A New Methodology for Storing Consistent Fuzzy Geospatial Data in Big Data Environment

Besma Khalfi, Cyril de Runz, Sami Faiz, Herman Akdag

To cite this version:

Besma Khalfi, Cyril de Runz, Sami Faiz, Herman Akdag. A New Methodology for Storing Consistent Fuzzy Geospatial Data in Big Data Environment. IEEE transactions on big data, 2021, 7 (2), pp.468-482. 10.1109/TBDATA.2017.2725904. hal-01856186

HAL Id: hal-01856186
https://hal.science/hal-01856186
Submitted on 18 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A New Methodology for Storing Consistent Fuzzy Geospatial Data in Big Data Environment

Besma Khalfi, Cyril De Runz, Member, IEEE, Sami Faiz, and Akdag Herman, Member, IEEE

Abstract—In this era of big data, as relational databases are inefficient, NoSQL databases are a workable solution for data storage. In this context, one of the key issues is the veracity and therefore the data quality. Indeed, as with classic data, geospatial big data are generally fuzzy even though they are stored as crisp data (perfect data). Hence, if data are geospatial and fuzzy, additional complexities appear because of the complex syntax and semantic features of such data. The NoSQL databases do not offer strict data consistency. Therefore, new challenges are needed to be overcome to develop efficient methods that simultaneously ensure the performance and the consistency in storing fuzzy geospatial big data. This paper presents a new methodology that tackles the storage issues and validates the fuzzy spatial entities’ consistency in a document-based NoSQL system. Consequently, first, to better express the structure of fuzzy geospatial data in such a system, we present a logical model called Fuzzy GeoJSON schema. Second, for consistent storage, we implement a schema-driven pipeline based on the Fuzzy GeoJSON schema and semantic constraints.

Index Terms—Spatial databases, Data Storage Representations, Schema and subschema, Fuzzy set, imprecision, consistency, big data, NoSQL systems, JSON.

1 Introduction

Since the emergence of the era of big data, a classic consideration of the associated issues is the four Vs: volume, velocity, variety, and veracity. The last one is clearly less studied and is intrinsically linked with data quality issues, such as consistency, accuracy, imprecision, correctness, etc. In a geographic context, the significant growth of geospatial data is mainly due to the increasing volume of data produced from many sources, such as geo-located sensors, location-based social media data, and volunteer geographic information (VGI) [1]. This expansive volume, with varying formats, raises big data storage and big data analysis issues and poses additional challenges in verifying data quality [2]. To ensure a better quality of geographic data, it is important to consider its imperfect nature, study it, and integrate it into the analyzing process [3].

Because they are voluminous and heterogeneous, imperfections affect spatial data even more, in both quantitative and qualitative characteristics. This means adapting the classical representation methods, storage, and processing as well as giving special consideration to models using this data. This paper considers a type of imperfection called vagueness according to [4] in the GIS community and called imprecision in the fuzzy set theory community [5]. This paper does not consider error, uncertainty, ambiguity, inaccuracy, or incompleteness.

Geographic data are imprecise by nature; their qualification and quantification are either linked to approximations or to subjective assessments [6]. Based on fuzzy sets theory [7] used to represent imprecise data, many approaches have been suggested to model this kind of imperfection in relational databases [8].

In the relational database management system (RDBMS) context, some models have worked on extending the entity-relationship model [9] and others have focused on extending the object-based model [10]. For spatial data, F-Perceptory [11] is an approach that proposes the object modeling of imprecise spatio-temporal data in object/relational databases. It represents the fuzzy geographic data as a set of connected geographic objects with membership degrees. The approach defines a set of constraints that ensure the fuzzy object consistency. Consistency in RDBMS is maintained by integrity constraints. However, for big data environments, RDBMS are not scalable.

The real needs for scalability and handling data with performance have led to the appearance of NoSQL systems, such as MongoDB and Cassandra, for storing and analyzing big data. These systems have rapidly emerged and are widely used despite relying less on data schema and data quality. The NoSQL systems are non-relational systems that store and analyze data in various formats [12]. They are distributed systems that exceed the traditional relational systems capabilities in storage capacities, scaling level, and data heterogeneity [13]. The compensation of their scalability is the fact that, in general, NoSQL databases do not provide strict consistency [2] and are known only for BASE properties.

This paper focuses on the storage (structure and consistency) of fuzzy geospatial data in document-oriented NoSQL databases (big data environment). This paper does not consider the modeling and storing of fuzzy geospatial objects in relational databases as introduced in [8], or [11] or the fuzzy spatial data analysis as proposed in [14], [15], and [16].
Understanding and fixing the requirements of how the data will be stored and managed by the system, allows more chance of handling the data consistency and quality. Even the schema-less is an asset, as any system needs to make some assumptions about the data structure and add more code to manipulate this data, which implies a strong effect on data processing quality.

Our research goal is to succeed in benefiting from the best of both relational and NoSQL databases, namely, combining the flexibility and scalability of the NoSQL systems to store and process fuzzy spatial big data, with the possibility of imposing a data structure model, to ensure data consistency as found in the relational model. We propose a new methodology for storing fuzzy geospatial objects, which are consistent and thus correctly modeled to be compliant with big data processing. This methodology is based on the introduction of a logical schema and of a system that checks the consistency of data against the proposed schema.

The remainder of this paper is structured as follows. In the Context and Definitions section, we perform a general overview of fuzzy geospatial data and consistency issues in NoSQL systems. The Methodology section describes our solutions based on the Fuzzy GeoJSON schema and semantic constraints to validate fuzzy geospatial datasets in the document-oriented NoSQL databases. The Fuzzy GeoJSON schema is presented in the Fuzzy GeoJSON Schema section and semantic constraints are discussed in the Constraints section. The Detection Inconsistent Data Experiments section demonstrates the results, and the Performance Experiments section presents the performance results. The Discussion section discusses our proposed solutions and presents a comparative study. Finally, the Conclusions and Future Work section presents the conclusions and future work.

2 CONTEXT AND DEFINITIONS

First, geospatial data are characterized by the location information in a geo-referenced space, where each geographic entity is characterized by its shape (geometry) and location (spatial dimension). Geospatial data, as classic data, are imprecise by nature.

This section, first illustrates the imprecision in geospatial data using some examples. It introduces, then, the fuzzy set theory as well as the assumptions for the imprecise spatial data formalism. Finally, it discusses the consistency challenge in NoSQL databases.

2.1 Imprecision in Geospatial Data

In most existing databases, users consider data to be precise. Even if it is imperfect, it is modeled and stored under the assumption of crisp data. For this, users either eliminate it or make it artificially precise. This assumption gives a far-from-perfect vision of reality and seems inappropriate for many complex applications, such as geographic and environmental information systems whereby data are inherently imprecise or fuzzy, for example, the geographic object locations, the limits between regions, and some spatial relationships [17]. Rather than simplifying the models, the best solution is to represent the fuzziness more explicitly.

