Evaluating Preprocessing and Interpolation Strategies to Create Moving Regions from Real-World Observations

Rogério Luís C. Costa, Enrico Miranda, Paulo Dias and José Moreira
IEETA, University of Aveiro
Aveiro, 3810-193
Portugal
{rogeriocosta,enrico.miranda,paulo.dias,jose.moreira}@ua.pt

ABSTRACT

Spatio-temporal data may be used to represent the evolution of real-world objects and phenomena. Such data can be represented in discrete time, which associates spatial information (like position and shape) to time instants, or in continuous time, in which the representation of the evolution of the phenomena is decomposed into slices and interpolation functions are used to estimate the intermediate position and shape at any time. The use of a discrete model may seem more straightforward, but a continuous representation provides potential gains in terms of data management, including in compression and spatio-temporal operations.

In this work, we study the use of the continuous model to represent deformable moving regions captured at discrete snapshots. We use a dissimilarity distance-based strategy to select the observations that should be used to define the time slices of the continuous representation, thus transforming data acquired at discrete steps into a continuous model. We also study how the use of geometry simplification algorithms and simplification levels may impact on moving regions interpolation quality.

We evaluate our proposals using a dataset composed by thousands of aerial bush-fires images. After applying object simplification and slice decomposition, we use two region interpolation algorithms to generate in-between observations and compare them with geometries representing real images. The results prove the effectiveness of our proposals and their importance in terms of interpolation accuracy.

CCS Concepts

•**Information systems** → *Temporal data; Spatial-temporal systems;*

Keywords

spatio-temporal data, continuous representation, moving regions

1. INTRODUCTION

Spatio-temporal data may be used to represent the evolution of real-world objects and phenomena. Such data is frequently stored in a discrete format, which associates spatial information (like position and shape) to time instants. This is a straightforward approach, specially because real-world data is mostly captured in discrete snapshots. But real-world objects can also be modeled in spatio-temporal databases using a continuous representation, called *Moving Objects* representation. In this case, the representation of the moving objects is decomposed into slices and interpolation methods are used to represent their evolution (including changes in shape and position) during a time slice while maintaining context-specific constraints [8].

We are particularly interested in using the continuous model to represent real-world entities with application to studies of the propagation of forest fires. We use real videos of controlled burns captured by drones to validate the potential of moving regions representation. An initial challenge is to select the observations that lead to time slices in a way that in-between observations can be adequately estimated using interpolation methods. A few existing algorithms already exist to create representations of the behavior of moving regions between observations, but the selection of key observations for continuous representation of spatio-temporal data in databases is an open issue in this context.

In [1], we propose a dissimilarity distance based model to select which observations should be used to organize the representation of the moving objects in slices. We experimentally evaluate such model, using interpolation functions to generate in-between observations. The results of the interpolations are compared to the original data extracted from videos. To further validate our approach, we also compare the dissimilarity distance approach with the use of uniform sampling (where geometries are selected at equidistant time intervals) in terms of interpolation quality and data compression ratio (i.e. the number of time slices necessary to represent a large number of observations).

This work expands our previous work [1] by evaluating how distinct simplification algorithms affect the quality of moving regions representations and by comparing the performance (in terms of similarity metrics) of two region interpolation methods of the spatio-temporal databases literature. These interpolation methods are evaluated in various scenarios, composed by variations on the observation selection strategy, as well as on the geometry simplification levels and algorithms.

As most of existing works use synthetic data and little has been done in the spatio-temporal databases literature on representing real-world data as moving regions, there has been no prior evaluation on how distinct geometry simplification methods and sampling strategies affect interpolation. Also, there is no prior comparison of moving region interpolation algorithms in terms of similarity between interpolation results and real-world data.

Hence, another goal we achieve with this work is to identify the parameters (in terms of time slice specification, geometry simplification and interpolation algorithm) that should be considered when simulating the phenomena evolution during time frames with no real information (i.e. in our case, to simulate the burned area evolution during periods of time that were not filmed by the drones).

In the next section we present some background and related work. Then, section 3 presents slice definition algorithms. In section 4, we describe the preprocessing steps we take until we got our *burned area* dataset and its alternative representations (obtained through the use of distinct geometry simplification algorithms and levels). Section 5 presents experimental. Section 6 concludes the paper and describe future work.

2. BACKGROUND AND RELATED WORK

In this section, we present some concepts and related work.

2.1 Representing moving regions in spatio-temporal databases

Moving regions are representations of real-world objects in which the evolution of their position, shape and extent over time is important. A common approach to represent such kind of object in a database is to store its position and shape over time in a discrete way, i.e., to record the object's geometry and position at specific timestamps. This is a straightforward approach, as real-world observations are commonly acquired as discrete snapshots, which is followed in many works [25, 11, 17, 24]. Although easy, the use of discrete observations may lead to accuracy, storage and performance issues, depending on the spatial and the temporal resolutions.

