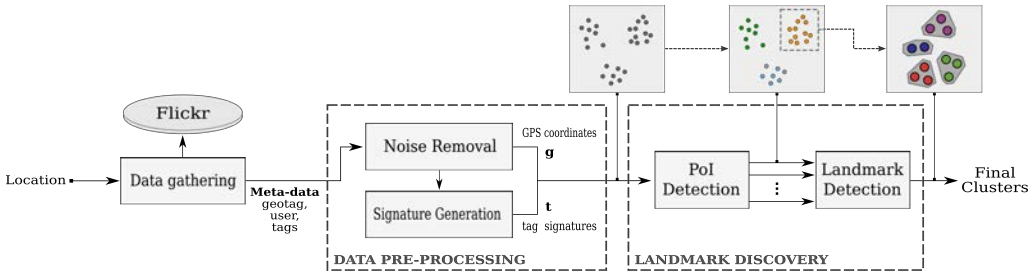


Figure 3.6: *Pipeline of the complete system. It is divided in three main blocks: data gathering, which accesses the online platform to conform the dataset (Section 3.2.1); data pre-processing, which removes inherent noise (Section 3.2.2) and generates the bag-of-words model signatures (Section 3.2.2); and landmark discovery, which performs* PoI *and landmark discovery through hierarchical density-based clustering (Sections 3.2.3 and 3.2.3, respectively).*

3.2.1. Data gathering

Given an area of interest, we aim to automatically collect a dataset of user-generated contents, that will be then 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, associated to the area, which contains the images' geo-location and associated text (tags, descriptions, titles, etc.). As it was discussed in Section 2.2.1, we found the ideal platform for our purposes is Flickr. The Flickr API [7] 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 dissertation, 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.

3.2.2. 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.

[7] Flickr: https://www.flickr.com/services/api/

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 $P = \{\boldsymbol{p_1}, \boldsymbol{p_2}, ..., \boldsymbol{p^*_N}\}$, 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.

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* [113] approach. It is worth mentioning that alternative techniques, such as the more advanced *word2vec* [114], 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 **g** and a textual (based on the descriptive tags) signature vector **t**, as illustrated in Fig. 3.6.

(a) Euskadi (b) Valencia

Figure 3.7: *Example of a region with multiple PoIs (a) and a single PoI (b). Both locations are displayed at the same scale.*

3.2.3. Landmark Discovery (*K+VDBSCAN*)

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 *KDBSCAN* described in Section 3.1.1 will be employed. The second module, *Landmark Detection*, consists on a multi-scale analysis of each identified location using the algorithm *VDBSCAN* (discussed in Section 3.1.2), which will ultimately identify each independent touristic landmark. In the remainder of this document, the complex algorithm yielded by the application of these two modules will be referred to as *KDBSCAN*.

VDBSCAN 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 *KDBSCAN*.

Fig. 3.7 displays an example of a region that is clearly a single conurbation (Valencia) and one that contains several isolated villages (coastal area of Euskadi).

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 3.2.2). For now, however, we ignore the tag signature and simply work with the geodesic coordinates. The *KDBSCAN* algorithm is applied to these set of coordinates yielding N clusters as the module's output, each one representing an individual *PoI*.

Landmark Detection Module

This second module aims to discover, within each detected *PoI*, its most prominent landmarks. To this end, it makes use of *VDBSCAN*. Due to the aforementioned reasons, in *VDBSCAN* we concurrently use geodesic coordinates and tag signatures to describe the data samples. Consequently, the distance between two points $d_p(\boldsymbol{p_1}, \boldsymbol{p_2})$, i.e., the distance used in Eqs. (3.5), (3.7) and (3.9), is computed as a weighted linear combination of two independent distances (both normalized between 0 and 1), one regarding each considered feature:

$$d_p(\boldsymbol{p_1}, \boldsymbol{p_2}) = \gamma \cdot \hat{d}_g(\boldsymbol{p_1}, \boldsymbol{p_2}) + (1 - \gamma) \cdot d_t(\boldsymbol{p_1}, \boldsymbol{p_2}) \tag{3.10}$$

where $\hat{d}_g(\boldsymbol{p_1}, \boldsymbol{p_2})$ stands for a normalized version of the geodesic distance; $d_t(\boldsymbol{p_1}, \boldsymbol{p_2})$ 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:

$$\hat{d}_g(\boldsymbol{p_1}, \boldsymbol{p_2}) = min\left(\frac{d_g(\boldsymbol{p_1}, \boldsymbol{p_2})}{f_d},\ 1\right) \tag{3.11}$$

where $\hat{d}_g(\boldsymbol{p_1}, \boldsymbol{p_2})$ is the geodesic distance separating $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$, 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:

$$d_t(\boldsymbol{p_1}, \boldsymbol{p_2}) = 1 - \frac{\mathbf{t}_1^T \mathbf{t}_2}{\|\mathbf{t}_1\|_2 \|\mathbf{t}_2\|_2} \tag{3.12}$$

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

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 a [0,1] interval, are in general small, specially in the last iterations. In those cases, the textual distance becomes relevant.

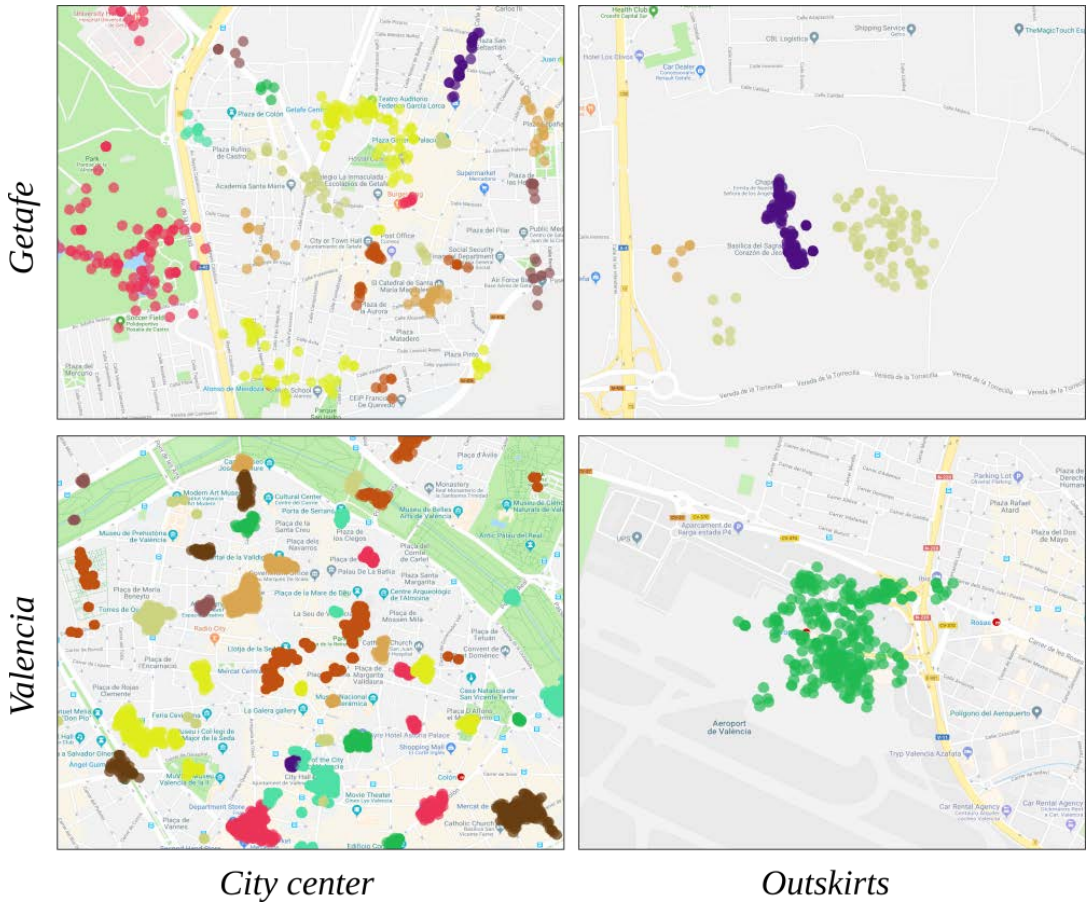


Figure 3.8: *VDBSCAN results in urban regions. At the bottom, Valencia; at the top, Getafe. The left side represents centric zones, while the right hand side represents places in the outskirts. Data belonging to different clusters is represented by different colors.*

Another aspect of this module that is worth mentioning is that the relevance concept used to sort the sub-clusters in *VDBSCAN* (line 11 in Alg. 3; Section 3.1.2) 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. 3.8 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 4. A series of experiments were conducted to reflect the performance of the proposed algorithms regarding simple landmark discovery (*VDBSCAN*) and landmark discovery with prior *PoI* detection (*K+VDBSCAN*).

Algorithm 3 *VDBSCAN* 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).