In the spatial context, geographers admit that data are imprecise; their qualification and quantification are either linked to approximations or to subjective assessments. In reality, populating geographic databases always ends with some imprecision.

To describe the imprecision of geospatial data, we present examples of fuzzy distribution for some spatial phenomena.

A first example is the continuous distribution of city centers. The center of a city is usually represented on a map, on a global scale, by point-type geometry. However, if we consider the fact that a city is expanding all the time, it becomes difficult to precisely indicate its center. For this, a representation that shows all the possible boundaries that form the center borders is needed. Therefore, instead of considering the center to be point geometry, it is represented by fuzzy point geometry. Under the fuzzy constraint, the distribution of the city center changes from unique coordinates to a possibility interval [11].

A second imprecision issue is how to determine where a forest begins (see Fig. 1). It is a classic spatial example, which studies the delineation of forest boundaries. How to determine the threshold of trees to consider the forest boundaries or how to represent the outlying trees as being partial members of the forest.

Therefore, for any given approximate model, various boundaries can be represented, and then we may have a distribution of the outlying trees. The real boundary values can be located at various positions. The forest is modeled as fuzzy polygon geometry instead of considering it polygon geometry.

In general, geospatial objects whose contours are easy to locate are often represented as points, lines, or surfaces. However, this structure type is not suited to the representation of objects or phenomena whose boundaries are imprecise.

2.2 Fuzzy Set Theory for Fuzzy Geospatial Data

Research in data imperfection affirm that the uncertainty can be resolved by statistical and probabilistic models while other imperfection aspects, such as missing information, imprecision, and use of natural language, cannot be solved by a probabilistic process. The fuzzy set theory presented by [7] is embedded in many approaches to deal with imperfect information, essentially for the imprecision quantification.

2.2.1 Basic Principles of Fuzzy Set Theory

In this section, we mention some basic concepts, definitions and important properties of fuzzy set theory, found in literature, which are relevant to the focus of this work.

![Fig. 1. Issues related to delineation of forests boundaries](image-url)
**Definition 1** (partial membership). Let $U$ be a universe of discourse whose elements/objects are denoted by $e$ and $E$ be a set in $U$ and $\mu_E$ be its membership function. The element $e$ has partial membership in $E$ and is assigned to a membership function that takes values in the interval $[0,1]$. Additionally, $E$ is formally defined by its membership function $\mu_E$ as follows:

$$\mu_E : U \rightarrow [0,1]$$

where for each $e \in U$ and $\mu_E(e)$ denotes the membership degree of $e$ to the set $E$.

The generalization of partial membership provides a means to define a fuzzy set. An example is depicted in Fig. 2.

**Definition 2** (fuzzy set). Let $U$ be a universe of discourse whose elements/objects are denoted by $e$ and $E$ be a set in $U$ and $\mu_E$ be its membership function. If all the elements $e$ in $E$ have a membership function that maps $U$ to the membership space $[0,1]$. Thus, $E$ is called a fuzzy set and $e$ is called a fuzzy element, whose value belongs to a fuzzy set. A fuzzy set $E$ is uniquely defined in $U$ by the following set of pairs:

$$E = \{(e, \mu_E(e))|e \in U\}$$

To further investigate a fuzzy set by considering the membership function, we may discretize it such that only a finite number of cuts need to be considered to represent the fuzzy set. Each cut is associated with a degree membership value. Fig. 3 gives an example of a discretized fuzzy set.

**Definition 3** ($\alpha$-cut). For any fuzzy set $E$ defined on $U$, an $\alpha$-cut $E_\alpha$ of $E$ is a crisp set of elements for which the value of the membership function $\mu_E$ of $E$ is greater than $\alpha$, that is:

$$E_\alpha = \{e|\mu_E(e) \geq \alpha, e \in U, 0 < \alpha \leq 1\}$$

By extension, the 0-cut of $E$ is equal to the support of $E$:

$$E_0 = \{e|\mu_E(e) > 0, e \in U\}$$

**Definition 4** (connected $\alpha$-cut). An $\alpha$-cut $E_\alpha$ is connected if and only if we cannot make a bipartition of $E_\alpha$ with two non-empty open spaces. That is, for each pair of points belonging to $E_\alpha$, there is a path connecting them included in $E_\alpha$. An example of a connected $\alpha$-cut is depicted in Fig. 4.

**Definition 5** (connected fuzzy set). A fuzzy set $E$ is connected if and only if for all $\alpha$ ($0 \leq \alpha \leq 1$), $E_\alpha$ is connected. An example of a connected fuzzy set is depicted in Fig. 5.

A means of comparing fuzzy sets is through their standardization. Thus, we may say that a fuzzy set is normalized if there is an element whose membership degree is equal to 1. In **Definition 6**, we introduce the height function that is essential to normalize fuzzy sets:

**Definition 6** (height of fuzzy set). The height of a fuzzy set $E$ is the maximum value of the membership function $\mu_E$:

$$H(E) = Max_{e \in U}(\mu_E(e))$$

**Definition 7** (normalized fuzzy set). A fuzzy set $E$ is said normalized if and only if $H(E) = 1$. That is, an $e$ exists whose membership degree is equal to 1 or that the cut $E_1$ is different from the empty set. The previous statement can be encoded as follows:

$$H(E) = 1 \Leftrightarrow \exists e \in U, \mu_E(e) = 1 \Leftrightarrow E_1 \neq \emptyset$$

In this work, the first approximation we consider is that fuzzy sets must be connected and normalized.

### 2.2.2 Assumptions for Imprecise Geospatial Data

Spatial objects can be represented in vector mode as a point, line, or polygon. To model geospatial objects precisely within a vector mode, the spatial distribution of objects must be discretely defined, which means a spatial feature with clearly defined boundaries. At a refined scale, rivers,
roads, lakes, agricultural land, and forests are examples of
geospatial objects; they are continuous in nature, but in gen-
eral, they are illustrated with arbitrarily defined boundaries,
which are normally fuzzy (see Fig. 6).

Based on the fuzzy set theory, using the membership
function and the \(\alpha\)-cut concept, as described in Fig. 7, each
geospatial object (point, line, or polygon) may be identified
as a set of features with a principal feature in the center
with membership degree equal to 1 surrounded by other
features with membership degree lower than 1. Therefore,
we provide in the following the feature definition:

**Definition 8** (feature). A feature is a clearly bounded
portion of space. It can have a zonal, linear, or point
geometry. Then, we consider \(\alpha_i\), with \(i \in [1, n]\), where \(n\) is
the number of \(\alpha\)-cuts chosen. In a spatial context, for each fuzzy
geospatial object \(E\), a feature \(G\) may take its geometry in
\{Point, Line, Polygon\} geometries.