As real-world moving objects may change their position and shape continuously over time, there are applications where it is also necessary to represent these properties at any time, including between observations. In [6], the authors propose using Abstract Data Types (ADT) to represent moving regions in spatio-temporal databases. Then, Forlizzi et al. [8] define a discrete data model to decompose the representation of moving regions into fragments called slices. Each slice contains the representation of a moving region at a given time instant and an interpolation function to estimate its evolution during a time interval. Thus, this data model not only allows to represent the evolution of real-world moving objects continuously over time, but it also provides a compact representation because a single slice can represent many observations.

Although there are several works on modeling and querying spatio-temporal data using continuous representations [29, 20, 22], and prototypes that provide such functionalities [9, 4], to the best of our knowledge there is no in-depth work that evaluates the use of such model over real-world data, as well as discussing how to select the observations to be used to decompose the movement of the objects into slices and what are the main issues to be considered.

2.2 Moving regions and interpolation

Most spatio-temporal research considers moving objects as points [22], but there is an increasing demand for applications that must deal with other types of moving data, such as moving regions [22]. This means that region interpolation methods are needed to represent the objects evolution within a time interval (slice).

Existing proposals on moving objects interpolation in the spatio-temporal databases literature include [23, 16, 15, 12, 4], but, there is no consensus on a method that provides realistic interpolations for complex geometries. Each method has its own constraints and uses, and may generate unrealistic interpolations, e.g., in the presence of noisy data or when dealing with geometries with concavities. Therefore, the use of a method in a certain context must be validated (by visual inspection or using metrics) in order to verify the quality of the geometries generated during an interpolation.

In this paper, we use two region interpolation methods. The first one is PySpatioTemporalGeom [15]. This algorithm deals with complex regions (i.e. regions with multiple faces and, possibly, holes) and focuses on providing guarantees of correctness, aiming to generate valid regions (i.e. regions with no-self intersections in the boundaries) for all input cases, even though generated representations are not so likely to be a close approximation of input geometries for all time instants. We use geometric similarity functions to evaluate interpolation results, comparing them to real data segmented from videos. The second region interpolation is the one provided by the Secondo DBMS [9, 10] when using Secondo's *interpolate2* algebra. These two methods differ mainly in the way they deal with concavities.

2.3 Geometry similarity functions

We use some metrics to compare geometries (A and B) and to select the observations that should be used to create a continuous representation of a moving region using the sliced representation. The first metric is the Jaccard Index (JI), which measures the area similarity. It is defined as the ratio between the intersection and the union of two polygons, as denoted in Equation 1. In the implementation of our distance-based algorithm, we use the Jaccard Distance. It is the complement to the Jaccard Index, as denoted in Equation 2. A Jaccard Index near 1 or a Jaccard Distance near 0 means that the geometries are similar.

$$JI(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

$$JD(A,B) = 1 - JI(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

Other metrics we use are the Haussdorff Distance and the

Haussdorff Similarity.

Haussdorff Distance measures the greatest distant between a pair of closest points where one point belongs to A and the other to B. This definition can be seen on Equation 3 [13]. We use the Euclidean Distance to measure the distance between points. The Jaccard Index allows to assess whether two geometries are similar as a whole, while the Haussdorf distance is better to detect local outliers, i.e., specific parts of the geometries that differ significantly.

$$HD(A,B) = max(sup_{a\in A} inf_{b\in B} d(a,b), \\ sub_{b\in B} inf_{a\in A} d(a,b)) \quad (3)$$

The Haussdorff Similarity between two geometries is computed by dividing the Haussdorff Distance between them by the diagonal distance across an envelope specified over the combined geometries. Haussdorff Similarity is a normalization (in the range [0, 1]) of the Haussdorff Distance and higher values represent higher similarity between the geometries.

2.4 Simplification Algorithms

The contours obtained on the raw data are just the coordinates for the border pixels, and thus are raster data. To convert the contours into a vector representation, we have used two well-known simplification algorithms.

The first is the Douglas-Peucker (DP) algorithm [3] that works as follows. A line segment is defined between two points (reference vertices) from a polyline (or the contour of a polygon) to be simplified. Then, the farthest point from this line is included in the simplified geometry, as long as the distance of the point to the line segment is below a given threshold (ξ). This process is applied recursively on all resulting sub line segments until there are no points over ξ distance from the line segment.

The second is the Visvalingam-Whyatt (VW) algorithm [27]. For every point in a polyline, a triangle is built between this point and the previous and next points. Then, the area of each triangle is calculated, and the middle point of the triangle with the smallest area is removed. The area of the adjacent triangles is recalculated, until all triangles have areas above the tolerance level [21].