function V-DBSCAN($P, \varepsilon_0, \eta_\varepsilon$)
  C_0 = create-empty-cluster(0)
  C_0 = combine-unique(P,C_0)
  $\varepsilon = \varepsilon_0$
  C = create-set{C_0}
  unchanged = 0
  while *unchanged* < L **do**
    $C_{old} = C$
    for i **from** 0 **to** $|C| - 1$ **do**
      S = dbscan(C_i, ε)
      S = sort-by-relevance(S)
      for j **from** 0 **to** $|S| - 1$ **do**
        $S_j^{\complement}$ = remove-points-from-cluster(S_j, C_i)
        if $Dist(S_j, S_j^{\complement}) > MInt(S_j, S_j^{\complement})$ **then**
          C = add-cluster-to-set(S_j, C)
          S = remove-cluster-from-set(S_j, S)
        end if
      end for
    end for
    if $C_{old} == C$ **then**
      unchanged = *unchanged* + 1
    else
      unchanged = 0
    end if
    ε = update(ε,η_ε) // *see Eq. (3.3)*
  end while
  return C
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, ε, 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, ε, 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
```

CHAPTER 4

EXPERIMENTS ON *KDBSCAN* & *VDBSCAN* FOR LANDMARK DISCOVERY

In this chapter, we explain the experiments carried out to assess the performance of *KDBSCAN* and *VDBSCAN* in the discussed application context: landmark discovery. First, in Section 4.1, the evaluation methods employed are described. After that, in Section 4.2, the datasets against which the algorithms will be tested are detailed, along with the experimental setup. To conclude, the results of the experiments are broken down and analyzed in Section 4.3.

4.1. Evaluation metrics

Clustering techniques are usually unsupervised 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 manually identified main landmarks of the 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 the automatically generated set of clusters C (groups of data samples), and a ground-truth vector GT with the GPS locations of the real landmarks. 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 $TH_d = 50m$.

However, we need to take into account the fact that a cluster could be aligned with several GT locations, and vice versa. Potential multiple alignments are processed as follows: in the case that a GT location aligns with multiple clusters, we prioritize the alignment of the GT location with the cluster that, among those that are not yet assigned to a GT location, has the highest cardinality. Alternatively, if a cluster can be aligned with several GT locations, we prioritize the location that, among those that are not yet assigned to a cluster, is closest to the cluster centroid. In other words, the alignment process is started with the cluster with the highest cardinality, and the collection is traversed in order until all clusters are processed or all the GT locations have been assigned.

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

Table 4.1: *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).*

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 [115].

4.2. Dataset and experimental setup

4.2.1. Dataset

We made our dataset publicly available [8]. 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), with approximately 3.7K inhabitants registered in 2020's population census; two medium-sized cities, Jerez de la Frontera (Andalucia) and Getafe (Madrid), with approximately 213K and 185K inhabitants respectively; and a large city, Valencia (Comunitat Valenciana), with approximately 800K inhabitants. 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, with several coastal touristic villages). For each location, the data gathering module (see Section 3.2.1) 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 of the dataset are displayed in Table 4.1.

4.2.2. 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 4.3). 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,

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

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 ($\sim 10\%$).

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

Scale-space parameters Parameters ε_0, η_ε and (just for *VDBSCAN*) 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 *KDBSCAN*, values of $\varepsilon_0 = 200\ m$ and $\eta_\varepsilon = 0.1$ were set. Regarding *VDBSCAN*, it is worth mentioning that, unlike the case of *KDBSCAN*, ε is no longer a spatial magnitude, as we are using the aforementioned combined distance metric $d_p(\boldsymbol{p_1}, \boldsymbol{p_2})$. 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 *KDBSCAN*: 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 *VDBSCAN* 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 r_C (which represents the size of the cluster) takes low values due to its normalization.

4.3. 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 discovery. 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 *KDBSCAN* in those regions with more than one *PoI*.

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

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 [107] to detect road lanes from GPS data, while in [20] and [19], DBSCAN and Mean-Shift are used, respectively, to find meaningful locations based on GPS data (see Section 2.3). Other algorithms have been particularly designed to deal with variable-density sample populations (*HDBSCAN* [29], Hierarch. Agglom. [116]). Finally, some algorithms were specifically proposed to tackle the task of landmark discovery (P-DBSCAN and P-DBSCAN with adaptive density [26]).

The results obtained for the six considered locations are displayed in Table 4.2, 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 4.2.a shows the results obtained in the regions with a single *PoI*, whereas Table 4.2.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 (*KDBSCAN* and *VDBSCAN*) and using just *VDBSCAN*.

In addition, Fig. 4.1 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 (this can be appreciated in Fig. 4.1). In other words, they divide the sample space into Voronoi-cells, rather than detecting relevant independent locations within an enclosed space. This limitation comes from the fact that the evaluation technique was designed for information retrieval and not for cluster alignment.

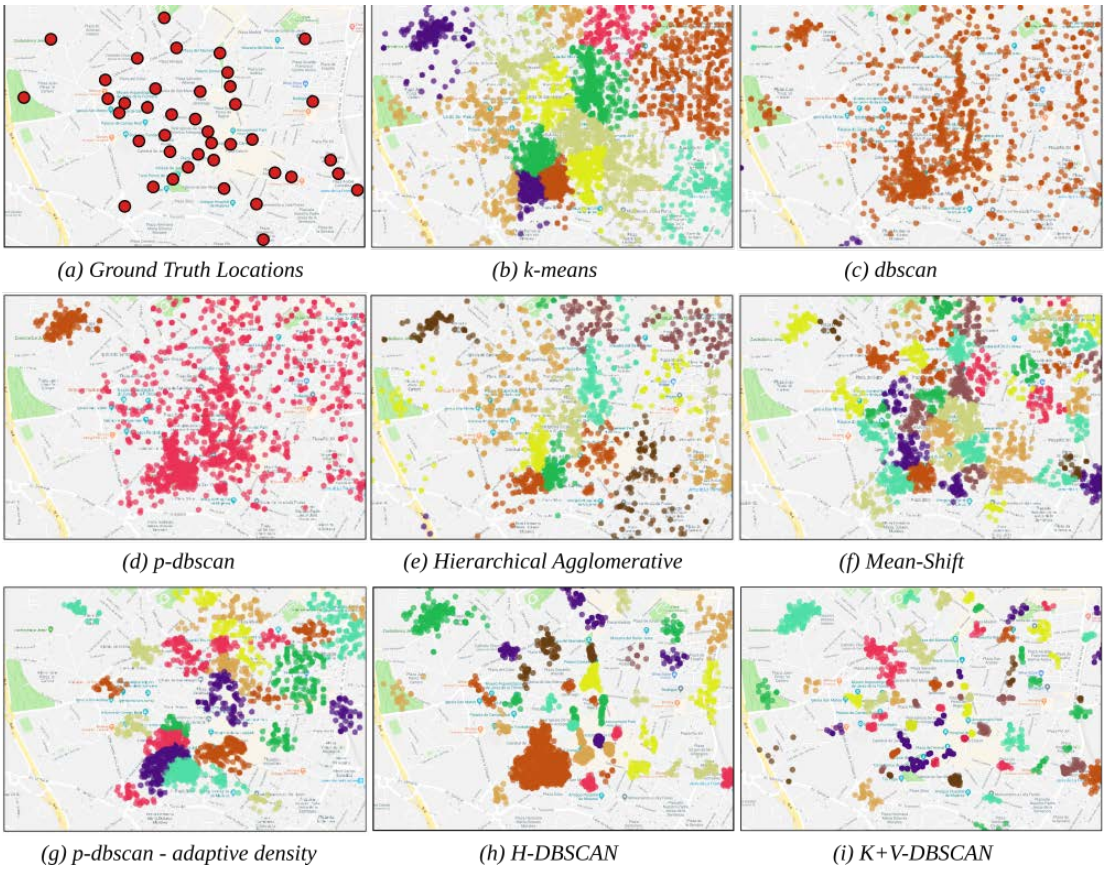


Figure 4.1: *Visual comparison between clustering algorithms in a central area of Jerez de la Frontera. Note that, for each cluster, a maximum of 1000 points are displayed.*

In contrast, Hierarchical Agglomerative clustering and *HDBSCAN* take density variations into consideration. The former is particularly tricky to adjust, as the presented 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. *HDBSCAN*, on the other hand, produces a set of clearly defined, heavily discriminated clusters (see Fig. 4.1.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 discovery: P-DBSCAN and our proposal. Although P-DBSCAN takes into account additional information other than the spatial coordinates, i.e., the user who posted the picture (see Section 2.3), it does not consider variations of density, so its performance is generally poor. Indeed, its Adaptive Density 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

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
HDBSCAN (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
VDBSCAN	0.53	**0.73**	**0.54**	**0.61**	**0.60**
K+VDBSCAN	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
HDBSCAN (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
VDBSCAN	0.67	0.59	0.63
K+VDBSCAN	**0.75**	**0.77**	**0.76**

(b) *Multi-PoI regions.*

Table 4.2: *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+VDBSCAN) and without (VDBSCAN) the* PoI *Detection Module are displayed.*

without the *PoI* Detection Module. We can clearly see the impact that *KDBSCAN* 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 *KDBSCAN* and *VDBSCAN* (*KDBSCAN*).

The visual comparison in Fig. 4.1 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 *HDBSCAN*, 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 *VDBSCAN*, which takes advantage of the assumption made in Section 3.2.3 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. Results in Table 4.2 show large differences depending on the location, ranging from 0.23 to 0.73. 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 specialized on a 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 *KDBSCAN* 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 (*KDBSCAN* and *VDBSCAN*) 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 discovery. *VDBSCAN* 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 *KDBSCAN*, *VDBSCAN* 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 (*KDBSCAN*), 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, *VDBSCAN* can be used independently over each detected group.

Assessment for automatic generation of travel guides

A very direct application of our landmark discovery 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. 4.2 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, *KDBSCAN* provided the most robust result. In the case of Valencia, for instance, the precision did not drop until the 25^{th}

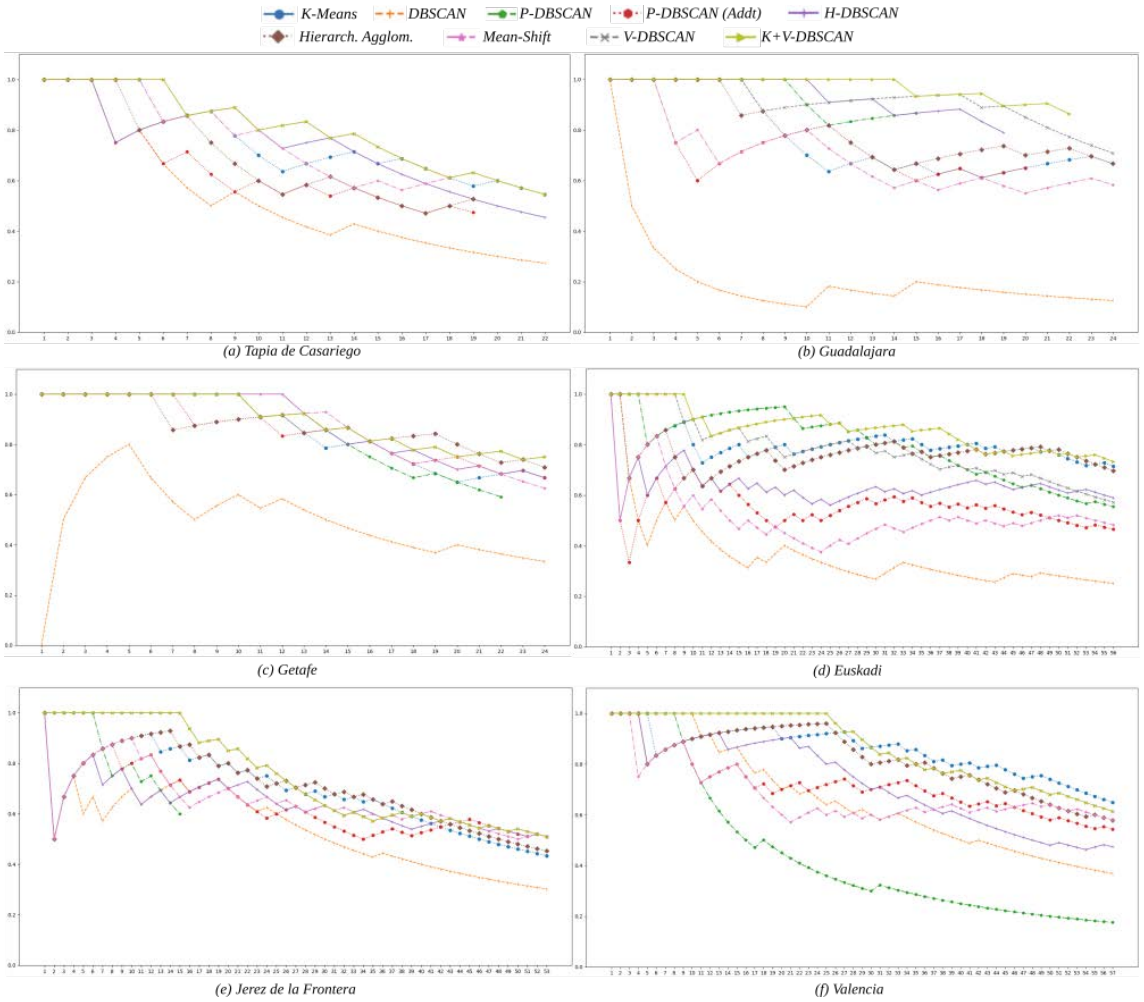


Figure 4.2: *Variation of the precision when providing up to $K = 1, 2, ..., G$ clusters as the output of the system, where G is the length of the GT location list.*

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 *HDBSCAN* 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 other 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 platform

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 4.3: *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.*

TripAdvisor [9]. Table 4.3 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 4.1). Therefore, we can infer that TripAdvisor is often insufficient to provide a significant landmark list (this is particularly remarkable for less mainstream places). In the remaining columns of Table 4.3, 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.

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

CHAPTER 5

A BENCHMARK FOR DENSITY-BASED CLUSTERING ALGORITHMS

There are several problems that affect most scenarios where the application of density-based clustering algorithms is of interest. Two of the most important ones, as it was already mentioned in Chapter 2, are: (1) the absence of labeled data, which prevents the use of external metrics to evaluate an algorithm's performance, and (2) the lack of meaningful internal metrics tailored to evaluate performance in databases where density, shape and size variability are important factors.

In this chapter, we present a solution to address these two problems. First, in Section 5.1, a system for automatic *synthetic data generation with flexible parametrization (SynFlex)* is presented. This system is designed to be flexible and versatile. Its parameters control different aspects of the data (density, shape, dimensionality, etc.) and allow the user to produce realistic, fully-labeled databases. Nevertheless, non-synthetic, labeled databases are still scarce, which narrows the applicability of external metrics mostly to synthetic data. To tackle this, Section 5.2 of this Chapter presents a novel internal metric, *FLAG*, tailored to assess the performance of clustering algorithms applied to data distributions where density and shape variability are relevant factors to be taken into account. In contrast to other popular metrics, which (as seen in Section 2.2.3) are usually based on distance-based compactness, the proposed index (*FLAG*) focuses on density fluctuation within data clusters.

To conclude this chapter, we integrate the two aforementioned contributions to build a benchmark for density-based clustering algorithms (Section 5.3). In other words, a set of both synthetic and non-synthetic databases with different parametrizations and characteristics are generated. A series of experiments based on these databases are then proposed to thoroughly assess the performance of clustering algorithms and, moreover, to analyze changes in their behaviour and performance caused by the varying characteristics of the data.

It is worth mentioning that, since there is a considerable amount of variables in this chapter, we have included a reference (Table 5.2) at the end of the chapter for the sake of comprehensiveness. Note that the notation guide included in Section 1.4 can be consulted as well.

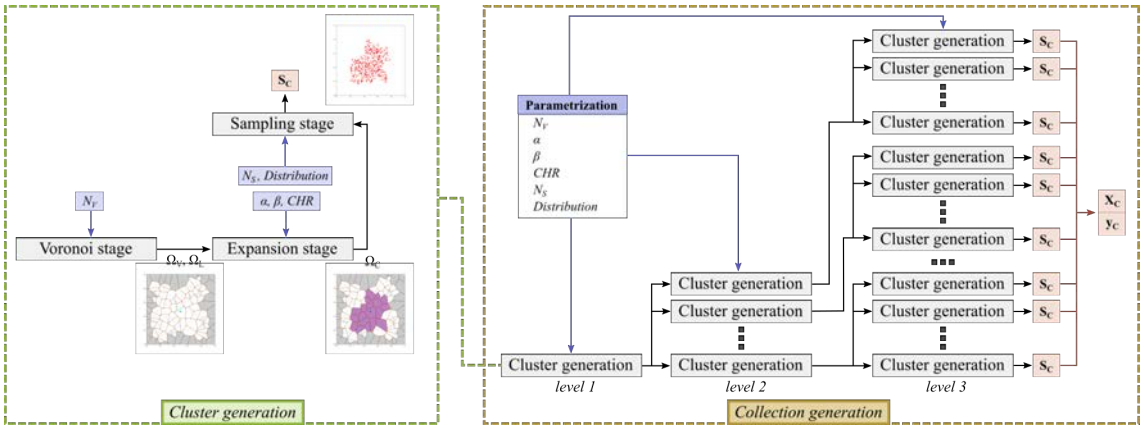


Figure 5.1: *Diagram for the SynFlex system. On the left, the cluster generation model; on the right, the collection generation model with 3 hierarchy levels.*

5.1. Synthetic data generation with flexible parametrization: *SynFlex*

As discussed in Section 2.2.1, most generic data generation algorithms either require reference datasets or focus on linear transformations to generate differently shaped clusters. However, real data is often governed by non-linear transformations and arbitrary or hard to define distributions. In this section, we introduce a synthetic data generation with flexible parametrization (*SynFlex*) to produce fully-labeled, realistic data that can be used to evaluate density-based clustering algorithms. In fact, *SynFlex* will be later used to build a benchmark to analyze the behavior and performance of certain clustering algorithms (see Section 5.3).

SynFlex follows an iterative probabilistic data generation process that can be divided into two different modules: (1) **cluster generation**, which describes the generation process and parametrization (shape, density, sample cardinality, dimensionality) of a single, isolated data cluster, and (2) **collection generation**, which defines the whole sample space through a hierarchical generation of clusters and establishes the relationships between them (cluster cardinality, hierarchy level, noise presence). Therefore, the latter makes use of the former to generate the distribution. A pipeline of the system is displayed in Fig. 5.1 and, in the subsequent sections, both modules will be explained.

5.1.1. Cluster generation

A single cluster is generated, confined to a normalized hyper-cubic sample space, with sides of unitary length. Through the modification of several parameters, the cluster can have variable distribution, shape, and density.

The generation of the cluster, as it is displayed in Fig. 5.1, can be expressed as 3 separate steps: (1) the Voronoi stage, (2) the expansion stage, and (3) the sampling stage. In the first stage, the hyper-cube is divided in Voronoi cells that can potentially become

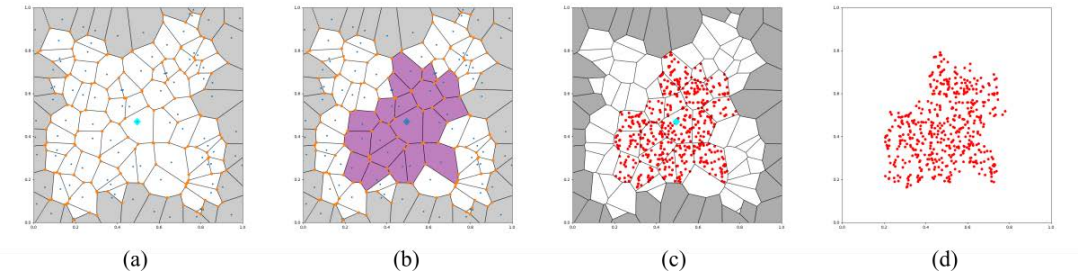


Figure 5.2: *A 2D example of the stages representing the cluster generation process: (a) Voronoi cells in hyper-cubic unitary space, (b) cells that conform the shape of the cluster after expansion, (c) samples confined by cells selected during cluster expansion, and (d) final cluster samples represented in isolation.*

part of the cluster (Fig. 5.2.a). In the second, certain cells are selected to conform the shape of the cluster (Fig. 5.2.b). In the third and final step, samples are drawn from a statistical distribution and confined to the shape established in step 2 (Fig. 5.2.c). The result of this process is a cluster of flexible cardinality, shape and density (Fig. 5.2.d) .

Voronoi Stage

The first step of this stage is the definition of a hyper-cubic unitary space. This hyper-cube is then divided into N_V cells by randomly sampling centroids from a uniform distribution throughout the space and establishing a Voronoi diagram based on those centroids (Fig. 5.2.a). The number of cells into which the hyper-cube is divided can be specified, but one has to take into account that when data dimensionality increases, more cells are needed to maintain the same expressive capabilities. Therefore, for our approach, a default value of this parameter, which depends only on data dimensionality, is used:

$$N_V = 100 \cdot 2^{D-1} \tag{5.1}$$

where D is the number of dimensions of the desired cluster.

Hence, the output of this stage is a set of N_V D-dimensional vectors $V = [\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_{N_C}}]$, each representing the centroid coordinates of a Voronoi cell. This can be represented as a Voronoi diagram like the one depicted in Fig. 5.3, and each cell can be easily referred to using the index $j : 1 < j < N_V$, which stands for the position of the cell j in the whole set of generated Voronoi cells. From this point on, let us refer to the set of all cell indices as Ω_V.

Note that the set of cells that collide with the limits of the hyper-cube (shaded cells in Fig. 5.3) are not considered as viable candidates to conform the shape of the cluster, since the space they contain surpasses the limits of the hyper-cube and, furthermore, some can have infinite extension. The indices of these limit cells constitute a fixed subset of Ω_V,

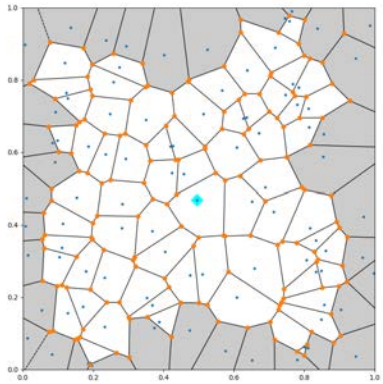


Figure 5.3: *Representation of Voronoi stage: Voronoi cells in hyper-cubic unitary space. Shaded cells are not considered as potentially eligible to conform the final shape of the cluster.*

which is henceforth referred to as Ω_L.

In the following stage, the cells that are part of the final cluster shape are selected.

Expansion Stage

The expansion stage plays a fundamental role in the data generation process. Once the candidate cells have been defined, the ones that will finally shape the cluster must be selected. Conceptually, this is done iteratively by choosing a transmitter cell and expanding the cluster shape appending one adjacent Voronoi cell at every iteration. Thus, the shape of the cluster depends on the selected transmitter cell and a direction of expansion at every iteration, which will determine the next cell (receptor cell) to be added to the cluster shape. To control this expansion, both the transmitter cell and the direction of expansion are subject to change. Specifically, they are chosen using random distributions controlled by certain expansion parameters. This approach allows us to generate clusters with diverse characteristics like arbitrary shape and varying internal densities.

