Analysing Temporal Evolution of Complex Data Using Similarity Queries

Isis C. O. S. Fogaça, Renato Bueno

Federal University of São Carlos (UFSCar)
São Carlos – SP – Brazil
{isiscarolyne@gmail.com, renato@ufscar.br}

Abstract.

Regardless of the data domain, there are applications that must track the temporal evolution of data elements. Based on the instances present in the database, the goal is to estimate the state of a given element at a different time instant from those available in the database. This kind of task is common in many database application domains, such as medicine, meteorology, agriculture, financial, and others. In content-based retrieval with complex data (such as images, sounds and videos), data are usually represented in metric spaces, where only the distances between elements are available. Without dimensional coordinates, it is not possible simply to add a time dimension for trajectory estimation in these spaces, as is the case in multidimensional spaces. In this article we propose to map the metric data to a multidimensional space so that we can estimate the element's status at a given time instant, based on known states of the same element. As it is not possible to create the complex data equivalent to its estimated position in mapped space, we propose to apply similarity queries using this position as query center. Then, we estimate how this element would be, retrieving the real data elements present in the database that are close to the estimate. In this article, in addition to the nearest neighbor query (k-NN), we propose to use two other queries: kAndRange and kAndRev. With both methods, we aim to prune non-relevant elements from the query results, retrieving only the elements that are really close to the estimates. We present experiments with different query scenarios, evaluating the effects of varying input parameters of the proposed queries.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Image databases; H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords: similarity queries, complex data, temporal evolution

1. INTRODUCTION

Complex data (such as images, videos, sounds and time series) commonly do not present the total ordering relationship as the conventional data (numbers and short texts). Because of this, they cannot be compared and retrieved at the same manner, so they are compared using their similarity. Considering images as an example of complex data, the Content-based Image Retrieval (CBIR) systems extract the features of the images, usually representing their shapes, texture or colors, and store these features in feature vectors that represent the images. In similarity queries, these feature vectors are compared using distance functions. The search space is in general represented in a metric space, where only the images and the distances between them are available.

The need to manage temporal information applies to many application domains in databases, such as medicine, agriculture, meteorology and others. However, one of the big challenges that we have in metric data is the analysis of the temporal evolution, since it is not possible to do this analysis as we do with multidimensional data. In multidimensional data, we can estimate an element´s position in

This work has been supported by the Brazilian research agencies FAPESP, CNPq and CAPES.

some time instant using its known positions in other time instants as reference, then we can trace this element's trajectory and finally represent its evolutionary behavior through a temporal axis variation. Thereby it is possible to estimate the element's position in different time instants. We cannot perform these estimates in metric spaces as we do not have the geometric information regarding the data, as dimensional axes or coordinates. Only the distances between the metric elements are available.

In this article, we propose mapping the metric data into multidimensional spaces and, once we have the data in this space, we can analyze their evolutionary behavior and estimate their trajectories over the time. However, we are not able to create an element in the metric space from the multidimensional space estimate. For example, it is not possible to rebuild an image from its estimated position in the mapped space. Therefore, to estimate how this element would be in the original space, we use the results from similarity queries performed over the estimated element in the mapped space, retrieving the real data elements present in the database that are close to the estimate.

We evaluated three types of similarity queries applied to the estimated position in the mapped space. Initially, we proposed the use of simple k-nearest neighbor queries (*k-NN*). However, when we use this query, we always get the same number of elements, retrieving elements that are not near enough from the estimated position and are not relevant to the user.

To provide better results to the user regarding the query results, we propose to use two other types of queries. In both cases, the goal is to retrieve only the elements that are really close to the estimates, decreasing the number of non-relevant elements in the query result. The proposed methods are based on the *kAndRange* query (an intersection between *k-NN* and range queries) and an approach based on *reverse k-NN* (the query where we retrieve all the elements that have the query center as one of the nearest neighbors). To both proposals, we present in this article some of the experiments that demonstrated the increased precision of the query results when we compare them to the *k-NN* queries.

This article is an extended version of [de Sousa Fogaça and Bueno 2020] presented at the 35th Brazilian Symposium on Databases. In this article we present new experiments performed to evaluate the effects of varying the input parameters of the proposed queries: the radius for the *kAndRange* query and the number of nearest neighbors at each step of the *kAndRev* query (k_1 and k_2). We also present new experiments in scenarios with few relevant elements (less than the value of k). This strategy allows us to evaluate the ability of pruning non-relevant elements with the *kAndRange* and *kAndRev* queries.

In Section 2 we present some basic concepts and related works. Section 3 describes the proposed approach to analyze the evolution of metric data over time and Section 4 presents the performed experiments. Finally, Section 5 presents the conclusions of this article.

2. BASIC CONCEPTS AND RELATED WORKS

2.1 Metric Spaces and Similarity Queries

Complex data do not present the total ordering relationship as we find in conventional data. That is why many relational operators are not applicable to them. Even equality comparison in general is not useful. These data domains are usually represented in metric spaces and their elements are compared by their similarity [Chávez et al. 2001].

A metric space is defined as $\langle \mathbb{S}, d \rangle$, where $\mathbb{S}$ represents the elements and d is the distance function, or metric, defined as $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$. This distance function must provide the properties of symmetry, non-negativity and triangular inequality [Chávez et al. 2001]. In metric spaces we have only the data and the distances between them.

The most common similarity query operators are the *range query* and the *k-nearest neighbors query* (*k-NN*). Both queries are performed from a query center. In a range query, we retrieve all the elements

that are inside a radius defined in the query. In *k-NN* query, the k nearest neighbors from query center are returned. There are many variations for these operators, such as *reverse k-NN* [Tao et al. 2006] and *kAndRange* [Arantes et al. 2004], that are used in this work.

2.2 Metric-temporal space

In [Bueno et al. 2009], a metric-temporal model was proposed, where complex data are compared by similarity using temporal information. The metric-temporal model projects the metric and the temporal distances separately, generating a metric component and a temporal one. The metric component is defined as a metric space $\langle \mathbb{S}, d_s \rangle$, where $\mathbb{S}$ represents the dataset that belong to the domain application, and $d_s : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ is a metric that calculates the dissimilarity between the elements in the domain. The temporal component is defined as a metric space $\langle \mathbb{T}, d_t \rangle$ where $\mathbb{T}$ represents the time measures and $d_t : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{R}^+$ is the metric to calculate the dissimilarity between two elements of $\mathbb{T}$. The metric-temporal space is defined as a pair $\langle \mathbb{V}, d_v \rangle$ where $\mathbb{V} = \mathbb{S} \times \mathbb{T}$ and $d_v : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{R}^+$ represents the metric between the elements from the metric-temporal space. The d_v metric is composed by d_s and d_t aggregation in a metric product, where we need to define the contribution of metric and temporal components to calculate the similarity.