Using the feature definition permit to give the fuzzy
geospatial object definition:

**Definition 9** (fuzzy geospatial object). Let us consider \(O\)
a fuzzy spatial object defined as a set of \(n\) \(\alpha\)-cuts. Each \(\alpha\)-cut
is defined as a feature associated with a membership degree
\(\alpha\). Therefore, \(O\) is considered as being a set of features with
their own membership degree values. A fuzzy spatial object
\(O\) has the definition as follows:

\[
O : \{O_{\alpha_i}, i \in [1, n], \alpha_i \in [0, 1]\} \Leftrightarrow \\
O : \{(G_i, \alpha_i), i \in [1, n], \alpha_i \in [0, 1]\}
\]

where \(\forall g \in G_i; \mu_O(g) \geq \alpha_i\). Here \(G_i\) is the geometry of
the feature \(i\) with \(\mu_O\) the membership function value that is
greater than \(\alpha_i\).

In vector mode, to model fuzzy geospatial objects, fuzzy
set theory and \(\alpha\)-cut concept are useful bases from which
we can discretize imprecise boundaries. Therefore, \(\alpha\)-cuts
allow storing different imprecision values on fuzzy sets,
presenting this imprecision on a scale of values between
impossible (\(\alpha = 0\)) and very possible (\(\alpha = 1\)).

Consistency requirements for fuzzy geospatial objects
modeling have led to completing the \(\alpha\)-cut definition with
additional constraints, which are related to some complex
properties and conditions concerning the set of features (\(\alpha\)-
cuts). The first constraint that must be met is to verify that
each fuzzy spatial object is composed of \(n\) features with at
least two features.

**Definition 10**. Let a fuzzy spatial object \(O\) defined by its
set of features \(\{O_{\alpha}\}\). The lower number of features is two. For
accuracy, the following condition is obtained:

\[
\min_{O_{\alpha} \in O} (O_{\alpha}) = 2
\]

The second constraint that must be considered is to verify that for each feature \(O_{\alpha}\), the fuzzy object geometry
must be a polygon type, except in the case of the fuzzy point
and fuzzy line objects. Let us see the examples below.

**Example 1.** Let us consider a fuzzy point \(P\) with its \(\alpha\)-cuts
\(P_{\alpha}\). Each feature geometry \(G(P_{\alpha})\) is defined as follows:

\[
G(P_{\alpha}) = \begin{cases} 
\text{Polygon} & \text{if } \alpha_i \neq 1 \\
\text{Point} & \text{if } \alpha_i = 1.
\end{cases}
\]

**Example 2.** Let us consider a fuzzy line \(L\) with its \(\alpha\)-cuts
\(L_{\alpha}\). Each feature geometry \(G(L_{\alpha})\) is defined as follows:

\[
G(L_{\alpha}) = \begin{cases} 
\text{Polygon} & \text{if } \alpha_i \neq 1 \\
\text{Line} & \text{if } \alpha_i = 1.
\end{cases}
\]

Thus, the following definition gives a rule to verify the
feature geometries.

**Definition 11** (feature geometry). In the general case, let us
consider a fuzzy object \(O\) with its \(O_{\alpha}\). Each feature geometry
\(G(O_{\alpha})\) is defined as follows:

\[
G(O_{\alpha}) = \begin{cases} 
\text{Point} & \text{if } G(O) = \text{Fuzzy Point and } \alpha_i = 1; \\
\text{Line} & \text{if } G(O) = \text{Fuzzy Line and } \alpha_i = 1; \\
\text{Polygon} & \text{Otherwise.}
\end{cases}
\]

The third constraint is added to verify that, for each
fuzzy object, its set of features \(O_{\alpha}\) forms a connected
and normalized fuzzy set, meaning that, first, the features are
related geometric shapes, therefore, for all features with
\(\alpha (0 \leq \alpha \leq 1)\), \(\{O_{\alpha}\}\) is connected (see Definition 3).
Second, the maximum membership degree must be equal to
1, this means that \(H(O) = \max_{O_{\alpha} \in O} (\mu(O_{\alpha})) = 1\). Finally,
regardless of the feature \(O_1\) with a degree \(\alpha_1\), all features
about the fuzzy set having a higher degree than \(\alpha_1\) must
be included in \(O_1\); we give the following definition of the
inclusion constraint:

**Definition 12** (included cuts). Let \(O\) be a fuzzy spatial
object. For each \((O_{\alpha_1} \in O, O_{\alpha_2} \in O)\) with \((G_1, \alpha_1)\) and
\((G_2, \alpha_2)\), respectively, the inclusion relation between \(O_{\alpha_1}\)
and \(O_{\alpha_2}\) is defined as follows:

\[
\text{IF } \alpha_1 \geq \alpha_2 \text{ Then } G_1 \subset G_2
\]

Using these considerations may determine a closer fuzzy
spatial objects modeling for the reality and has a significant
influence on the input data quality.
2.3 Data Consistency in NoSQL Databases

Big data appeared with the production of vast data that could not be dealt with by relational systems. It offers new efficient tools able to store, process, and analyze data. Big data are mainly defined by the 3Vs: volume, variety, and velocity. Veracity is an additional dimension that has been added to describe data integrity and quality [2].

Consistency is one of many dimensions that research work has focused on as well as accuracy, relevance, and completeness, to evaluate data quality in databases. In this case, many methods have been proposed, such as data validation and domain analysis ([18], [19], [20]).

Data consistency is a basic feature of relational databases and a challenge for NoSQL databases. Relational databases have been the leading model for data storage, retrieval, and management. They still offer the perfect solution where the data size is not big; one machine is sufficient to store data, and data consistency is mandatory. Today, NoSQL databases complement relational databases that tackle scalability and performance limits. The NoSQL databases are unavoidable data storage for big data due to the set of programming languages used for developing applications over these new systems.

Data consistency is one of the biggest challenges in managing non-relational databases because a schema-less design makes the storage consistency very doubtful and the access control requirement very complex. According to the CAP theorem defined by Eric Brewer, any database management system (DBMS) can satisfy only two of three properties, which are availability, consistency, and partitioning. The consistency property requires that the DBMS provides the same data version at the same time for all users. The availability property means that the DBMS should give data to users at any time. With the tolerance of the network partitions property, the system can be divided over many computers [21]. The NoSQL databases follow the CAP theorem [22] and have the BASE properties: basically available, soft-state, and eventual consistency; this means an application basically works all the time (basically available), does not have to be consistent all the time (soft-state), but will be in some known state eventually (eventual consistency). In general, NoSQL databases do not have data model restrictions. They do not need a fixed semantic and structure for data before storing it and do not require ACID3 constraints.