The expansion process is depicted in Fig. 5.4. On the left, we can appreciate the order of appendment of each cell (the later the appendment, the lighter the tone) until the final cluster shape (on the right) is reached. The details of the process of expansion are rigorously explained in the remaining of this section.

The first step of this stage is to define the variables involved in the process. As we briefly introduced in the previous paragraph, three variables define each state of the process: (1) the transmitter cell index (t_n), (2) the direction of expansion vector ($\boldsymbol{d_n}$), and (3) the receptor cell index (r_n). From the transmitter's vicinity, the receptor is chosen using the direction as a reference.

We can use this state to define a First-order Markov Model, which expresses the state

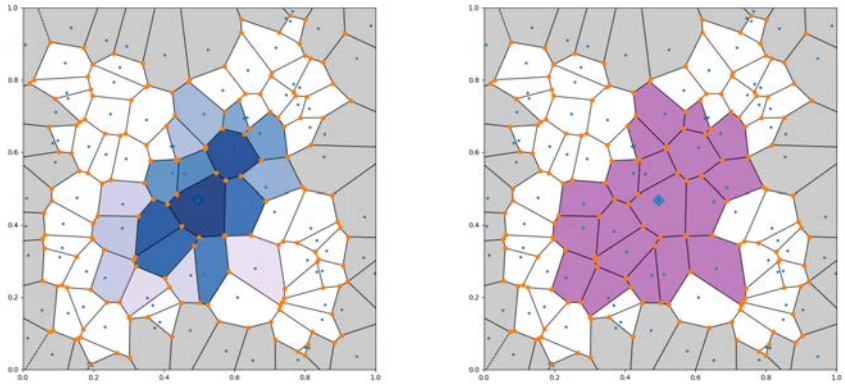


Figure 5.4: *Representation of Expansion stage: Order of the expansion (left), where dark-blue cells represent older appendments, whereas the lighter the tone, the newer the appendment; and final shape of the cluster (right).*

$\boldsymbol{x_n}$ of the cluster at each instant n. This means that the following state of the cluster at a given instant depends solely on the current state ($p(\boldsymbol{x_n}|\boldsymbol{x_{n-1}})$). Thus, the cluster expansion occurs by changing the Markov state every time instant:

$$n \in \mathbb{N} \ : \ 1 \leq n \leq N_C \tag{5.2}$$

where N_C is the maximum number of Voronoi cells that can belong to the cluster shape after the expansion, or, in other words, the stopping criterion, which will be further explained later on. Note that, since n has a finite range, the initial state (x_1) needs to be memory-less, so special conditions, which will be specified later, need to be applied.

For a cluster with D dimensions, each state $\boldsymbol{x_n}$ is defined by the three random variables defined above ($x_n = [t_n, \boldsymbol{d_n}, r_n]$). Following this approach, we can express the probability of a certain state $\boldsymbol{x_n}$ as a product of three conditionally independent distributions:

$$p(\boldsymbol{x_n}|\boldsymbol{x_{n-1}}) = p(t_n = j \mid \boldsymbol{i_{n-1}}) \cdot p(\boldsymbol{d_n} \mid \boldsymbol{d_{n-1}}) \cdot p(r_n \mid t_n, \boldsymbol{d_n}, \boldsymbol{i_{n-1}}) \tag{5.3}$$

where $\boldsymbol{i_n}$ is a vector of length N_V that records the order of adding time instant of each Voronoi cell to the final cluster shape. Particularly, $i_{n,j} = m$ if cell j was added to the cluster shape at time instant $m < n$ and $i_{n,j} = 0$ if cell j has not been added. This means that this *adding instant vector* will be initialized as a vector of zeros and will be updated with every iteration, letting us track of the cluster shape along the expansion process. It is important to remember that only one cell can be added at a given time instant, so each position of vector $\boldsymbol{i_n}$ that is greater than zero must be a unique timestamp.

In the subsequent paragraphs, we will motivate and define the distributions chosen for each element in Eq. (5.3).

The *transmitter cell* index (t_n) is the index of the Voronoi cell whose vicinity will be

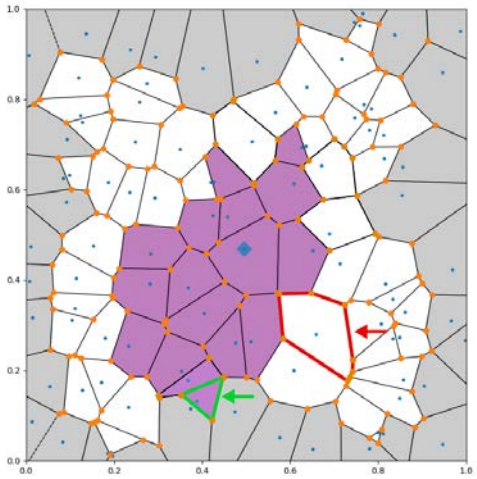


Figure 5.5: *Representation of two cells that are not valid transmitter (green and red outlines): for the green cell, even though it belongs to the cluster, its vicinity contains either limit cells or already added cells. Conversely, the red cell has a valid vicinity, but has not been added to the cluster shape.*

considered as potential candidates to be added to the cluster. We understand the vicinity of a cell as the cells that share at least D vertices with it (2 vertices, i.e., a line, for a 2D case; 3 vertices, i.e., a plane, for a 3D case). The transmitter cell needs to be selected among those that have already been added to the cluster shape. However, before the expansion begins, there are no added cells, so the first transmitter cell can be arbitrarily chosen from those that do not collide with the limits ($t_0 \notin \Omega_L$). In our approach, the default initial transmitter cell (t_1) is the one that lies closer to the hyper-cube's center, in order to maximize expansion possibilities in any direction.

After the expansion begins, at each time instant n the transmitter cell is subject to be updated. Any cell that has already been added to the cluster shape can potentially become the new transmitter cell as long as its vicinity contains at least one cell that can still be added to the cluster, i.e., a cell that does not already belong to the cluster and is not a limit cell (see Fig. 5.5).

The update of the transmitter cell is controlled by parameter α and depends on the instants in which the different cells have been added to the cluster (i.e., time instant vector $\boldsymbol{i_n}$). The range of α is $(-\infty, +\infty)$. Conceptually, choosing a lower value ($\alpha < 0$) will result in electing a transmitter cell with a lower appendment time instant (older cell), whereas choosing a higher value of this parameter ($\alpha > 0$) will conversely mean electing a cell with a higher appendment time instant (recently added cell). Particularly, the probability that the cell j is selected as the new transmitter cell at instant n can be expressed as:

$$p(t_n = j \mid \boldsymbol{i_{n-1}}) \propto \begin{cases} (i_{n-1,j})^{\alpha} & \textbf{if } i_{n-1,j} > 0,\ \Omega_{E_n} \cap \Psi_j \neq \{\} \\ 0 & \textbf{otherwise} \end{cases} \tag{5.4}$$

where Ψ_j is the set of cells in the vicinity of cell with index j and Ω_{E_n} is the set of cells that have not been added to the cluster shape ($i_{n,j} = 0 \ : \ \forall j \in \Omega_{E_n}$) and are not limit cells ($\Omega_{E_n} \cap \Omega_L = \{\}$).

It is worth noticing that the probability in Eq. (5.4) is defined with respect to $\boldsymbol{i_{n-1}}$, since the adding instant vector is updated at the end of each iteration. Furthermore, note that it is defined as a proportion, so the final values of the distribution are normalized such that $\sum_j p(t_n = j) = 1$. Thus, for a cell to be selected as a transmitter, it must have already been added to the cluster shape ($i_{n-1,j} > 0$). Once the transmitter has been fixed, two elements are remaining to define $p(\boldsymbol{x_n}|\boldsymbol{x_{n-1}})$. The receptor cell (i.e., the cell to be appended to the cluster shape at time instant n) must be selected and, to this end, the direction of expansion must be fixed first.

The *direction of expansion* vector ($\boldsymbol{d_n}$) is a D-dimensional normalized vector that represents the direction from a given point $\boldsymbol{p}$ to another $\boldsymbol{p'} = \boldsymbol{p} + \boldsymbol{d_n}$ and it determines which of the potential cells will be considered to be added to the cluster shape (receptor cell). Its initial value is arbitrary. For our approach, it is initially chosen randomly from a D-dimensional multivariate uniform distribution ($\boldsymbol{d_1} \sim \mathcal{U}_D[0, 1]$). Then, during algorithm execution, the probability of it being updated at each time instant n follows a Bernoulli distribution controlled by a parameter β with range $[0, 1]$, such that:

$$p(\boldsymbol{d_n} = \boldsymbol{\delta} \mid \boldsymbol{d_{n-1}}) = \begin{cases} \beta & \textbf{if } \boldsymbol{\delta} = \boldsymbol{\delta^*} \\ 1 - \beta & \textbf{if } \boldsymbol{\delta} = \boldsymbol{d_{n-1}} \end{cases} \tag{5.5}$$

where $\boldsymbol{\delta^*} \sim \mathcal{U}_D[0, 1]$ is a uniform D-dimensional random vector that stands for the new candidate for direction of expansion.

Therefore, this distribution works as follows: by setting high values of β we can enforce changes in the direction of expansion (choosing new directions uniformly at random). On the other hand, low values of β will result in less changes of direction. An example of the direction of expansion vector is shown in blue in Fig. 5.6. The value of the direction vector, as mentioned before, controls which cell in the vicinity of t_n is set as the receptor cell. The specific criterion is explained below.

The *receptor cell* index (r_n) is the index of the Voronoi cell selected to be included in the cluster shape at time instant n. The direction of expansion $\boldsymbol{d_n}$ marks the most likely cells to be selected among those belonging to the vicinity of t_n. Once again, some special conditions apply for the initial state. Since a cell has already been added at $n = 1$ (the first transmitter cell) there cannot be another addition until the next iteration for the definition of adding instant vector $\boldsymbol{i_n}$ to be consistent (each added cell index must have a unique timestamp). Conceptually, this can be understood as if we set $r_1 = t_1$.

After the initial state ($n > 1$), we can define the probability of a cell to be selected as

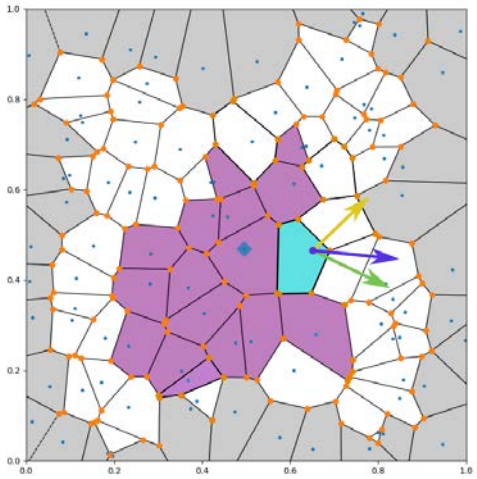


Figure 5.6: *Representation of a transmitter cell (cyan), from which the direction vector in set (blue). From the two eligible (white) cells adjacent to the transmitter, the vectors representing the candidates for the direction of expansion are shown. The bottom cell (green vector) is more likely to be selected as the receptor cell than the top cell (yellow vector), since it is more similar to the direction vector.*

the receptor cell using the following distribution:

$$p(r_n = j \mid t_n, \boldsymbol{d_n}, \boldsymbol{i_{n-1}}) \propto \begin{cases} S_{cos}(\boldsymbol{L_j} - \boldsymbol{L_{t_n}}, \boldsymbol{d_n}) & \textbf{if } i_{n-1,j} = 0\ ,\ j \notin \Omega_L \\ 0 & \textbf{otherwise} \end{cases} \tag{5.6}$$

where and $\boldsymbol{L_j}$ contains the spatial coordinates of cell j in the D-dimensional space and $S_{cos}(\boldsymbol{a}, \boldsymbol{b})$ stands for the cosine similarity of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Let us also remember that Ω_L is the fixed set of limit cells. Hence, $(\boldsymbol{L_j} - \boldsymbol{L_{t_n}})$ stands for a vector that sets a potential direction of expansion (represented in yellow and green in Fig. 5.6), and $\boldsymbol{d_n}$ is the desired direction (represented in blue and green in Fig. 5.6).

Thus, the cells in the vicinity whose centers lie in a direction that is similar to $\boldsymbol{d_{n-1}}$ are more likely to be selected as r_n, as long as they are eligible cells, i.e., they are not limit cells ($j \notin \Omega_L$) and have not been added to the cluster shape yet ($i_{n-1,j} = 0$). Note that, like t_n, the probability of r_n is defined as a proportion, so it is normalized analogously.

Thus, the expansion and, inherently, the final shape of the cluster are heavily conditioned by the selection of α and β. In Fig. 5.7 these potential shape variations are displayed.

Finally, as mentioned above, a stopping criterion needs to be set. The expansion process carries on by adding a new cell every iteration until the maximum number of iterations is reached. Let us remember that, as it is expressed in Eq. (5.2), this number is fixed by the maximum number of Voronoi cells that can conform the final cluster shape: N_C.

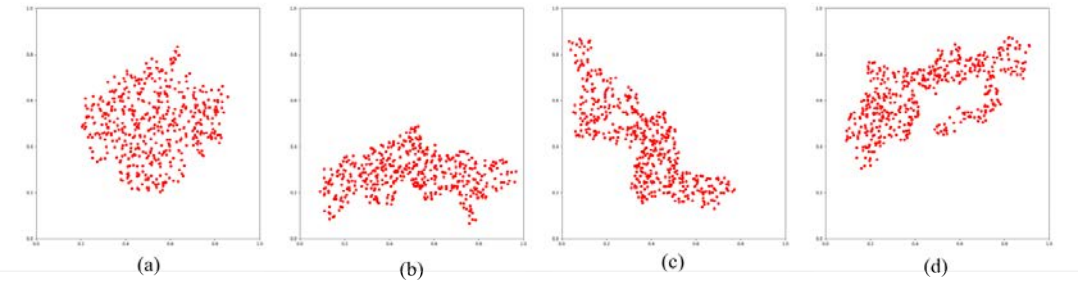


Figure 5.7: *Potential shape variations for a SynFlex cluster: (a) low (left) vs high (right) values of α for a constant $\beta = 0.5$; (b) low (left) vs high (right) values of β for a constant $\alpha = 0$.*

The value of N_C can be adjusted through the use of a parameter: the *Cluster-to-Hypercube-Ratio* (CHR):

$$N_C = (|\Omega_V| - |\Omega_L|) \cdot CHR \tag{5.7}$$

Let us remember that Ω_V and Ω_L are, respectively, the set of all Voronoi cells and the set of limit cells (cells that collide with the limits of the hyper-cube). Then, from Eq. (5.7), we can understand CHR as a ratio between the maximum number of Voronoi cells that can conform the final cluster shape (N_C) and the total number of those which can potentially become part of the cluster shape after excluding the limit cells ($|\Omega_V| - |\Omega_L|$). Additionally, note that since, necessarily, $0 < N_C < (|\Omega_V| - |\Omega_L|)$, it yields that $0 < CHR \leq 1$ must hold.

After the expansion is finished, we can obtain the set of cells that conform the definitive cluster shape by simply checking the positive indices of adding instant vector $\boldsymbol{i_n}$:

$$\Omega_C = \bigcup_{\{j\} \in \Omega_V} j \ : \ i_{n,j} > 0 \tag{5.8}$$

Therefore, the output of this stage is the set of cells Ω_C, which define the shape of the cluster. This is illustrated in the right-hand side of Fig. 5.4.

The pseudo-code in Alg. 5 is provided as a summary of the complete expansion stage, which produces the cluster shape. The following stage consists on generating the samples that will fill this shape.

Sampling Stage

This final sampling stage is devoted to draw samples from a distribution to fill the cluster shape defined by Ω_C. The process to achieve this is quite straightforward.

First, a distribution is chosen and its parameters (or the rules to select them) are set. Then, we draw samples from that distribution, with the restriction that they must lie inside

Algorithm 5 Expansion stage of cluster generation.

Input: A D-dimensional set of Voronoi cells Ω_V, of which limit cell set Ω_L is a subset, expansion parameters (α, β) and a stopping criterion parameter CHR.
Output: A D-dimensional cluster cell set Ω_C.

function EXPAND(Ω_L, α, β, CHR)
  $N_V = |\Omega_V|$
  $N_L = |\Omega_L|$
  $N_C = (N_V - N_L) \cdot CHR$
  $n = 1$
  $\boldsymbol{i_n} = \mathbf{0}$
  t_n = choose-initial-transmitter(Ω_L)
  $r_n = t_n$
  $i_{n,r_n} = n$
  $\boldsymbol{d_n}$ = choose-initial-direction(D)
  while $n \leq N_C$ **do**
    $n = n + 1$
    t_n = update-transmitter($\boldsymbol{i_{n-1}}$) // *see Eq. (5.4)*
    $\boldsymbol{d_n}$ = update-direction(D, $\boldsymbol{d_{n-1}}$) // *see Eq. (5.5)*
    r_n = select-receptor($t_n, \boldsymbol{d_n}, \boldsymbol{i_{n-1}}$) // *see Eq. (5.6)*
    $\boldsymbol{i_n} = \boldsymbol{i_{n-1}}$
    $i_{n,r_n} = n$
  end while
  Ω_C = get-cluster-shape($\boldsymbol{i_n}$)
  return Ω_C
end function

a cell belonging to the shape ($j \in \Omega_C$). The number of samples to draw is adjustable parameter (N_S) that must be set.

Therefore, the output of this stage is, like in the Voronoi stage, a total of N_S D-dimensional coordinate vectors representing the samples of the cluster $S_C = [\boldsymbol{s_1}, \boldsymbol{s_2}, ..., \boldsymbol{s_{N_S}}]$. In this thesis, a Normal distribution was chosen to draw the data samples:

$$S_C \sim \mathcal{N}(\mu = \boldsymbol{l_0}, \sigma = \sigma_C) \quad : \quad j_{s_i} \in \Omega_C \quad \forall\, (i \in \mathbb{N} \ : i < N_S) \tag{5.9}$$

where $\boldsymbol{l_0}$ is set as a vector representing the location of the initial transmitter cell, σ_C is a parameter to be adjusted, which controls the internal density variation of the cluster (see Fig. 5.8), and j_{s_i} is the index of the cell that contains each sample $\boldsymbol{s_i} \in S_C$, this is:

$$j_{s_i} = \arg \min_{h \,:\, \boldsymbol{v_h} \in V} (\|\boldsymbol{s_i} - \boldsymbol{v_h}\|) \tag{5.10}$$

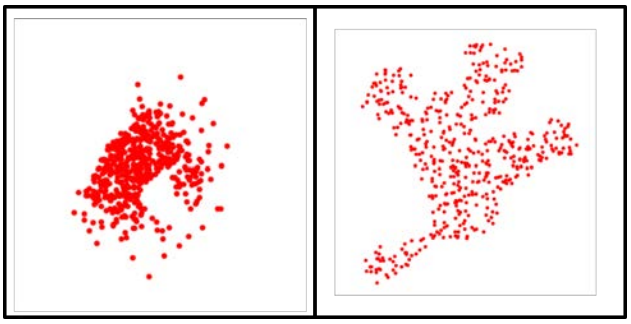


Figure 5.8: *Representation of the effect of $\sigma_C = 0.1$ (left) and $\sigma_C = 5.0$ (right) for a SynFlex cluster. Regardless of the shape, note that for a low σ_C, the cluster density decreases radially. For high enough values, the distribution tends to be uniform.*

Parameter	Brief description	Default value
D	Dimensionality	-
N_V	Nº Voronoi cells	$100 \cdot 2^{D-1}$
N_S	Nº samples in the cluster	-
CHR	Cluster-to-Hypercube-Ratio	0.25
α	Choice of t_n	-
β	Change in $\boldsymbol{d_n}$	-
Distribution	Type of distribution	$\mathcal{N}(\mu = l_o, \sigma = \sigma_C)$
l_o	Mean of Distribution	Initial transmitter cell
σ_C	Variance of Distribution	-

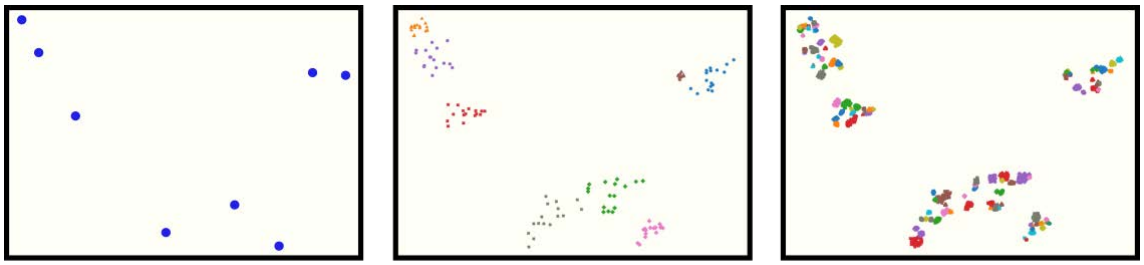
Table 5.1: *Summary of cluster generation parameters. Parameters derived from choice of distribution are represented under a horizontal dashed line. Parameters with no default value need to be set by the user.*

where let us remember that V is the Voronoi cell vector. Therefore, all drawn samples are confined to the shape defined in the previous stage.

This set of cluster samples S_C is, as mentioned above, the output of this last stage and, therefore, the output of the cluster generation module.

To recapitulate, through the three stages explained above (Voronoi, expansion, sampling), we are able to generate a cluster with arbitrary shape and variable internal density. The flexibility of the system so far includes the parameters summarized in Table 5.1. For some of these parameters, a sensible default value has been set, while others need to be adjusted by the user to determine the nature of the desired cluster.

This cluster generation system can be used to generate a realistic sample space with a flexible number of clusters with different characteristics, as it will be explained in Section 5.1.2.



(a) (b) (c)

Figure 5.9: *Representation of a hierarchically organized collection with various levels that are recursively generated: (a) 1 level, 8 elements (8 samples), (b) 2 levels, 16 elements at each of the 8 existing elements (128 samples) (c) 3 levels, 64 elements at each of the existing 128 elements (8192 samples).*

5.1.2. Collection generation

The main goal of the *SynFlex* generator is to obtain a sample space where data is susceptible to be grouped into clusters of arbitrary shape, size and density. This module of the system uses the cluster generation module to obtain such a sample space.