The metric-temporal model allows to introduce temporal information to the similarity queries. Therefore, by considering the data distribution in the metric space that represents the metric component, we can approximate the elements that have more temporal similarity and put away others that have bigger temporal distance. However, it is not possible to analyze evolutive behavior of data over the time, verifying for example the trajectory, once the data are still in a metric space, making analysis and geometric estimates based on dimensional coordinates impossible.

3. ANALYSIS OF TEMPORAL EVOLUTION OF COMPLEX DATA

In the approach proposed in this work, we intend to analyze the evolution of metric data over time. We aim to estimate how an element will be in a defined instant of time from other information existent in the database. To make it possible, we proposed to map the metric data into a multidimensional space, performing interpolations and extrapolations of values in their positions in space. After mapping the data, each element can be represented in a position inside the multidimensional space. Then we can perform estimates and queries in the mapped space considering also a temporal dimension.

It is not possible to estimate the complex data in an instant of time (in a metric space), but its position inside the multidimensional space where the metric data were mapped. It is not possible to generate the equivalent complex data in the metric space from this position (like rebuild an image). The proposal is to perform similarity queries using the estimated position as a query center and then retrieve the elements existent in the database that are closer to the estimate.

Let's consider the monitoring of a patient using medical exam images as an example. These images are available in a database that contains images of many patients in different phases of treatment. They are represented in Figure 1 as points in a bidimensional space, considering that this is already the multidimensional space where we mapped the metric data. For patient P_A we have two images: one image taken with 2 months of treatment ($t = 2$) and another one taken with 4 months of treatment ($t = 4$). We also have images for patient P_B when the treatment was started ($t = 0$) and with 12 months of treatment ($t = 12$).

In the mapped space, from the positions of the mapped elements that represent the patient's P_A images with $t = 2$ and $t = 4$, we estimate the image position regarding patient P_A with 8 months of treatment. Then we perform a similarity query using this estimated position as a query center (*5-NN* in the case of Figure 1), returning the closest mapped elements, and retrieving the images associated with them. Similar approach was applied to patient P_B, estimating the image position with 15 months of treatment.

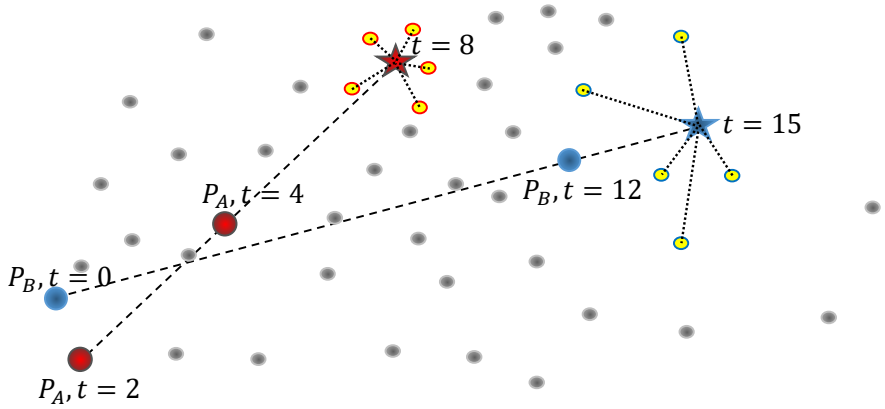

Fig. 1. Example of k nearest neighbors query using the estimated positions of patients P_A and P_B as a query center.

In the experiments that we show in this article, we used only 2 elements of reference (two images of the same object in different timestamps) to estimate the position of this object in a third timestamp using linear interpolation/extrapolation.

In the experiments presented in [Paiva et al. 2020], another work in which we present visualization techniques for queries using temporal estimates, the use of more reference elements (2, 3 and 4) was evaluated. However, the estimates using only two elements provided better results in interpolation and extrapolation.

In this work, three different similarity queries applied to the estimate of the element in mapped space were appraised: *k-NN*, *kAndRange* and an adaptation of the *reverse k-NN*. In the next Section we discuss the metric data mapping into multidimensional space, and in Section 3.2 we present the different proposals for the similarity query over the estimated position.

3.1 Metric data mapping into a multidimensional space

In a multidimensional space we can estimate the elements' trajectories based on their values at each coordinate (dimension) and the time values associated to them. This is not possible in metric data, where only distances between the pairs of elements are available. In this research, we propose to map this data into a multidimensional space to estimate the temporal evolution of metric data. To do this, we have to define which method will be used to perform the mapping and also the number of dimensions of the mapped space.

There are several methods to perform the metric data mapping for multidimensional spaces presented in literature [Hjaltason and Samet 2003]. In this article we used and evaluated two methods: the Fastmap [Faloutsos and Lin 1995], whose complexity is $O(kN)$, and the Multidimensional Scaling (MDS) [Cox and Cox 2008], that is a more costly method, but capable of maintaining more faithfully the distribution of distances between the elements in the mapped space.

To define the number of dimensions in the multidimensional space that the metric data will be mapped to, we used the concept of intrinsic dimension of the dataset, by seeking the number of dimensions that are needed to immerse the elements in a multidimensional space keeping the distances between them [Chávez et al. 2001]. The intrinsic dimension indicates the minimum number of attributes that are necessary to represent a dataset [Sousa et al. 2007]. There are a variety of algorithms that can be applied to estimate the intrinsic dimension of datasets in literature [Bustos et al. 2015; Traina Jr. et al. 2010; Traina Jr. et al. 2010]. In this article, the intrinsic dimension was estimated using the distance exponent proposed on [Traina Jr. et al. 2000].

3.2 Similarity queries applied to the estimated position

As discussed earlier, in multidimensional space we can estimate the position of an element at a certain instant of time based on the known positions of the same element at other times that exist in the database. However, it is not possible to create the complex data in the metric space from the estimate. In this article, we propose to conduct similarity queries using the estimated position in the mapped space as a center, in order to return the elements that are close to the estimate.

3.2.1 ***k-NN queries.*** The first proposal is the query to the nearest neighbors (*k-NN*). However, we need to consider that, in this type of query, all the k nearest neighbors are retrieved, no matter their distances from the estimated element. Hence, on this approach, non-relevant elements can be returned to the user. Consider here the same example of patients monitoring described in Section 3, where we estimated the positions for P_A in $t = 8$ and for P_B in $t = 15$. These estimated positions are used as a query center to find the 5 nearest neighbors, as illustrated in Figure 1. The 5 images returned for patient P_A are visibly near the estimate in $t = 8$. However, we can also see that some of the images retrieved for P_B in $t = 15$ are relatively farther. Since the 5 closest elements are returned disregarding how far they are from the element estimated, the distant elements can be returned and probably they are not relevant for the query.

