Spatial Information Retrieval in Digital Ecosystems: A Comprehensive Survey

Anderson Chaves Carniel
accarniel@utfpr.edu.br
Federal University of Technology - Paraná
Dois Vizinhos, PR, Brazil

ABSTRACT

Spatial information retrieval is a common task of digital ecosystems due to the popularity of collecting and storing spatial information and phenomena in the world of the Internet of Things (IoT). *Spatial relationships* play an important role in this context by specifying how two or more spatial objects are related or connected. Examples of spatial relationships include *topological relationships* (e.g., intersect, overlap, contains), *metric relationships* (e.g., nearest neighbors), and *direction relationships* (e.g., cardinal directions like north and south). Many works in the literature have proposed definitions and implementations of spatial queries based on specific types of spatial relationships. Hence, a holistic view of these works is important to understand their applicability and relations. This paper advances in the literature by providing a *comprehensive survey* of the implementations and types of spatial queries that can be used by digital ecosystems. We present a novel characterization based on spatial relationships to define topological-based, metric-based, and direction-based spatial queries. For each type of spatial query, we present its intuitive and formal definitions together with possible strategies of implementation. Further, we identify *hybrid spatial queries* as combinations of two or more spatial relationships, and *spatial joins* as generalization cases. In addition, we present some equivalences between some types of queries. As a result, we point out future research topics in spatial information retrieval.

CCS CONCEPTS

• **Information systems → Data access methods**; *Information retrieval*; **Geographic information systems**; • **General and reference → Surveys and overviews**.

KEYWORDS

Spatial information retrieval; Spatial relationship; Topological relationship; Metric relationship; Direction relationship; Spatial join; Hybrid spatial queries; IoT

ACM Reference Format:
Anderson Chaves Carniel. 2020. Spatial Information Retrieval in Digital Ecosystems: A Comprehensive Survey. In *12th International Conference on Management of Digital EcoSystems (MEDES '20), November 2–4, 2020, Virtual Event, United Arab Emirates.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3415958.3433038

1 INTRODUCTION

Spatial information retrieval is a common task required by modern and advanced applications belonging to digital ecosystems. The interest is to enrich data analysis by representing, storing, and managing spatial or geometric phenomena [19, 24, 37, 38]. In the world of the Internet of Things (IoT), systems that share resources and collect spatial data are very popular. For instance, *spatial crowdsourcing* has attracted the attention of many researchers when making available huge amounts of spatial data that can be retrieved by spatial queries. This has led to the development of architectures to deal with spatial data in the context of IoT [13], benefiting several applications like location-based services, smart cities, and mobile recommendation systems.

Spatial database systems and Geographical Information Systems (GIS) are powerful tools for retrieving spatial phenomena often modeled as instances of *spatial data types* (e.g., points, lines, and regions). *Spatial relationships* are commonly expressed as spatial operations in spatial queries by specifying how two or more spatial objects are related or connected. Examples of spatial relationships include *topological relationships* (e.g., intersect, overlap, contains) [31, 38], *metric relationships* (e.g., nearest neighbors) [12, 42], and *direction relationships* (e.g., cardinal directions like north and south) [9, 35]. In fact, the importance of spatial relationships to retrieve spatial objects from spatial databases is indicated by several works in the literature (see Section 2).

Due to the importance of spatial information retrieval, works in the literature survey approaches that focus on defining and optimizing types of spatial queries. For instance, the authors in [14] survey several spatial index structures that are employed to optimize spatial queries based on topological relationships. Another example is the work in [10] that defines how spatial relationships can be embedded in SQL-like spatial queries. Further, the authors in [42, 45] propose taxonomies for spatial queries based on metric relationships. Other examples are surveys dedicated to deal with *spatial joins* [3, 21], a common operation that combines two spatial datasets according to a spatial relationship.

However, existing surveys in the literature face the main problem of focusing on defining types of spatial queries on *one* type of spatial relationship only. This limits us to comprehend the applicability and flexibility of spatial information retrieval. Further, there is a lack of identifying possible equivalences in the different types of queries. Finally, a general view of the optimized techniques, which can be applied in different scenarios, is missing.

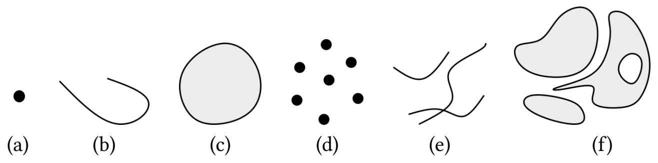

Figure 1: Examples of spatial objects: a simple point object (a), a simple line object (b), a simple region object (c), a complex point object (d), a complex line object (e), and a complex region object with a hole (f).

This paper fulfills this gap in the literature by presenting a comprehensive study on the implementations and types of spatial queries that are frequently issued in digital ecosystems. We pursue three objectives. The first objective encompasses the characterization of spatial queries by considering three types of spatial relationships. As a result, we classify spatial queries in the following types: (i) *topological-based spatial queries*, (ii) *metric-based spatial queries*, and (iii) *direction-based spatial queries*. For each type of spatial query, we present its intuitive and formal definitions together with possible strategies of implementation. The intuitive and formal definitions help us to understand the purpose of a type of spatial query. From the comprehension of the formal definitions, we can even observe that the same implementation can solve different types of queries. This is useful to understand how to optimize queries by rewriting them in other formats.

The second objective relates to the identification of types of spatial queries that are derived from the aforementioned characterizations. Some spatial queries are very simple in their intuitive and formal definitions. On the other hand, applications may need more specialized queries that are obtained from the combination of different spatial relationships. This leads to *hybrid spatial queries*. Further, *spatial joins* are defined as generalizations of the described queries.

Finally, the last objective is to establish future research topics. This is possible thanks to the comprehensive review conducted in this paper.

This paper is organized as follows. Section 2 summarizes needed basic concepts. Section 3 surveys the identified types of queries and their implementations. Section 4 discusses challenges and future trends. Section 5 concludes the paper and presents future work.

2 AN OVERVIEW OF SPATIAL DATABASE SYSTEMS

This section provides needed concepts from spatial data types (Section 2.1), spatial relationships (Section 2.2), and how spatial queries are commonly processed (Section 2.3).