The simplest way to do this would be to generate a given number of clusters with random locations. However, this would limit our analysis, since all the expressiveness would lie on the cluster generation module. Therefore, it is interesting to use a richer strategy: by introducing a hierarchical approach, we are able to extend the previous module's expressiveness to the generation of a collection of clusters.

Therefore, this module is designed as a hierarchical generative process, as it can be observed in Fig. 5.1. The process is defined recursively. First, at the top level (l=1), a single cluster is generated with the desired characteristics (see Fig. 5.9.a). In the next level (l=2), a new cluster can be generated in the location of each of the samples that conform the cluster generated at the previous level (see Fig. 5.9.b). This process can be repeated as many times as needed generating as many levels of hierarchy as required (l=3, l=4, l=5, etc.). As it can be observed in Fig. 5.9.c, this data distribution has density variability throughout the sample space, with areas where clusters are packed together and areas where the distance between clusters is much larger.

Each cluster is first generated in a normalized environment and, then, a scaling factor is applied before placing it in its corresponding location (the D-dimensional coordinates of an element in the previous hierarchy level). The maximum scale that a cluster can have is limited by the nearest sample in the collection (excluding noise samples). This means that, at each new added depth level, each element of the previous level is traversed in a random order, and the maximum scale for the cluster that is placed in that position is calculated according to the nearest sample, even if that sample has been generated in the current level.

This is illustrated for a two-dimensional case in Fig. 5.10, where we can see that

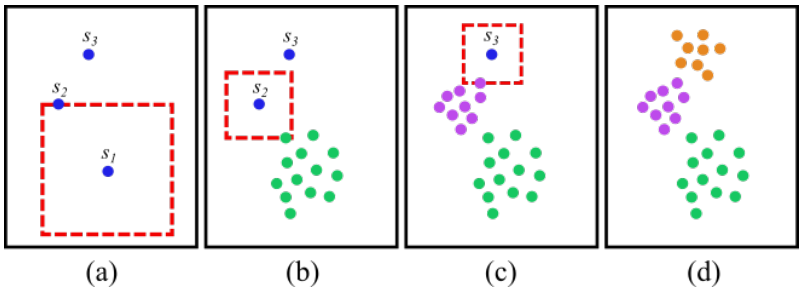


Figure 5.10: *Representation of the scaling process: (a) Sample vector (blue samples) from previous hierarchy level and max scale of cluster located in* $\boldsymbol{s_1}$ *(red dotted square), (b) new cluster samples located in* $\boldsymbol{s_1}$ *(green) and max scale of cluster located in* $\boldsymbol{s_2}$*, (c) new cluster samples located in* $\boldsymbol{s_1}$ *(green) and* $\boldsymbol{s_2}$ *(purple) and max scale of cluster located in* $\boldsymbol{s_2}$*, and (d) new cluster samples from current hierarchy level (*$\boldsymbol{s_1}$*: green,* $\boldsymbol{s_2}$*: purple, and* $\boldsymbol{s_2}$*: orange).*

sample set $S = [\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_2}]$ contains samples from a previous hierarchy level (Fig. 5.10.a). In the new level, we generate a cluster in each of these samples' locations. We can see that the maximum scale of the cluster located in sample $\boldsymbol{s_1}$ is limited by the nearest sample $\boldsymbol{s_2}$. Specifically, the maximum side length of each hyper-cube, λ_{MAX}, is twice the Chebyshev distance to the nearest sample:

$$\lambda_{MAX}(\boldsymbol{s_c}) = 2 \cdot \underset{1 \leq i \leq D}{max}(|s_{c,i} - s_{n,i}|) \tag{5.11}$$

where D is the dimensionality of the data, $\boldsymbol{s_c}$ is the current sample (from the previous hierarchy level) in whose location a cluster is being generated and $\boldsymbol{s_n}$ is the nearest relevant sample in the collection to $\boldsymbol{s_c}$, regardless of the level it belongs to. This can be observed in Fig. 5.10.b, where it is clear that $\boldsymbol{s_2}$ is closer to $\boldsymbol{s_2}$ than to $\boldsymbol{s_1}$, but $\lambda_{MAX}(\boldsymbol{s_2})$ is limited by a new sample in the cluster located in the position of $\boldsymbol{s_1}$ rather than by $\boldsymbol{s_2}$. A similar thing happens when the cluster at $\boldsymbol{s_2}$'s position is limited by a new sample rather than by $\boldsymbol{s_2}$ (Fig. 5.10.c,d).

Controlling the scale of the clusters is essential to analyze density variations, since the size of the region a cluster occupies conditions its internal density. For this reason, the final scale (λ_S) of the generated cluster, although limited by λ_{MAX}, is determined by an adjustable parameter: the scale factor $f_\lambda \in [0, 1]$. Hence:

$$\lambda_S = f_\lambda \cdot \lambda_{MAX} \tag{5.12}$$

Additionally, this module offers the possibility of adding noise samples at each generation level. Noise samples are generated with the same distribution and characteristics as the relevant samples, and their quantity is controlled by the *Element-to-Noise Ratio* (*ENR*) parameter. This ratio affects each hierarchy level independently and, therefore, it can change from one level to the next. For instance, if we have a 2-level cluster collection

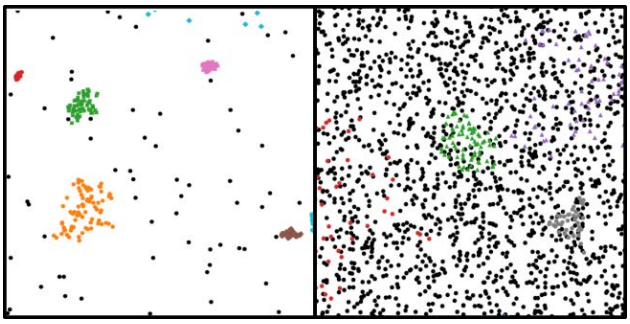


Figure 5.11: *Representation of the impact of ENR = 0.035 (left) and ENR = 0.005 (right). The lower the ENR, the stronger the noise presence (noise samples in black).*

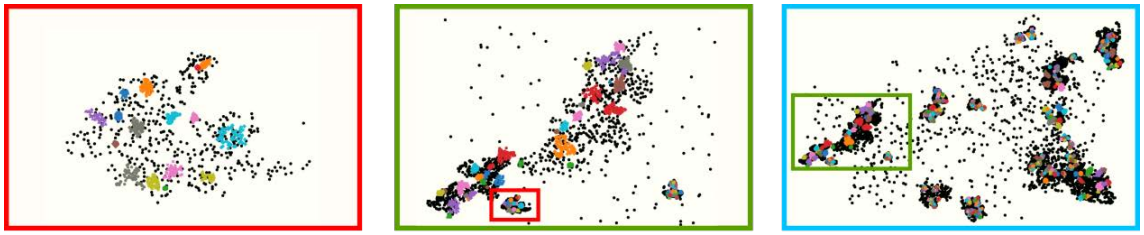


Figure 5.12: *Representation of a 2-D cluster collection at each of the 3 hierarchy levels. With the world coordinate analogy: (a) level 3 (city), (b) level 2 (region), and (c) level 1 (country).*

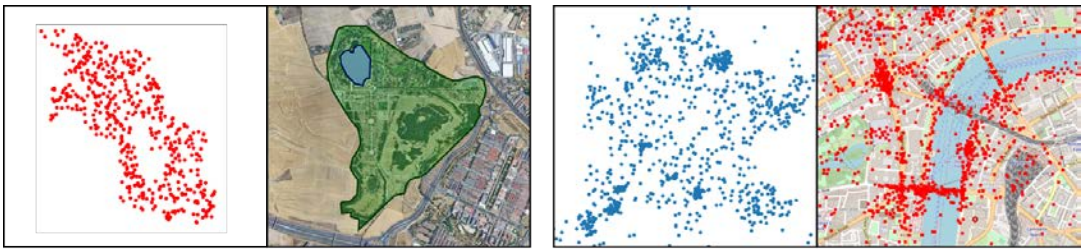
and we specify $ENR = 0.1$ in level 1, we will have 10 noise samples per every cluster in level 2. The specific values of these parameter across all levels will determine the global ratio of samples to noise in the collection, i.e., the *Signal-to-Noise Ratio* (SNR). In the example specified before ($ENR = 0.1$), generating clusters of 15 samples each yields:

$$SNR = 0.1 \frac{clusters}{noise\ samples} \cdot 15 \frac{relevant\ samples}{cluster} = 1.5 \frac{relevant\ samples}{noise\ samples} \tag{5.13}$$

A visual example of the effect of the ENR is shown in Fig. 5.11.

Let us reflect on the last paragraphs with a representative example. We can imagine a set of geographical world coordinates representing important landmarks within a small town. This is represented in Fig. 5.12.a. If we zoom out of the area, we could start seeing a region with several towns of different shapes and sizes, each containing their respective landmarks (Fig. 5.12.b). Zoom out a little further and we could distinguish a country containing several regions (Fig. 5.12.c). This illustrates a collection with 3 hierarchy levels.

In sum, this module gives us control over three more flexible parameters: (1) the number of hierarchy levels of the collection, i.e., the depth of the generative process; (2) the number of elements in the clusters at each hierarchy level, which determines the final



(a) (b)

Figure 5.13: *Examples of similarities between SynFlex and real-world data: (a) SynFlex cluster with arbitrary shape (left) and an area with no samples within it compared to a park(Parque Polvoranca, Community of Madrid, Spain) that has an inaccessible small lake inside (right); (b) 2-level SynFlex cluster collection (left) and a partial map of London with the Thames river where samples are relevant photos (right).*

number of clusters in the collection; and (3) the SNR of the sample space, controlled by the value of the ENR at each hierarchy level.

It is worth mentioning that, when generating a collection, the value for the parameters of the cluster generation need to be provided as well. This can be done by specifying a unique value of each parameter for all clusters in that level (e.g., $N_S = 32,\ \alpha = 20,\ \beta = 0.2$) or by specifying a distribution from which to draw the value of the parameter (e.g., $N_S \sim \mathcal{N}(\mu = 60, \sigma = 15),\ \alpha \sim \mathcal{U}(-50, 50),\ \beta \sim \mathcal{U}(0.2, 0.8)$).

The output of this module can be expressed similarly to the previous, using a set of samples $\boldsymbol{X_C}$, this time representing the whole collection of clusters. Furthermore, a label vector $\boldsymbol{y_C}$ can be defined as a vector that maps the samples with their corresponding cluster.

Through the process defined in this section, we are able to generate synthetic databases with realistic distributions whose characteristics can be flexibly controlled by specifying a few interpretable parameters. The system was designed to express realistic datasets. A clear influence when developing the flexibility of the system was the application context in which this thesis is focused: landmark discovery. In this sense, the similarities between real geographic data and *SynFlex* data are quite illustrative. For instance, the ability to produce clusters with arbitrary shapes emulate the conditions in which data distributions are generated in real-word scenarios, where obstacles may be present. The combination of the cluster generation module with a hierarchical approach and the noise adding mechanism allow for the imitation of certain characteristics of real data that are hard to replicate. This can be observed in Fig. 5.13. In real datasets, there are often physical limitations that hinder the presence of data on certain regions (e.g., a lake or a river). On the other hand, it is common to have some samples in certain regions even when there is not a nearby cluster (user uploading pictures of their own interest). This can be observed in Fig. 5.13.b, where some isolated samples (noise) are present in between the main agglomerations (clusters). In Section 6.2, we will further revise the realism and applications of this data generation system.

5.2. Internal clustering evaluation: Fluctuation-Agglomeration index (*FLAG*)

In this section, we provide a novel metric for internal clustering evaluation: the *Fluctuation-Agglomeration* (*FLAG*) index.

Even if we are able to generate a realistic synthetic database, the fact that real-world fully-labeled databases are scarce remains true. Therefore, it is essential to find an accurate method to evaluate clustering algorithms without access to detailed labels. However, as we have seen in Section 2.2.3, most internal metrics [94]–[97] are conceived as generic evaluation techniques and, for the reasons discussed in that section, they are not tailored to be applied in databases with significant density variations. Furthermore, those that are designed to be applied in scenarios as those envisaged in this research [98], [99] do not accurately reflect the performance of the algorithms. This last argument is further developed and proven in Section 6.3.

Therefore, we define *FLAG*, a novel internal evaluation technique that outperforms all the relevant metrics in the current literature (see Section 6.3). As we have seen, most internal metrics rely on a comparison between intra-cluster dispersion (ideally low) and inter-cluster dispersion (ideally high). However, unlike most of the metrics discussed in Section 2.2.3, which rely on sample separation to compute intra-cluster dispersion, our metric focuses on density *fluctuation*. Furthermore, let us remember that, when computing inter-cluster dispersion, most of the discussed metrics rely on cluster centroid positions to calculate absolute distances between clusters, which works appropriately for globular clusters, but is not representative of arbitrarily shaped clusters or clusters with different scales. Instead, our metric will take into account the distance between cluster boundaries to compute distances and gain notion of the *agglomeration* of the compared clusters taking into account their respective scales.

Fluctuation: First, we resort to graph theory and define the notion of intra-cluster dispersion as the fluctuation of the Minimum Spanning Tree (MST) representing the cluster. To this end, we compute the branch vector $\boldsymbol{b_i}$ of a cluster C_i as the vector that contains the values (length) of the edges of the Minimum Spanning Tree (see Fig. 5.14):

$$\boldsymbol{b_i} = [b_{i,1}, b_{i,2}, ..., b_{i,N_S-1}] \tag{5.14}$$

where N_S is the number of samples in cluster C_i.

Then, the *fluctuation* F_i of cluster C_i can be expressed as:

$$F_i = 1 + \frac{\sigma_i}{\mu_i} \tag{5.15}$$

where σ_i and μ_i are, respectively, the standard deviation and the mean of branch vector $\boldsymbol{b_i}$. Thus, the fluctuation is computed as the variation of distances between adjacent cluster samples (σ_i). This variation, however, needs to be normalized by the cluster's own

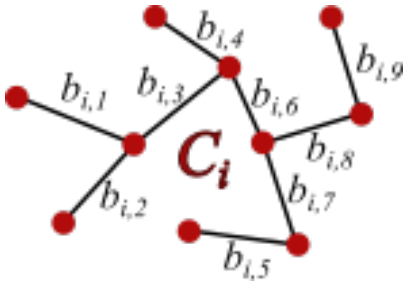


Figure 5.14: *Graph representing the MST of Cluster C_i and its corresponding branch vector $\boldsymbol{b_i}$.*

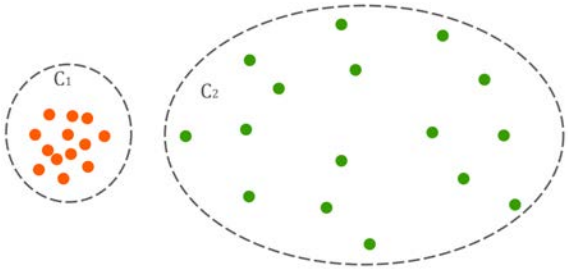


Figure 5.15: *Graphical representation of two clusters with different scales.*

internal distribution (μ_i), since computing absolute values would penalize more dispersed clusters. Furthermore, it is convenient to avoid that the value for the fluctuation fully determines the final *FLAG* metric value in cases with low cardinality (if $|C_i| = 2$, then $\sigma_i = 0$), which leads to the sum of a constant value (if σ_i, then $F_i = 1$).

Intuitively, since we want our clusters to be as evenly distributed as possible, F_i is desired to have a small value ($F_i \approx 1$).

Agglomeration: Secondly, we need to define the notion of inter-cluster dispersion. This notion is defined for each cluster as the degree of agglomeration of other clusters in the region where it is located. Ideally, we want every cluster to be grouped so that the distance to its neighbouring clusters is larger than the distance between the samples that conform it. However, this causes a scale conflict. This conflict is illustrated in Fig. 5.15, where two clusters with different scales can be observed. At first glance, it is clear that these clusters have been properly grouped. However, if we measure the absolute distance between clusters C_1 and C_2 and we compare it to their corresponding internal distances, C_1 yields a satisfactory result whereas C_2 does not.

To overcome this issue, we define the relative distance between to clusters C_i and C_j as:

$$r_{i,j} = \frac{d_{i,j}}{\min(\mu_i, \mu_j)} \tag{5.16}$$

where $d_{i,j}$ is the distance between clusters C_i and C_j. This distance, as it was partially mentioned above, is calculated as the smallest euclidean distance between two samples that do not belong to the same cluster. Again, μ_i is the mean of C_i's branch vector.

From the relative distance in Eq. (5.16), we can compute the *agglomeration* relative to cluster C_i as follows:

$$A_i = \sum_{\substack{1 \leq j \leq |C| \\ j \neq i}} \frac{1}{r_{i,j}} \tag{5.17}$$

where $|C|$ is the cardinality of the cluster collection, i.e., the total number of clusters.

By considering the inverse of this relative distance we give more relevance in the final value of the agglomeration to the clusters that are closer to the one being analyzed.

Since we want our clusters to be separated by a distance that is large enough to consider them as independent, their relative distances should be high and, hence, A_i is desired to have a low value. Note that the distance only needs to be higher than the average distance of the most compact cluster (for instance, the left cluster in Fig. 5.15), thereby solving the conflict presented above.

FLAG: Finally, we could estimate the quality of each cluster as a simple product between *fluctuation* (Eq. (5.15)) and *agglomeration* (Eq. (5.17)). Nevertheless, one could argue that not all clusters have the same importance in a collection. As we have mentioned before in this manuscript, a common method to determine relevance is by sheer popularity or, in this case, sample cardinality. Hence, the impact of each cluster C_i in the final metric is determined by its weight:

$$w_i = \frac{|C_i|}{N_T - N_N} \tag{5.18}$$

where $|C_i|$ is the cardinality of cluster C_i, and N_T and N_N are, respectively, the total amount of samples and the amount of noise samples in the sample space.

Note that noise samples are not taken into account when calculating the weight of each cluster. However, noise is an important factor that must not be neglected in the evaluation. For instance, a hypothetical algorithm that generates excessive noise and a scarce collection of perfectly separated clusters could, were noise to be neglected, obtain a better result than an algorithm that would risk classifying more points. Thus, we define the noise factor f_N as the ratio between the total amount of points and the amount of relevant points:

$$f_N = \frac{N_T}{N_T - N_N} \tag{5.19}$$

Therefore, this factor is designed to penalize algorithms that generate excessive noise

and, in consequence, higher values of f_N should hinder the value of the metric more than lower ones.

To conclude, the final value of the *FLAG* index is computed as the sum of the weighted products of *fluctuation* and *agglomeration* for each cluster, and then modulated by the noise factor as follows:

$$FLAG = f_N \cdot \sum_{1 \leq i \leq |C|} w_i \cdot F_i \cdot A_i \tag{5.20}$$

where let us remember that $|C|$ is the cardinality of the cluster collection, i.e., the total number of clusters.

This metric, as it will be later explored in Section 6.3, is a powerful instrument for evaluating clustering algorithms in datasets where labels are not available and, furthermore, where variability of density and shapes are important factors to be taken into account.

5.3. Combining datasets and metrics into a benchmark for density-based clustering

As it was discussed at the beginning of this chapter, we aim to collect a series of synthetic and non-synthetic databases with the goal of building a benchmark for density-based clustering algorithms. Particularly, the synthetic datasets were generated using *SynFlex* (see Section 5.1) to reflect the behaviour of a tested algorithm in scenarios with varying characteristics.