To improve the user's satisfaction with the performed queries, decreasing the quantity of elements that are possibly not relevant in the analysis, we propose to use two other queries on the estimated position: *kAndRange* [Arantes et al. 2004] and an approach of *reverse k-NN* [Tao et al. 2006].

3.2.2 ***kAndRange queries.*** The query operator *kAndRange* [Arantes et al. 2004] not only searches for the nearest neighbors, but also allows us to delimitate the maximum distance wanted for the returned objects. In other words, it returns the nearest neighbors if they are within a certain range radius.

In this proposal, a *kAndRange* query is performed from the estimate in mapped space, and the range radius is set according to the domain of application, with the purpose of preventing distant elements to be returned, according to the data distribution. This range radius should reflect the average radius of a *k-NN* query on the dataset. That value can be estimated by a sample, or else it can be estimated considering the fractal distribution of the dataset [Vieira et al. 2007; Traina Jr. et al. 2010].

To illustrate the use of *kAndRange* queries, in Figure 2 is presented an example with the same scenario presented in Figure 1 but limiting the result of the *k-NN* query by the average radius of a *5-NN* query in the dataset. In that case, only two images would be returned for P_B.

3.2.3 ***kAndRev: Approach of the Reverse k-NN queries.*** In a reverse k nearest neighbor query (*Reverse k-NN*), all the elements of the database are evaluated and those elements that have the query center as one of their k nearest neighbors are returned [Tao et al. 2006]. In this work, we used an approach from *Reverse k-NN* operator, by evaluating only the elements returned in the *k-NN* query that was applied to the estimated position.

The approach proposed is carried out in two steps. In the first step, the k_1*-NN* query is done using the estimated element as the query center. In the second one, we perform k_2*-NN* queries using each one of the elements returned in the first step as a query center. In case an element returned in the first step does not have the estimated element as one of its k_2-nearest neighbors, it is considered as a non-relevant element, and is then discarded. Otherwise, the element is considered relevant.

Although the same k value can be used for the first (k_1) and second (k_2) steps of *kAndRev* query, these values can be defined separately, aiming to prioritize precision or recall of queries, as presented in experiments.

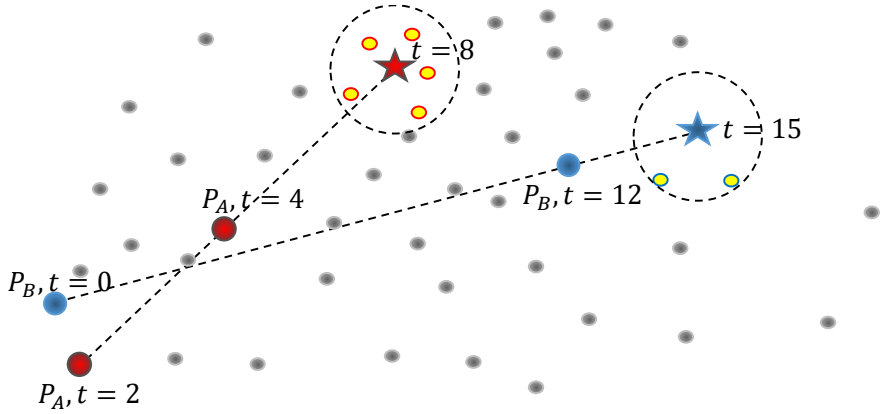

Fig. 2. A *kAndRange* query example, with $k = 5$. For P_A, the five nearest neighbors would be retrieved, but for P_B, only two of them.

4. EXPERIMENTS

To execute the experiments, we used the ALOI[1] image dataset [Geusebroek et al. 2005], which consists of a set of 1,000 objects, photographed at 72 angles of vision, with a 5-degree variation. From this set, we use the images of all 1,000 objects, but at 10 angles of rotation, varying from 0 to 45 degrees. Each rotation angle was intended to represent the variation of time, representing 10 time instants with a difference of 5 time units. The images are classified according to the object photographed. Each image is identified individually by its identification class (ID) and its time, as in the example illustrated in Figure 3.

The feature vectors that were extracted from these images represent the gray levels histograms in 256 bins. The choice of a feature vector that can be represented, without mapping, in a multidimensional space occurred just to verify the influence of mapping on precision of estimates. Which means that the feature vector itself can be used in the estimate of trajectories, allowing us to evaluate possible distortions due to the mapping.

4.1 Evaluation of data mapping

The goal of this first set of experiments is to evaluate the quality of the mapping, checking to see if the data's proximity distribution is maintained in multidimensional space. The data was mapped using the *Fastmap* [Faloutsos and Lin 1995] and *MDS* [Cox and Cox 2008].

To define the number of dimensions of mapped space, we considered the intrinsic dimension of the dataset [Traina Jr. et al. 2000]. The value we found was 9, but in order to favor the maintenance of the original distribution of the data considering mapping methods approximations, we considered 10 dimensions. We also performed data mappings into 3 dimensions in order to check the results

[1]available at https://aloi.science.uva.nl/

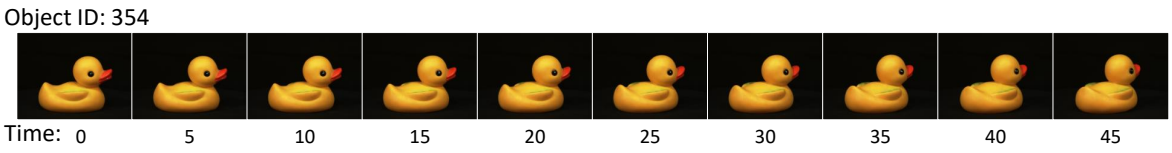

Fig. 3. Example of the ALOI image dataset.

obtained in an even lower dimensionality and easily visualizable case.

To evaluate the quality of mapping, we verified if the distribution of the elements in the original space was held in the mapped space. For every single element of each dataset (original and mapped spaces), we performed a search for the 10 nearest neighbors (considering that we have 10 images of each of the 1,000 photographed objects). The answers of the queries were evaluated through precision and recall curves, displayed in Figure 4, where the answers of the original dataset are considered to be the correct ones. For both mappings, the *MDS* algorithm reaches better results and, as expected, mapping for 10 dimensions had better results than for 3 dimensions, being this selected to continue to the next experiments. The average precision for data mapping in 10 dimensions with the *MDS* algorithm was close to 98%.

From the data mapping, all the experiments of the next sections were performed using the three datasets: original, mapped to 10 dimensions with MDS and mapped to 10 dimensions with Fastmap. Throughout the text, they will be named respectively as *Hist256*, *MDS10* and *Fastmap10*. In all experiments, we used the Euclidean distance function.

4.2 *k-NN* query evaluation

The estimates were performed at different time instants, in the following way: by having two instances of the same object in different times, we estimated the values of every coordinate in a third time instant, through linear interpolation/extrapolation. Using this estimated position as a query center, we performed a query for the 10 nearest neighbors, in order to return images close to the estimate at that time point. To evaluate the quality of the estimates, the results were compared to the 10 nearest neighbors of the real element at the same time used for the estimate, since the element exists in the original dataset (that is the real image of the object in the time instant used for the estimate).