2.1 Spatial Data Types

Spatial database systems and GIS store, query, and handle spatial or geographic information represented by homogeneous geometries in the Euclidean space. For modeling them, formal definitions of *spatial data types* for *points*, *lines*, and *regions* have been provided in the literature [19, 38]. *Spatial objects* are instances of these data types and can assume *simple* or *complex* structures, as shown in Figure 1. Simple spatial objects are single-component objects (Figures 1a, b, and c), whereas complex spatial objects contain finitely many components that allow users to represent the reality and complexity of geographic phenomena (Figures 1d, e, and f). Examples are the locations of ATMs, trees, and hydrants represented as point objects, rivers, streets, and highways expressed as line objects, and countries, states, and airports shaped as region objects. The context of this paper is the retrieval of point, line, and region objects. Hence, the manipulation of other types of spatial information, such as spatiotemporal databases [37] and fuzzy spatial databases [6, 7], goes beyond the scope of this paper.

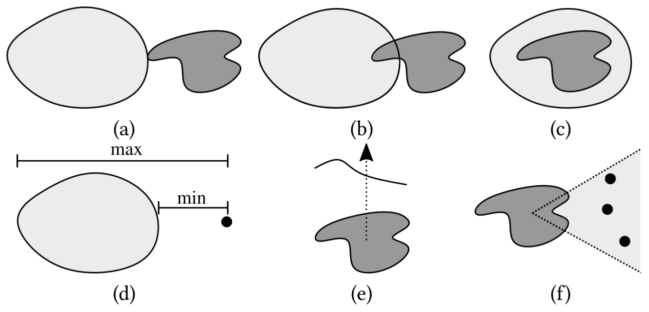

Figure 2: Examples of spatial relationships: two region objects *meeting* (a), two *overlapping* region objects (b), *containment* relation of two region objects (c), minimum and maximum *distances* between a point object and a region object (d), a line object in the *north* of a region object (e), and the *visible angle* of a region object (f).

2.2 Spatial Relationships

Spatial relationships are used as conditions and predicates in spatial queries and describe how a spatial object is related, connected to another spatial object. For this, different interpretations are conceivable, including *topological relationships* [31, 38], *metric relationships* [12, 42], and *direction relationships* [9, 35]. Topological relationships are often characterized by employing the 9-intersection model and variants. They make use of point sets and point set topology to provide a complete collection of mutually exclusive topological relationships for each combination of simple and complex spatial data types. The model is based on the nine possible intersections of boundary, interior, and exterior of two spatial objects, according to the topologically invariant criteria of emptiness and non-emptiness. Further, this model can be extended to deal with the dimensions of the intersections [28, 39]. As a result, nine topological relationships are commonly defined in the literature [31, 38]; they are: *intersect*, *overlap*, *meet*, *disjoint*, *equal*, *inside*, *contains*, *covers*, and *coveredBy*. The constellations depicted in Figure 2 provide some examples of topological relationships.

Metric relationships are characterized by the extraction of numerical values from geometric properties of spatial objects [12]. It includes type-dependent operations like the length of line objects and the area of region objects [38]. In addition, metric relationships make use of minimum and maximum distance functions (e.g., Euclidean and Manhattan) to evaluate the spatial proximity of spatial objects [42], such as shown in Figure 2d. Hence, these relationships provide us the notion of *nearness* among spatial objects.

Direction relationships (also known as directional relationships) are characterized by the relative direction of two spatial objects [9, 35]. Intuitively, a spatial object has the same direction as another spatial object, if the latter is within a given angular range of the former [23]. Further, the angle can be associated with the visible area of a spatial object considering its spatial orientation [30]. Hence, directions can be either predefined by using cardinal directions like *north*, *east*, *west* or *south* and their intermediate directions [35] or user-defined angles for location-based services [18]. Examples for these two situations are given in Figures 2e and f. To compute such cases, a direction-relation matrix [9] can be employed, which is based on the point set topology in a similar way as the 9-intersection model.

Different spatial relationships can be further combined in spatial queries. This combination can be obtained by linking different conditions using logical operators (e.g., or, and, not), and by specifying new types of spatial queries.

2.3 Steps for Spatial Query Processing

Spatial index structures are often employed to reduce the search space of spatial queries by avoiding spatial objects that certainly do not belong to the final answer of the query. Examples include the R-tree [20] and variants like the R*-tree [1] (see [14] for a survey of other spatial index structures). Intuitively, a spatial index structure groups objects spatially close in well-defined data structures. Due to the high complexity of representing spatial objects, spatial objects are approximated to a simpler geometric format in the spatial index structures. An example of approximation is the *Minimum Boundary Rectangle* (MBR). The main benefit is that the evaluation of spatial relationships by using MBRs is much faster than processing costly computational geometric algorithms [33]. However, approximations introduce an area that does not belong to the original spatial object.

To handle this situation, two steps are needed when processing spatial queries [14]. The first step, called *filter*, uses a spatial index structure (or approximations) to reduce the number of spatial objects to be precisely evaluated. The second step, called *refinement*, gets these candidates as input and accesses the original spatial object to evaluate the spatial relationship of the query; thus, it excludes the false candidates and returns the final answer of the query (e.g., as shown in Figure 3b). Note that the filter step can be sufficient to process some types of queries, such as the *Containment Range Query* (Figure 3d).

3 TYPES OF SPATIAL QUERIES

In this section, we provide a taxonomy for characterizing spatial queries by taking into account the employed spatial relationship. Four main types are identified: (i) *topological-based spatial queries* (Section 3.1), (ii) *metric-based spatial queries* (Section 3.2), (iii) *direction-based spatial queries* (Section 3.3), and (iv) *hybrid spatial queries* (Section 3.4) that employ two or more different types of spatial relationships. Further, we consider *spatial joins* (Section 3.5) as a general case of the identified types since it can apply any spatial relationship to associate two or more spatial datasets.

Table 1: Some variants of the topological-based spatial queries. Commonly, the search object of a range query is a rectangle or square, called *query window*. Further, the search object can also be a circle in range queries.