There are several data behaviours that can affect an algorithm's result. When working with datasets with high density variability and arbitrary data formations (the type of data for which density-based clustering algorithms would be applied), we have identified the following as the primary factors to be studied: (1) the shape of the clusters to be discovered, (2) the density variability of the sample space, (3) the amount of noise in the distribution and (4) the depth of the hierarchy in the data generation process and (5) the dimensionality of the data.

Then, a series of databases exploring each of these factors is generated. This is done by exploring different values of the parameters that control the *SynFlex* generation system. For instance, when analyzing the effect of changing the shape of the clusters, different values for parameters α and β must be explored (see Section 5.1.1). To this end, we first fix some default values for all adjustable parameters. Then, for each behaviour factor, the corresponding parameters are tweaked to generate databases with the desired characteristics.

Furthermore, two additional studies are presented: (6) a difficulty study, which combines several factors of impact creating four levels of clustering difficulty and (7) a study of unlabeled real-world databases using the *FLAG* index proposed in Section 5.2. The

objective of these last two experiments is to provide a realistic performance evaluation of the algorithms in generic scenarios.

The details of the parameter adjustment for each of the six mentioned synthetic studies, as well as the characteristics of the real-world datasets of the seventh study, will be thoroughly described and further explained in Section 6.4, where the benchmark will be tested against popular clustering algorithms from the literature discussed in this manuscript (Section 2.3) and the ones proposed in [1], which are also presented and validated in Chapters 3 and 4.

Variable	Brief description
D	Data dimensionality
CHR	Cluster-to-Hypercube Ratio
N_V	Nº Voronoi cells in hyper-cube
N_C	Nº Vor. cells that form the clsuter shape
N_S	Nº samples in the cluster
V	**Vor. stage output:** set containing coordinates of Vor. cells
Ω_V	Set of total Vor. cells
Ω_L	Set of limit Vor. cells
Ω_{E_n}	Set of valid Vor. cells at time instant n
x_n	Markov state at time instant n
t_n	Transmitter cell index at time instant n
$\boldsymbol{d_n}$	Direction of expansion vector at time instant n
r_n	Receptor cell index at time instant n
$\boldsymbol{i_n}$	Adding instant vector at time instant n
Ψ_j	Set of cells in the vicinity of cell j
α	Choice of t_n
β	Change in $\boldsymbol{d_n}$
Ω_C	**Exp. stage output:** set of cells forming the shape
l_o	Mean of Distribution (Normal)
σ_C	Variance of Distribution (Normal)
S_C	**Sam. stage output:** set of cluster samples
λ_{MAX}	Max. scale of the cluster
λ_S	Final scale of the cluster
f_λ	Scale factor
ENR	Element-to-Noise Ratio
$\boldsymbol{b_i}$	Branch vector of cluster C_i
F_i	Fluctuation of cluster C_i
F_i	Fluctuation of cluster C_i
$r_{i,j}$	Relative distance between clusters C_i and C_j
A_i	Agglomeration of cluster C_i
w_i	weight of cluster C_i
f_N	Noise factor

Table 5.2: *Summary of the variables used in this chapter and a brief explanation of each of them. Over the dashed lines, the variables corresponding to SynFlex; below, those belonging to FLAG.*

CHAPTER 6

VALIDATION EXPERIMENTS ON THE PROPOSED BENCHMARK AND ASSOCIATED TOOLS (*SYNFLEX*, *FLAG*).

In this chapter, we present the experiments carried out to verify the usefulness of the tools proposed in Chapter 5. To this end, we have studied the behaviour of several clustering algorithms that, as we discussed in Section 2.3, are often used in the scenarios proposed in this research. Particularly, we consider five algorithms with diverse characteristics from the literature: (1) K-Means, (2) Mean-Shift, (3) Agglomerative Hierarchical Clustering, (4) DBSCAN, (5) *HDBSCAN*; and the two algorithms proposed in Chapter 3: *KDBSCAN* and *VDBSCAN*.

For the experiments where labels were available, we made use of an external metric to evaluate the performance of each algorithm. We will first present the experiment through which we chose this metric (Section 6.1). Then, the validation experiments carried out for the *SynFlex* generator and the *FLAG* index are presented in Sections 6.2 and 6.3. To close the chapter, the benchmark experiments (Section 6.4) are described and thoroughly discussed.

6.1. Evaluation metrics for labeled data

The metric employed when evaluating an algorithm is important, especially when the context of application of the algorithm is known, since not all clustering metrics are suitable for the task, as it was discussed in Section 2.2.2. The following metrics were considered for this experiment: Adjusted Rand Index, Fowlkles-Mallows, Adjusted Mutual Information, and V-Measure (along with its components Completeness and Homogeneity). The choice is supported by a simple experiment, for which a real-world database was fully-labeled. This database consists in a set of geo-located points centered in the city of *Getafe* (Madrid, Spain). Similarly to the ones discussed in Chapter 4, this dataset was extracted from Flickr, and it includes 2827 data samples that correspond to the GPS coordinates where the photographs were taken. The samples were manually labeled into 20 different classes, each one representing a landmark (plus a a 21^{st} category representing noise).

Thus, in order to choose the optimal external metric to evaluate clustering, the *Getafe* database has been previously clustered using the six algorithms described above. Each algorithm has a key parameter to adjust, which has been thoroughly swept to check the influence on the performance measured by each metric. Fig. 6.1 shows a comparison of these results for each algorithm in the *Getafe* dataset.

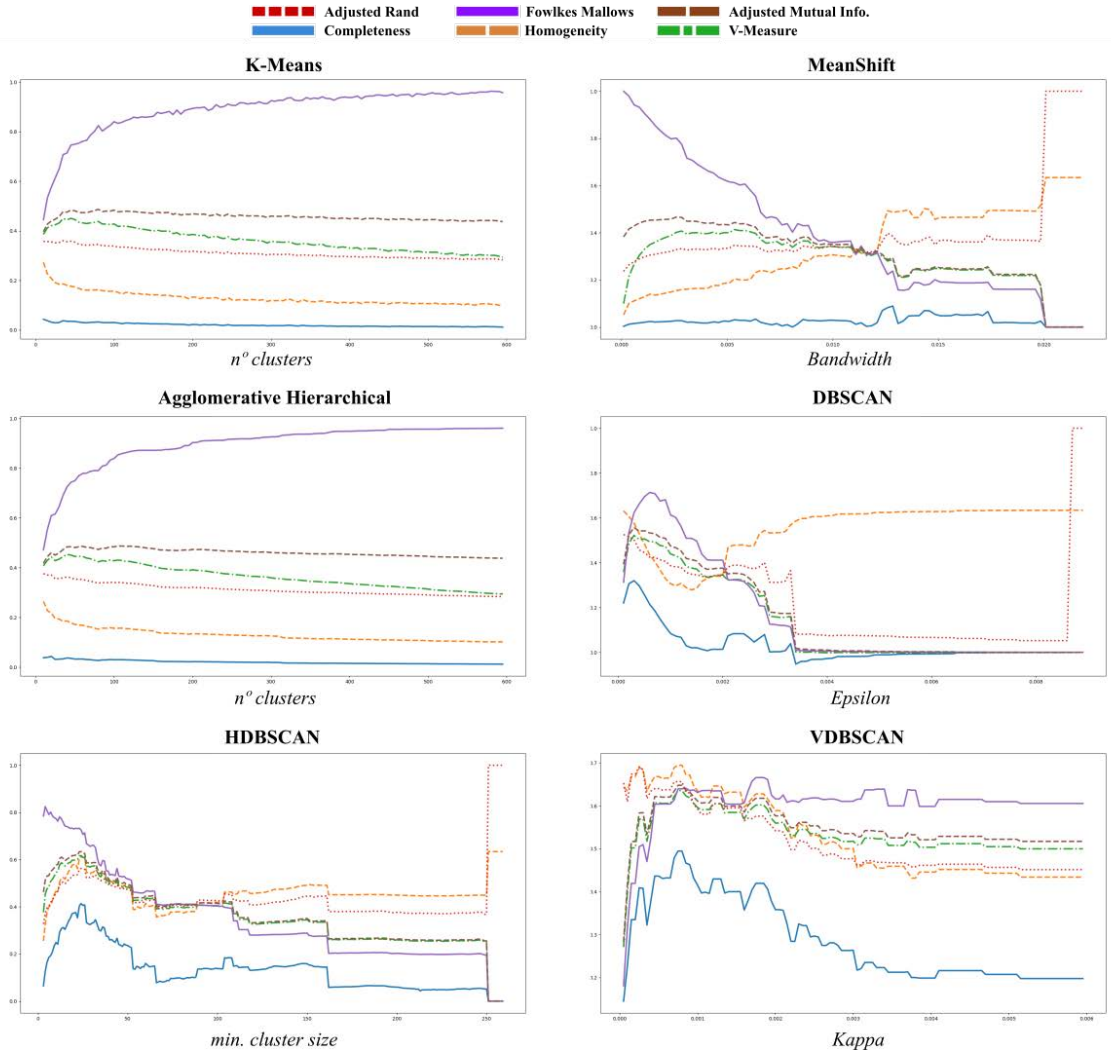


Figure 6.1: *Parameter sweep (x-axis) for six clustering algorithms and the results obtained for the considered external metrics.*

The first thing we can notice when observing Fig. 6.1 is that some metrics produce erratic results when the parameters take extreme values. Fowlkes-Mallows index tends to return the maximum value when the lowest number of clusters is detected. This means that, for algorithms with parameters defining an area-of-effect (e.g., the bandwidth for Mean-Shift; the ε for DBSCAN), Fowlkes-Mallows index reaches its maximum when the parameter is at its largest value and, therefore, the number of clusters tends to a minimal value. Likewise, for algorithms where the parameter is the number of clusters to be generated (K-Means, Agglomerative Hierarchical), the highest value is obtained when the parameter is minimal. This behaviour, although rather expected (see Section 2.2.2), is undesired and, in consequence, prevents the adoption of this metric in our scenario. Similarly, the Completeness score returns the highest values for small values of the bandwidth and ε parameters. Although it does provide a local maximum when analyzing different values of the parameter for K-Means, the observed increase in performance is not substantial and, furthermore, it still presents an erratic behaviour for the rest of the algorithms.

The results produce by the homogeneity index are radically different from the other two, returning better results for high number of clusters and low bandwidth for Mean-Shift. A local maximum can be observed in the Homogeneity score when analyzing different values of ε in DBSCAN, but this is not enough to realistically reflect the results of a certain clustering in this scenario. However, these two complementary scores (as it is mentioned in Section 2.2.2) were conceived to be parts of a whole, more representative index: the V-Measure.

The V-Measure and the other two depicted metrics (Adjusted Rand Index and Adjusted Mutual Information) provide more stable results. Moreover, they share most of the local maxima (even if they do not always share the absolute maxima) for most of the tested algorithms.

The Adjusted Rand Index provides generally smaller values than the other two, and it penalizes harshly those algorithms that cannot identify noise samples (K-Means, Mean-Shift, Agglomerative). The V-Measure provides, in average, higher performance values than the other two. This is not necessarily a good thing, since, in fact, it has a narrower range of values and is therefore less discriminative. Finally, the Adjusted Mutual Information score provides a compromise between the previous two, since it is more permissive with no-noise algorithms while still providing an acceptable range of results. For this reason, the external results in this Chapter will be provided using the Adjusted Mutual Information score (AMI) as the evaluation metric, comparing the ground truth labels with the predicted labels resulting from applying each clustering algorithm.

Therefore, the first required validation is centered on *SynFlex*, and it is explained in the following section.

6.2. Assessment of the synthetic data generator with flexible parametrization (*SynFlex*)

The *SynFlex* generator is defined in Section 5.1 as a tool capable of generating datasets that can be used to emulate realistic scenarios for tasks like landmark discovery. In order to justify this claim, we have assessed the potential of *SynFlex* to automatically generate a database with similar properties to the real-world *Getafe* dataset.

In order to set the appropriate parameters in *SynFlex* that led to a dataset similar to *Getafe*, we proceeded as follows: first, we obtained the number of clusters ($N_S(level\ 1) = 20$) and the range of the cluster cardinalities ($N_S(level\ 2) = \mathcal{U}(17, 103)$) from the *Getafe* database. Moreover, a similar SNR (~ 0.6) was desired, resulting on setting $ENR = 0.01128$ (88.65 noise samples per cluster) in level 1. For the rest of the parameters, a random range of values was heuristically set. To account for possible variations, 10 different databases were produced with these characteristics, and the six clustering algorithms discussed above were applied to each of the sets, with the aim of assessing if the results were in concordance with those obtained using the original dataset. An instance of a synthet-

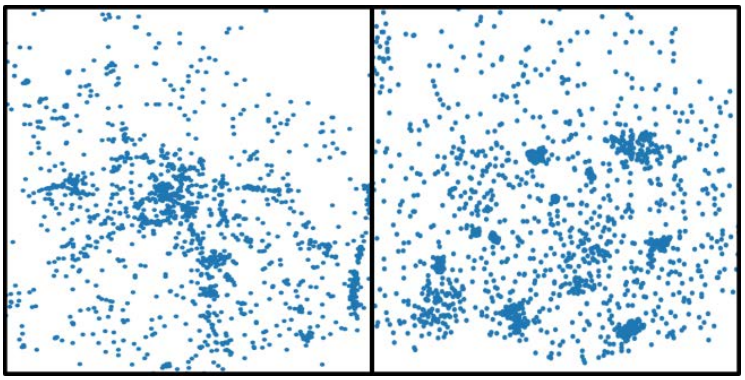


Figure 6.2: *Visual comparison between the Getafe dataset (left) and an equivalent dataset generated using SynFlex (right).*

ically generated dataset using this parametrization is compared to the *Getafe* dataset in Fig. 6.2, where one can observe that the data distributions are visually similar.

It is important to mention that fixing the algorithms' parameters following a cross-validation strategy was not practical, since there were 10 different databases with different characteristics and a fair comparison required that all executions shared a generic parametrization. Thus, the parameters were heuristically set taking as a reference the parametrization for the *Getafe* database and the distribution of each synthetic set. Particularly, the median of all the distances between samples in the distribution was obtained. Let us call this ϕ_D. Then, for each algorithm, an adjustable, algorithm-specific parameter k is set and used in all databases. After that, each algorithm's key parameter is fixed from these two. For instance, for the ε in DBSCAN:

$$\varepsilon = k \cdot \phi_D \tag{6.1}$$

For those algorithms that require an integer parameter (e.g., the number of clusters), the nearest integer to $k \cdot \phi_D$ is set as the parameter's value. Note that, since this strategy for setting the parameters depends on the unique distribution of each dataset, it can be used for any potential parametrization of the *SynFlex* generation system. In fact, this strategy is later used in Section 6.4 to fix the parameters of the algorithms to be tested on the benchmark.

The average results obtained for the 10 synthetic datasets are compared to those obtained by the *Getafe* dataset (see Table 6.1). At first glance, one can clearly see that the performance results are very similar, both in ranking and absolute value. In fact, the only noticeable difference is between *VDBSCAN* and *HDBSCAN*, which, despite being the top 2 ranked algorithms, their position is exchanged. This could be due to the fact that *VDBSCAN* makes the assumption of a radially decreasing density from the distribution center (see Section 3.1.2), which may not always hold in the case of synthetic databases. Despite this small variation, the results are similar enough to demonstrate the capability of

Algorithm	*Getafe*	Synthetic
K-Means	0.443	0.430 (0.03)
Mean-Shift	0.413	0.397 (0.03)
Agglom. Hier.	0.453	0.443 (0.03)
DBSCAN	0.523	0.539 (0.1)
HDBSCAN	0.620	0.596 (0.03)
VDBSCAN	0.636	0.581 (0.03)

Table 6.1: *Results for the Getafe dataset and average (standard deviation) results for the analogous synthetic datasets. Results are expressed as the Adjusted Mutual Information for each of the six prposed algorithms.*

SynFlex to generate synthetic datasets that look realistic and share properties with real databases. This observation supports the use of synthetic datasets for the evaluation of density-based clustering algorithms.

6.3. Assessment of *FLAG* as an internal metric to evaluate density-based clustering algorithms

A realistic automatic data generator like *SynFlex* is a powerful tool to study the behaviour of clustering algorithms. However, as it has been mentioned, its existence does not change the fact that there is an important lack of real-world labeled databases. In consequence, the definition of the *FLAG* index is vital to evaluate those cases in which obtaining labels is either impractical or impossible. Nevertheless, since internal metrics are not based on any ground-truth, they need to be verified for the purposes at hand.

In this section, we present the results of the fully labeled *Getafe* dataset for the six clustering algorithms that were already used in Section 6.2. The results shown are those yielded by four of the most popular internal metrics in the literature: Dunn index [94], Davies-Bouldin score [95], Calinski-Harabasz score [96], and Silhouette score [97]. Additionally, two scenario specific metrics are used: DBCV [98] and SV index [99]. These two metrics, as we discussed in Section 2.2.3, were designed to evaluate data distributions where density variation is an important factor to be accounted for. The results are then compared to the ones provided by the *FLAG* index and, of course, to those obtained by the selected external metric: Adjusted Mutual Information (*AMI*). Let us remember that two of the discussed metrics (Davies-Bouldin and the proposed *FLAG* index) express better clustering results with lower values, whereas higher values are preferred for the rest of the indices.

Table 6.2 shows the results of our experiment. Although a glance at them reveals the superiority of the FLAG index to mimic the behaviour of the external metric, it is worth noticing that different metrics produce values with very different ranges and varying orders of magnitude. Hence, a fair comparison is not straightforward. In such scenario, despite considering absolute values, it is more meaningful to compare the rankings of

Algorithm	Dunn	Dav-Bould	Cal-Hbz	Silhouette	DBCV	SV	FLAG	AMI
K-Means	0.0096	0.70	4782.1	0.51	-0.65	1.10	5.63	0.443
Mean-Shift	0.0147	0.67	2859.1	0.50	-0.71	1.42	6.44	0.413
Agglom. Hier.	0.0154	0.71	4325.8	0.48	-0.55	1.11	3.93	0.453
DBSCAN	0.0002	3.01	34.4	-0.20	-0.49	2.15	1.99	0.523
HDBSCAN	0.0011	2.27	122.2	0.01	-0.70	1.19	1.61	0.620
VDBSCAN	0.0012	1.46	184.9	-0.14	-0.65	1.64	0.48	0.636

Table 6.2: *Performance for the six algorithms in the Getafe dataset. Results are expressed as four generic internal metrics (Dunn, Davies-Bouldin, Calinski-Harabasz and Silhouette), 2 specific density-based internal metrics (DBCV and SV), the proposed FLAG index and the Adjusted Mutual Information score.*

Algorithm	Dunn	Dav-Bould	Cal-Hbz	Silhouette	DBCV	SV	FLAG	AMI
K-Means	3	2	1	1	3	6	5	5
Mean-Shift	2	1	3	2	6	3	6	6
Agglom. Hier.	1	3	2	3	2	5	4	4
DBSCAN	6	6	6	6	1	1	3	3
HDBSCAN	5	5	5	4	5	4	2	2
VDBSCAN	4	4	4	5	4	2	1	1
Rank coeff.	Dunn	Dav-Bould	Cal-Hbz	Silhouette	DBCV	SV	FLAG	AMI
Kendall τ	-0.33	-0.60	-0.33	-0.60	0.20	0.33	1.0	-
Spearman ρ	-0.60	-0.77	-0.60	-0.77	0.14	0.43	1.0	-
Pinto Weighted	-0.59	-0.77	-0.61	-0.78	0.02	0.44	1.0	-

Table 6.3: *Performance ranking for the six algorithms in the Getafe dataset. Results are expressed as four generic internal metrics (Dunn, Davies-Bouldin, Calinski-Harabasz and Silhouette), 2 specific density-based internal metrics (DBCV and SV), the proposed FLAG index and the Adjusted Mutual Information (AMI) score. Three ranking correlation coefficients (Kendall's τ, Spearman's ρ and Pinto Weighted Rank) are shown for the ranking of every metric with respect to the AMI ranking.*

the algorithms obtained using different metrics. Table 6.3 shows equivalent results to Table 6.2, albeit this time in terms of the position that each algorithm got using each evaluation technique.

In addition to comparing internal metrics to the external reference (*AMI*), we made use of three different ranking coefficients: Kendall's τ [117], Spearman's ρ [118] and Pinto Weighted Rank [119].

Kendall's τ and Spearman's ρ are well-known ranking coefficients. Both of them consider all positions of a ranking to be equivalent and, hence, every position variation is equivalent as well. However, when comparing clustering algorithms, it is useful to assign more relevance to the top positions of the ranking. Therefore, we have also made use of the Pinto Weighted Rank, which considers higher positions of the ranking to be more relevant when calculating the final value of the coefficient.

The values of the three ranking coefficients are expressed in the range $[-1, +1]$, where higher values represent a better correlation with the reference ranking. When observ-