In the Figure 5 we see an example of evaluation of the estimates, where: for an object X in timestamps 5 and 15, we estimated this same object in timestamp 25, using this third timestamp estimate as the center for a 10-NN query. Then, we also perform a *10-NN* query using the real object X in timestamp 25, that exists in the original dataset. The results of both queries are compared in order to know how close the results were. In other words: how the estimate is close to the real object.

The estimates were performed for the 1,000 objects of the dataset. Four levels of time distance were considered between the instants of the elements used for estimates. These levels have been identified as 1, 2, 3 and 4 for the *past*, *intermediate* and *future* times, and they represent, respectively, a difference of 20, 15, 10 and 5 units of time between the reference elements, as shown in Figure 6. The estimates evaluation for each dataset are presented in precision and recall graphics, shown in Figures 7, 8 and 9. Despite variations in precision levels, the behavior of the curves was similar to the same query in

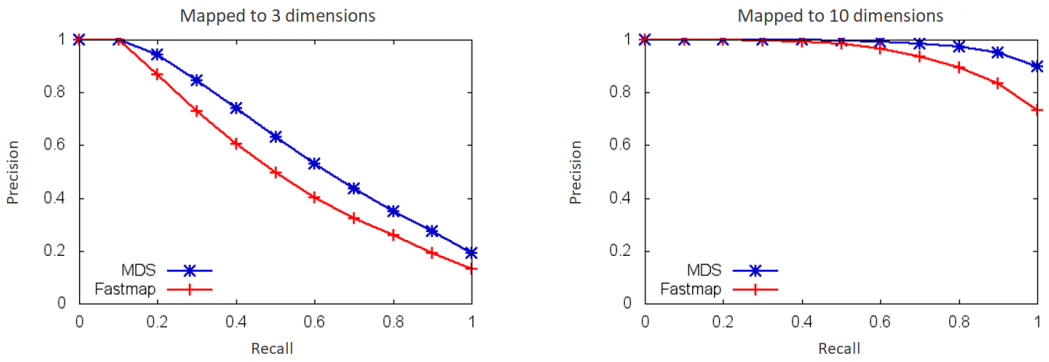

Fig. 4. Evaluation of the quality of data mapping for 3 and 10 dimensions with *MDS* and *Fastmap* algorithms.

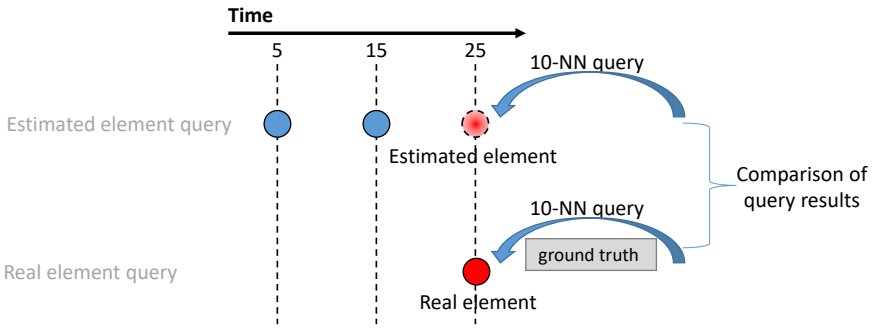

Fig. 5. Evaluation of the estimates.

different spaces. To *MDS10*, the average precision varied from 64.5% (*Past 1*) to 92.7% (*Intermediate 4*).

To perform the remaining experiments, we opt to use the estimates regarding *Future 4*. In Figure 10 it is possible to see the comparison between the estimates results in *Future 4* for *Hist256*, *MDS10* and *Fastmap10*, where we can note that the results for *Hist256* and *MDS10* are very similar, with an average precision of 83% and 81% respectively. If we consider only the initial levels of recall, that difference is even smaller.

4.3 *kAndRange* query evaluation

To evaluate the quality of the results of the similarity queries using the estimate as a center of the query, in the following experiments we compared the answers of *k-NN*, *kAndRange* and *kAndRev* queries. However, to evaluate whether the returned elements are truly relevant semantically, in all follow queries we considered to be relevant the returned elements that belong to the same class of the query center. That is, those elements who correspond to images of the same object used to generate the estimate.

Initially we present the results of the experiments with *kAndRange* queries. For all queries, the value of k was defined as 10, and the radius was determined by calculating the average of the *10-NN* query radius considering all the elements of the set: pre-calculated average of the distances of the

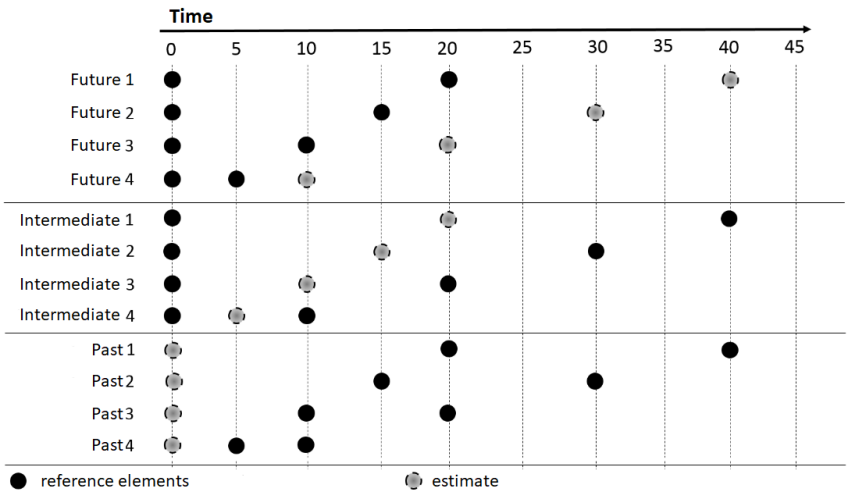

Fig. 6. Time intervals for estimates.

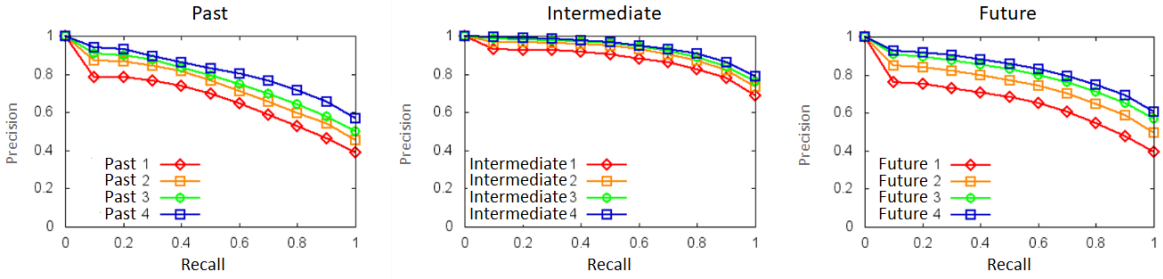

Fig. 7. Evaluation of estimates performed on dataset *Hist256*.