Topological-based Spatial Query	Search Object (s)	Topological Relationship (t)
Exact Match Query	any spatial object	equal
Adjacency Query	any spatial object	meet
Point Query	simple point object	intersect
Group Point Query	complex point object	intersect
Range Query	simple region object	any
Intersection Range Query	simple region object	intersect
Containment Range Query	simple region object	contains
Enclosure Range Query	simple region object	inside

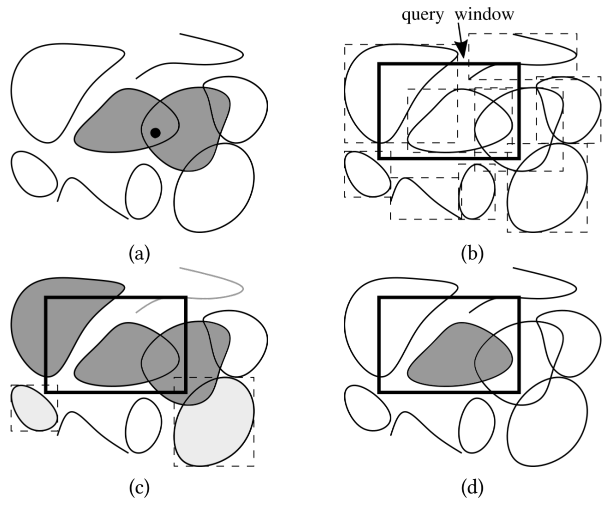

Figure 3: Examples of topological-based spatial queries: a *Point Query* (a), an example of a query window in a *Range Query* and the use of MBRs represented as dashed lines (b), an *Intersection Range Query* with the false candidates illustrated in gray (c), and a *Containment Range Query* (d). The final answer for the queries in (a), (c), and (d) are the spatial objects in dark gray.

Let D be a spatial dataset in which we aim at posing spatial queries. Let further $s \in \{point, line, region\}$ be a search object. This section studies and comprehends each spatial query type by (i) describing its application, (ii) providing formal definitions, and (iii) summarizing optimized data structures and algorithms for its implementation.

3.1 Topological-based Spatial Queries

We generalize this category as the *Object Query* (or *spatial selection*). Hence, this query serves as a basis to define other kinds of topological-based spatial queries (Table 1).

Descriptions and Applications. Intuitively, the *Object Query* retrieves all spatial objects from D that satisfy a given topological relationship for the search object s. Table 1 shows some possible variants of the *Object Query* frequently studied in the literature (e.g., [3, 14]). Figure 3 depicts some examples of these queries. Many spatial database and GIS applications have employed variants of the topological-based spatial queries to answer important issues, such as location-based services [47], disaster management tools [48], scientific spatial models [43], and wireless sensor networks [40].

Formal Definitions. Let $t \in \{intersect, overlap, meet, disjoint, equal, inside, contains, covers, coveredBy\}$. The *Object Query* can be formally defined as $OQ(D, s, t) = \{o \mid o \in D \wedge t(s, o) = true\}$. This definition can be viewed as a general template for topological-based spatial queries by varying the values for s and t as shown in Table 1 [14]. For instance, the *Point Query* assumes that s is a simple point object and $t = intersect$ in order to retrieve all spatial objects having the point s (Figure 3a). Further, if s is a complex point object, then we obtain a *Group Point Query*. Another example is the *Range Query*, which adopts a fixed geometric format for s, and does not require a specific topological relationship for t (Figure 3b). A further specialization of a *Range Query* is, for example, the *Intersection Range Query*, which assumes that $t = intersect$ to retrieve all spatial objects having at least one point in common with s (Figure 3c).

Implementations. The use of spatial index structures, such as the R-tree and its variants, has been a widely studied technique to process topological-based spatial queries. In this context, several works in the literature provide analysis and specific optimizations for these spatial queries. Due to space restrictions, this paper mentions some of them only. The authors in [22] analyze the selectivity and the number of index nodes accessed when processing range queries with different query window sizes. In [43], a spatial index based on the R-tree has been designed to optimize the performance of range queries for scientific spatial models. The execution of range queries using R-trees and its variants has been also studied in different environments, such as disks and Solid State Drives, by using FESTIval [5], a PostgreSQL/PostGIS extension that implements topological-based spatial queries. The authors in [36] leverage parallel algorithms to execute multiple range and point queries in flash memory, a persistent memory widely used in IoT applications. In [32], topological-based spatial queries are empirically evaluated by using parallel and distributed data processing frameworks to deal with a huge volume of spatial data.

3.2 Metric-based Spatial Queries

We identify three types of spatial queries belonging to metric-based spatial queries: (i) spatial queries with *unary operators*, (ii) spatial queries with *binary operators*, and (iii) the retrieval of *nearest neighbors* from a given input.

Descriptions and Applications. Metric-based spatial queries with unary and binary operators extract real values from geometric properties of spatial objects that are used in conventional predicates with relational operators. Unary operators have type restrictions on their operand. For instance, it is possible to compute the *length* of line objects only. Similarly, we yield the *area* of region objects only.

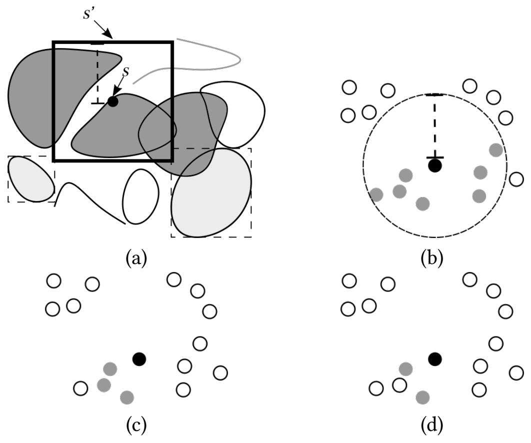

Figure 4: Examples of metric-based spatial queries: a distance-based spatial query with the Chebyshev distance, a distance-based spatial query using the Euclidean distance (b), a *k-Nearest Neighbor Query* with $k = 3$ (c), and a *Reverse k-Nearest Neighbor Query* with $k = 3$ (d). The resulting objects are denoted in dark gray. Note that the queries in (a) and (b) can also be specified as *Intersection Range Queries*.