Some complete comparative studies on those simplification methods in terms of quality and performance include [28] [26] [21].

In this work, we tested DP and VW with different configurations to study how they influence on the selection of observations used to represent a moving region, and thus, on the number of slices and on the boundary of time intervals in a sliced representation. In order to evaluate the impact of the chosen algorithm over interpolation quality, we also compared interpolation results obtained over geometries generated using both algorithms.

3. IDENTIFYING TIME SLICES

In this section, we present two methods to create the time slices necessary to build a continuous representation from a set of discrete observations. The first one is a straightforward approach based on uniform sampling. The second one is our proposal that uses a dissimilarity function to determine which observations should be chosen to define the boundaries of the slices.

3.1 Uniform Sampling

Let *observations* be a collection of geometries ordered by the observation time. A straightforward approach to choose the observations is to use uniform sampling (i.e. to select observations at fixed equidistant time intervals). The algorithm 1 presents how to use uniform sampling. The variable δ represents the step size and *numObservations* is the number of tuples in the collection. This is a simple method that we use as reference for comparison.

Algorithm 1 Uniform sampling

Input:
observations: set of observations
δ: step size
numObservations: number of observations
Output:
set of selected observations
Method:
$i \leftarrow 1$
while $i < numObservations$ **do**
$observations(i).selected \leftarrow true$
$i \leftarrow i + \delta$
end while
$observations(numObservations).selected \leftarrow true$

3.2 Distance-based slices

Let t_1 and t_2 represent two time instants ($t_1 < t_2$) and o_1 and o_2 represent the position and shape of a moving object at time t_1 and t_2, respectively. Let's call *distance* the dissimilarity level between o_1 and o_2. The maximum length of the time interval (i.e. time slice) defined by $[t_1, t_2]$ is constrained by the existence of a function f that is able to represent the evolution of a moving object o from o_1 to o_2 in $[t_1, t_2]$ [8]. Therefore, there should be an upper bound value α for the distance between o_1 and o_2 that a given interpolation function f can represent. So, whenever the distance between o_i and o_j is greater than α, then o_i and o_j should be in distinct time slices.

Let *observations* be a collection of tuples, containing four properties: (i) *time* (which identifies the time when an observation was made), (ii) *obs* (which is the actual geometry representation), (iii) *beginInterval* (which identifies if the observation opens a time interval) and (iv) *endInterval* (which identifies if the observation closes a time interval). Such collection is ordered by the *time* property. The procedure to define which the observations to be selected (to create the time slices) from the collection *observations* is detailed in algorithm 2.

The function *Distance* is the dissimilarity function and represents the distance between two observations, and α is the threshold used to determine when a new time slices should be created. The function *Distance* may be the Jaccard Distance, the Hausdorff Distance, the Fréchet distance, or any

Algorithm 2 Distance based slice identification

Input:
observations: set of observations
Distance: dissimilarity function
α: tolerance threshold
numObservations: number of observations
Output:
time slices identified in the set of observations
Method:
$i \leftarrow 1$
$j \leftarrow i + 1$
while $j \leq numObservations$ **do**
$d_{i,j} \leftarrow Distance(observations(i).obs,$
$observations(j).obs)$
if $d_{i,j} > \alpha$ **then**
$observations(i).beginInterval \leftarrow true$
$observations(j-1).endInterval \leftarrow true$
$i \leftarrow j$
end if
$j \leftarrow j + 1$
end while
$obs(i).beginInterval \leftarrow true$
$obs(j-1).endInterval \leftarrow true$

other distance. A combination of two or more distances is also possible. The choice of the *Distance* function and of the threshold value α are application dependent.

4. BURNT AREA REPRESENTATIONS

In this section, we present the preprocessing steps we took to create alternative discrete representations of the evolution of a burnt area.

4.1 From aerial images to WKT representations

Our study is based on videos on fire spread acquired during a controlled burn by an Unmanned Aerial Vehicle (UAV) equipped with an RGB camera. The burn took place at Pinhão Cel, in the north of Portugal, in 2019, and is part of studies on the estimation of gas emissions to the atmosphere.

In this paper we are interested in the burned area evolution. The video we use here has approximately 15 minutes and was recorded using a 25 frames per second rate. Then, it contains more than 22,500 snapshots on fire spread and on burnt area.

We applied segmentation techniques to identify the object of interest (i.e. burnt area) in each video frame. After segmentation, we generated a WKT (Well Known Text) representation of all pixels on the contour of the segmented burnt area for each frame.

Figure 1 presents an example of a video frame and the corresponding segmented burnt area. The used video and the WKT representations can be downloaded from [18]) the source-code of the segmentation software is available at [2]). We used SPT Data Lab [5] to identify invalid geometries (mostly related to self-intersections) and to turn them into valid ones.