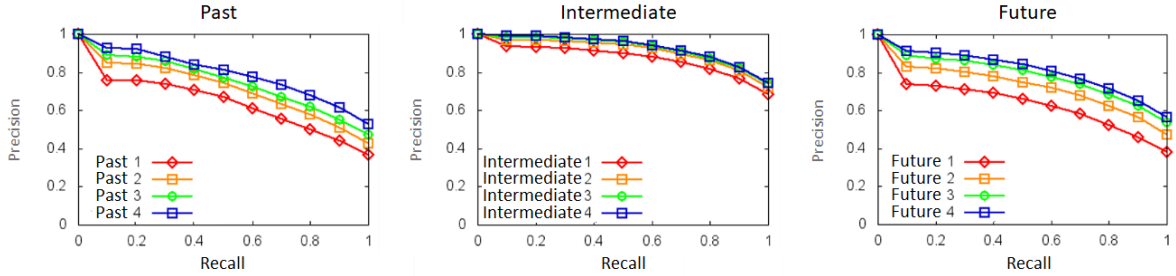

Fig. 8. Evaluation of estimates performed on dataset *MDS10*.

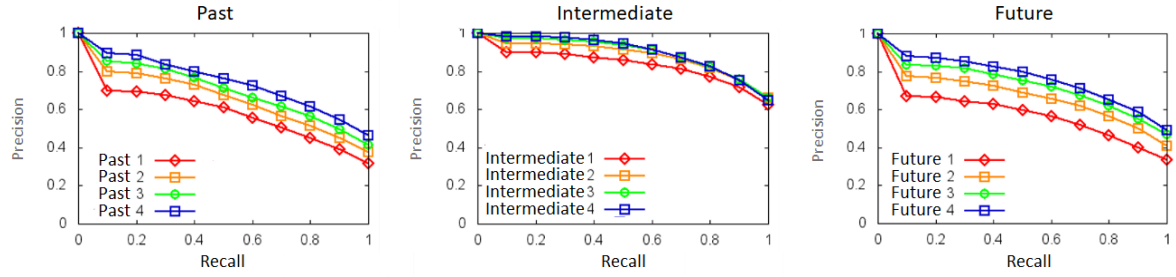

Fig. 9. Evaluation of estimates performed on dataset *Fastmap10*.

tenth element returned in *10-NN* queries.

In Figure 11 we present the total of images retrieved and the total of relevant images retrieved (images of the same object used for estimate). The results are presented to each search space (*Hist256*, *MDS10* and *Fastmap10*). We verified that, by performing *k-NN* queries, (1,000 *10-NN* queries), 72,1%, 70.7% and 65.6% of the 10,000 return elements were relevant to *Hist256*, *MDS10* and *Fastmap10* respectively.

The limitation of the answer set by using *kAndRange* has decreased the number of non-relevant elements retrieved (false positives). Despite the reduction of the total of relevant retrieved (true positive), the precision was increased for every case analyzed. By comparing the *k-NN* query to *kAndRange*, the precision of queries increased from 72.1% to 79% in *Hist256*, from 70.7% to 77.5% in *MDS10* and from 65% to 72.4% in *Fastmap10*. We can note that the difference between the columns "Total Retrieved" and "Relevant Retrieved" is greater for the queries using only *k-NN* for all the three cases. It means that in the *kAndRange* queries the number of elements returned is closer to the number of relevant ones. For the three datasets, respectively, among the 1,000 queries performed using the estimates, 568, 535 and 480 returned only relevant images.

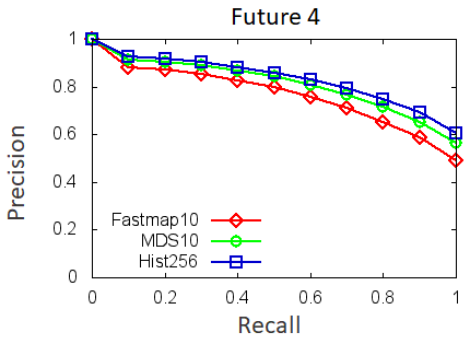

Fig. 10. Estimate evaluation comparison to *Future 4* in the datasets *Hist256*, *MDS10* and *Fastmap10*.

To exemplify the exclusion of non-relevant images of the query answers, the results of *k-NN* queries compared to the results of *kAndRange* are shown in Figure 12. Queries for 5 estimates A, B, C, D and E performed in the dataset *MDS10* are shown. For each one of the estimated elements, the images retrieved for the 10 nearest neighbors are shown. The images highlighted by rectangles refer to those returned by the *kAndRange* queries. The identifiers placed in each image correspond to the ID of the object photographed and the time, respectively. In the examples presented in Figure 12, *kAndRange* held the relevant images retrieved in the *10-NN* query and excluded all (except E) the non-relevant elements (images of other objects).

4.3.1 ***kAndRange input parameter: expanding radius.*** In order to evaluate the effects of varying the input parameter of *kAndRange* queries, we repeated the experiments by varying the radius values. The *kAndRange* queries were executed with the radius value increased in 10%, 20%, 30%, 40%, 50% and 100%. In Figure 13, the graphics showing the results using the dataset *MDS10* are displayed. The results for *Hist256* and *Fastmap10* were similar.

It is possible to see that, as we increase the radius, more relevant and non-relevant elements are retrieved. However, the number of non-relevant elements increases faster. Therefore, despite the increment on recall levels, the results precision decreases. Thus the definition of the radius value depends on the query goals and the specification of the application and data domain.

4.3.2 ***kAndRange: scenario with few relevant elements.*** In order to verify the effectiveness of pruning the non-relevant elements of the *kAndRange* query results, in this experiment we aim to check this method's behavior when we do not have many relevant elements in dataset: the number of relevant elements present in dataset is smaller than the number of neighbors (k) fetched in the queries

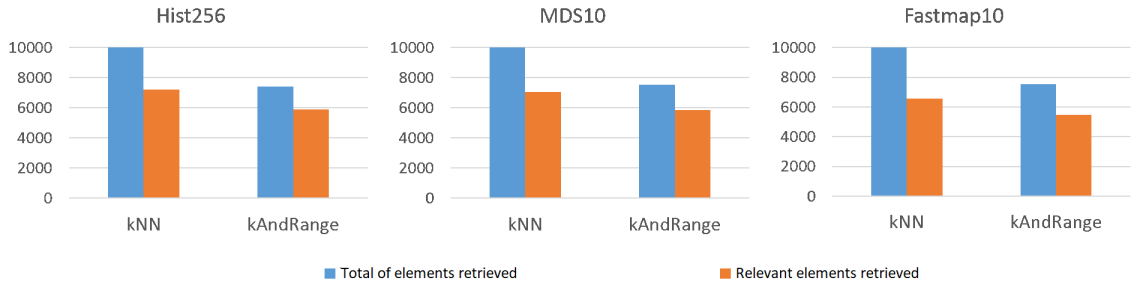

Fig. 11. Comparative between the results of the *k-NN*, and *kAndRange* queries, for the datasets *Hist256*, *MDS10* and *Fastmap10*.