ing Table 6.3, one can notice that the three ranking coefficients yield similar results. As it is expected, the density-based metrics (DBCV and SV index) show better correlation with the ranking provided by the external metric than the other four generic internal metrics. Moreover, *FLAG* offers a perfect correlation with the considered external metric, which proves that it is better tailored to evaluate clustering algorithms in databases where density, shape and size variations are strong factors to be taken into account. In consequence, we selected *FLAG* as the evaluation metric of choice when dealing with unlabeled databases. This is the case of real-world databases obtained to for the benchmark, which are described in the following section.

6.4. Experiments using the proposed benchmark

In the previous sections, we have demonstrated the usefulness of *SynFlex* as a tool for the generation of realistic data. Moreover, we have assessed the viability of *FLAG* as an internal metric to evaluate clustering algorithms in scenarios where density-based clustering is usually applied.

This section contains a detailed description of the process followed to create the databases for the benchmark presented in Section 5.3. The goal of this benchmark is to establish a systematic, deep comparison between clustering algorithms in scenarios with vastly varying densities.

In this context, as it was mentioned in Section 5.3, we have identified five main factors that impact the nature of the data distributions when we want to attempt the clustering of the data: (1) shape, (2) density, (3) noise, (4) hierarchy depth, and (5) dimensionality. This led to several experiments where appropriate parametrizations of *SynFlex* were proposed to study each factor. Furthermore, two additional experiments were included in the benchmark, namely: (6) a difficulty experiment and (7) a real-world experiment.

6.4.1. Setup protocol

In order to assess the impact of each factor, some default values for the parameters were proposed, which are listed in Table 6.4. Then, to study each factor, the corresponding parameters were modified and the impact of these changes was studied, keeping the default values for the rest. Let us remember that, as we expressed in Section 5.3, for each parametrization, the model was executed 10 times to produce more stable results. A brief summary of the experiments that are described in the remaining of this section is included in Table 6.5.

It is also worth mentioning at this point that all datasets used for the experiments have been made publicly available [10] for the sake of replicability and to promote future research in the field using this benchmark.

[10] https://github.com/plasavall/benchmark

Parameter	Brief description	Default value
Depth	Nº Hierarchy levels	2
f_λ	Scale factor	$\mathcal{U}(0.05, 1)$
ENR (*level* 1)	Element-to-Noise Ratio	0.025
N_S (*level* 1)	Nº el. in the super-cluster	32
D	Dimensionality	2
N_V	Nº Voronoi cells	$100 \cdot 2^{D-1}$
N_S (*last level*)	Nº samples in the clusters	$\mathcal{U}(40, 80)$
CHR	Cluster-to-Hypercube-Ratio	0.25
α	Choice of t_n	$\mathcal{U}(-100, +100)$
β	Change in $\boldsymbol{d_n}$	$\mathcal{U}(0, 1)$
Distribution	Type of distribution	$\mathcal{N}(\mu = l_o, \sigma = \sigma_C)$
l_o	Mean of Distribution	Initial transmitter cell
σ_C	Variance of Distribution	0.5

Table 6.4: *Default value given to the SynFlex parameters. The first three (over the first dashed line) are set for the collection generation, while the rest are set for the cluster generation. Note that the parameters below the second dashed line are directly dependent on the choice of distribution.*

Factor studied	Parameters involved	Evaluation metric
Shape	α, β	AMI
Density	σ_{N_S}, σ_{f_λ}	AMI
Noise	*ENR*	AMI
Depth	N_S, *Depth*	AMI
Dimensionality	D	AMI
Difficulty	N_S, *Depth*, *ENR*, α, β, f_λ	AMI
Real-World	-	FLAG

Table 6.5: *Summary of the experiments. Over the dashed lines, the experiments corresponding to the five identified impact factors; below, the two additional experiments proposed.*

6.4.2. Results of the benchmark experiments

In this section, we break down the results for the seven experiments described at the beginning of Section 6.4. First, each experiment is properly motivated. For the experiments using SynFlex datasets, the parametrization details are presented and discussed. Furthermore, for those databases that can be represented in 2 dimensions we include reference figures to better understand the features of the data. To conclude, the results for each experiment are analyzed, and we include relevant conclusions drawn from them.

Shape experiment

The shape of a cluster is determined by parameters α and β, described in Section 5.1.1. The former controls which Voronoi region dictates the immediate expansion area of the cluster, whereas the latter controls the direction of that expansion. Low values of α will yield shapes closer to a globular cluster, while high values will produce a worm-like

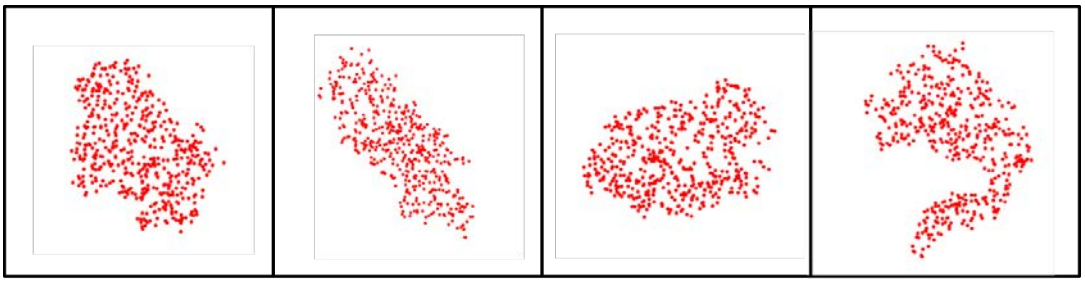


Figure 6.3: *Cluster examples for each parametrization: (a) globular-linear clusters, (b) worm-linear clusters, (c) globular-random clusters, and (d) worm-random clusters.*

Parameter	globular-linear	worm-linear	globular-random	worm-random
α	-100	+100	-100	+100
β	0	0	1	1

Table 6.6: *Parametrization for the shape experiment.*

expansion of the cluster. On the other hand, low values of β will force the expansion in a single direction, while large values of this parameter will produce random changes in expansion direction, leading to irregularly-shaped clusters.

In this experiment, datasets with four different configurations are built altering the values of these two parameters with respect to the default parametrization. Particularly, the four parametrizations are: (1) globular-linear clusters, (2) worm-linear clusters, (3) globular-random clusters, and (4) worm-random clusters. The values for the α and β parameters are shown in Table 6.6. The rest of the parametrization remains in its default values. An example of a cluster generated with each parametrization is displayed in Fig. 6.3.

Clustering is then performed over this data using the six compared algorithms. Average results, in terms of Adjusted Mutual Information, obtained from running the algorithms in 10 replications of each parametrization are shown in Fig. 6.4.

From the figure, we can draw some conclusions. First, shape variations do not seem to affect the performance of K-Means and the Agglomerative Hierarchical Clustering algorithm, since the presence of noise is probably the most important factor in reducing their performance (as it is shown in Section 6.4.2). While the impact of noise is also strong for Mean-Shift, in this case a change in behaviour related to shape can observed. Shapes closer to globular offer better performances than worm-like shapes. The opposite happens for DBSCAN: it seems to be better are recognising the clusters among the noise when they have distinctive shapes. A worthy aspect to mention about DBSCAN is the high variation between results, which could be due to its notorious limitation to handle data with different scales. Finally, *HDBSCAN* and *K+VDBSCAN* offer the best results for all parametrizations, and a slight improvement can be observed for both of them when the

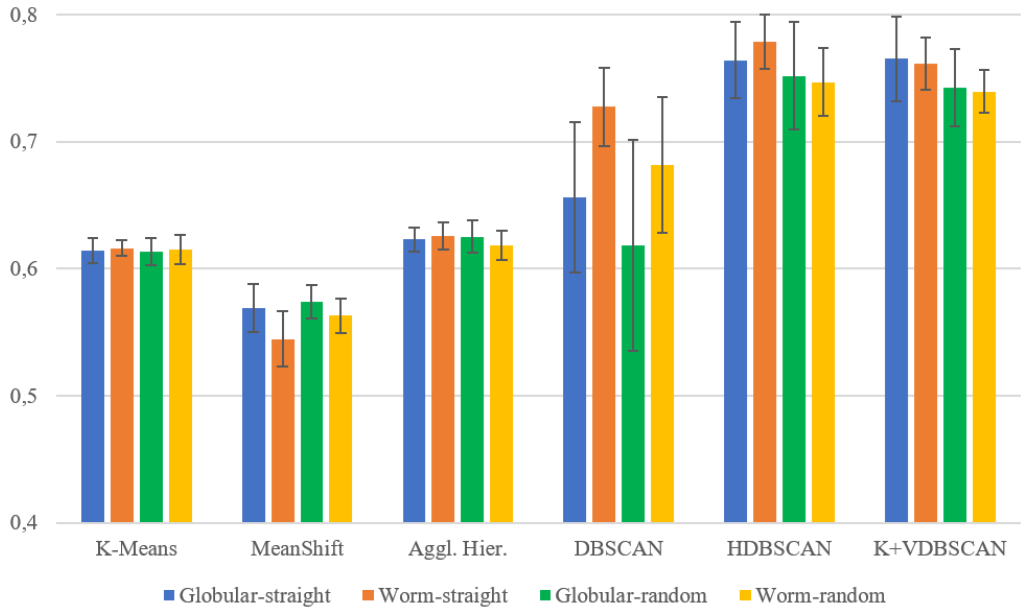


Figure 6.4: *Results for the shape experiment in terms of AMI. The black segments represent the variation in the results through the 10 instances for each parametrization (their height is the value of the standard deviation).*

changes in the direction are minimal (low β).

It is worth mentioning that, as it can be observed in Fig. 6.4, *HDBSCAN* shows a slightly better performance than *K+VDBSCAN*. This swap with respect to the results obtained in Chapter 4 is recurrent among the experiments presented in this chapter, and further supports the idea that the use of additional metadata (e.g., the descriptive tags used in Chapter 4), which is not present in these datasets, is crucial to the task of landmark discovery. This issue will be further discussed in Section 7.2.

Density experiment

The variations in the density within a data distribution are undoubtedly a factor to consider when developing a clustering algorithm. In fact, this was one of the main motivations to develop density-based clustering algorithms originally (see Section 2.1).

According to the definition of the *SynFlex* generation system, we could control the internal density variation within a cluster by choosing an appropriate statistical distribution. For instance, when choosing a Normal distribution to sample data inside a cluster, this variation is controlled by σ_C. However, since these variations depend more on the nature of each individual cluster and not the whole sample space, focusing on inter-cluster variations is more appropriate to study the behaviour of the algorithms. In consequence, variations in density from one cluster to another are no longer controlled by the data distribution, but by their cardinality (set by N_S) and size (set by f_λ).

Therefore, to study the effects of these variations, 4 parametrizations were defined

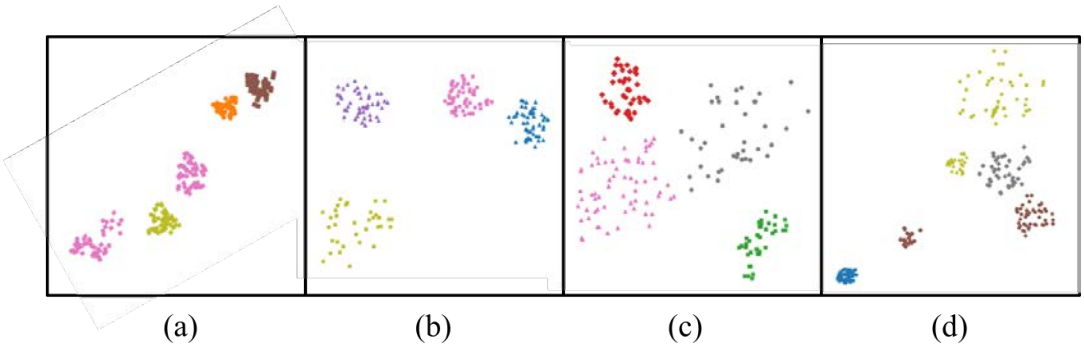


Figure 6.5: *Cluster examples for each parametrization: (a) no density variation, (b) low density variation, (c) medium density variation, and (d) high density variation.*

Parameter	no variation	low variation	med. variation	high variation
σ_{N_S}	0	5	10	15
σ_{f_λ}	0	0.1	0.2	0.3

Table 6.7: *Parametrization for the density experiment.*

where values of these two parameters were changed to sequentially increase the variations in density. In particular, we sampled these parameters using a Normal distribution with a fixed mean ($\mu_{N_S} = 60$, $\mu_{f_\lambda} = 0.7$) and an increasing standard deviation. Then, the four parametrizations were: (1) no density variation, (2) low density variation, (3) medium density variation, and (4) high density variation. The values of the parameters are reflected in Table 6.7. The rest of the parameters were set to their default values. Again, an example of clusters generated with each parametrization is displayed in Fig. 6.5.

Fig. 6.6 shows the results obtained in this experiment. Observing the results, one can notice that modified density-based clustering algorithms (*HDBSCAN* and *K+VDBSCAN*) experiment a slight increase in performance from no density variation to low density variation. From then, a steady decrease is observed, but they always remain over AMI values of 0.7, whereas the rest of the algorithms do not surpass the 0.6 barrier. The only exception to this is DBSCAN, for we which we can observe a sudden increase in performance for the high density variation parametrization. As it happened in the previous section, it is worth mentioning that DBSCAN presents the highest standard deviation in the performance values, so this could also be related to the scaling issues that DBSCAN faces. As for non-density-based clustering algorithms, we can see that density variability is not a strong factor affecting their performance: K-Means and Agglomerative Hierarchical yield AMI performances around 0.6. Nevertheless, Mean-Shift presents the worst results systematically ($AMI < 0.55$). This is probably because, even though it is based on kernel density estimation, it cannot identify noise samples or outliers (all samples were drawn to each of the local maxima of the distribution).

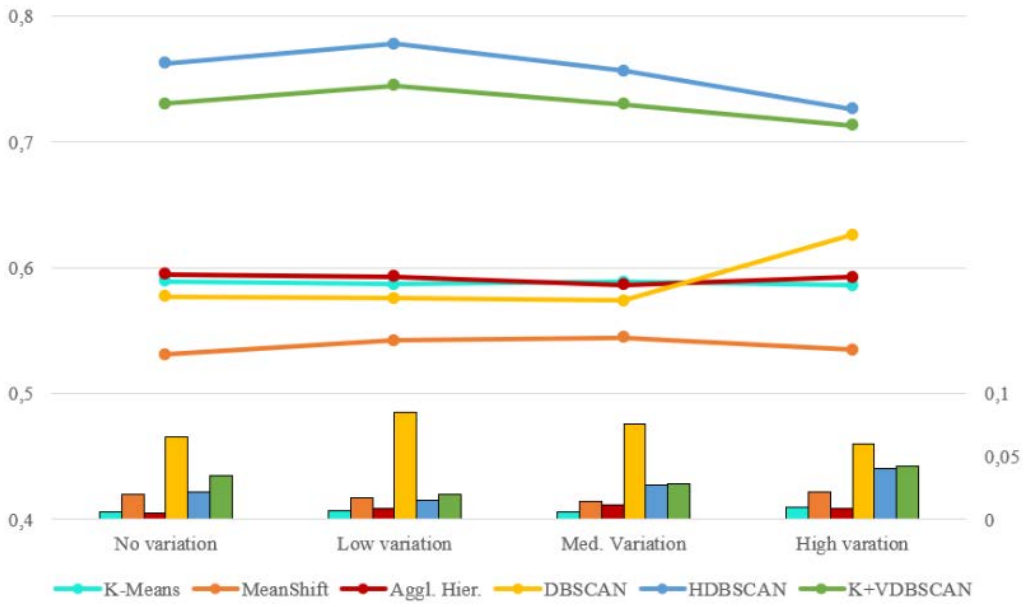


Figure 6.6: *Average results for the density experiment in terms of AMI (left axis). The bars represent the variation in the results through the 10 instances for each parametrization, i.e., their height is the value of the standard deviation (right axis).*

Parameter	no noise	weak noise	mod. noise	strong noise
ENR	∞	0.035	0.015	0.005
$Ns/cl.$	0	29	67	200

Table 6.8: *Parametrization for the noise experiment. The second row shows the approx. amount of noise samples per generated cluster* ($1/ENR$).

Noise experiment

Noise is a problematic factor to analyze. Some algorithms, like K-Means, are designed to neglect the possibility of noise identification. This is a clear disadvantage when the presence of noise or outliers is a reality, which is often the case in real-world databases.

The presence of noise in the *SynFlex* sets is easily controlled with the *ENR* parameter. The default parametrization includes 2 levels of hierarchy, and since any noise generated in the last level would be indistinguishable from the cluster samples themselves, the noise is generated in level 1. Analogously to the previous sections, four parametrizations were tested: (1) no noise, (2) weak noise, (3) moderate noise, and (4) strong noise. The values of the *ENR* for each parametrization are shown in Table 6.8. The rest of the parameters were set to their default values. An example of the noise presence resulting from each parametrization is depicted in Fig. 6.7

The performance of the selected algorithms for the discussed parametrizations is shown in Fig. 6.8. As it can be clearly observed, there is a significant decrease in performance when the amount of noise increases. Most of the density-based clustering algorithms (*DBSCAN*, *HDBSCAN*, *K+VDBSCAN*) obtain decent results ($AMI > 0.65$) up to

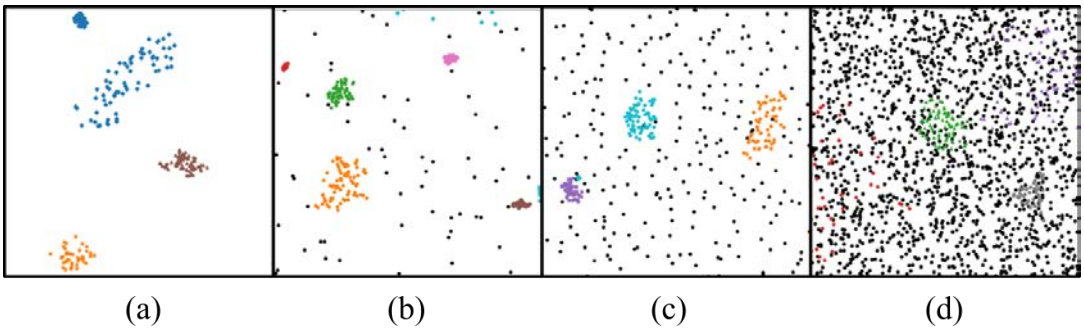


Figure 6.7: *Visual examples (at the same scale) of generated data for each parametrization: (a) no noise, (b) weak noise, (c) moderate noise, and (d) strong noise.*

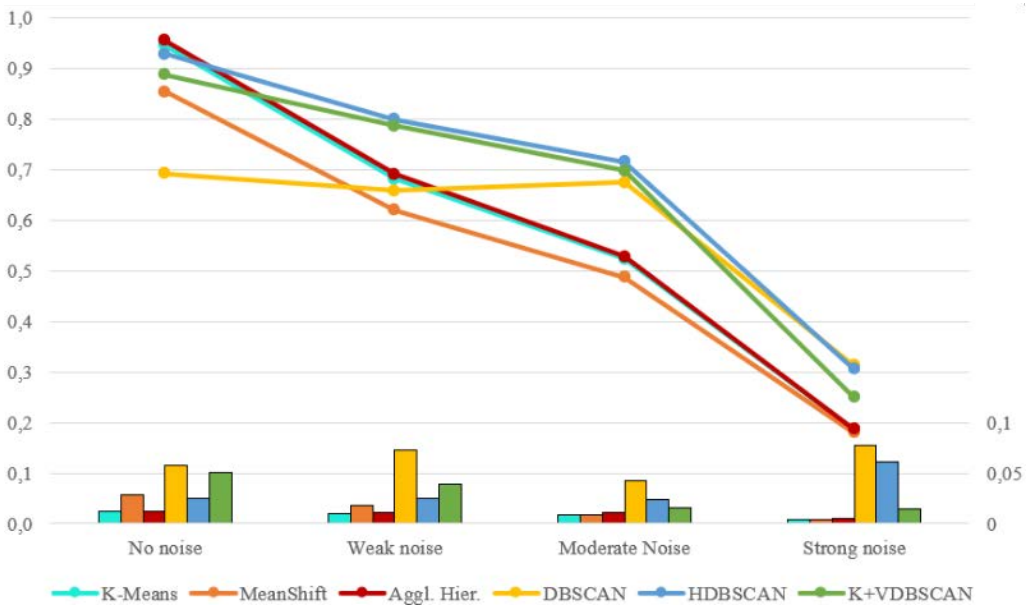
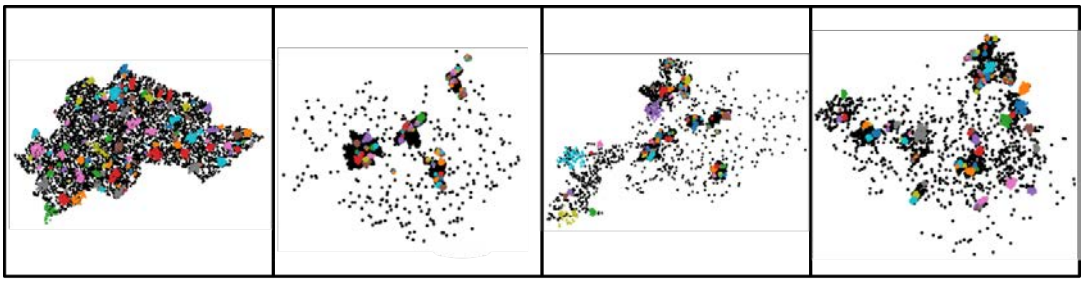


Figure 6.8: *Average results for the noise experiment in terms of AMI (left axis). The bars represent the variation in the results through the 10 instances for each parametrization, i.e., their height is the value of the standard deviation (right axis).*

the scenario with highest levels of noise, where their performance drops dramatically. The other algorithms present a steady decrease, with notable AMI performances when there is no noise (around 0.9) and falling under $AMI < 0.2$ when the noise is strong. These significant changes in performance suggest that noise is one of the most relevant factors in density-based clustering.

Experiment on the depth of the hierarchical model

The depth of the hierarchy used to generate the data distribution is, like noise, a troublesome factor to study. This is mainly because it is difficult to determine the correct scale for analysis. For instance, a given clustering partition of a data distribution with 5 hierarchy levels could have correctly identified the clusters on the second-last level and, thus,



(a) (b) (c) (d)

Figure 6.9: *Visual examples of generated data for each parametrization: (a) a 2-level parametrization, (b) a 3-level parametrization, (c) a 4-level parametrization, and (d) a 5-level parametrization.*

Parameter	2-level	3-level	4-level	5-level
N_S (level 1)	128	8	4	2
N_S (level 2)	-	16	4	4
N_S (level 3)	-	-	8	4
N_S (level 4)	-	-	-	4
Nº clusters	128	128	128	128
N_S (last level)	$\mathcal{U}(40, 80)$			

Table 6.9: *Parametrization for the hierarchy depth experiment. All databases contain the same final number of clusters (128).*

its comparison with the true labels would be diminished.

For this experiment, we have studied how the algorithms adapt to different depths in data generation. To this end, 4 parametrizations with a sequentially increased number of hierarchy levels were fixed: (1) a 2-level parametrization, (2) a 3-level parametrization, (3) a 4-level parametrization, and (4) a 5-level parametrization. The rest of the parameters were set to their default values. An example of a dataset with each configuration is shown in Fig.6.9

To control the depth, the *SynFlex* generator offers the possibility of directly adding cluster levels over the existing ones. However, in order to be able to compare the scenarios, the same number of clusters on the last level generated is kept the same. Table 6.9 reflects this parametrization in terms of the hierarchy depth and the number of elements per level (N_S).

Fig. 6.10 shows the results obtained by the selected algorithm pool in the databases for each parametrization. Note that a seventh algorithm is included in the comparison: *VDBSCAN* (without *KDBSCAN*). This is done to further reflect the usefulness of *KDBSCAN* in identifying different regions of interest before applying *VDBSCAN* (see Chapter 3). For the rest of the experiments, this comparison is unnecessary, since there is only 2 levels of hierarchy and, therefore, there were no super-cluster divisions.

Indeed, when observing the results, a clear difference is observed between *K+VDBSCAN*

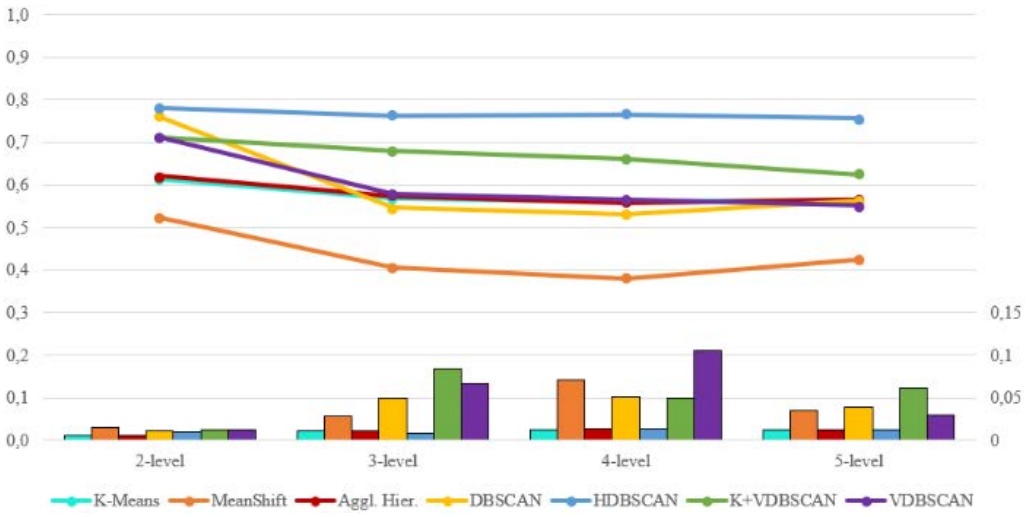


Figure 6.10: *Average results for the depth experiment in terms of AMI (left axis). The bars represent the variation in the results through the 10 instances for each parametrization, i.e., their height is the value of the standard deviation (right axis).*

and *VDBSCAN*, with the performance of the latter dropping when depth is increased (from 0.71 to 0.58), whereas the former only suffers from a slight decrease (from 0.71 to 0.68). Despite this fact, *HDBSCAN* remains to be the most robust against hierarchy depth changes, maintaining its results around a decent margin ($0.75 < AMI < 0.79$).The remaining algorithms decrease their performance with the increased depth, with the exception of Mean-Shift and DBSCAN, which experiment a slight increase for the last parametrization (5-level hierarchy depth). In the case of DBSCAN, however, it is worth mentioning that this slight increase (from 0.53 to 0.56) is not enough to recover from its initial drop (0.76 to 0.54 from 2-level to 3-level hierarchy depth).

Dimensionality experiment

In this document, we are mostly studying the behaviour of algorithms in two-dimensional environments. However, data often presents more than 2 dimensions, which may affect the behaviour of clustering techniques.