Hence, we can retrieve line (region) objects whose lengths (areas) satisfy a given condition. The use of binary operators permits us to retrieve spatial objects within a certain *minimum or maximum distance* from a search object. Different distance functions are conceivable, such as the Euclidean distance, Manhattan distance, and Chebyshev distance. Distance functions can also be leveraged to find all the *nearest neighbors* of a search object [42]. Commonly, the number of spatial objects is limited to $k \in \mathbb{N}$, leading to a *k-Nearest Neighbor Query*. A popular variation of this query is the *Reverse k-Nearest Neighbor Query*, which finds all the spatial objects that have the search object s as one of their k-nearest neighbors. Other variants can also be defined [42, 45]. Figure 4 depicts some examples of metric-based spatial queries. These queries can be applicable in distinct scenarios, such as location-based services looking for buildings near a user [47].

Formal Definitions. Let $m \in \mathbb{R}$ and $\phi \in \{\leq, <, =, \neq, >, \geq\}$. Metric-based spatial queries that retrieves all region objects whose areas satisfy a given condition can be defined as $UNQ(D, \phi, m) = \{o \mid o \in D \wedge o \in region \wedge area(o)\ \phi\ m\}$. Similarly, we can select line objects according to their lengths by overloading UNQ, i.e., $UNQ(D, \phi, m) = \{o \mid o \in D \wedge o \in line \wedge length(o)\ \phi\ m\}$. The aforementioned queries make use of unary operators. Let *dist* be a function that computes a particular distance type (e.g., Euclidean, Manhattan, Chebyshev). We are able to define metric-based spatial queries with binary operators as $BNQ(D, s, dist, \phi, m) = \{o \mid o \in D \wedge dist(o, s)\ \phi\ m\}$. Here, some relations to the topological-based queries (Section 3.1) can be obtained. By applying a distance function to the search object s, we yield a new spatial object, named s', which contains all the points whose distance from s respects the distance function. Hence, this corresponds to $OQ(D, s', intersect)$, as shown in Figure 4a. Additionally, distance functions are used by *k-Nearest Neighbor Queries*

(Figure 4c). Let $k \in \mathbb{N}$. This query can be formally defined as $kNNQ(D, s, k, dist) = \{o \mid o \in F = \{o_1, o_2, ..., o_{k-1}, o_k\} \land F \subseteq D \land \forall q \in D - F\ \forall 1 \le i \le k : dist(o_i, s) \le dist(q, s)\}$. Its popular variant, the *Reverse k-Nearest Neighbor Query* (Figure 4d), can be formally expressed as $RkNNQ(D, s, k, dist) = \{o \mid o \in D \land s \in kNNQ(D \cup \{s\}, o, k+1, dist)\}$.

Implementations. Spatial index structures also play an important role to optimize metric-based spatial queries, as discussed in several works in the literature (e.g., [14, 42]). Although these queries can be applied to spatial objects of any type, approaches have focused on handling point objects only. In addition, they assume the minimum Euclidean distance among the objects. Thus, the final answer of a metric-based spatial query can be directly obtained by using spatial index structures instead of using the same two-step query processing as the topological-based spatial queries (Section 3.1). For instance, parallel algorithms to deal with metric-based spatial queries with a binary operator using hierarchical index structures [36]. Variants of the *Nearest Neighbor Query*, such as the *k-Nearest Neighbor Query* and the *Reverse k-Nearest Neighbor Query*, have been widely studied in the literature. The authors in [32] empirically study *k-Nearest Neighbor Queries* provided by Spatial Analytics Systems. An approximation method for index structures is proposed in [8] to optimize *Reverse k-Nearest Neighbor Queries*. These queries can also be optimized by neural network models, as proposed in [2].

3.3 Direction-based Spatial Queries

We identify two types of direction-based spatial queries: (i) spatial queries based on (*inter*)*cardinal directions* (e.g., north, south, west, east, and their intermediate directions), and (ii) the retrieval of spatial objects in a particular *angular range*.

Descriptions and Applications. Direction-based spatial queries have been widely studied in the context of location-based services due to the increasing use of mobile applications. Intuitively, direction-based spatial queries aim at retrieving all spatial objects considering a particular direction of the user, such as the four cardinal directions and the intermediate directions [44, 47]. Further, these queries can be generalized to accept any particular angular range when capturing spatial objects, called *Angle-constrained Spatial Query*. Figure 5a depicts an example of direction-based spatial query. Optionally, the distance can be combined with the direction, leading to *Hybrid Spatial Queries* (Section 3.4). Applications of direction-based spatial queries include the processing of user-defined spatial patterns with direction relationships [24].

Formal Definitions. Let c be a direction relationship (e.g., *north*, *south*). Let $\alpha \in]0, 360]$ denote an angle in the space. A spatial query based on the *(inter)cardinal directions* (Figure 5a) can be formally defined as $DCQ(D, s, c) = \{o \mid o \in D \land c(s, o) = true\}$. This can be further formally generalized when using angles, such as $ACQ(D, s, \alpha, \phi) = \{o \mid o \in D \land dir(s, o)\ \phi\ \alpha\}$ where dir is a function that extracts the direction angle of two spatial objects. In this case, any angle can be defined instead of using fixed angles (e.g., *north* would correspond to $\alpha = 90$) and thus, the *Angle-constrained Spatial Query* is formed. The range of acceptable angles can also be

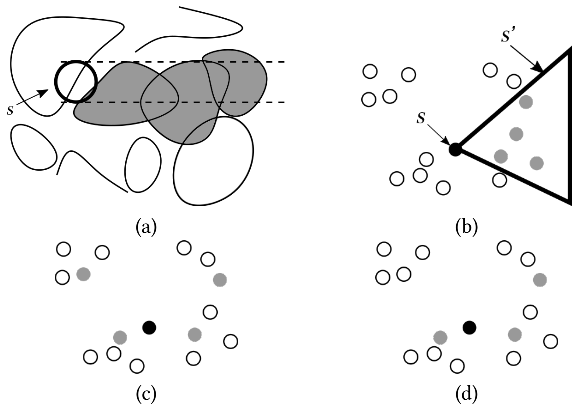

Figure 5: Examples of direction-based and hybrid spatial queries: a direction-based spatial query retrieving objects in the east (a), a *Direction and Distance-based Query* and its correspondence to an *Object Query* (the formed triangle) (b), a *Direction-based Surrounder Query* (c), and a *Reverse Direction-based Surrounder Query* (d). The objects that answer these queries are denoted in dark gray.

specified by extending this definition with other conditions. For instance, $ACQ(D, s, \alpha_1, \alpha_2) = \{o \mid o \in D \land \alpha_1 < dir(s, o) < \alpha_2\}$.