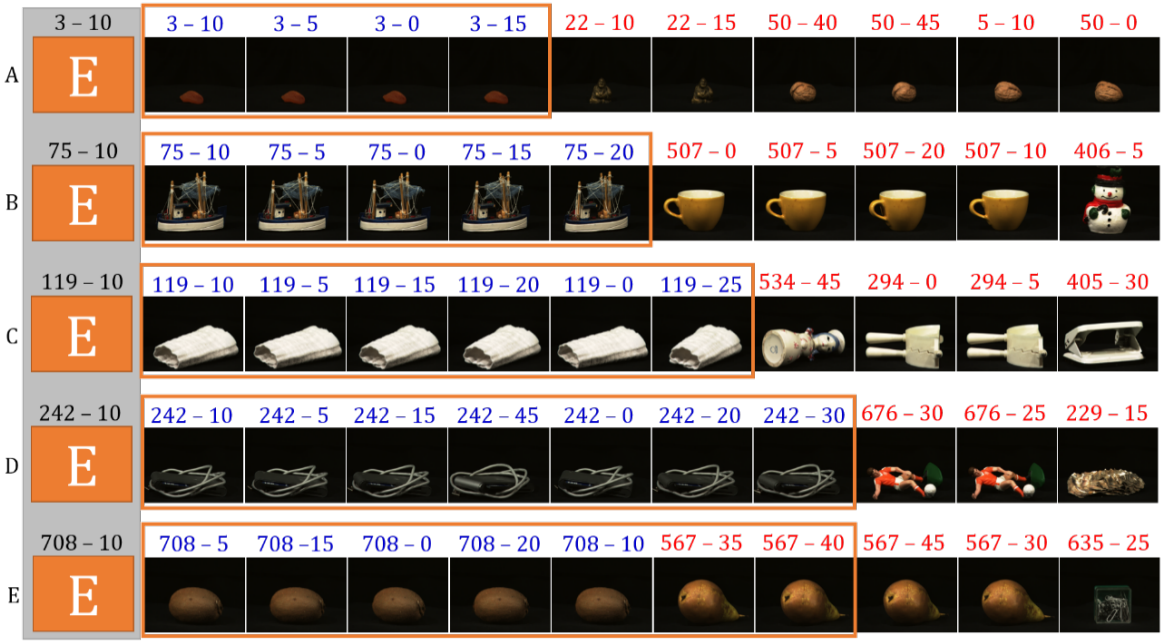

Fig. 12. *k-NN* queries compared to *kAndRange* queries.

To do this, we removed from the dataset all the images of the query object except the two ones that are used as reference for the estimate. The same estimates and *kAndRange* queries were done. But, for each estimated element, now we have in the dataset only two relevant images: those ones that were used as a reference for the estimate.

Considering the 1,000 *kAndRange* queries performed, it was returned approximately 4.5 elements for *Hist256*, 4.5 for *MDS10* and 6.5 for *Fastmap10*, with precision of 36.2%, 36.1% and 25.6% respectively. It is worth mentioning that the maximum precision for 10-NN query would be 20%. For the three datasets, respectively, 518, 493 and 294 queries retrieved only relevant images. However, the return of images related to other objects can be explained due to the fact that the images are similar if we

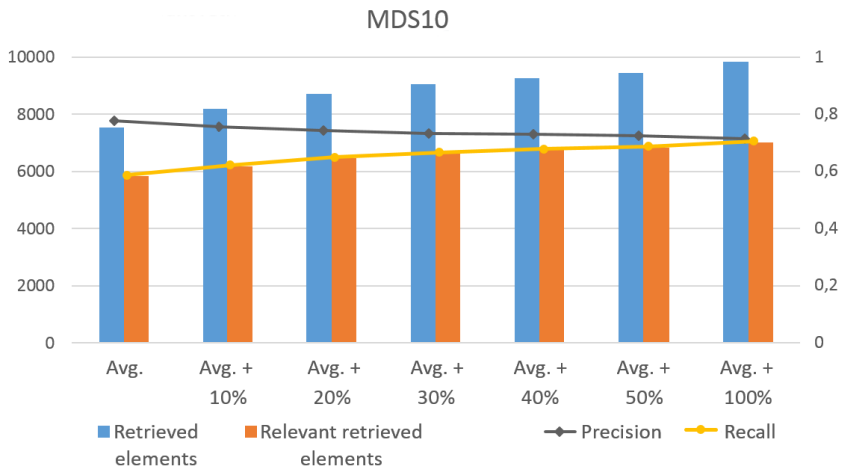

Fig. 13. Examples of *kAndRange* queries varying the radius value

Fig. 14. Examples of *kAndRange* queries in a scenario with only two relevant elements: visually similar images.

consider only the color features, and then images from other objects that are visually similar can be retrieved, as we show with some examples in Figure 14. In all examples displayed, only the first and the second retrieved images correspond to the estimated object (their indicators are highlighted): the only two images of this object present in the dataset.

4.4 *kAndRev* query evaluation

In this first experiment with *kAndRev* queries, the same k value (10) was used in the first and second step (k_1 and k_2). We will refer to this query as $k_{10}AndRev_{10}$.

In Figure 15 we present the total of retrieved images and the total of relevant retrieved images, considering *Hist256*, *MDS10* and *Fastmap10*, comparing with *10-NN* and *kAdnRange*. The limitation of the answer set by using $k_{10}AndRev_{10}$ further decreased the number of non-relevant elements retrieved. Comparing to *kAdnRange*, the precision of the queries increased from 79% to 89.2% in *Hist256*, from 77.5% to 85.9% in *MDS10* and from 72.4% to 81.9% in *Fastmap10*.