The *SynFlex* generation system allows us to choose the dimensionality of the data with a single parameter D. For this experiment, 4 parametrizations were set with increasing dimensionality: (1) a 2-D parametrization, (2) a 3-D parametrization, (3) a 4-D parametrization, and (4) a 5-D parametrization. The rest of the parameters were set to their default values.

The performance of the selected algorithms for the discussed parametrizations is shown in Fig. 6.11. As it can be observed, an increase in dimensionality does not have a strong effect in most of the algorithms. A slight performance decrease can be observed for Mean-Shift in 4 and 5 dimensions, while K-Means and Agglomerative Hierarchical

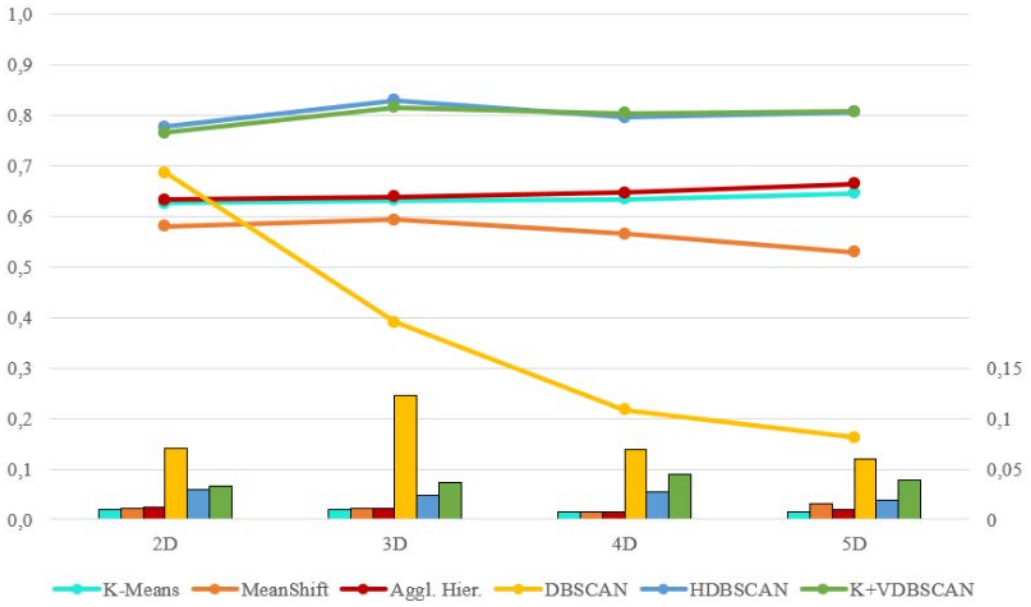


Figure 6.11: *Average results for the dimensionality experiment in terms of AMI (left axis). The bars represent the variation in the results through the 10 instances for each parametrization, i.e., their height is the value of the standard deviation (right axis).*

experiment a slight progressive increase, always under the $AMI = 0.7$ barrier. In *HDBSCAN* and *K+VDBSCAN*, slim improvements are observed for 3D and 5D, but all results are circling a 0.8 value for the AMI. The only exception is DBSCAN, which suffers a steep performance loss with every increase in dimensionality, reaching an AMI value under 0.2 for 5D. This suggests that, as dimensionality increases, the effect of density variations becomes more relevant (i.e., a single value of ε is unable to correctly identify the clusters). Therefore, the obtained results support the idea that the modifications of the original DBSCAN algorithm for *HDBSCAN* and *K+VDBSCAN* not only serve the purpose of adapting it to variable densities, but also make them more robust against dimensionality changes.

Difficulty experiment

After analyzing the performance impact obtained from changes in the five discussed factors, we present this additional experiment studying the generic concept of *difficulty*. Four parametrizations were defined with an increasing difficulty by changing some of the parameters from their default values to values that have posed complications in the previous experiments. Table 6.10 defines these parametrizations. The parameters that do not appear in the table were set to their default values.

Once more, the performance of the selected algorithms for the discussed parametrizations is shown in Fig. 6.12. As it is expected, the harder the scenario, the worse the performance. Like it happened with the noise experiment, the modified density-based clustering algorithms (*HDBSCAN*, *K+VDBSCAN*) are the only ones that hold AMI results over 0.6

Parameter	Trivial	Easy	Moderate	Hard
Depth	2	2	3	3
f_λ	0.9	$\mathcal{U}(0.05, 1)$	$\mathcal{U}(0.05, 1)$	$\mathcal{U}(0.05, 1)$
ENR (*level* 1)	∞	0.04	0.025	0.005
N_S (*level* 1)	32	32	4	4
ENR (*level* 2)	-	-	0.025	0.005
N_S (*level* 2)	-	-	8	8
N_S (*lastlevel*)	60	$\mathcal{U}(40, 80)$	$\mathcal{U}(40, 80)$	$\mathcal{U}(40, 80)$
α	-100	-100	$\mathcal{U}(-100, +100)$	$\mathcal{U}(-100, +100)$
β	0	0	$\mathcal{U}(0, 1)$	$\mathcal{U}(0, 1)$

Table 6.10: *Values given to the SynFlex parameters for the difficulty experiment. Unless it is specified otherwise, the parameter value is the same for all hierarchy levels.*

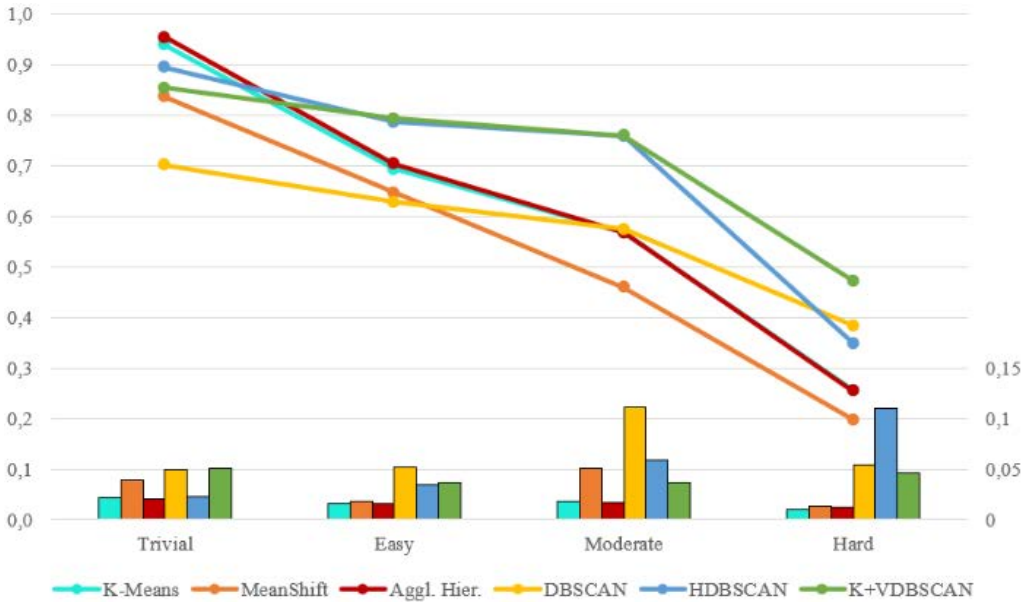


Figure 6.12: *Average results for the difficulty experiment in terms of AMI (left axis). The bars represent the variation in the results through the 10 instances for each parametrization, i.e., their height is the value of the standard deviation (right axis).*

until the last parametrization (hard scenario). This further supports the point of noise being one of the most influential analyzed factors. However, the value of *ENR* in the stronger noise parametrization ($ENR = 0.005$) is also used in the hard parametrization of this experiment, and while there is still a dramatic performance drop for every algorithm, modified density-based clustering algorithms suffer less in this experiment than in the one depicted in Section 6.4.2. Particularly, *K+VDBSCAN* yields an $AMI = 0.25$ for the noise experiment, whereas the hard parametrization of this experiment yields an $AMI = 0.47$. This result is probably caused by the increased dimensionality, since it matches with the one obtained in the dimensionality experiment, where modified density-based algorithms experimented a slight increase in performance for the 3D databases. Additionally, DBSCAN suffers from an even greater performance drop in this experiment from the third parametrization (moderate scenario) than in the noise experiment, which

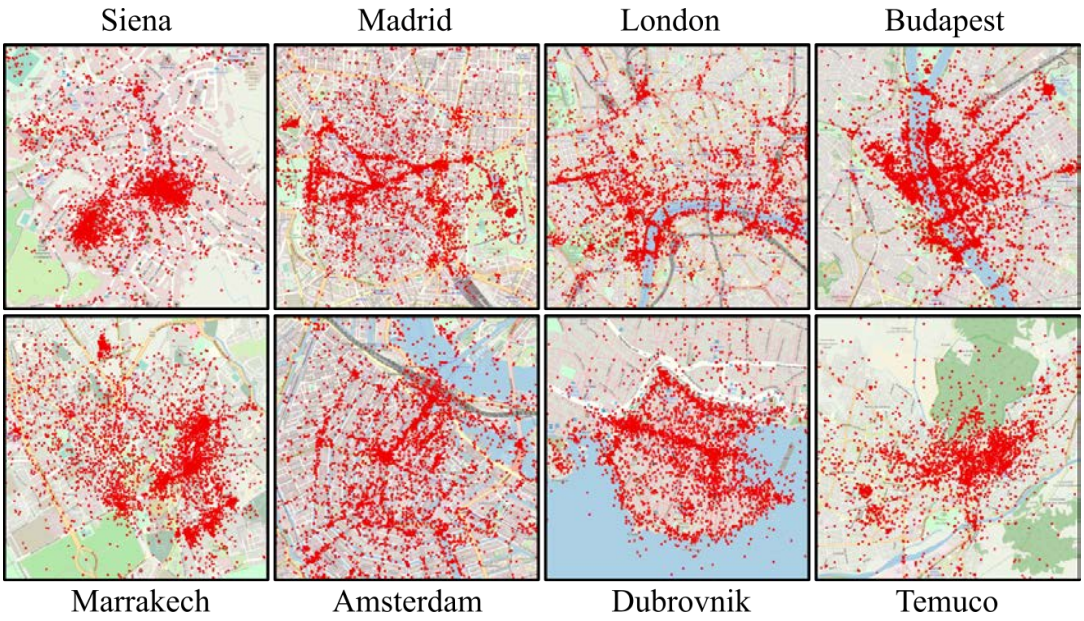


Figure 6.13: *Representation of each sample space over their corresponding location's map. Note that each location is representing in the scale that favoured its visualization.*

also matches with the significant decrease in performance in the dimensionality experiment when $D > 2$.

Real-World experiment

To conclude this benchmark, we have also explored the behaviour of the algorithms in several real-world databases. Similar to the *Getafe* dataset and the ones used in Chapter 4, these databases were obtained from the Flickr web platform, and they represent geographical coordinates from 7 different real regions: (1) Siena (Italy), (2) Madrid (Spain), (3) London (United Kingdom), (4) Budapest (Hungary), (5) Marrakech (Morocco), (6) Amsterdam (The Netherlands), (7) Dubrovnik (Croatia) and (8) Temuco (Chile). These represent areas of diverse nature attending to factors like population densities, tourism, shape and size. For the purposes of this experiment, the assembled databases have default sizes depending on the size of the region. Particularly, small regions (Temuco, Siena and Dubrovnik) contain 4,000 samples; while the rest, which are sections of larger cities, contain 6,000 samples. A map of each location is displayed in Fig 6.13. Let us remember that these databases, as well as the ones in the previous experiments, have been made publicly available [11] for further research.

It is important to remember that these databases were unlabeled and, therefore, cannot be evaluated using a external metric like the Adjusted Mutual Information score. Therefore, we have used of the proposed *FLAG* index (Section 5.2) to assess their performance. These results are shown in Fig. 6.14. Let us remember that lower FLAG values represent

[11] https://github.com/plasavall/benchmark

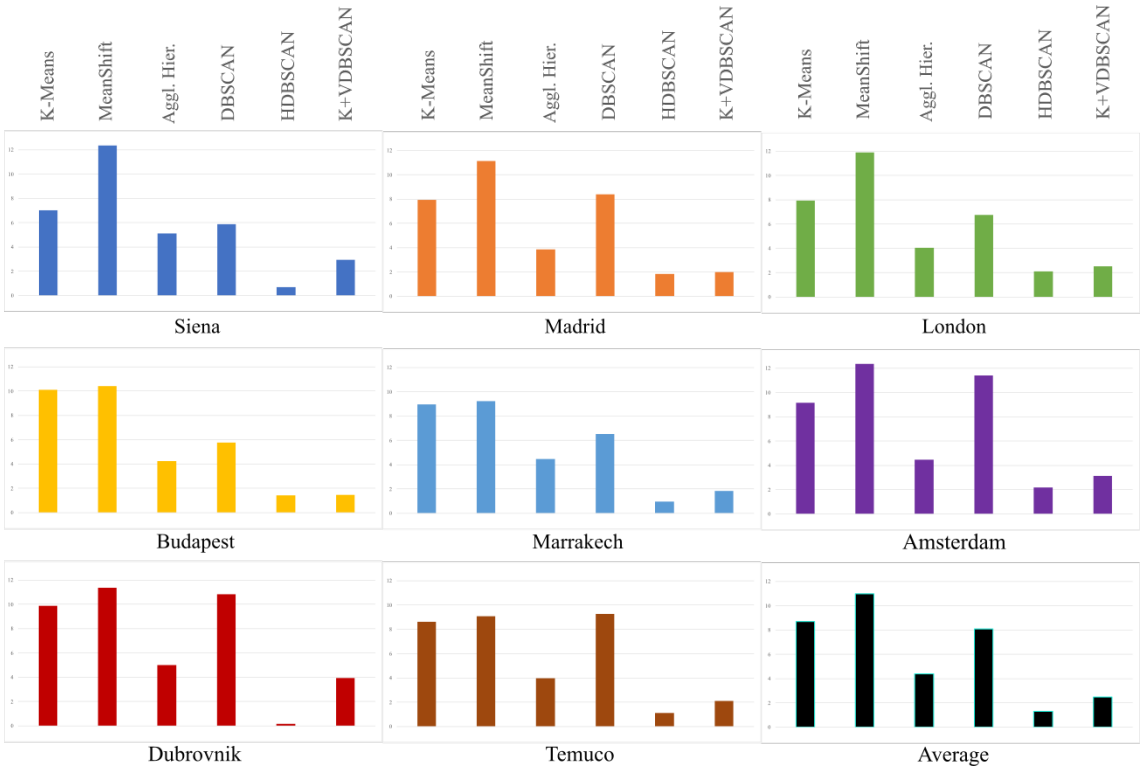


Figure 6.14: *Results for the Real-World experiment in terms of FLAG for each tested region.*

better performances (see Section 5.2).

One can clearly observe that Mean-Shift systematically offers the worst performance results, returning values of $FLAG > 9$ and often going over the 11 threshold, for an average $FLAG = 10.98$. This is in accordance with the results of the previous, supervised experiments, where Mean-Shift was often the last contender in the comparisons. Likewise, K-Means has similar results, yielding marginally better results than Mean-Shift, but yielding second-last best results in most scenarios (average $FLAG = 8.70$). The exception to this is, once again, DBSCAN, which offers moderately worse results than K-Means in Madrid, Amsterdam, Dubrovnik and Temuco (average $FLAG = 9.97$) but significantly better results for the rest of the locations (average $FLAG = 6.23$). This erratic behaviour was observed in the supervised experiments as well, where DBSCAN usually obtained high standard deviations in the results of the 10 databases for each parametrization, suggesting a high variability in the performance even in supposedly similar scenarios.

An interesting contrast with respect to the supervised experiments is the results obtained by the Agglomerative Hierarchical clustering, which usually ranked 4^{th} under DBSCAN in said experiments, but obtains stable, decent performance results here, averaging a $FLAG = 4.39$ for all the analyzed real world scenarios. One of the reasons that could cause this is the inability to identify noise samples by this algorithm, which has a penalizing effect when labels are available, but is not as significant with internal metrics.

Once again, *HDBSCAN* and *K+VDBSCAN* yield the best results in every scenario, with the former performing slightly better than the latter (an average $FLAG = 1.31$ versus $FLAG = 2.49$).

This experiment is useful to further analyze the behaviour of algorithms in scenarios with variable density and clusters with arbitrary shapes and sizes. Of course, as we discuss in Section 7.2, it would be interesting to study this using fully labeled databases, but obtaining them is not trivial, so the availability of an internal metric with this level of correlation with the external ones (see Section 6.3) for these scenarios is a substantially up to the mark alternative.

6.4.3. Conclusions about the experiments using the benchmark

Finding the appropriate algorithm to perform a certain task is challenging. Through the experiments described in the previous section, we have seen that different algorithms obtain varying performances depending on the scenario. In this last section of the chapter, we gather some conclusions that may help a potential user decide which algorithm to choose based on the results obtained in the proposed benchmark.

When the presence of noise is a factor for the application context, it is important to choose an algorithm that does not neglect this factor. For instance, algorithms like K-Means and Mean-Shift assign every sample to a category, so they are not a viable option when the databases have high levels of noise. On the other hand, density-based clustering algorithms are able to identify noise samples, so they are more suitable for this scenario.

Similarly, if the scenario requires the identification of clusters with arbitrary shapes, the use of density-based clustering techniques is highly recommended, since they are able to identify groups of points based on local density variations rather than distance to a centroid. Therefore, classic approaches that favour the identification of globular clusters (e.g., Mean-Shift) are not recommended in this case.

When analyzing the impact of dimensionality, we have observed that non density-based approaches, although not obtaining great results, remain stable as dimensionality increases. This, however, is not the case of DBSCAN, which experiments a steep decrease in its performance with the increase of dimensionality. As a result, in these cases we should go one step further and make use of advanced density-based clustering algorithms like *HDBSCAN* and *K+VDBSCAN*, which are more robust against the increase in dimensionality.

To conclude, the experiment carried out using real-world databases for landmark discovery revealed that the use of advanced density-based clustering algorithms is essential in such a scenario, since it combines many problematic characteristics, namely: the presence of noise, varying inter-cluster density and arbitrary shape clusters.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1. Conclusions

One of the main goals of this work was to find clustering algorithms able to adapt to data distributions with complex density variations. In this dissertation, we proposed two novel density-based clustering algorithms: 1) *KDBSCAN* 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) *VDBSCAN* is designed to exploit data distributions where density decreases radially from a center of mass.

Additionally, 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 in the challenge of building more sustainable tourism models in popular destinations, as well as attracting 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 (the dataset obtained from Flickr). First, *KDBSCAN* was used to discriminate independent conurbations (*Places of Interest*) within a certain region. After that, we applied *VDBSCAN* 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 discovery (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 noticing that when algorithms *KDBSCAN* and *VDBSCAN* were designed, certain considerations were taken into account, the main one being the availability of metadata in the samples (i.e., descriptive tags) to be combined with the geographical information. Nevertheless, later experiments (Chapter 6) showed that, in the absence of this additional metadata, *HDBSCAN* often proved to be a more effective method for landmark discovery. This emphasises the relevance of the proposed combined distance and,

furthermore, opens up potential lines of research that will be discussed in the next section.

Another important problem considered in this thesis was the lack of proper evaluation techniques in the context of density-based clustering. To tackle this issue, we have proposed a synthetic data generation system with flexible parameters (*SynFlex*) that allows to model different characteristics regarding the nature of the data (density, shape, dimensionality, hierarchy, noise, etc.). This system was proven to be able to flexibly produce realistic datasets in the context of landmark discovery, but its high flexibility in the parametrization makes it potentially applicable to a wide range of fields. Also related to the evaluation demand, an internal index based on intra-cluster fluctuation and inter-cluster agglomeration (*FLAG*) was proposed. This index showed significant correlation with the external metric used to evaluate annotated data in this context (Adjusted Mutual Information), outperforming other generic, well-known indices (Dunn, Davies-Bouldin, Calinski-Harabasz and Silhouette) as well as some task-specific metrics (SV, DBCV).

To conclude, these two contributions were combined to build a new evaluation benchmark for density-based clustering. The benchmark consists of real and synthetic data gathered with the goal of providing a systematic evaluation of the performance of clustering algorithms regarding five crucial factors: shape, density, noise, hierarchy and dimensionality. A set of well-known clustering algorithms have been assessed using the benchmark, and the obtained results outlined some of their strengths and weaknesses. Particularly, noise was proven to be a dominant factor when analyzing the algorithms' performance, with DBSCAN and its modifications (*HDBSCAN* and *K+VDBSCAN*) being less affected by its presence. Also worth mentioning is the swap in performance at the top of the ranking with respect to the experiments on landmark discovery (Chapter 4). In those experiments, we were dealing with spatial and textual features and, while *K+VDBSCAN* showed a better performance when additional textual information was available, *HDBSCAN* performed slightly better in most generic scenarios with only spatial information.

In summary, we have demonstrated the usefulness of our benchmark and we firmly believe that it will become a powerful tool to support the development of new algorithms and techniques in the field of density-based clustering.

7.2. Future work

Although the research conducted for this dissertation allowed us to extract some significantly important conclusions, there are still some potential lines of research that could further enhance the impact of this thesis.

It is worth mentioning that the system presented and tested in Chapters 3 and 4, respectively, was designed to be integrated to another system, which would 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. The technical background to achieve this system is further explored in [120], [121].

Nonetheless, there are still a few lines of work that could be explored regarding the research performed for the development of density-based clustering algorithms. 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 time-consuming, so this should be done only if an exceptionally robust input for the subsequent processing blocks of the project was required. Additionally, some blocks proposed in our system could be used for alternative tasks. Particularly, *KDBSCAN*'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.

On the other hand, regarding the research on evaluation techniques discussed in Chapters 5 and 6, the main line of research that we identify is the inclusion of additional experiments to the evaluation benchmark. Currently, regarding real-world data, only spatial datasets are gathered to perform testing, and even though synthetic data offers a significant sense of generality, it would undoubtedly be positive to explore other real-world data-types. Furthermore, the synthetic data generation for the experiments in Chapter 6 was designed to take into account the factors that affect density-based clustering algorithms, but additional synthetic experiments could be proposed to analyze other type of behaviours, shifting the paradigm and centering the analysis in supervised learning (studying classification or regression algorithms, for instance) or on alternative types of unsupervised learning algorithms.

Furthermore, the experiments on the benchmark also showed that, in the absence of additional metadata, *HDBSCAN* proved to be more competitive than other clustering alternatives, so it would be interesting to pursue a line of research that combined this method with complementary algorithms, such as the proposed *KDBSCAN*, to further improve its performance.

Finally, we are confident that the proposed benchmark can significantly boost research in the field, as it offers a fair, systematic and complete evaluation mechanism. For instance, deep learning models for metric learning, very useful for clustering tasks, could be trained using large, labeled, synthetic datasets generated using *SynFlex*. Furthermore, the benchmark can be the basis to approach scenarios where algorithms provide poor performances (e.g., scenarios with a strong noise presence) and attempt to design and develop new clustering algorithms.