In the literature, several solutions intend to keep the relational model advantages with NoSQL databases. As a first option, some studies choose to keep structured data (objects and arrays) in a relational database, while unstructured data are stored in a NoSQL database. Others proposed more effective custom-developed solutions. Model and data migration, which are processed automatically or semi-automatically from relational models to NoSQL, are proposed in [23], [24]. The approach presented in [23] keeps the data model semantically identical to the original (relational), and all relationships are adequately represented without data loss or distortion. They defined an automatic and transparent data and model migration from MySQL to MongoDB. However, the authors of [25] proposed to alternate data storage and processing between a relational database and NoSQL database; they decided which system is better to run certain datasets, and they used two models for their solution.

Considering the significant data diversity in NoSQL databases, a migration solution from SQL to NoSQL does not seem to be a feasible option, at least given the high implementation costs [26]. Some new methodologies defined in [27], [28], and [29] have emerged to provide some improvements in the big data field. They propose to keep the consistency and ACID properties of RDBMS in a NoSQL-based environment. These solutions aimed to provide transaction ACID properties and consistency, but they do not address the problem of consistency storage and data structure. In [30], the authors explore the JSON format advantages with RDBMSs. They provide an automated mapping layer called Argo for storing and querying JSON data in a relational system. Argo presents a simple and intuitive schema for mapping a JSON data model to a relational schema. Self-describing data formats (like JSON format) are a satisfactory solution to store and transfer data, but even with their common use, many solutions, such as [26] underlined the usefulness of integrating metadata information and standardized definitions of domains mainly for specific variables and uncertainties.

The schema-less concept in NoSQL databases is proposed to overcome the relational data model inflexibility. It offers an unixed definition of the stored data in rows. Moreover, JSON is a text format for describing structured data [31]. It is the most popular format for data processing adopted by document-oriented NoSQL databases. The JSON document is built on two structures: a collection of name/value pairs and an ordered list of values. A schema language for JSON has been proposed in [32] to overcome the schema-less model weaknesses. It integrates syntax for table/objects and structured data and has emerged as a powerful tool for validating the structure of JSON data; it has been adopted by various databases like MongoDB.

In addition, the JSON schema offers many advantages: (i) the possibility to add more key-value properties in the document due to the Boolean property “additional-properties = true” and (ii) the possibility to modify the document structure so the user can act directly in schema.

For specific databases such as geospatial databases, to provide a consistent schema for representing fuzzy data in a relational structure, the relational data model has been extended using geographical primitives of points, lines, polygons, and other discretized continuous fields [33]. Many suggestions have been developed for modeling geographical data, like CONGOO, POLLEN, and PERCEPTORY, but they did not consider their fuzzy nature. More recently, based on Fuzzy UML and Perceptrory pictograms, a modeling approach called F-Perceptory was defined in order to model fuzzy geospatial data ([11], [34]).

To store fuzzy geospatial data in relational databases, one possible method is to map the fuzzy model to the relational model using its associated UML model. Many transformation rules must be applied to the fuzzy entities, fuzzy attributes, and relations to get the relational model. As the object nature is spatial and imprecise, some other

2. CAP: consistency, availability, partition tolerance  
3. ACID: atomicity, consistency, isolation, and durability
constraints must be applied to compensate for this specific nature and the complex structure of such data. Additional constraints are then added to provide integrity on the fuzzy spatial object structure. However, in the geospatial big data context, the authors of [35] determined the RDBMS limitations for fuzzy geospatial big data handling, even though they ensure consistency.

In fact, NoSQL databases are useful tools in dealing with geospatial big data. Unfortunately, because they do not support transactions, data integrity, or durability, NoSQL databases do not ensure data quality. Most do not require structural constraints on data and have only unknown schema. Without an explicit schema, it is very difficult to be sure of managing consistent fuzzy data. Moreover, accessing and analyzing this data without some idea of the schema or structural constraints does not result in a meaningful and interesting analysis. Therefore, because metadata are considered the main quality report method ([36], [37]), providing a data structure schema solves the schema-less limits.

To manage fuzzy geospatial big data, it is essential to develop approaches for their storage. In the following section, we present our solution for handling the consistency of fuzzy geospatial data in non-relational systems.

3 METHODOLOGY

Considerable research work has focused on geospatial data management in relational databases, but few are interested in fuzzy geospatial data [11]. However, relational systems suffer when scaling up to very large data volumes. Currently, innovative solutions are proposed, such as using NoSQL systems with geospatial object indexing, but, to the best of our knowledge, there is no approach for managing fuzzy geospatial data in NoSQL databases.

To store fuzzy spatial data as coherently as possible, data must validate several constraints relating to their complex structure and required semantic values. Hence, our proposal aims to assist users in correctly storing fuzzy geospatial data in NoSQL databases by adding integrity data control. This leads to providing a specific schema to validate the fuzzy geospatial data structure and a set of developed functions to validate the required semantic characteristics.

Our methodology consists of an automatic checking system to provide an efficient method for consistent storage of fuzzy spatial data within NoSQL databases. Fig. 8 shows the process architecture based on a format, structure, and semantic validators used to identify conformant and non-conformant fuzzy data.

The objective pursued by this methodology is using a document-oriented NoSQL database for storing fuzzy data. Document storage offers the best representation means to define complex data, such as fuzzy geospatial data. Based on an aggregation approach, it can improve data access performance and reduce latency. In addition, it is not concerned with high performance, but rather to ensure big data storage and good query performance.

A document database is a document collection in JSON format. The JSON format contains implicit structural information whereby each document presents a structure of either a single object or a collection of objects. Object properties are name/value pairs, which include basic, multi-valued, and more complex data types. With JSON format, there is no need to define a complex join, as with the relational model. Using the JSON format allows us to match the fuzzy spatial data modeling needs and to give more flexibility in defining other specific structures. MongoDB is an open-source document-oriented database used for a wide variety of applications. It is built for scalability, performance, and high availability. It also enables high speed access of data via its internal memory storage. MongoDB supports indexing over embedded objects and arrays. It provides geospatial index types, covering GeoJSON [38] objects to support the geospatial queries. Its schema flexibility is one of the most useful features in making different models and prototype versions and in rapidly building applications [39]. MongoDB is suitable for storing fuzzy geospatial big data. As depicted in 8, the first constraint that fuzzy data must validate is to follow the JSON format used by document databases such as MongoDB.

To encode fuzzy geospatial data structure correctly, we propose the Fuzzy GeoJSON schema. It is based on the GeoJSON format [38] that defines geometries (objects with their spatial extents), features (geometries with additional properties) and a feature collection. The Fuzzy GeoJSON schema extends GeoJSON in order to provide the required structure of fuzzy geometries, fuzzy features, and fuzzy feature collection. It assigns several constraints applied on types, number of values, and ranges of values for each element in the schema. As illustrated in Fig. 8, the second control checks fuzzy data structure against the Fuzzy GeoJSON schema.
Section 2 gives the theoretical bases of the Fuzzy GeoJSON schema and Section 4 demonstrates its definition detail.

