Temporal Evolution of Complex Data

Isis C. O. S. Fogaça, Renato Bueno

Federal University of São Carlos (UFSCar)
São Carlos – SP – Brazil

Abstract. *Monitoring the temporal evolution of data is essential in many areas of application of databases, such as medicine, agriculture and meteorology. Complex data are usually represented in metric spaces, where only the elements and the distances between them are available, which makes it impossible to represent trajectories considering a temporal dimension. In this paper we propose to map the metric data to multidimensional spaces so that we can estimate the element's status at a given time, based on known states of the same element. As it is not possible to create the complex data equivalent to its estimated position, we propose to apply similarity queries using this position as query center. We evaluated three types of similarity queries: k-NN, kAndRange and kAndRev.*

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 position in some time instant using its known positions in other time instants as a 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 distance between the metric elements are available.

In this paper, 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 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 paper some of the experiments that demonstrated the increased precision of the queries results when we compare them to the *k-NN* 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 paper.

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 $\langle S, d \rangle$, where S represents the elements and d is the distance function, or metric, defined as $d : S \times S \rightarrow R^+$. 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 research.

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 $\langle S, d_s \rangle$, where S represents the data set that belong to the domain application, and $d_s : S \times S \rightarrow R^+$ is a

metric that calculates the dissimilarity between the elements in the domain. The temporal component is defined as a metric space $\langle T, d_t \rangle$ where T represents the time measures and $d_t : T \times T \rightarrow R^+$ is the metric to calculate the dissimilarity between two elements of T. The metric-temporal space is defined as a pair $\langle V, d_v \rangle$ where $V = S \times T$ and $d_v : V \times V \rightarrow R^+$ 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 for example 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 necessary to highlight that 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 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.

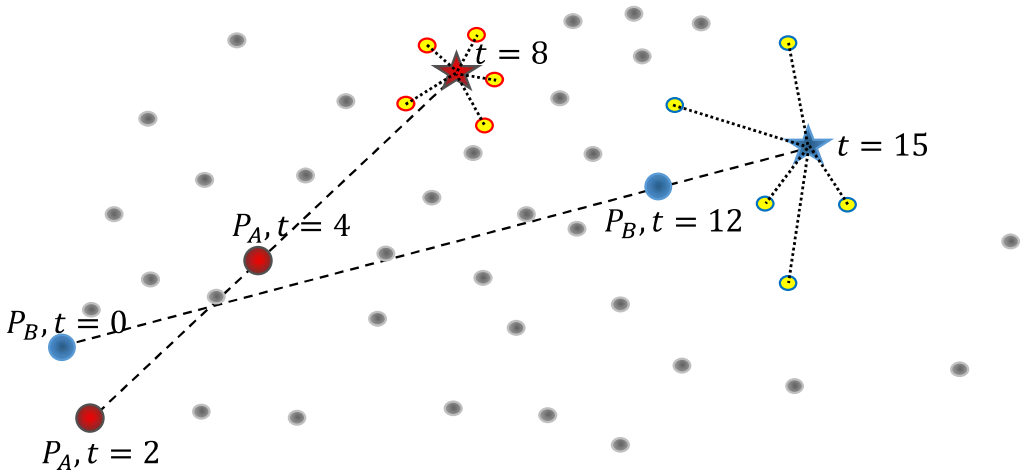

Figure 1. Example of k nearest neighbors query using the estimated positions of patients P_A and P_B as a query center.

In this research, 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 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 paper 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 data set, 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 data set [Sousa et al. 2007]. There are a variety of algorithms that can be applied to estimate the intrinsic dimension of data sets in literature [Bustos et al. 2015]. In this paper, 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 paper, 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, the 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 data set. That value can be estimated by a sample, or else it can be estimated considering the fractal distribution of the data set [Vieira et al. 2007].