Figure 1: Video frame example with corresponding segmented burned area and database representation

4.2 Creating the simplified representations

Geometries extracted from the video using segmentation algorithms had on average almost 2,000 points, which could lead to performance issues during the simulations (the entire dataset is composed of almost 43 million points). Also, not all the vertices are required to represent the geometries' contours.

Then, we applied the Douglas-Peucker (DP) algorithm and the Visvalingam-Whyatt (VW) simplification algorithms to the burnt area representations. In both methods, a *tolerance level* can be specified, but such tolerance has distinct meanings and effects on each method (see Section2.4). Therefore, using the same tolerance in both methods is not a guarantee of achieving the same simplification level. Hence, we evaluated both methods with several tolerances and verified the simplification level achieved (i.e. number of points per geometry after simplification). Results are in figure 2.

We aim to use geometries simplified by both methods and verify if the simplification method and level influence on interpolation quality. Consider the dash horizontal lines in figure 2: using DP with a tolerance of 1.2 and VW with a tolerance of 2 leads to approximate 67% of simplification, while using DP with a tolerance of 8 and VW with a tolerance of 10, leads to a simplification of approximate 94%.

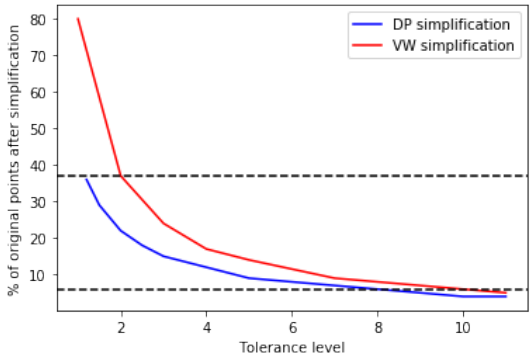

Figure 2: Simplification level evolution - DP and VW algorithms

Let's call DP1.2 and DP8 for the geometry sets resulting from the use of DP simplification with tolerances of 1.2 and 8, respectively, and let VW2 and VW10 be the geometry sets resulting from the use of VW simplification with tolerances of 2 and 10, respectively. Figure 3 presents a burnt area representation with the original WKT data and its simplified representations in DP1.2, VW2, DP8 and VW10. It is possible to notice small differences on the contours between the original geometry and the simplified ones.

Then, we verified the similarity between the simplified geometries and the original ones in terms of Jaccard Index and Hausdorff Similarity. Figure 4 and 5 present how the average Jaccard Index and the average Hausdorff Similarity, respectively, evolve for several tolerance levels and using the Douglas-Peucker and the Visvalingam-Whyatt algorithms.

Geometries simplified with Visvalingam-Whyatt algorithm have a higher similarity with the original ones in terms of Jaccard Index, while geometries simplified using the Douglas-Peucker algorithm have a higher similarity with the original ones in terms of Hausdorff Similarity. But the smallest average Jaccard Index value was 0.965 and the lowest Hasdorff Similarity value was 0.984, indicatinge that simplified geometries were (in average) highly similar to original ones.

The simplification level obtained using DP1.2 and VW2 was 36% and 37%, respectively, while the simplification level obtained using DP8 and VW10 was 6%. Tables 1 and 2 present the similarity between each original geometries and its simplified versions using the Jaccard Distance and the Hausdorff Distance for such configurations.