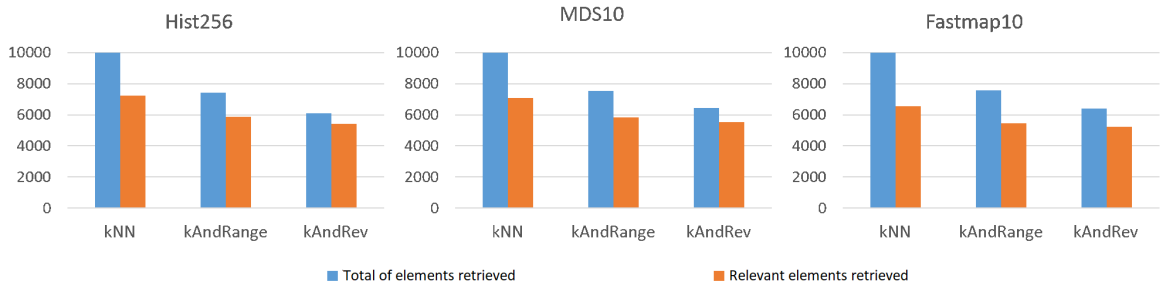

Fig. 15. Comparative between the results of the *k-NN*, *kAndRange* and *kAndRev* queries, for the datasets *Hist256*, *MDS10* and *Fastmap10*.

Fig. 16. $k_{10}AndRev_{10}$ queries compared to *k-NN* queries.

In Figure 16, some examples of $k_{10}AndRev_{10}$ queries are shown to illustrate the exclusion of non-relevant images compared to *10-NN* queries. The queries were executed using the dataset *MDS10*. For each one of the 5 estimates (A, B, C, D and E) we show the images returned by the *10-NN* query. The images returned by $k_{10}AndRev_{10}$ are highlighted by the rectangle. For the element B, for example, only 3 of the 10 neighbors retrieved by *10-NN* query are relevant. When we applied the $k_{10}AndRev_{10}$ query, only these 3 elements were retrieved.

4.4.1 ***kAndRev input parameter: varying (k_1 and k_2) values***. The intention of this experiment was to verify the effect on *kAndRev* queries when varying the k_1 (firts step) and k_2 (second step) values. In the first verification, we kept the k_1 value static ($k_1 = 10$) and varied k_2 between the values 5, 10, 20, 30, 40 and 50. The results of this experiment for *MDS10* are shown in Figure 17. They are similar for the three datasets.

With $k_2 = 5$, for example, the number of elements returned decreased considerably. However, almost all the elements returned are relevant, by achieving precision levels of 96.1% for *Hist256*, 93.4% for *MDS10* and 89.5% for *Fastmap10*. However, the recall was quite reduced, varying between 22.5% and 24.5% considering the three datasets.

In a second verification, we varied the k_1 value between 5, 10, 20, 30, 40 and 50 and kept the k_2 value static ($k_2 = 10$). The results of the queries using the set *MDS10* are displayed in Figure 18. Compared to the previous experiment varying k_2 (Figure 17), the variations on precision and recall levels were smaller when varying k_1. Taking as example the results of $k_1 = 5$, from the 5,000 images that could be returned from the queries, 4,212 images were retrieved for *Hist256*, 4,284 for *MDS10* and 4,272 for *Fastmap10*, with precision levels of 94.4%, 92.2% and 89% respectively.

Hence, as the definition of radius values on *kAndRange* queries, the choice of k_1 and k_2 values also depends on the goals of the queries (more precision or more recall) and the specification of data domain and application. The search for suitable values depends on further experiments and is not within the scope of this work.

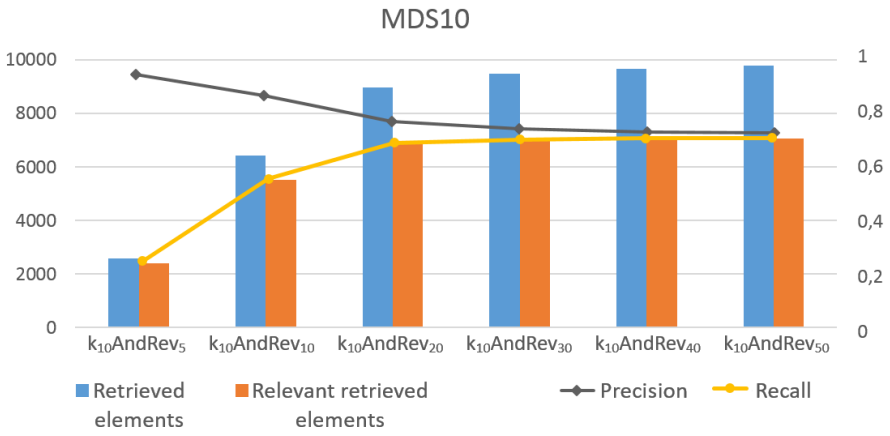

Fig. 17. Examples of *kAndRev* queries varying k_2 values

4.4.2 ***kAndRev: scenario with few relevant elements.*** As we did with the *kAndRange* queries in Section 4.3.2, in this experiment we intend to verify the efficiency of the *kAndRev* method in pruning non-relevant elements. We removed all the images of the estimated object from the dataset, except the two images used as reference. For the 1,000 *kAndRev* queries performed, we retrieved 4.6 elements for *Hist256*, 5.2 for *MDS10* and 5.4 for *Fastmap10*, with precision of 38.1%, 33.7% and 31% respectively.

4.5 *Experiments results summary*

Tables I, II and III shows a summary of the results obtained with *k-NN* (*10-NN*), *kAndRange* (pre-calculated *10-NN* average radius) and *kAndRev* (k_1=k_2=10) queries considering Future 4. It is worth noting that in all experiments we performed 1,000 queries, corresponding to the 1,000 photographed objects in dataset.

By comparing the results of the queries, *kAndRange* and *kAndRev* provided precision levels higher than *k-NN* query, even though with lower recall. The *kAndRev* queries brought the best results, whose precision was higher than 89% for the set *Hist256*, and returning only relevant elements in 705 of the 1,000 executed queries.

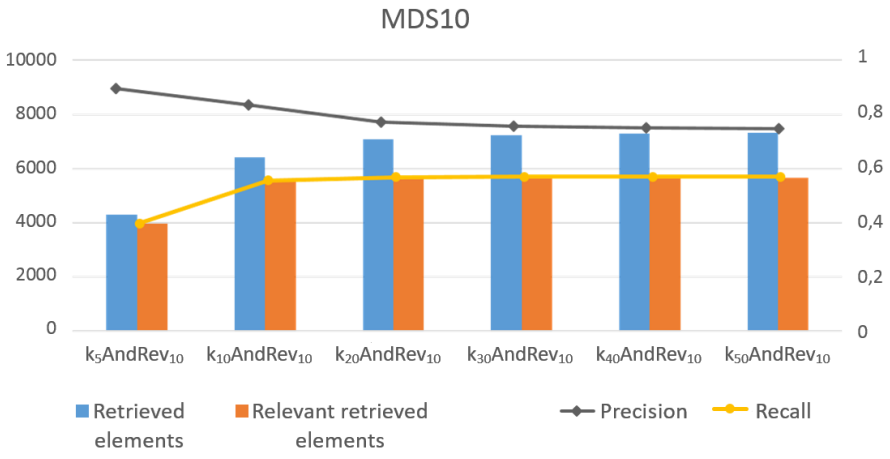

Fig. 18. Results of *kAndRev* queries varying k_1 values

Table I. *k-NN* results.