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 data set. 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 nearest neighbors

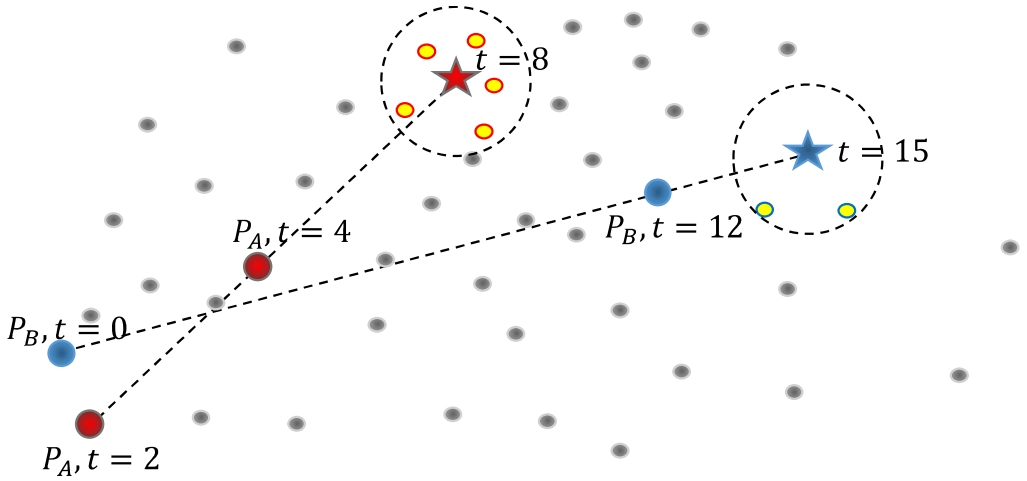

Figure 2. A *kAndRange* query example, with $k = 5$. For P_A, the five nearest neighbors would be retrieved, but for P_B, only two of them.

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-NN* query is done using the estimated element as the query center. In the second one, we perform *k-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 nearest neighbors, it is considered as a non-relevant element, and is then discarded. Otherwise, the element is considered relevant.

Although we use the same k value for the first and second steps of *kAndRev* query in the experiments that are presented in this paper, these values can be defined separately, aiming to prioritize precision or revocation of queries.

4. Experiments

To execute the experiments, we used the ALOI image data set [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.

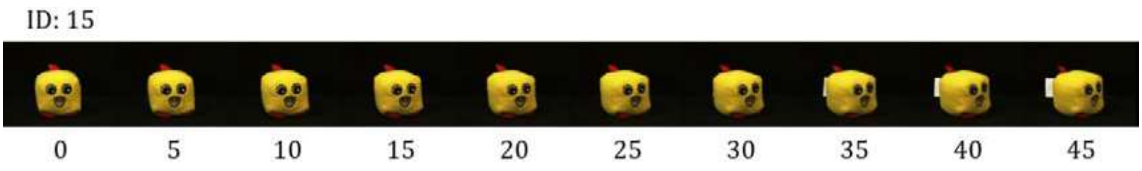

Figure 3. Example of the ALOI image data set.

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 very feature vector 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 data set [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, we considered 10 dimensions, in a strategy that is similar to the one used in [Traina Jr. et al. 2007]. We also performed data mappings into 3 dimensions in order to check the results 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 data set (original and mapped spaces), we performed a search for the 10 nearest neighbors. The answers of the queries were evaluated through precision and recall curves, displayed in Figure 4, where the answers of the original data set 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%.

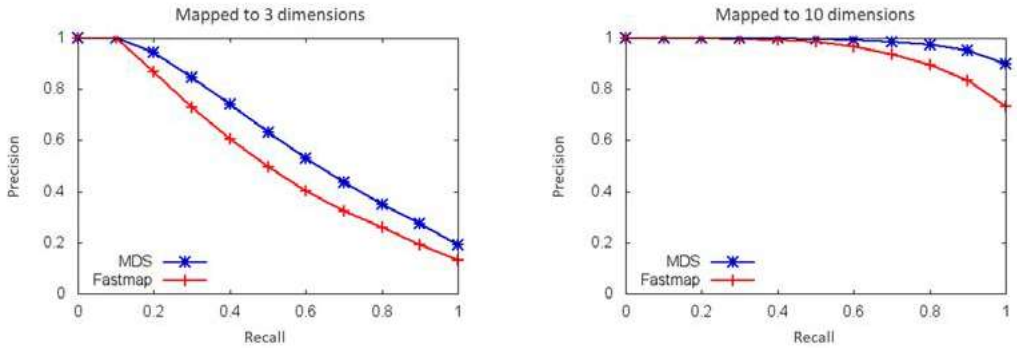

Figure 4. Evaluation of the quality of data mapping for 3 and 10 dimensions with *MDS* and *Fastmap* algorithms.