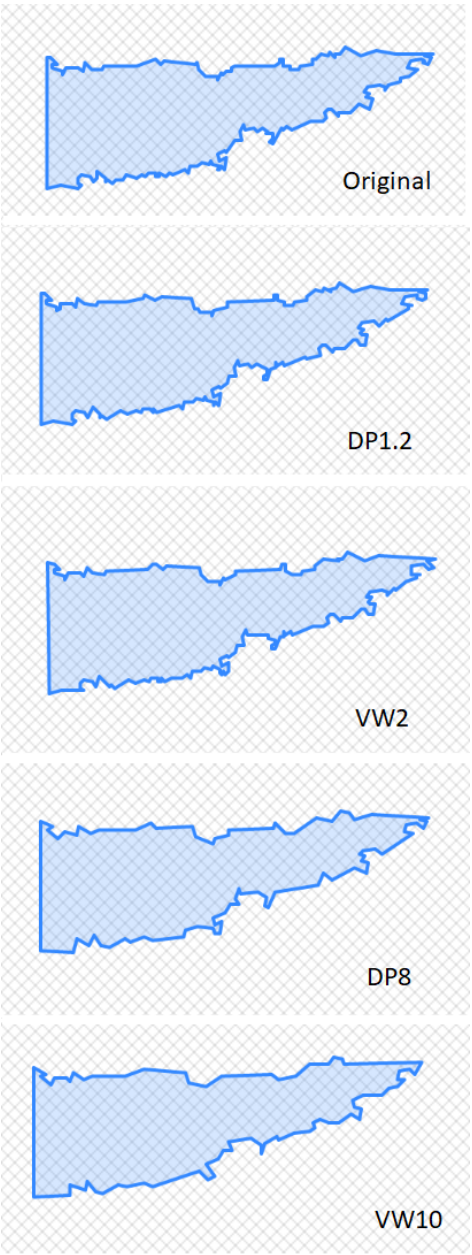

Figure 3: Original x Simplified geometries for the same time instant

Table 1: Jaccard Distance - Simplified dataset x Original dataset

	DP1.2	VW2	DP8	VW10
min	0.001	0.000	0.001	0.009
max	0.016	0.028	0.107	0.219
avg	0.005	0.004	0.027	0.023

The values of the Jaccard Distance in Table 1 are considerably low. But comparing configurations with similar simplifications levels (i.e. comparing DP1.2 with VW2 and DP8 with VW10), one can notice that the datasets generated with the Visvalingam-Whyatt had a greater variation (i.e. max value - min value) in the Jaccard Distance than the ones generated with the Douglas-Peucker algorithm.

A similar behaviour in terms of metric variation can be observed in Table 2. Also, the average values of Hausdorff Distance obtained when using the Visvalingam-Whyatt algorithm are significantly higher than those obtained using Douglas-Peucker algorithm. This means that there are local deformations on the contours of some simplified geometries in the Visvalingam-Whyatt datasets.

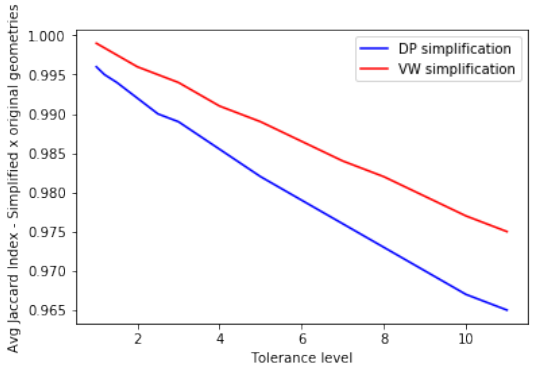

Figure 4: Average Jaccard Index - original x simplified geometries - DP and VW algorithms

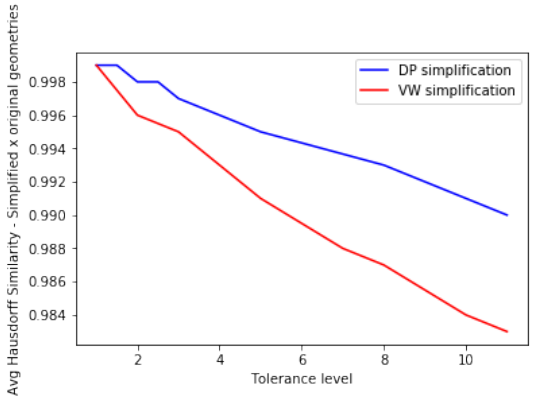

Figure 5: Average Hausdorff Similarity - original x simplified geometries - DP and VW algorithms

Hence, the use of Visvalingam-Whyatt led to geometries that are on average slightly more similar to original ones than the geometries generated by the Douglas-Peucker algorithm in terms of Jaccard Distance, but the use of the Visvalingam-Whyatt generated a greater (min-max) variation in the similarity level and also let to more local deformations on the contours of the geometries.

We also looked for anomalies and inaccurate representations. Used data represents the evolution of the region burnt in a controlled fire. Hence, the initial representation of the burnt area should be contained in its following, that should be contained in the third representation, and so on, until the last one. As we are dealing with areal images from fire spread, one can expect that some inconsistencies may happen, due to the presence of heavy smoke, for instance. We made simple consistency checks (i.e. looking for geometries whose area are less than 85% of the area of the first observation or greater than 115% of the area of the last observation) and identified some anomalies. The number of inaccurate geometries in the original and simplified datasets are presented in table 3. The number of geometries in the simplified datasets that were considered to have a smaller area than acceptable is less or equal to the corresponding statistic in the original dataset, while the number of geometries identified as too big in the simplified datasets is greater or equal to the same statistic in the original dataset.

Table 2: Hausdorff Distance - Simplified dataset x Original dataset

	DP1.2	VW2	DP8	VW10
min	1.2	0.0	6.9	7.8
max	1.2	176.4	8.0	473.2
avg	1.2	4.1	7.9	17.8

Table 3: Datasets and consistency checks

Dataset	Geometries removed due to		Number of remaining geometries
	first geometry check	last geometry check	
Original	393	147	21992
DP1.2	389	149	21994
VW2	393	147	21992
DP8	377	155	22000
VW10	391	150	21991

We removed the inaccurate geometries from the datasets. The resulting data is used in the experiments described in the following section.