To be consistent, fuzzy geospatial data need to validate other required semantic features. These semantic features are not overcome by the Fuzzy GeoJSON schema. Therefore, we add a third validator to semantically validate fuzzy geospatial data (Fig. 8). Semantic constraints are defined in Section 2 and developed solutions to them will be discussed in Section 5.

Consequently, if fuzzy geospatial data respect all the JSON format, the fuzzy structure, and the semantic feature constraints, they will be stored within the database for subsequent lecture, processing, and interpretation.

4 Fuzzy GeoJSON Schema

The JSON format and GeoJSON schema are both starting-points in defining the Fuzzy GeoJSON schema. The JSON format is used to define structural constraints of different JSON object types. The GeoJSON schema is used to add syntax descriptions for geospatial objects. The Fuzzy GeoJSON introduces syntax descriptions and constraints for fuzzy geospatial objects.

4.1 General Structure

As a main representative means for geospatial data, GeoJSON is a lightweight format for representing geographical features; it is endowed with all the basic geometries, feature and feature collections. Based on the GeoJSON specification, the Fuzzy GeoJSON schema is formulated; it involves integrating extensions in the GeoJSON features definitions in order to consider the imprecise nature of spatial data.

The Fuzzy GeoJSON schema is based on the json-schema-core specification and the json-schema-validation specification. The Fuzzy GeoJSON schema specifies terms used to identify objects, data types, and data structures defined in the IETF RFC 4627. The key words used to add constraints are to be interpreted as described in IETF RFC 2119.

Since the term feature is used by both the GeoJSON format and the Fuzzy GeoJSON schema, it is necessary to precisely determine its definition. In the GeoJSON format, each geospatial object has a geometry that denotes its shape, but it also has an identity and attributes. Thus, GeoJSON considers a feature to be an object with geometry and some other properties.

In the geographic domain, a feature is a spatial region with clearly defined boundaries. In our case, each \( \alpha \)-cut (see Definition 8 above) is a feature defined by a geometry and other non-spatial properties. Therefore, the Fuzzy GeoJSON schema uses feature to define one \( \alpha \)-cut for a fuzzy geospatial object. In this way, a fuzzy geospatial object that is naturally composed of a set of features (see Definition 9 above) is called a fuzzyFeature.

To show the general structure more clearly (see Fig. 9), some details in the schema have been replaced by \{···\} blocs and will be specified later. As defined with GeoJSON, the geometries sub-section gives definitions shape and coordinates characteristics of the basic geometries (point, line, and polygon). Using this, the sub-section definitions gives the new definitions of fuzzy geospatial objects structure. In the following, we discuss the definitions sub-section.

4.2 Simple Fuzzy Features Schema

The Fuzzy GeoJSON schema provides two main generic classes to define fuzzy data structure. These are fuzzyFeature and fuzzyFeatureCollection. The fuzzyFeature is the common super-class defined for all fuzzy geospatial object types either with simple or complex structure (see Fig. 10).

Fuzzy features can be summarized in two main categories: simple and composite. The schema provides fuzzyPoint, fuzzyPolygon and fuzzyLineString structures with which to describe fuzzy simple geospatial objects. For composite ones, the Fuzzy GeoJSON schema studies two sub-types: (1) composite objects with homogeneous geometries (a collection of one dimensional shapes); we can identify multiFuzzyPoint, multiFuzzyPolygon and multiFuzzyLineString and (2) composite objects with heterogeneous structures such as alternative, facultative and multiple geometries.

Regarding the fuzzyFeatureCollection, it denotes a collection of fuzzyFeature that is a fuzzy database for GIS applications consisting of a collection of fuzzy geospatial objects (see Fig. 11).

In this section, the schema detail of single fuzzy features is presented. Depending on the geometry type of the geospatial object, three simple features can be used to model single contiguous shapes: fuzzy point, fuzzy lineString, and fuzzy polygon.
Before describing the schema structure, we give some general and common characteristics. For each object type, geometry and a degree membership are defined. The geometry has two properties: the shape type that may be Point, Polygon, or LineString and the coordinates that give the object location. Associated with the geometry type, the coordinate property has a specific structure; for point geometry, it is a position (an example is illustrated by Fig. 12). For polygon geometry, it is a polygon that is a linear ring array. Finally, for line geometry, it is a lineString structure, which is a position array [38]. The second required property is the membership degree; its value domain must be a numeric in the closed interval [0,1].

However, the fuzzy feature structure must have, in general, a type giving an idea about its geometric form and cuts introducing its fuzzy spatial boundaries. After this, depending on the geometry and the degree membership value of its cuts, each fuzzy geospatial object has a specific definition. In the following, the schema for each type is provided.

4.2.1 Fuzzy point geometry schema

Here, we describe the Fuzzy GeoJSON schema of a fuzzy point object. As illustrated in Fig. 13, a fuzzy point is an object with two properties: (i) a type that must take a value as a FuzzyPoint and (ii) cuts that must be a pointCut array with “minItems”=2. This implies that the cuts property has at least two α-cuts specifications related to the constraint defined earlier in Definition 10. The schema considers type and cuts properties as required (“required”: [“type”,“cuts”]) and does not support any additional properties to describe the fuzzy object (“additionalItems”: false).

A pointCut presents a one α-cut structure for a fuzzy point object. As described in Fig. 14, it is a JSON object that requires a type as a Feature, a geometry, and a set of other information (properties). In particular, the pointCut must have a degree defining the α-cut membership degree to the fuzzy point. Depending on the membership degree value, the α-cut geometry can have a value of either the Point or the Polygon type. Therefore, if the degree is equal to 1, the geometry is necessarily a Point; otherwise, the geometry is a Polygon.

4.2.2 Fuzzy line geometry schema

A fuzzyLineString describes the fuzzy line object structure. As shown in Fig. 15, it is a JSON object, which requires a type and a set of cuts. Using the key “enum”, the object type must be FuzzyLine (“type”: “enum”: [“FuzzyLine”]). The set of α-cuts is specified as a lineCut array. Each lineCut presents one feature for the fuzzy line object.