From the data mapping, all the experiments of the next sections were performed using the three data sets: 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 nearest 10 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 data set (that is the real image of the object in the time instant used for the estimate).

The estimates were performed for the 1,000 objects of the data set. 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 elements, as shown in Figure 5.

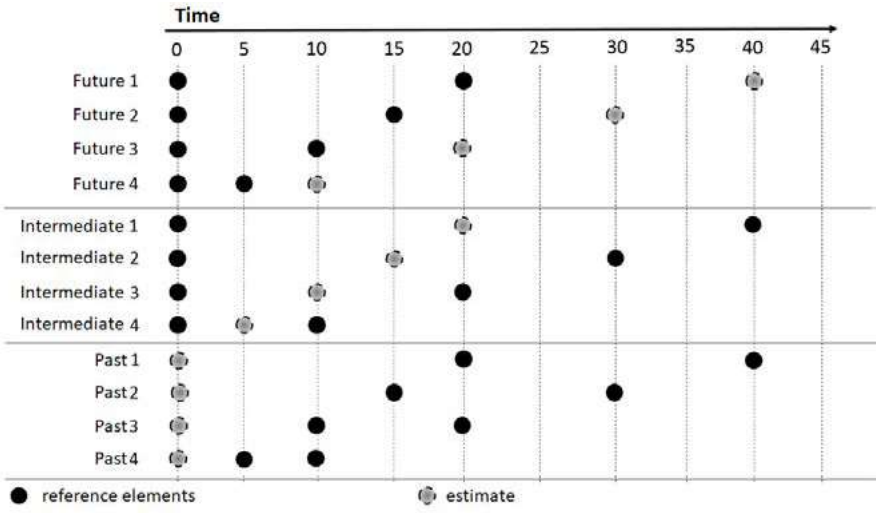

Figure 5. Time intervals for estimates.

The estimates evaluation for each data set are presented in precision and recall graphics, shown in Figures 6, 7 and 8.

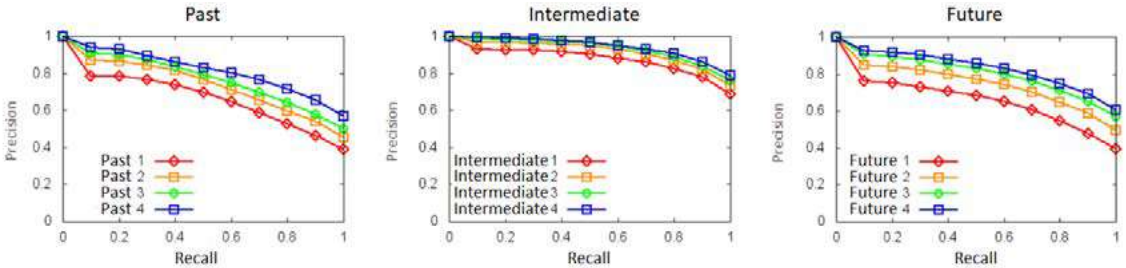

Figure 6. Evaluation of estimates performed on data set *Hist256*.

Despite variations in precision levels, the behavior of the curves was similar to the same query in different spaces. To *MDS10*, the average precision varied from 64.5% (*Past 1*) to 92.7% (*Intermediate 4*).

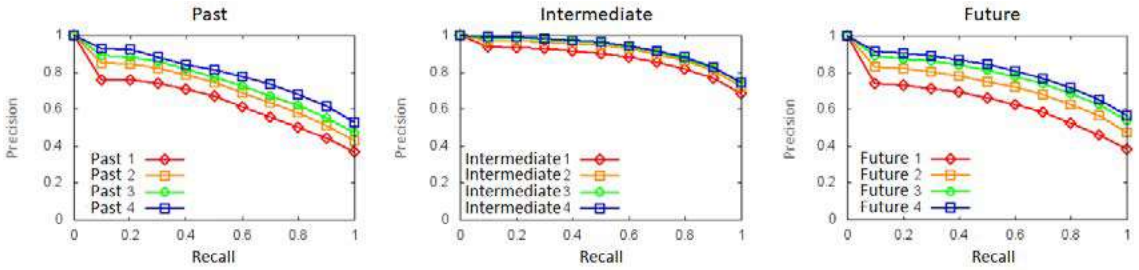

Figure 7. Evaluation of estimates performed on data set *MDS10*.