5. EXPERIMENTAL EVALUATION

This section presents an experimental evaluation on how to create moving regions to represent the burned area in a controlled forest fire, using a sliced representation. We evaluate geometry simplification, key observation selection and region interpolation methods.

Experiments were executed using the implementation of PySpatioTemporalGeom [15] algorithm available at [14], version 4.1.3 of Secondo (using the virtual machine appliance available at [7]), and SPT Data Lab [5]. The current version of SPT Data Lab is available at [2] is capable of executing several geometry and spatio-temporal related operations, including applying simplification algorithms, selecting key representations, comparing datasets and generating moving regions database scripts.

5.1 Selecting the key geometries

We used the distance-based and uniform sampling algorithms described in Section 3 to select key observations. These are the observations to be stored in the spatio-temporal database system and that would be used as input to region interpolation methods.

5.1.1 Distance-based observation selection

Firstly, we experimentally created time slices for all the five variations (original, DP1.2, VW2, DP8 and VW10) of our dataset using the dissimilarity distance-based algorithm. We use the *Jaccard Distance* to measure the dissimilarity between two observations (o_i, o_j), as defined on Section 2.3, because we want to study how the region (i.e. geometric area) evolves (i.e. grows) over time.

The algorithm was evaluated using several values for α. Figure 6 presents the results.

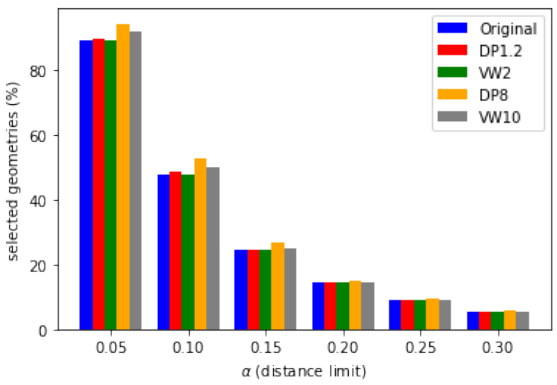

Figure 6: Number of geometries selected to create slices per dataset and value of α

The number of selected geometries decreases significantly as α increases. With $\alpha = 0.05$, about 90% of original geometries were selected for most configurations, while only 5.5% of original observations were selected when $\alpha = 0.3$.

When α is small, the number of geometries selected in the datasets with high simplification levels (i.e. DP8 and VW10) was a little greater than the one in the other datasets. This can be explained as the simplification methods introduced slight modifications that increased the dissimilarity between observations. As the threshold value increases, the results obtained over distinct datasets tend to converge.

Although the number of selected geometries is similar for each value of α, there is no guarantee that the geometries selected in a dataset represent the same time instant than the ones selected in the other datasets. Table 4 presents some statistics on the the number of slices created on some of the tested configurations. When $\alpha = 0.30$, some big slices were defined, being the biggest one of more than 2 minutes (i.e. 3,325 observations). But single observation slices are present in all tested configurations and represent almost all the slices created when $\alpha = 0.05$. Also, when comparing datasets with a similar level of simplification, one can verify that the number of single geometry slices is greater for datasets simplified using Douglas-Peucker algorithm than for those created using the Visvalingam-Whyatt algorithm. Visvalingam-Whyatt simplification aims to maintain the same area of the original geometry, but it smooths the contours, while Douglas-Peucker algorithm aims to maintain a contour similar to the original one (but that can make, for instance, the simplified geometry to be contained in the original one, thus reducing the area of the simplified geometry).

Single slice observations were also generated in all configurations, even when using $\alpha = 0.30$. That means those observations are significantly different from their previous and following ones. Considering that the time interval between frames in the original video is 0.04 seconds, such abrupt changes are probably due to noise or other anomalies in the data, like exemplified in Section 5.4.

Table 4: Sizes of slices x Dataset

	α	DP1.2	VW2	DP8	VW10
Single geom. slices	0.05	10336	10199	13519	12147
	0.30	109	108	123	121
Average size of slices	0.05	1.5	1.5	1.3	1.4
	0.30	33.9	34.1	32.7	33.6
Size of biggest slice	0.05	12	12	9	10
	0.30	3325	3325	3325	3325

5.1.2 *Observation selection using uniform sampling*

When selecting the step size to be used in the uniform sampling strategy, the user is, in fact, specifying a compression ratio to be used and defining the number of slices to be created. The dissimilarity distance-based strategy, on the other hand, provides no guarantee on the compression ratio and depending on the value of α, the number of slices can vary drastically.