Implementations. Similarly to the other types of queries, spatial index structures are often employed to speed up direction-based spatial queries. In this sense, the use of MBRs to accelerate the processing of direction relationships has been studied in the literature. The authors in [34] discuss how the R-tree can be developed to deal with direction-based spatial queries. Further, the work in [26] leverages the cardinal relations of MBRs in the form of a rectangle algebra for spatial reasoning. The authors in [25] also deploy studies on the semantics of MBRs when processing direction-based spatial queries using R-trees. In this case, the MBR of the search object is rotated, resulting in object-based direction queries. Another example is the work in [46], which employs the filter and refinement steps to process direction-based spatial queries. Its filter step leverages the combination of MBRs with a cone-based model and quad-trees (surveyed in [14]), whereas the refinement step computes the direction relationship.

3.4 Hybrid Spatial Queries

The combination of different spatial relationships in the same spatial query provides a restricted filter when retrieving a set of spatial objects from a given spatial dataset. In this section, we identify some types of spatial queries characterized by this behavior.

Descriptions and Applications. Spatial queries that combine metric and direction relationships have attracted the increasing interest of applications in the world of the IoT. A first combination is the *Direction and Distance-based Query*, which is a hybrid spatial query that retrieves all the objects located in a particular direction (or acceptable angles) and limited to a given distance in relation to a search object. Other combinations pick a variation of the *Nearest Neighbor Query* and apply the notion of direction.

The *Direction-based Surrounder Query* (also known as *Direction-aware Nearest Neighbor Query*) retrieves the nearest objects (i.e., minimum distance) around a spatial object from different directions [18, 23]. It provides the *diversification* of the results if compared to the single-use of direction and/or metric relationships [17]. Further, the concept of *domination* plays an important role. A spatial object dominates another spatial object if they are direction close and the first one is located at a shorter distance from the search object. Optionally, the number of spatial objects returned by the spatial query can also be indicated by k, leading to the *Direction-based k-Nearest Neighbor Query*. Another variation is the *Reverse Direction-based Surrounder Query*, which finds all the spatial objects that have the search object s as one of their direction-based nearest neighbors [16]. Figures 5b, c, and d depict examples of hybrid spatial queries. Examples of applications include digital ecosystems composed of mobile devices that use the aforementioned queries to recommend buildings and services for users [16, 18].

Formal Definitions. By using the functions *dist* and *dir* (Sections 3.2 and 3.3), we are able to define hybrid spatial queries as follows. The *Direction and Distance-based Query* can be defined as $DDQ(D, s, m, \alpha, dist) = \{o | o \in D \wedge dir(s, o) = \alpha \wedge dist(s, o) < m\}$. Note that the angle constraint could be a range in a similar way to the *Angle-constrained Spatial Query*. This query has a correspondence to topological-based queries (Section 3.1). When applying an angle and distance to the search object s, we yield a new spatial object, called s', that represents all points whose direction from s respect a given angular range and distance. Hence, by using s', we can specify $OQ(D, s', intersect)$. Figure 5b shows an example of this case, where the formed s' is a triangular-shaped search object.

To define the other hybrid spatial queries, we define the *dominance relationship* of two spatial objects $o_1, o_2 \in D$ with respect to the search object s as $dist(s, o_1) < dist(s, o_2) \wedge |dir(s, o_1) - dir(s, o_2)| \leq \theta$. This means that o_1 dominates o_2, denoted as $o_1 \prec_\theta o_2$, since o_1 is nearest from s and the directions of o_1 and o_2 are in an acceptable angular range indicated by θ with respect to s. Intuitively, the *Direction-based Surrounder Query* (Figure 5c) retrieves all the spatial objects that are not dominated by any other object. This can be formally defined as $DBSQ(D, s, \theta) = \{o | n \in \mathbb{N} \wedge o \in G = \{o_1, o_2, ..., o_{n-1}, o_n\} \wedge G \subseteq D \wedge \forall 1 \leq i \leq n \nexists q \in D - G : q \prec_\theta o_i\}$. Some variants are possible as follows. If s is a point object, then we obtain the *Point Direction-based Surrounder Query*. If s is a query window, then we obtain the *Range Direction-based Surrounder Query*. Further, we can limit the number of retrieved objects by considering the k nearest objects only, that is, $kDBSQ(D, s, \theta, k) = \{o | \wedge o \in G = \{o_1, o_2, ..., o_{k-1}, o_k\} \wedge G \subseteq DBSQ(D, s, \theta) \wedge \forall q \in DBSQ(D, s, \theta) - G \, \forall 1 \leq i \leq k : dist(o_i, s) \leq dist(q, s)\}$. Another variant is the *Reverse Direction-based Surrounder Query*, which retrieves the spatial objects that also see the search object as their direction-based nearest neighbors (Figure 5d). Formally, it can be expressed as $RDSQ(D, s, \theta) = \{o \,|\, o \in D \wedge s \in DBSQ(D \cup \{s\}, o, \theta)\}$.

Implementations. The R-tree has been widely used to reduce the search space of the aforementioned queries, such as the application of heuristics to traverse in its index nodes [23]. The authors in [30] present algorithms to prune spatial objects during the processing of *Visible k-Nearest Neighbor Queries*, which take into account the angle of visibility when retrieving objects. Search algorithms in [18] do not employ index structures. Instead, such algorithms are based on the notions of direction order and adjacent objects to speed up *Direction-based Surrounder Queries*. The authors in [16] leverage R*-trees to deal with angular boxes. In summary, they transform the search object with the angular range into a region object which in turn is the search object of a topological-based spatial query (as previously discussed). The true candidates are then selected based on the directional closeness condition. In [17], the R-tree is employed to discard MBRs and two strategies to process *Direction-based Surrounder Query* are proposed. The first strategy yields the objects that satisfy the conditions of the query, whereas the second one is much faster but may include false candidates (i.e., an approximate algorithm).