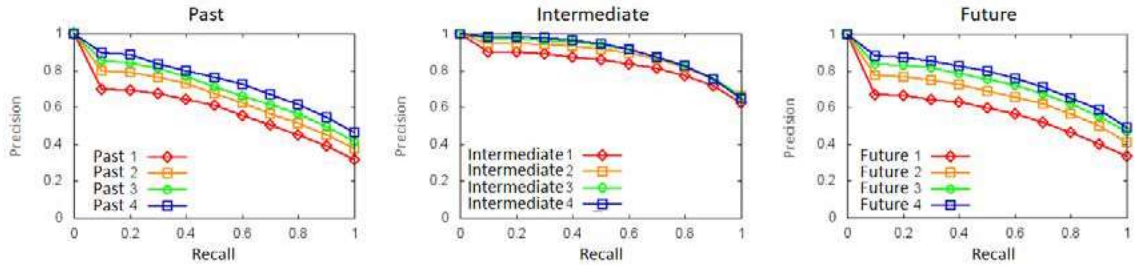

Figure 8. Evaluation of estimates performed on data set *Fastmap10*.

To perform the remaining experiments, we opt to use the estimates regarding *Future 4*. In Figure 9 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.

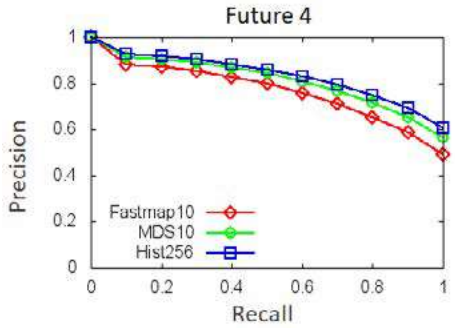

Figure 9. Estimate evaluation comparison to *Future 4* in the data sets *Hist256*, *MDS10* and *Fastmap10*.

4.3. Comparative evaluation between *k-NN*, *kAndRange* and *kAndRev*

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 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.

For all queries, the value of k was defined as 10. To perform *kAndRange*, the maximum range 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 tenth element returned in *10-NN* queries. For the *kAndRev* queries, the same k value (10) was used in the first and second step.

In Figure 10 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.

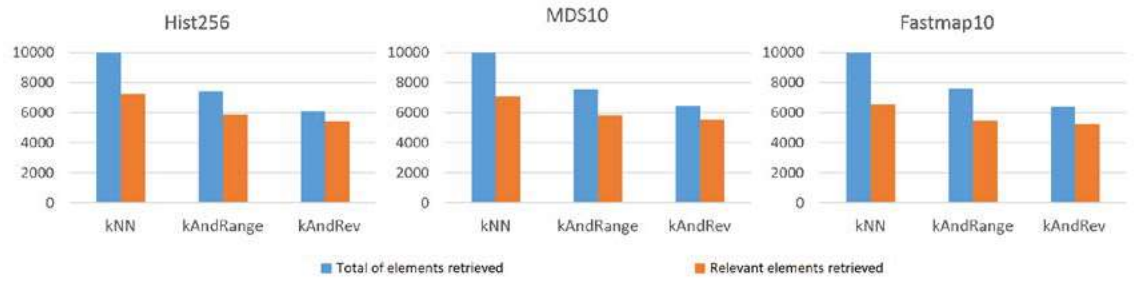

Figure 10. Comparative between the results of the *k-NN*, *kAndRange* and *kAndRev* queries, for the data sets *Hist256*, *MDS10* and *Fastmap10*.

The limitation of the answer set by using *kAndRange* and *kAndRev* 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*. If we consider the *kAndRev* queries, the precision of the queries has increased to 89.2% in *Hist256*, 85.9% in *MDS10* and 81.9% in *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* and *kAndRev* queries are shown in Figure 11. Queries for 4 estimates A, B, C, and D performed in the set *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 (A and B) and by *kAndRev* queries (C and D). The identifiers placed in each image correspond to the ID of the object photographed and the time, respectively. In the examples presented in Figure 11, both strategies (*kAndRange* and *kAndRev*) held the relevant images retrieved in the *10-NN* query and excluded all (except B) the non-relevant elements (images of other objects). The B and D queries were carried out with the same query center, to illustrate the differences in the results of these two strategies.