In order to provide a fair comparison between the two strategies, we considered the average compression ratio (i.e. number of geometries after applying key selection algorithm divided by the number of geometries in the dataset) obtained for each value of α and used such ratio to specify the step size to be used in uniform sampling. For instance, table 5 present of the compression ratio obtained for several values of α when using the distance-based approach and the corresponding values of step size and compression ratio when using uniform sampling on DP8 and VW10 datasets.

Table 5: Key selection strategies and compression ratio - DP8 and VW10 datasets

Distance based			Uniform sampling	
α	DP8 ratio	VW10 ratio	step size	DP8 and VW10 ratio
0.10	1.9	2.0	2	2.0
0.15	3.8	4.0	4	4.0
0.20	6.7	6.9	7	6.9
0.25	10.8	11.0	11	10.8

We use the step sizes presented at table 5 to create the slices that we use as input to region interpolation functions (together with the slices created using the distance-based strategy), as we present in the following sections.

5.2 Evaluating PySpatioTemporalGeom

After identifying the time slices to be created, we used the implementation of PySpatioTemporalGeom algorithm available at [14] to estimate the shape of the intermediate geometries within each slice. Then, we computed the Jaccard Index and Hausdorff Similarity to determine the similarity between each geometry generated using the interpolation algorithm and its corresponding one in the original dataset.

5.2.1 *Evaluating geometry simplification levels*

To evaluate how geometry simplification level affects interpolation results, we compare similarity metrics obtained

when using datasets simplified with the same algorithm, but with distinct simplification levels. Figure 7 presents the similarity results in terms of Jaccard Index for datasets DP1.2 and DP8, while figure 8 presents the similarity results for datasets VW2 and VW10.

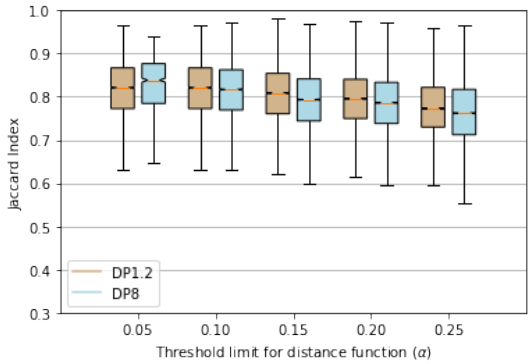

Figure 7: Jaccard Index - simulated geometries x simplified data - DP1.2 and DP8 - PySpatioTemporalGeom interpolation

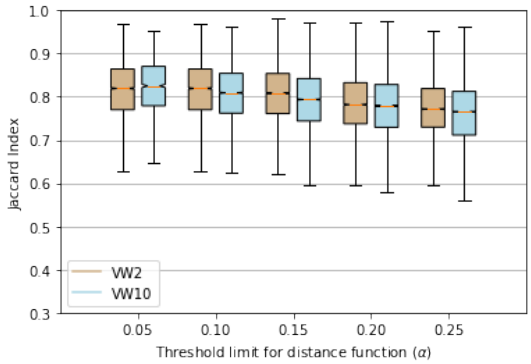

Figure 8: Jaccard Index - simulated geometries x simplified data - VW2 and VW10 - PySpatioTemporalGeom interpolation

For both evaluated geometry simplification algorithms and for the smallest tested value of α, low simplification levels have worse results (in terms of similarity between simulated and real data) than higher ones. But for other values of α, low simplification levels lead to better results than higher ones.

The results described in figures 7 and 8 represent only the successfully simulated geometries. In all configurations, there existed observations that could not be successfully generated by the use of the interpolation algorithm, mostly because the interpolation program generated invalid or empty geometries (validity tests use the *IsValid* method of the *Geometry* class available at the JTS Topology Suite [19], which verifies if a geometry is topologically valid according to the Open GIS Consortium Simple Features Specification). The number of unsuccessfully simulated timestamps increases as the time slice increases, as represented in figure 9. Also, the number of unsuccessfully simulated timestamps is significantly greater for the datasets with smaller simplification levels, which indicates that the extremely detailed contour with a great number of points may lead to more invalid geometries.

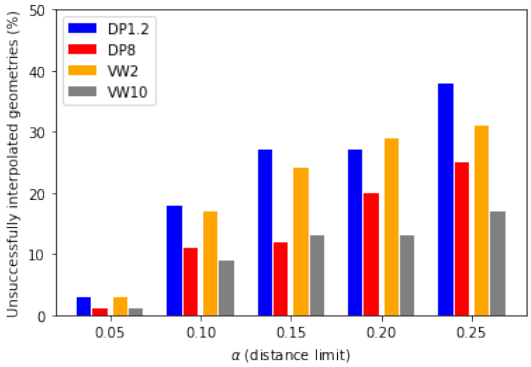

Figure 9: Unsuccessfully simulated timestamps

The interpolation execution time also varies significantly depending on the geometry simplification level. The average time to create interpolation results using DP8 was of 21% of the elapsed time when using DP1.2. When using the VW10 dataset, the average elapsed time to execute all the interpolations was of near 16% of the elapsed time when using the VW2 dataset.