The lineCut models a one α-cut structure for the fuzzy line object. The structure is similar to that of pointCut. As shown in Fig. 16, it consists of three required attributes: a type, a geometry, and additional non-spatial information. The cut type is always a Feature. However, depending on the membership degree value that must be defined in the properties, the geometry may have either a LineString or a Polygon value. According to the constraint defined in Defini-
Fig. 15. Fuzzy line schema

Fig. 16. $\alpha$-cut schema for line geometry

Fig. 17. Fuzzy Polygon schema

Fig. 18. $\alpha$-cut schema for polygon geometry

4.2.3 Fuzzy polygon geometry schema

A Fuzzy polygon object can be defined as a JSON object with two required properties: type and cuts. As presented by the schema in Fig. 17, the type must be a FuzzyPolygon and the cuts must be an polygonCut array with at least two cuts. The model does not support additional properties.

Each polygonCut requires a type as a Feature, a geometry property, which must have the Polygon value, and additional information presented by the properties key. As illustrated by the schema in Fig. 18, properties part consists of key-value attributes that require a degree to define the membership degree of the cut to the fuzzy polygon object. The geometry is can take only the Polygon value independently of the membership degree value.

4.3 Composite Fuzzy Features Schema

Each geometry single type is excellent for representing homogeneous regions, but often, the real world contains regions that are much more complex. Many GIS modeling designs consider the concept of composite geometries. Consequently, composite geometries are becoming the basis for geographic modeling needs [40]. The Fuzzy GeoJSON schema provides required elements to define several composite geometries. It allows expressing modeling needs in terms of collection, alternative, facultative, and multiple geometries. They are described in detail in [34]. The Fuzzy GeoJSON schema introduces the JSON structure of these composite fuzzy features associated with the necessary integrity constraints.

4.3.1 Fuzzy aggregation geometry schema

The collection geometry describes a spatial object that is an aggregation of one simple shape type. For example, a street in the real world might consist of several road segments [34]. Based on the type shape, the Fuzzy GeoJSON schema distinguishes three aggregation categories: (1) MultiFuzzyPoint object: a collection of fuzzy point geometries, (2) MultiFuzzyPolygon: a collection of fuzzy polygon geometries, and (3) MultiFuzzyLineString: a collection of fuzzy line geometries.

In this section, the MultiFuzzyPoint example is studied and the corresponding schema is given in Fig. 19. The fuzzy point collection is an object whose type is MultiFuzzyPoint. Because it is composed of a set of fuzzy points, MultiFuzzyPoint requires the features key presented as an array of features of fuzzyPoint type.

The main criterion for distinguishing between MultiFuzzyPoint, MultiFuzzyPolygon, and MultiFuzzyLineString, is the definition of the features element. In the case of Mul-
tiFuzzyPolygon, the features is an array of fuzzyPolygon elements; however, in the case of MultiFuzzyLineString, the features is an array of fuzzyLineString elements. The constraint ("minItems": 2) indicates that each collection is composed of at least two fuzzy features.

### 4.3.2 Fuzzy facultative geometry schema

A facultative geometry is potentially used to model an object that may be defined with or without spatial attributes. A dataset presenting facultative geometries may contain objects with a shape and others without. The data provider will, according to its own rules, provide or not provide the geometric attribute.

For example, for a facultative polygon building, the data provider can give the geometric attribute (fuzzy polygon) of a building object only if its area is greater than 500 m²; if not, the building object is considered a non-spatial object. Three types of facultative geometry can be expressed: fuzzyFacultativePoint, fuzzyFacultativeLine and fuzzyFacultativePolygon.

An example presenting the schema of a fuzzy facultative polygon object is given in Fig. 20. The schema defines a JSON object with a type element that must be a FuzzyFacultativePolygon, a cuts element that is a polygonCut array and, finally a properties element to define additional information. The schema allows us to make finer distinctions between the fuzzyPolygon and the fuzzyFacultativePolygon since with a fuzzyPolygon, type and cuts must be defined (see Fig. 17); however, if the object cannot have spatial information, then cuts becomes unnecessary. Thus, with fuzzyFacultativePolygon, type and properties are mandatory, while cuts is optional.

### 4.3.3 Fuzzy alternative geometry schema

According to [34], the fuzzy alternative geometry introduces objects that can be represented by two possible spatial representations, but each object has only one of the two, never both. For example, a shopping center can be either markets or malls. Based on the area criterion, two groups may be distinguished; some regions with fuzzy point features (markets) and some others with fuzzy polygon features (malls).

As illustrated in Fig. 21, a fuzzy alternative object requires the FuzzyAlternativeGeometry type, a set of cuts, and various non-spatial properties. According to the schema, cuts can have one value among the three following ones: pointCuts, polygonCuts or lineCuts. Each object in the database has one type of cuts so it cannot have two different representations. The oneOf keyword is used to validate the given value (cuts) against exactly one of the given subschemas [31]. This constraint avoids storing multiple representations for the same object in the database.

### 4.3.4 Fuzzy multiple geometry schema

The fuzzy multiple geometry presents geometries that can be represented by at least a spatial representation, meaning that each object may have one or more representative geometries. The result of the queries can be one or two representations for the same object. Fig. 22 shows the associated schema to this type, where each object can have one or more cuts type, then the object may have multiple representations. The anyOf keyword is used to answer this complexity, meaning that the given value of cuts may be valid against any (one or more) of the given subschemas [31].

As mentioned earlier, in addition to the definition of cuts, we also need to define some other constraints to validate the consistency of the structure and content of fuzzy geospatial data. We give, in the following, the semantic constraints not overcome by the Fuzzy GeoJSON schema.
5 SEMANTIC CONSTRAINTS

The Fuzzy GeoJSON schema defines the structure constraints at the conceptual level, but it cannot overcome constraints at the logical model. Therefore, given a fuzzy dataset \( E \), we are interested in checking the semantic consistency of all fuzzy objects \( O^E = \{O_1, \ldots, O_n\} \) that exist in \( E \) where each fuzzy object \( O \) has a geometry \( G_O \) which is \{Point, Line, Polygon\} and is composed of a finite set of cuts \( C^O = \{C_1, \ldots, C_m\} \). Each cut \( C \) is defined by \( (G_C, \alpha_C) \) where \( G_C \in \{\text{Point}, \text{Line}, \text{Polygon}\} \) is the cut geometry type and \( \alpha_C \in [0, 1] \) is the degree membership of the cut \( C \) to the fuzzy object \( O \).

In this section, we give the formalism used to validate the remaining semantic constraints presented by the definitions 7, 11 and 12 (see Section 2.2).

5.1 Cut Consistency constraint

Related to the Definition 11, a reliable cut is measured according to its degree membership and the geometry type of the related fuzzy object. Within a range \([0, 1]\), the \( \alpha \) degree gives an estimated idea about the possibility that the cut presents the object with boundaries as realistically as possible. A consistent cut, denoted by \( C \in C^O \), is defined by \( (O, \text{consistentCut}(C)) \) where \( O \) is the related fuzzy object and the function \( \text{consistentCut}(C) \) verifies that (1) for a membership degree \( \alpha_C = 1 \), the cut geometry \( (G_C) \) must be the same as the geometry \( G \) of the related fuzzy object or (2) for a membership degree \( \alpha_C \neq 1 \), the cut geometry \( G_C \) must be of type \text{Polygon}.