dataset	Retrieved elements	Relevant retrieved	Average precision	Average recall	Queries retrieving only relevant elements
Hist256	10000	7219	72.19%	72.19%	255
MDS10	10000	7075	70.75%	70.75%	236
Fastmap10	10000	6561	65.61%	65.61%	182

Table II. *kAndRange* results.

dataset	Retrieved elements	Relevant retrieved	Average precision	Average recall	Queries retrieving only relevant elements
Hist256	7430	5878	79.11%	58.78%	568
MDS10	7544	5845	77.48%	58.45%	535
Fastmap10	7560	5471	72.37%	54.71%	480

Table III. *kAndRev* results

dataset	Retrieved elements	Relevant retrieved	Average precision	Average recall	Queries retrieving only relevant elements
Hist256	6083	5428	89.23%	54.28%	705
MDS10	6421	5515	85.89%	55.15%	607
Fastmap10	6407	5247	81.89%	52.47%	539

We also evaluated the variation in the input parameters for the queries *kAndRange* (varying the radius) and *kAndRev* (varying k1 and k2). It is clear that these parameters can be defined as a strategy to prioritize the precision or the recall of the queries, according to the goals of the queries and the characteristics of the dataset.

In the experiments performed with few relevant elements (less than the k value set in the queries), both *kAndRange* and *kAndRev* showed efficiency in pruning the non-relevant elements, then returning less elements.

5. CONCLUSION

Complex data usually presents high dimensionality or are non-dimensional, and therefore represented in metric spaces, where only the data elements and distances between them are available. The absence of dimensional coordinates makes the representation of trajectories impossible in these spaces only with the addition of a temporal dimension, as can be carried out in multidimensional spaces. In this article, we proposed the mapping of complex data represented in metric spaces into multidimensional spaces, with the goal of making the temporal evolution analysis of this data possible. The number of dimensions in the multidimensional space was defined using the concept of intrinsic dimension of the dataset. The experiments presented showed that the data distribution was satisfactory maintained in the mapped space.

Since it is not possible to create the complex data equivalent of its estimated position in multidimensional space, we proposed to apply the similarity query using this position as a query center. Three kinds of similarity queries were appraised, all of them relating to the search for the nearest neighbors of the estimated position.

We evaluated the application of the *k-NN* query initially. The experiments presented showed that the results of the queries using the estimated positions reached an average precision of up to 92.7% in the mapped space. However, the direct definition of a fixed number of elements in the predicate of *k-NN* query can result in the retrieval of non-relevant elements to the user, especially in the case when the search is conducted in an area with few elements close. With the purpose of retrieving only

elements that are really close to the estimated position, and then relevant, the application of two other queries were proposed: *kAndRange* (intersection of *k-NN* and *Range queries*) and an approximation of the *Reverse k-NN*, *kAndRev*, (only the elements retrieved for *k-NN* that have the query center as one of the k nearest neighbors are returned).

The experiments showed that the proposed methods increased the precision of the answers compared to the queries to the nearest neighbors. In experiments with dataset variations, in scenarios with few relevant elements (less than the value of k), *kAndRange* and *kAndRev* queries were efficient in pruning non-relevant elements. Experiments were also conducted to evaluate the effects of varying the input parameter in both methods. The results showed that these parameters can be set to prioritize the precision or recall of queries, according to the goals of the application.

REFERENCES

Arantes, A., Vieira, M., Traina Jr., C., and Traina, A. Efficient algorithms to execute complex similarity queries in RDBMS. *Journal of the Brazilian Computer Society* vol. 9, pp. 5–24, 04, 2004.

Bueno, R., Kaster, D. S., Traina, A. J. M., and Traina Jr., C. Time-aware similarity search: a metric-temporal representation for complex data. In *Proceedings of the International Symposium on Spatial and Temporal Databases*. Springer, Aalborg, Denmark, pp. 302–319, 2009.

Bustos, C., Navarro, G., Reyes, N., and Paredes, R. An empirical evaluation of intrinsic dimension estimators. In *Proceedings of the International Conference on Similarity Search and Applications*. Springer International Publishing, Cham, pp. 125–137, 2015.

Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. Searching in metric spaces. *ACM Computing Surveys* 33 (3): 273–321, Sept., 2001.

Cox, M. A. A. and Cox, T. F. pp. 315–347. In , *Multidimensional Scaling*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 315–347, 2008.

de Sousa Fogaça, I. C. O. and Bueno, R. Temporal evolution of complex data. In *Proceedings of the Brazilian Symposium on Databases*. SBC, Porto Alegre, RS, Brasil, pp. 25–36, 2020.

Faloutsos, C. and Lin, K.-I. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD Record* 24 (2): 163–174, May, 1995.

Geusebroek, J., Burghouts, G. J., and Smeulders, A. W. M. The amsterdam library of object images. *International Journal of Computer Vision* 61 (1): 103–112, 2005.

Hjaltason, G. R. and Samet, H. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (5): 530–549, 2003.

Paiva, C. E., Malaquias Jr, R. D., and Bueno, R. Visualization of similarity queries with trajectory estimation in complex data. In *Proceedings of the International Conference Information Visualisation*. Vienna, Austria, pp. 92–97, 2020.

Sousa, E. P. M., Traina Jr., C., Traina, A. J. M., Wu, L., and Faloutsos, C. A fast and effective method to find correlations among attributes in databases. *Data Mining and Knowledge Discovery* 14 (3): 367–407, 2007.

Tao, Y., Yiu, M. L., and Mamoulis, N. Reverse nearest neighbor search in metric spaces. *IEEE Transactions on Knowledge and Data Engineering* 18 (9): 1239–1252, 2006.

Traina Jr., C., Traina, A. J. M., and Faloutsos, C. Distance exponent: A new concept for selectivity estimation in metric trees. In *Proceedings of the IEEE International Conference on Data Engineering*. IEEE Computer Society, San Diego, California, USA, pp. 195, 2000.

Traina Jr., C., Traina, A. J. M., and Faloutsos, C. Fast feature selection using fractal dimension - ten years later. *Journal of Information and Data Management* 1 (1): 17–20, 2010.

Traina Jr., C., Traina, A. J. M., Wu, L., and Faloutsos, C. Fast feature selection using fractal dimension. *Journal of Information and Data Management* 1 (1): 3–16, 2010.

Vieira, M. R., Traina Jr., C., Traina, A. J. M., Arantes, A. S., and Faloutsos, C. Boosting k-nearest neighbor queries estimating suitable query radii. In *Proceedings of the International Conference on Scientific and Statistical Databases Management*. IEEE Computer Society, pp. 1–10, 2007.