BIBLIOGRAPHY

[1] E. Pla-Sacristán, I. Gonzalez-Diaz, T. Martinez-Cortes, and F. Diaz-de-Maria, "Finding landmarks within settled areas using hierarchical density-based clustering and meta-data from publicly available images," *Expert Systems with Applications*, vol. 123, pp. 315–327, 2019.

[2] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[3] H. E. Driver and A. L. Kroeber, *Quantitative expression of cultural relationships*, 4. Berkeley: University of California Press, 1932, vol. 31.

[4] M. Ester, "Density-based clustering," *Data Clustering*, pp. 111–127, 2018.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, 1996, pp. 226–231.

[6] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[7] P. Viswanath and V. S. Babu, "Rough-dbscan: A fast hybrid density based clustering method for large data sets," *Pattern Recognition Letters*, vol. 30, no. 16, pp. 1477–1488, 2009.

[8] X. Chen, W. Liu, H. Qiu, and J. Lai, "Apscan: A parameter free algorithm for clustering," *Pattern Recognition Letters*, vol. 32, no. 7, pp. 973–986, 2011.

[9] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, 2008, pp. 312–321.

[10] W. Chen, M. Ji, and J. Wang, "T-dbscan: A spatiotemporal density clustering for gps trajectory segmentation.," *International Journal of Online Engineering*, vol. 10, no. 6, 2014.

[11] J. Yang, Y. Guo, Z. Yang, L. Yang, and S. Xie, "Estimating number of speakers via density-based clustering and classification decision," *IEEE Access*, vol. 7, pp. 176 541–176 551, 2019.

[12] Q. Ye, W. Gao, and W. Zeng, "Color image segmentation using density-based clustering," in *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, IEEE, vol. 2, 2003, pp. II–401.

[13] W. He and Z. Liu, "Motion pattern analysis in crowded scenes by using density based clustering," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, 2012, pp. 1855–1858.

[14] E. Bernard, P. Naveau, M. Vrac, and O. Mestre, "Clustering of maxima: Spatial dependencies among heavy rainfall in france," *Journal of Climate*, vol. 26, no. 20, pp. 7929–7937, 2013.

[15] I. El-Feghi, M. A. Sid-Ahmed, and M. Ahmadi, "Automatic localization of craniofacial landmarks for assisted cephalometry," *Pattern Recognition*, vol. 37, no. 3, pp. 609–621, 2004.

[16] B. Elias, "Extracting landmarks with data mining methods," in *International Conference on Spatial Information Theory*, Springer, 2003, pp. 375–389.

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

[18] I. Cenamor, T. de la Rosa, S. Núñez, and D. Borrajo, "Planning for tourism routes using social networks," *Expert Systems with Applications*, vol. 69, pp. 1–9, 2017.

[19] L. Cao, J. Luo, A. C. Gallagher, X. Jin, J. Han, and T. S. Huang, "A worldwide tourism recommendation system based on geotaggedweb photos.," in *ICASSP*, Citeseer, 2010, pp. 2274–2277.

[20] J. Tang and L. Meng, "Learning significant locations from gps data with time window," in *Geoinformatics 2006: GNSS and Integrated Geospatial Applications*, International Society for Optics and Photonics, vol. 6418, 2006, 64180J.

[21] W. Feng *et al.*, "Streamcube: Hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, IEEE, 2015, pp. 1561–1572.

[22] V. Frias-Martinez, V. Soto, H. Hohwald, and E. Frias-Martinez, "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)*, IEEE, 2012, pp. 239–248.

[23] S. Papadopoulos, C. Zigkolis, Y. Kompatsiaris, and A. Vakali, "Cluster-based landmark and event detection for tagged photo collections," *IEEE MultiMedia*, vol. 18, no. 1, pp. 52–63, 2011.

[24] Y.-T. Zheng *et al.*, "Tour the world: Building a web-scale landmark recognition engine," in *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, IEEE, 2009, pp. 1085–1092.

[25] I. Lee, G. Cai, and K. Lee, "Exploration of geo-tagged photos through data mining approaches," *Expert Systems with Applications*, vol. 41, no. 2, pp. 397–405, 2014.

[26] S. Kisilevich, F. Mansmann, and D. Keim, "P-dbscan: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geotagged photos," in *Proceedings of the 1st international conference and exhibition on computing for geospatial research & application*, ACM, 2010, p. 38.

[27] R. Ji, Y. Gao, B. Zhong, H. Yao, and Q. Tian, "Mining flickr landmarks by modeling reconstruction sparsity," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 7, no. 1, p. 31, 2011.

[28] X. Zhou, C. Xu, and B. Kimmons, "Detecting tourism destinations using scalable geospatial analysis based on cloud computing platform," *Computers, Environment and Urban Systems*, vol. 54, pp. 144–153, 2015.

[29] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[30] S. Louhichi, M. Gzara, and H. Ben-Abdallah, "Unsupervised varied density based clustering algorithm using spline," *Pattern Recognition Letters*, vol. 93, pp. 48–57, 2017.

[31] M. Zhang, "Forward-stagewise clustering: An algorithm for convex clustering," *Pattern Recognition Letters*, vol. 128, pp. 283–289, 2019.

[32] U. Von Luxburg, R. C. Williamson, and I. Guyon, "Clustering: Science or art?" In *Proceedings of ICML workshop on unsupervised and transfer learning*, JMLR Workshop and Conference Proceedings, 2012, pp. 65–79.

[33] P. Franti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, no. 12, pp. 4743–4759, 2018.

[34] P. Zacharis, G. West, G. Dobie, T. Lardner, and A. Gachagan, "Data-driven analysis of ultrasonic inspection data of pressure tubes," *Nuclear Technology*, vol. 202, no. 2-3, pp. 153–160, 2018.

[35] H. Mittal, A. C. Pandey, M. Saraswat, S. Kumar, R. Pal, and G. Modwel, "A comprehensive survey of image segmentation: Clustering methods, performance parameters, and benchmark datasets," *Multimedia Tools and Applications*, pp. 1–26, 2021.

[36] M. P. Sinka and D. W. Corne, "A large benchmark dataset for web document clustering," *Soft computing systems: design, management and applications*, vol. 87, pp. 881–890, 2002.

[37] H. Homburg, I. Mierswa, B. Möller, K. Morik, and M. Wurst, "A benchmark dataset for audio classification and clustering.," in *ISMIR*, vol. 2005, 2005, pp. 528–31.

[38] M. C. de Souto, A. L. Coelho, K. Faceli, T. C. Sakata, V. Bonadia, and I. G. Costa, "A comparison of external clustering evaluation indices in the context of imbalanced data sets," in *2012 Brazilian Symposium on Neural Networks*, IEEE, 2012, pp. 49–54.

[39] K. Kishida, "Empirical comparison of external evaluation measures for document clustering by using synthetic data," IPSJ SIG Technical Report, Tech. Rep., 2014.

[40] R. M. Aliguliyev, "Performance evaluation of density-based clustering methods," *Information Sciences*, vol. 179, no. 20, pp. 3583–3602, 2009.

[41] G. O. Campos *et al.*, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," *Data mining and knowledge discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[42] M. Hassani and T. Seidl, "Using internal evaluation measures to validate the quality of diverse stream clustering algorithms," *Vietnam Journal of Computer Science*, vol. 4, no. 3, pp. 171–183, 2017.

[43] I. Gurrutxaga, J. Muguerza, O. Arbelaitz, J. M. Pérez, and J. I. Martín, "Towards a standard methodology to evaluate internal cluster validity indices," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 505–515, 2011.

[44] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[45] M. G. Kendall, "Discrimination and classification," *Multivariate analysis*, vol. 1, 1966.

[46] V. Estivill-Castro, "Why so many clustering algorithms: A position paper," *ACM SIGKDD explorations newsletter*, vol. 4, no. 1, pp. 65–75, 2002.

[47] S. K. Uppada, "Centroid based clustering algorithms—a clarion study," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 6, pp. 7309–7313, 2014.

[48] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[49] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[50] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[51] M. Chegini, J. Bernard, P. Berger, A. Sourin, K. Andrews, and T. Schreck, "Interactive labelling of a multivariate dataset for supervised machine learning using linked visualisations, clustering, and active learning," *Visual Informatics*, vol. 3, no. 1, pp. 9–17, 2019.

[52] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & knowledge engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[53] A. Smiti and Z. Elouedi, "Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques," in *2012 IEEE 16th international conference on intelligent engineering systems (INES)*, IEEE, 2012, pp. 573–578.

[54] Chire, *Dbscan illustration — Wikipedia, the free encyclopedia*, [Accessed 25-June-2021], 2011. [Online]. Available: https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg.

[55] J. Esmaelnejad, J. Habibi, and S. H. Yeganeh, "A novel method to find appropriate epsilon for dbscan," in *Asian Conference on Intelligent Information and Database Systems*, Springer, 2010, pp. 93–102.

[56] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[57] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2013, pp. 160–172.

[58] Chire, *Optics — Wikipedia, the free encyclopedia*, [Accessed 25-June-2021], 2011. [Online]. Available: https://commons.wikimedia.org/wiki/File:OPTICS.svg.

[59] A. Ram, S. Jalal, A. Jalal, and M. Kumar, "A density based algorithm for discovery density varied cluster in large spatial databases," *International Journal of Computer Application*, vol. 3, no. 6, Jun. 2010. DOI: 10.5120/739-1038.

[60] A. Hinneburg, D. A. Keim, *et al.*, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, vol. 98, 1998, pp. 58–65.

[61] H. Rehioui, A. Idrissi, M. Abourezq, and F. Zegrari, "Denclue-im: A new approach for big data clustering," *Procedia Computer Science*, vol. 83, pp. 560–567, 2016.

[62] A. Hinneburg and H.-H. Gabriel, "Denclue 2.0: Fast clustering based on kernel density estimation," in *International symposium on intelligent data analysis*, Springer, 2007, pp. 70–80.

[63] O. R. Zaiane and C.-H. Lee, "Clustering spatial data in the presence of obstacles: A density-based approach," in *Proceedings International Database Engineering and Applications Symposium*, IEEE, 2002, pp. 214–223.

[64] Z. Ghaemi and M. Farnaghi, "A varied density-based clustering approach for event detection from heterogeneous twitter data," *ISPRS International Journal of Geo-Information*, vol. 8, no. 2, p. 82, 2019.

[65] G. H. Putri *et al.*, "Chronoclust: Density-based clustering and cluster tracking in high-dimensional time-series data," *Knowledge-Based Systems*, vol. 174, pp. 9–26, 2019.

[66] R. Tron, X. Zhou, C. Esteves, and K. Daniilidis, "Fast multi-image matching via density-based clustering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4057–4066.

[67] M. E. Celebi, Y. A. Aslandogan, and P. R. Bergstresser, "Mining biomedical images with density-based clustering," in *International conference on information technology: coding and computing (ITCC'05)-volume II*, IEEE, vol. 1, 2005, pp. 163–168.

[68] R. Amami and A. Smiti, "An incremental method combining density clustering and support vector machines for voice pathology detection," *Computers & Electrical Engineering*, vol. 57, pp. 257–265, 2017.

[69] C. C. Yang and T. D. Ng, "Analyzing and visualizing web opinion development and social interactions with density-based clustering," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 6, pp. 1144–1155, 2011.

[70] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 672–677.

[71] J.-O. Palacio-Niño and F. Berzal, "Evaluation metrics for unsupervised learning algorithms," *arXiv preprint arXiv:1905.05667*, 2019.

[72] J. W. Anderson, K. Kennedy, L. B. Ngo, A. Luckow, and A. W. Apon, "Synthetic data generation for the internet of things," in *2014 IEEE International Conference on Big Data (Big Data)*, IEEE, 2014, pp. 171–176.

[73] K. Houkjær, K. Torp, and R. Wind, "Simple and realistic data generation," in *Proceedings of the 32nd international conference on Very large data bases*, 2006, pp. 1243–1246.

[74] G. Soltana, M. Sabetzadeh, and L. C. Briand, "Synthetic data generation for statistical testing," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2017, pp. 872–882.

[75] J. Dahmen and D. Cook, "Synsys: A synthetic data generation system for healthcare applications," *Sensors*, vol. 19, no. 5, p. 1181, 2019.

[76] E. Lundin, H. Kvarnstrom, and E. Jonsson, "A synthetic fraud data generation methodology," in *International Conference on Information and Communications Security*, Springer, 2002, pp. 265–277.

[77] M. A. Whiting, J. Haack, and C. Varley, "Creating realistic, scenario-based synthetic data for test and evaluation of information analytics software," in *Proceedings of the 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*, 2008, pp. 1–9.

[78] Y. Pei and O. Zaiane, "A synthetic data generator for clustering and outlier analysis," 2006.

[79] R. J. Campello, "A fuzzy extension of the rand index and other related indexes for clustering and classification assessment," *Pattern Recognition Letters*, vol. 28, no. 7, pp. 833–841, 2007.

[80] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[81] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *International conference on artificial neural networks*, Springer, 2009, pp. 175–184.

[82] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[83] G. W. Milligan and M. C. Cooper, "A study of the comparability of external criteria for hierarchical cluster analysis," *Multivariate behavioral research*, vol. 21, no. 4, pp. 441–458, 1986.

[84] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American statistical association*, vol. 78, no. 383, pp. 553–569, 1983.

[85] S. Wagner and D. Wagner, *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[86] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

[87] J. Kreer, "A question of terminology," *IRE Transactions on Information Theory*, vol. 3, no. 3, pp. 208–208, 1957.

[88] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[89] I. Priness, O. Maimon, and I. Ben-Gal, "Evaluation of gene-expression clustering via mutual information distance measure," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–12, 2007.

[90] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *arXiv preprint arXiv:1110.2515*, 2011.

[91] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[92] N. Zahid, M. Limouri, and A. Essaid, "A new cluster-validity for fuzzy clustering," *Pattern recognition*, vol. 32, no. 7, pp. 1089–1097, 1999.

[93] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *2010 IEEE international conference on data mining*, IEEE, 2010, pp. 911–916.

[94] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[95] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[96] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[97] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[98] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*, SIAM, 2014, pp. 839–847.

[99] K. R. Žalik and B. Žalik, "Validity index for clusters of different sizes and densities," *Pattern Recognition Letters*, vol. 32, no. 2, pp. 221–234, 2011.

[100] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter.," *Icwsm*, vol. 11, no. 2011, pp. 438–441, 2011.

[101] J. Lin and R. G. Cromley, "Inferring the home locations of twitter users based on the spatiotemporal clustering of twitter data," *Transactions in GIS*, vol. 22, no. 1, pp. 82–97, 2018.

[102] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "Exploiting semantic annotations for clustering geographic areas and users in location-based social networks.," *The social mobile web*, vol. 11, no. 2, 2011.

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

[104] L. Kennedy, M. Naaman, S. Ahern, R. Nair, and T. Rattenbury, "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*, ACM, 2007, pp. 631–640.

[105] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002. DOI: 10.1109/34.1000236. [Online]. Available: http://dx.doi.org/10.1109/34.1000236.

[106] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[107] K. Wagstaff, C. Cardie, S. Rogers, S. Schrodl, *et al.*, "Constrained k-means clustering with background knowledge," in *ICML*, vol. 1, 2001, pp. 577–584.

[108] F. Nielsen, "Hierarchical clustering," in *Introduction to HPC with MPI for Data Science*, Springer, 2016, pp. 195–211.

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

[110] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*, Second edition. Hoboken, New Jersey: John Wiley & Sons, Inc, 2015.

[111] M. Llobera, "Building past landscape perception with gis: Understanding topographic prominence," *Journal of Archaeological Science*, vol. 28, no. 9, pp. 1005–1014, 2001.

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

[113] S. Gerard and J. M. Michael, *Introduction to modern information retrieval*, 1983.

[114] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[115] C. D. Manning, P. Raghavan, and H. Schutze, "Evaluation in information retrieval," in *Introduction to Information Retrieval*. Cambridge University Press, 2008, pp. 139–161. DOI: 10.1017/CBO9780511809071.009.

[116] E. M. Rasmussen, "Clustering algorithms.," *Information retrieval: data structures & algorithms*, vol. 419, p. 442, 1992.

[117] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[118] C. Spearman, "Demonstration of formulae for true measurement of correlation," *The American journal of psychology*, pp. 161–169, 1907.

[119] J. Pinto da Costa and C. Soares, "A weighted rank measure of correlation," *Australian & New Zealand Journal of Statistics*, vol. 47, no. 4, pp. 515–529, 2005.

[120] T. Martínez-Cortés, "Training deep retrieval models with noisy datasets," Ph.D. dissertation, Universidad Carlos III de Madrid, 2021.

[121] T. Martínez-Cortés, I. González-Díaz, and F. Díaz-de-María, "Training deep retrieval models with noisy datasets: Bag exponential loss," *Pattern Recognition*, vol. 112, p. 107 811, 2021.