Algorithm 1, with \( O(m) \) complexity where \( m \) is the number of cuts, gives pseudo-code to check which cuts are reliable for given degree and fuzzy object geometry type and tries to detect those that are inconsistent.

5.2 Fuzzy Object Consistency Constraint

Related to the definitions 7 and 12, a consistent fuzzy object must have a connected and normalized set of cuts. A connected and normalized fuzzy object denoted \( O \in O^E \), is defined by \( (C^O, \text{Norm}^O, \text{Inclus}^O) \) where \( C^O = \{C_1, \ldots, C_m\} \) presents a finite set of consistent cuts for the fuzzy object \( O \). The function \( \text{Norm}^O \) indicates that, \( \forall C \in C^O \) with a membership degree \( \alpha_C, \text{Max}(\alpha_C) = 1 \). The set of cuts is thus normalized. For each couple of cuts \( \{C_i, C_k\} \) with their respective geometry types and membership degrees \( \{(G_{C_i}, G_{C_k}), (\alpha_{C_i}, \alpha_{C_k})\} \), the function \( \text{Inclus}^O \) verifies the inclusion relation and indicates that if \( (\alpha_{C_i} > \alpha_{C_k}) \) then \( (G_{C_i} \subset G_{C_k}) \). If the inclusion constraint is violated or the set of cuts is not normalized, a message error is reported to the user. Algorithm 2 presents pseudo-code to verify fuzzy object consistency with \( O(n^2) \) complexity.

5.3 Fuzzy Database Consistency

The formalism that we have defined earlier allow us to define the following algorithm for checking fuzzy data for their structure and semantic.

Let \( E \) be a fuzzy dataset that is defined by \( (O^E, (C^O)^E, \text{json}^E, \text{fuzzyJson}^E, \text{consistentCut}^E, \text{consistentObject}^E) \) where \( O^E = \{O_1, \ldots, O_n\} \) is a finite set of fuzzy geospatial objects that exist in \( E \). The \((C^O)^E = \{C_1, \ldots, C_m\} \) is a finite set of cuts of each fuzzy geospatial object \( O \) over \( E \).

A fuzzy dataset is considered consistent if it does not violate the structure or semantic constraints. Regarding to the structure, the \( \text{json}^E \) function checks that the json structure of \( O^E \) is correct and then the \( \text{fuzzyJson}^E \) function verifies that \( O^E \) conforms to the Fuzzy GeoJSON schema. After, if no structure error exists, data can contain semantic inconsistency. It is therefore, \( \forall O \in O^E \), the \( \text{consistentCut}^E \) function checks that each cut does not represent a non-conformant semantic against the geometry and membership values. Finally, the \( \text{consistentObject}^E \) function checks that the set of cuts defines semantically a correct object \( O^E \).

Algorithm 2 Checking the fuzzy object consistency

Require: Initialize with \( O \)

if \( \text{norm}(O) \neq 1 \) then
return normalization error message and EXIT
else
for each \( C_i \) in \( C^O \) do
\( G_{C_i} \leftarrow \) the cut geometry
\( \alpha_{C_i} \leftarrow \) the cut degree membership
\( RC \leftarrow \) the remaining cuts of \( C^O \)
for each \( C_k \) in \( RC \) do
\( G_{C_k} \leftarrow \) the cut geometry
\( \alpha_{C_k} \leftarrow \) the cut degree membership
if \( \text{Inclus}((G_{C_i}, \alpha_{C_i}), (G_{C_k}, \alpha_{C_k})) \) is FALSE then
return inclusion error message and EXIT
end if
end for
end for
return TRUE
Algorithm 3 Checking the dataset consistency

Require: Initialize with $O^E$
1: if $\text{json}(O^E)$ is TRUE then
2: if $\text{fuzzy\text{-}json}(O^E)$ is TRUE then
3: for each $O$ in $O^E$ do
4: if $\text{consistent\text{-}cut}(O)$ is TRUE then
5: $\text{consistent\text{-}object}(O)$ is TRUE
6: end if
7: end for
8: end if
9: end if
10: store $O^E$ in database

Fig. 23. JSON Format error detection

Algorithm 3 presents the pseudo-code to verify the dataset consistency with $O(n(m + m^2))$ complexity. Regarding to the scalability requirements when the number of fuzzy objects grows, the algorithm can easily be parallelized among multiple clusters, where each of them must deal with a partition of objects. Moreover, the MongoDB database provides an automatic sharding, which makes the system scalable.

According to the results, fuzzy geospatial data can be stored as a consistent dataset within MongoDB database if they broadly respect the JSON format, the fuzzy structure, and the semantic constraints.

6 Detection Inconsistent Data Experiments

The methodology objectives are supposed to give fuzzy schema and semantic features for representing fuzzy geospatial data and to provide a comfortable way to apply the validation process to correctly store them in NoSQL databases. The principles of our methodology are illustrated by a prototype implementation developed in Java. The prototype includes an interface manipulation where the user introduces fuzzy data, visualizes them, and validates them against a fuzzy schema and constraints. On the top of the interface, there is a map to visualize datasets, and on the bottom, the running of the validation process appears in a log form. The crucial part of the prototype is related to the checking dataset, designed as a validation process.

In this section, we demonstrate the experiments results of the checking system using datasets from data.gouv.fr. Before, we developed a QGIS plugin to provide fuzzy data. In the following, some checking results are given.

6.1 Checking Syntax

The first stage in the validation process relies on the JSON format of data. Fig. 23 shows a format error example. The log panel displays an error related to a key definition. Fig. 24 shows a log panel result of a cut structure error. It is assumed that a fuzzy polygon must have only cuts with Polygon geometry. In the dataset, a cut is detected with a ‘Point’ geometry type and an error message is displayed.

6.2 Checking Semantics

In Fig. 26, the log panel displays an error relating to the normalization criterion. The fuzzy object number 195 does not build a normalized fuzzy set, which means that it does not have one $\alpha$-cut with a membership degree equal to 1.

Fig. 27 gives the result of connection error detection. Two fuzzy objects do not form a connected fuzzy set. The neighboring cuts for each object must be included among them according to the membership degree value.