5. Conclusion

Complex data usually presents high dimensionality or are non-dimensional, and therefore represented in metric spaces, where only the data and distances between them are available. The absence of dimensional coordinates makes the representation of

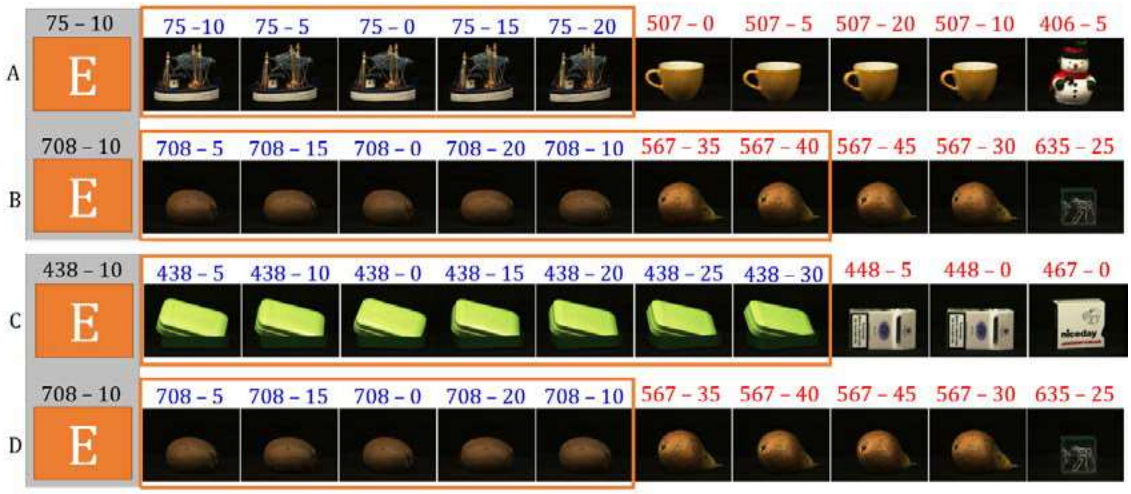

Figure 11. *k-NN* queries results compared to *kAndRange* (highlighted in A and B) and *kAndRev* (highlighted in C and D).

trajectories impossible in these spaces only with the addition of a temporal dimension, as can be carried out in multidimensional spaces. In this paper, 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 data set. 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.

Acknowledgment

This work has been supported by the Brazilian research agencies FAPESP (grants 2016/17078-0), CNPq and CAPES.

References

Arantes, A., Vieira, M., Traina Jr., C., and Traina, A. (2004). Efficient algorithms to execute complex similarity queries in RDBMS. *J. Braz. Comp. Soc.*, 9:5–24.

Bueno, R., Kaster, D. S., Traina, A. J. M., and Traina Jr., C. (2009). Time-aware similarity search: a metric-temporal representation for complex data. In *11th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2009)*, pages 302–319, Aalborg, Denmark. Springer.

Bustos, C., Navarro, G., Reyes, N., and Paredes, R. (2015). An empirical evaluation of intrinsic dimension estimators. In *Similarity Search and Applications*, pages 125–137, Cham. Springer International Publishing.

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

Cox, M. A. A. and Cox, T. F. (2008). *Multidimensional Scaling*, pages 315–347. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

Geusebroek, J., Burghouts, G. J., and Smeulders, A. W. M. (2005). The amsterdam library of object images. *Int. J. Comput. Vis.*, 61(1):103–112.

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

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

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

Traina Jr., C., Filho, R. F. S., Traina, A. J. M., Vieira, M. R., and Faloutsos, C. (2007). The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *VLDB J.*, 16(4):483–505.

Traina Jr., C., Traina, A. J. M., and Faloutsos, C. (2000). Distance exponent: A new concept for selectivity estimation in metric trees. In *16th International Conference on Data Engineering, San Diego, California, USA*, page 195. IEEE Computer Society.

Vieira, M. R., Traina Jr., C., Traina, A. J. M., Arantes, A. S., and Faloutsos, C. (2007). Boosting k-nearest neighbor queries estimating suitable query radii. In *19th International Conference on Scientific and Statistical Database Management, SSDBM*, page 10. IEEE Computer Society.