3.5 Spatial Joins

In this section, we comprehend how spatial relationships can be employed to combine and associate two or more spatial datasets, leading to *spatial joins*.

Descriptions and Applications. The spatial join is a typical operation when associating multiple spatial datasets and has been widely studied in the literature [3, 14, 21]. Considering our spatial dataset D and another spatial dataset E, a spatial join query retrieves the pairs of spatial objects $(o, o') \in D \times E$ that satisfy a given spatial relationship or a combination of spatial relationships. Table 2 shows examples of spatial join queries. Intuitively, the *Topological-based Join Query* finds all the pairs that satisfy a given topological relationship. For instance, all the overlapping pairs. The *Distance-based Join Query* retrieves all the pairs that are spatially separated according to a given distance. This query can further be specialized. The *k-Nearest Neighbors Join Query* retrieves the k closest neighbors of each spatial object of the first spatial dataset. In the same context, the *k-Closest Pairs Join Query* yields the k pairs with the minimum distances among all possible pairs. Finally, the *Direction-based Join Query* applies direction relationships to retrieve all the pairs with the same angular range. The spatial join query can be viewed as a general case of the other types of spatial queries in the sense that the search object s is a set of spatial objects. The combination of two or more spatial datasets is a common task of spatial database and GIS applications [3].

Formal Definitions. By using a topological relationship p and the functions *dist* and *dir* for computing the distance and direction of two spatial objects (Sections 3.1 to 3.3), we can define a formal template for spatial join queries as $D \bowtie E = \{(o, o') \,|\, o \in D \wedge o' \in E \wedge [\texttt{cond}]\}$, where $[\texttt{cond}]$ is replaced by a set of conditions as depicted in the third column of Table 2. This template can be further extended to deal with combinations of spatial relationships in a similar approach as the *Hybrid Spatial Queries* (Section 3.4).

Implementations. Spatial index structures have been also used as the main technique to optimize spatial joins, which are costly operations due to the complex nature of spatial objects and their relationships. The authors in [21] survey several techniques by varying how the spatial datasets are handled and stored in storage

Table 2: Some variants of spatial join queries. JQ stands for *Join Query.*

Spatial Join Query	Spatial Relationship	Conditions
Topological-based JQ	topological	$p(o, o') = true$
Distance-based JQ	metric	$dist(o, o')\ \phi\ m$
k-Nearest Neighbors JQ	metric	$o' \in kNNQ(D, o, k, dist)$
k-Closest Pairs JQ	metric	$F = \{(o_1, o'_1), (o_2, o'_2), \ldots, (o_{k-1}, o'_{k-1}), (o_k, o'_k)\} \wedge (o, o') \in F \subseteq D \times E \wedge$ $\forall\, 1 \leq i \leq k\ \forall\, 1 \leq j \leq k\ \forall\, (p_i, q_j) \in D \times E - F : dist(p_i, q_j) \geq dist(o_k, o'_k) \geq$ $dist(o_{k-1}, o'_{k-1}) \geq \ldots \geq dist(o_2, o'_2) \geq dist(o_1, o'_1)$
Direction-based JQ	direction	$dir(o, o')\ \phi\ \alpha$

devices. That is, these techniques cover the following cases: (i) non-indexed spatial datasets, (ii) only one spatial dataset is indexed, and (ii) both spatial datasets are indexed. Optimizations for spatial joins with metric relationships are proposed in [11, 15]. More precisely, in [11] the authors propose indexing and algorithms based on the R-tree for *k-Closest Pairs Join Query*, whereas in [15] algorithms for optimizing the *Distance-based Join Query* in SpatialHadoop and LocationSpark applications. Another survey highlighting future researches for optimizing spatial joins is presented in [3]. Further, the authors in [41] examine different ways to process spatial joins (possibly with temporal predicates) by using the main memory only, which is a common situation in the context of IoT.

4 DISCUSSIONS, CHALLENGES, AND FUTURE TRENDS

In this section, we discuss challenges and future trends behind concepts, definitions, and implementations of spatial queries. As a result, we identify four topics that can serve as a foundation for advances in spatial information retrieval in the world of IoT. These topics are: (i) the identification and abstraction of spatial relationships, (ii) the retrieval of spatial objects that are characterized by the spatial fuzziness or/and that vary over time, (iii) the correspondence and properties among the types of spatial queries, and (iv) the perspectives when optimizing spatial queries.

The first topic relates to the importance of spatial relationships when designing and implementing spatial queries. In fact, spatial queries often leverage topological, metric, and direction relationships, resulting in the types of spatial queries introduced in this paper. On the other hand, we can identify other spatial relationships that are not topological, metric, or direction relationships. For instance, a spatial relationship that computes the *similarity* of two spatial objects in terms of spatial data type, geometric format, and the number of points. Hence, future research topics may include the design of spatial queries based on other types of spatial relationships and conduct studies on their applicability.

With respect to the second topic, we are interested in how the time and the spatial fuzziness impact spatial information retrieval. This paper has dealt with the study and comprehension of spatial queries on *static* and *crisp* spatial objects since their locations do not change over time and their positions, boundaries, and interiors are well-known and precisely defined in the space. Despite the importance of static and crisp spatial objects in real spatial database and GIS applications, many spatial phenomena can *change over time* or can be characterized by *spatial fuzziness* [6, 7, 27, 37]. The definition of *spatiotemporal data types* and *fuzzy spatial data types* have adequately represented such phenomena. Hence, the definition of useful spatial queries for retrieving spatiotemporal objects and fuzzy spatial objects has been required in modern and advanced applications. This leads to the challenge of designing spatial queries that possibly combine: (i) space, (ii) time, and (iii) uncertainty. In general, current approaches in the literature (e.g., [45]) extend the queries that consider the space only (e.g., the *Range Query*) in order to handle either the space over time or the spatial fuzziness. On the other hand, there is a need for developing spatial queries with other combinations able to handle different types of objects. For instance, future research topics may include techniques to process spatial queries on a spatial dataset storing crisp and static spatial objects, spatiotemporal objects, and fuzzy spatial objects.