7 Performance Experiments

To evaluate the performance of the system for storing consistent fuzzy geographic big data, we proceed to a distributed checking using multiple clusters to share the full scan and store data. We used five clusters; each one has a 16 GB of RAM. Based on the CORINE Land Cover database (green land data), we created fuzzy data where each fuzzy object is represented by three cuts. We first characterize the storage process in terms of execution time for semantic checking and storage stage by varying data size. Second, we study the efficacy of the proposed checking process by error rates. Finally, we study the quality performance of the checking system in terms of recall and precision measures.
7.1 Storage Cost With Checking Process

The first sets of experiments assess the time cost of checking the fuzzy geographic data. In this case, we assume that data are error-free. The system incorporates 10 different data sizes to the system from 1000 fuzzy geographic objects (3000 cuts, 29 MB) to 192000 (576 000 cuts, 5560 MB). Fig. 28 shows the fuzzy data storage cost with checking process with varying data quantity.

MongoDB has a validation mechanism that enforces a schema on documents of a collection. We used this mechanism by introducing the Fuzzy GeoJSON schema to perform the structure checking of the input fuzzy data. Therefore, we consider the storage cost that combines the structure check. The storage time with varying data quantity progressively increases but is still significantly low (≃ 171 s for 576 000 cuts).

We define an additional validator on the collection level to perform the semantic checking. As seen in Fig 28, the storage and structure checking cost is more time-consuming than the semantic checking cost. The semantic checking consumes around 33% of the total time (semantic checking and storage). Therefore, according to the assumptions that MongoDB allows big data storage, our proposal permits dealing with fuzzy geospatial big data.

The system gives the advantage of having a pre-storage checking for a given input fuzzy data. The system stores data only if the checking stage does not return wrong data, thereby ensuring that the stored data are valid.

7.2 Comparison of Checking Cost by Error Rates

The second sets of experiments assess the checking cost with increasing error rate from 5% to 20% and varying the fuzzy object number from 1000 to 192 000. Fig. 29 shows the effect of the error rate on the checking cost. We observe that the execution times for the semantic checking between error-free data and erroneous data are similar. Even though the checking time increases with the data size, it does not consume time if the error rate increases.

Moreover, our validation system corresponds to the checking of the semantic rules, which are clear, logical, and precise. It detects all the errors without any false detection. Thus, the precision and recall measures are equal to 1. The system is used as a filter for all inconsistent data in order to store only the consistent data.

8 DISCUSSION

Classic representation methods explore the space in a crisp manner. With this, each system provides proper geospatial datasets with existing limits, creating new limits, or considering that some limits have no particular meaning and therefore eliminating them. Given these cases, traditional data-handling approaches are inadequate. Therefore, the development of conceptual and methodological approaches based on fuzzy descriptive models is a needed solution.

8.1 Fuzziness and Spatial Data

To illustrate the importance of the fuzzy model in capturing imprecise borders, let us consider the case of airport boundaries that are defined by four systems: OpenDataSoft, IGN, ESRI, and GISCO. Fig. 30-(a) illustrates a representative comparison between datasets produced by these systems. All of them are presented on OpenStreetMap, one of the most popular VGI. We limit the studied areas to two airports in France (Roissy Charles de Gaulle and the Bourget).

Displayed maps reveal clear differences in representing boundaries for the same airport areas (marked by red stars). Moreover, OpenDataSoft, IGN, ESRI, and GISCO are based on the GeoJSON model. Thus, they do not present boundaries as a set of α-cuts, and they cannot detect wrong data or limits. They do not consider fuzzy geospatial types and do not define a fuzzy schema. These different limits show the need for a new storage model for fuzzy geospatial data, such as that illustrated by Fig. 30-(b) using our fuzzy model with three α-cuts. Our tests, using a developed plugin under the QGis tool, indicate that the Fuzzy GeoJSON model is better at capturing fuzziness than crisp models.

Since fuzzy data have a complex structure, consistent storage is impossible without domain-specific knowledge and many constraint controls. The Fuzzy GeoJSON is required to solve many problems that involve predictive modeling, visualization issues that also use imprecise location as a key component. For example, the World of Disputed Territories application uses big data with a spatial component for detecting disputed territories. The system locates territories that are claimed by more than one country/occupying force. The description can be more relevant if the map was made using a fuzzy model because conflict locations cannot have

4. metrocosm.com/disputed-territories-map.html
Fig. 30. Representing imprecise areas: (a) different boundaries to represent the same objects in the crisp approach and (b) three cuts to represent imprecise objects in the Fuzzy GeoJSON approach

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>Fuzzy geospatial data handling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MADS</td>
</tr>
<tr>
<td>Fuzzy geospatial types</td>
<td>-</td>
</tr>
<tr>
<td>Implemented solution</td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>-</td>
</tr>
<tr>
<td>Relational/Object context</td>
<td>+</td>
</tr>
<tr>
<td>NoSQL system context</td>
<td>-</td>
</tr>
</tbody>
</table>

clearly defined borders. The Surging Seas application also uses big data with a spatial component for sea levels. It is a scientific projection of sea levels that could occur in this century. Some areas that will eventually be underwater have fuzzy limits.

8.2 Comparative Study

Different studies offer modeling opportunities to customize a classic approach for representing fuzzy geospatial concepts. Table 1 presents a comparison of the following fuzzy geospatial data modeling systems: MADS [41], GIS-UML [42], F-Perceptory [11], Schneider approach [8] and, Fuzzy GeoJSON. Schneider proposed a formal approach to deal with relational databases where the membership function of the fuzzy spatial object is defined using a mathematical formula, such as an interpolation function, and thus, it is subject to overinterpretation [43]. To consider reliable fuzzy degrees instead of functions, GIS-UML and F-Perceptory bring implemented solutions based on α-cuts but are only suitable for relational/object systems (e.g., spatial stores). F-Perceptory offers full consistency, where it can be used in any relational/object context but not in the NoSQL context. To the best of our knowledge, the Fuzzy GeoJSON methodology is the first proposal using a fuzzy data model to represent imprecise geospatial objects in NoSQL systems. The Fuzzy GeoJSON schema is the only solution that supports full consistency in the big data context.

In conclusion, the methodology and the tool proposed in this article contribute to data quality of big data in three aspects: first, fuzzy modeling of GeoData is a way to consider both the data and their imprecision that is impossible with classic and crisp GeoData; second, each fuzzy GeoData stored using this approach is consistent; third, the system is scalable.

9 Conclusions and Future Work

Simplifying fuzzy geospatial models or the missing data results in the spread, or even the increase, of errors. To deal with the quality storage of fuzzy geospatial big data, in this paper, we presented , a new methodology that allows us to store consistent (correct) fuzzy geospatial data compliant with NoSQL systems.

Through a specific consistency validation pipeline, this methodology includes an effective method in which fuzzy data meet a specified schema called Fuzzy GeoJSON and must satisfy higher semantic constraints, before being stored.

This validation methodology is the first step in providing a complete framework for dealing with fuzzy geospatial big data that may add querying and mining abilities. Our next research direction is in querying fuzzy spatial big data relating to topological relations.

References