The third topic is related to the identification of situations in which some types of spatial queries return the same result as other types of spatial queries. This paper has presented an initial study on possible correspondences, e.g., how to transform a metric-based or direction-based spatial query into a topological-based spatial query. This kind of transformation is interesting for optimizing a spatial query by rewriting it as a correspondent query that can be processed in a faster way. An example of this situation is given by the authors in [29] for the spatial join context. On the other hand, more theoretical studies on this topic are needed, which can result in a *transformation model* of spatial queries. A future research topic in this direction leads to exploit the identified correspondences when creating query execution plans in spatial database systems.

Finally, the fourth topic is the application of novel optimization strategies for spatial query processing. This paper has sketched some works in the literature that employ spatial index structures (e.g., R-tree and variants) to reduce the search space of spatial queries. Spatial indexing is a core issue in spatial database systems and GIS that has taken into account the constant evolution of the hardware. For instance, spatial index structures for (i) Solid State Drives (e.g., [4, 5]), (ii) parallel processing (e.g., [36]), and (iii) distributed environments (e.g., [32]). Spatial indexing based on machine learning models is another possibility of optimization when discovering data access patterns (e.g., [2]). Hence, future research topics may include the two perspectives to create spatial index structures: (i) modern hardware, and (ii) learned index structures.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have provided a comprehensive survey on typical spatial queries processed by spatial database and GIS applications. For analyzing their concepts and implementations, we have identified topological-based spatial queries, metric-based spatial queries, and direction-based spatial queries, which are respectively based on the well-known topological relationships (e.g., overlap, inside), metric relationships (e.g., distance), and direction relationships (e.g., north, south). We also defined Hybrid Spatial Queries as the combination of two or more different types of spatial relationships, and Spatial Join Queries as the combination of two or more spatial datasets. In addition, discussions on the challenges and future trends have addressed the importance of spatial relationships when developing spatial queries and their implementations. To the author's knowledge, this paper provides the first overview of types of spatial queries, considering different spatial relationships, that can be deployed in the world of IoT and digital ecosystems.

Future work will deal with three main topics, considering the future trends introduced in this paper. They are: (i) the analysis of spatial queries on spatiotemporal data by considering the combination of time and geographical location of objects [37], (ii) the comprehension of types of spatial queries for retrieving fuzzy spatial objects [6, 7], and (iii) the study of a taxonomy encompassing spatial queries on these different spatial data types in distinct environments (e.g., mobile applications, modern hardware).

REFERENCES

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. 1990. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *ACM SIGMOD Int. Conf. on Management of Data*. 322–331.

[2] M. Berrendorf, F. Borutta, and P. Kröger. 2019. k-Distance Approximation for Memory-Efficient RkNN Retrieval. In *Int. Conf. on Similarity Search and Applications*. 57–71.

[3] P. Bouros and N. Mamoulis. 2019. Spatial Joins: What's Next? *SIGSPATIAL Special* 11, 1 (2019), 13–21.

[4] A. C. Carniel, R. R. Ciferri, and C. D. A. Ciferri. 2019. A Generic and Efficient Framework for Flash-aware Spatial Indexing. *Information Systems* 82 (2019), 102–120.

[5] A. C. Carniel, R. R. Ciferri, and C. D. A. Ciferri. 2020. FESTIval: A versatile framework for conducting experimental evaluations of spatial indices. *MethodsX* 7 (2020), 1–19.

[6] A. C. Carniel and M. Schneider. 2016. A Conceptual Model of Fuzzy Topological Relationships for Fuzzy Regions. In *IEEE Int. Conf. on Fuzzy Systems*. 2271–2278.

[7] A. C. Carniel and M. Schneider. 2018. Spatial Plateau Algebra: An Executable Type System for Fuzzy Spatial Data Types. In *IEEE Int. Conf. on Fuzzy Systems*. 1–8.

[8] G. Casanova, E. Englmeier, M. E. Houle, P. Kröger, M. Nett, E. Schubert, and A. Zimek. 2017. Dimensional Testing for Reverse K-Nearest Neighbor Search. *VLDB Endowment* 10, 7 (2017), 769–780.

[9] S. Cicerone and P. Di Felice. 2004. Cardinal directions between spatial objects: the pairwise-consistency problem. *Information Sciences* 164, 1 (2004), 165–188.

[10] E. Clementini and P. Di Felice. 2000. Spatial Operators. *ACM SIGMOD Record* 29, 3 (2000), 31–38.

[11] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. 2000. Closest Pair Queries in Spatial Databases. *ACM SIGMOD Record* 29, 2 (2000), 189–200.

[12] M. J. Egenhofer and A. R. B. M. Shariff. 1998. Metric Details for Natural-Language Spatial Relations. *ACM Trans. on Information Systems* 16, 4 (1998), 295–321.

[13] K. A. Eldrandaly, M. Abdel-Basset, and L. A. Shawky. 2019. Internet of Spatial Things: A New Reference Model With Insight Analysis. *IEEE Access* 7 (2019), 19653–19669.

[14] V. Gaede and O. Günther. 1998. Multidimensional Access Methods. *ACM Comput. Surveys* 30, 2 (1998), 170–231.

[15] F. García-García, A. Corral, L. Iribarne, M. Vassilakopoulos, and Y. Manolopoulos. 2020. Efficient distance join query processing in distributed spatial data management systems. *Information Sciences* 512 (2020), 985–1008.

[16] X. Guo, Y. Ishikawa, Y. Xie, and A. Wulamu. 2017. Reverse direction-based surrounder queries for mobile recommendations. *World Wide Web* 20 (2017), 885–913.

[17] X. Guo and X. Yang. 2019. Direction-Aware Nearest Neighbor Query. *IEEE Access* 7 (2019), 30285–30301.

[18] X. Guo, B. Zheng, Y. Ishikawa, and Y. Gao. 2011. Direction-based surrounder queries for mobile recommendations. *The VLDB Journal* 20, 1 (2011).

[19] R. H. Güting. 1994. An Introduction to Spatial Database Systems. *The VLDB Journal* 3, 4 (1994), 357–399.

[20] A. Guttman. 1984. R-trees: A Dynamic Index Structure for Spatial Searching. In *ACM SIGMOD Int. Conf. on Management of Data*. 47–57.

[21] E. H. Jacox and H. Samet. 2007. Spatial Join Techniques. *ACM Trans. on Database Systems* 32, 1 (2007), 1–44.

[22] J. Jin, N. An, and A. Sivasubramaniam. 2000. Analyzing range queries on spatial data. In *Int. Conf. on Data Engineering*. 525–534.

[23] K. C. K. Lee, W.-C. Lee, and H. V. Leong. 2010. Nearest Surrounder Queries. *IEEE Trans. on Knowledge and Data Engineering* 22 (2010), 1444–1458.

[24] Y. Li, Y. Fang, R. Cheng, and W. Zhang. 2019. Spatial Pattern Matching: A New Direction for Finding Spatial Objects. *SIGSPATIAL Special* 11, 1 (2019), 3–12.

[25] X. Liu, S. Shekhar, and S. Chawla. 2003. Object-based directional query processing in spatial databases. *IEEE Trans. on Knowledge and Data Engineering* 15, 2 (2003), 295–304.

[26] Y.-S. Liu and Z.-X. Hao. 2005. The cardinal direction relations and the rectangle algebra. In *Int. Conf. on Machine Learning and Cybernetics*. 3115–3118.

[27] A. R. Mahmood, S. Punni, and W. G. Aref. 2019. Spatio-Temporal Access Methods: A Survey (2010 - 2017). *GeoInformatica* 23, 1 (2019), 1–36.

[28] M. McKenney, A. Pauly, R. Praing, and M. Schneider. 2005. Dimension-refined Topological Predicates. 240–249.

[29] E. Mella, M. A. Rodríguez, L. Bravo, and D. Gatica. 2019. Query rewriting for semantic query optimization in spatial databases. 23 (2019), 79–104.

[30] S. Nutanong, E. Tanin, and R. Zhang. 2010. Incremental Evaluation of Visible Nearest Neighbor Queries. *IEEE Trans. on Knowledge and Data Engineering* 22, 5 (2010), 665–681.

[31] OGC 2011. OpenGIS Implementation Standard for Geographic Information - Simple Feature Access - Part 1: Common Architecture. Open Geospatial Consortium. https://www.ogc.org/standards/sfa.

[32] V. Pandey, A. Kipf, T. Neumann, and A. Kemper. 2018. How Good Are Modern Spatial Analytics Systems? *VLDB Endowment* 11, 11 (2018), 1661–1673.

[33] D. Papadias, T. Sellis, Y. Theodoridis, and M. J. Egenhofer. 1995. Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-Trees. In *ACM SIGMOD Int. Conf. on Management of Data*. 92–103.

[34] D. Papadias, Y. Theodoridis, and T. Sellis. 1994. The retrieval of direction relations using R-trees. In *Int. Conf. on Database and Expert Systems Applications*. 173–182.

[35] D. J. Peuquet and Z. Ci-Xiang. 1987. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition* 20, 1 (1987), 65–74.

[36] G. Roumelis, P. Velentzas, M. Vassilakopoulos, A. Corral, A. Fevgas, and Y. Manolopoulos. 2019. Parallel processing of spatial batch-queries using xBR$^+$-trees in solid-state drives. *Cluster Computing* (2019).

[37] M. Schneider. 2017. Spatial and Spatiotemporal Data Types as a Foundation for Representing Space-Time Data in GIS. In *Handbook of Research on Geographic Information Systems Applications and Advancements*, S. Faiz and K. Mahmoudi (Eds.). IGI Global, 1–28.

[38] M. Schneider and T. Behr. 2006. Topological Relationships between Complex Spatial Objects. *ACM Trans. on Database Systems* 31, 1 (2006), 39–81.

[39] J. Shen, M. Chen, and X. Liu. 2018. Classification of topological relations between spatial objects in two-dimensional space within the dimensionally extended 9-intersection model. *Transactions in GIS* 22, 2 (2018), 514–541.

[40] R. I. Silva, D. F. Macedo, and J. M. S. Nogueira. 2014. Spatial query processing in wireless sensor networks – A survey. *Information Fusion* 15 (2014), 32–43.

[41] B. Sowell, M. V. Salles, T. Cao, A. Demers, and J. Gehrke. 2013. An Experimental Analysis of Iterated Spatial Joins in Main Memory. 6, 14 (2013), 1882–1893.

[42] D. Taniar and W. Rahayu. 2013. A taxonomy for nearest neighbour queries in spatial databases. *J. Comput. System Sci.* 79, 7 (2013), 1017–1039.

[43] F. Tauheed, L. Biveinis, T. Heinis, F. Schurmann, H. Markram, and A. Ailamaki. 2012. Accelerating range queries for brain simulations. In *Int. Conf. on Data Engineering*. 941–952.

[44] Y. Theodoridis, D. Papadias, and E. Stefanakis. 1996. Supporting direction relations in spatial database systems. In *Int. Symp. on Spatial Data Handling*.

[45] Y. Wang, X. Li, X. Li, and Y. Wang. 2013. A survey of queries over uncertain data. *Knowledge and Information Systems* 37, 1 (2013), 485–530.

[46] Z. Wang and H. Yan. 2012. Spatial queries based on direction relations with the case of vector rivers data. In *Int. Symp. on Geomatics for Integrated Water Resource Management*. 1–4.

[47] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. 2003. Location-Based Spatial Queries. In *ACM SIGMOD Int. Conf. on Management of Data*. 443–454.

[48] L. Zhigang, Liangtian, and Y. Wunian. 2010. Research of GIS-based urban disaster emergency management information system. In *Int. Conf. on Computer and Communication Technologies in Agriculture Engineering*. 484–487.