Note that the existence of unsuccessfully simulated timestamps does not indicate that region interpolation methods cannot be applied to generate in-between observations. Actually, we were able to get valid geometries for almost all timestamps using a *MakeValid* procedure over invalid interpolation results. Also, the average similarity metrics between *turned valid* geometries and real ones was similar to the average value of similarity metrics between valid simulated and real geometries. Hence, considering that the use of low simplification levels (i.e. datasets DP1.2 and VW2) led to much higher execution times and number of invalid geometries than the ones obtained when using higher simplification levels (i.e. datasets DP8 and VW10), and did not led to remarkable improvements in similarity metrics, the datasets with higher simplification levels (i.e. DP8 and VW10) seem to be a better choice for a practical use.

Nevertheless, the statistics we present in the remaining of this paper were computed considering only the geometries returned valid from interpolation programs, as we do not want to add a new (possible) source of error (i.e. the use of *MakeValid* over interpolation results) to the analyzes.

5.2.2 *Evaluating key observation selection*

Figures 10 and 11 present the evolution of the Jaccard Index between simulated geometries and the ones of DP8 and VW10 datasets, respectively.

The similarity between simulated results and real data decreases as the compression ratio increases. Also, distance-based geometry selection leads to significantly better results than uniform sampling for the lowest compression rates, but the similarity levels obtained with both selection strategies become closer as the compression ratio increases. Such be-

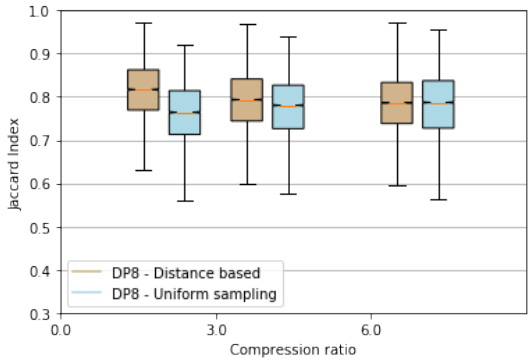

Figure 10: Jaccard Index - simulated geometries x real data - DP8 - PySpatioTemporalGeom interpolation

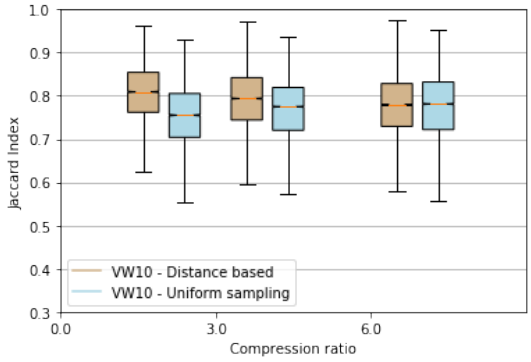

Figure 11: Jaccard Index - simulated geometries x real data - VW10 - PySpatioTemporalGeom interpolation

havior can also be observed in terms of Hausdorff Similarity, as represented in Table 6.

5.2.3 *Evaluating geometry simplification algorithms*

In order to evaluate the influence of the used geometry simplification algorithm on PySpatioTemporalGeom interpolation quality, we compare the similarity between simulated geometries and the ones simplified with each considered simplification algorithm. Figure 12 presents the Jaccard Index between simulated and simplified geometries when using DP8 and VW10 datasets. The values for the Hausdorff Similarity of both datasets are presented in table 6.

The similarity metrics obtained when using the dataset simplified with Douglas-Peucker algorithm are slightly better than the ones obtained when using the dataset simplified with the Visvalingam-Whyatt algorithm.

5.3 Evaluating Secondo Interpolation

We also evaluated the use of the Secondo interpolation to simulate in-between observations. We used the DP1.2, VW2, DP8 and VW10 datasets, and the slices created using several values of α (for distance-based geometry selection algorithm) and step sizes (for uniform sampling). We evaluated

Table 6: Hausdorff Similarity - Distance-based geometry selection x uniform sampling - PySpatioTemporalGeom interpolation

Dataset	Observation selection	Comp. ratio	min	avg	max
DP8	Dist.based	2	0.35	0.91	0.98
	Uniform		0.54	0.88	0.97
	Dist.based	4	0.65	0.90	0.98
	Uniform		0.56	0.89	0.97
	Dist.based	6.9	0.63	0.89	0.98
	Uniform		0.54	0.89	0.98
VW10	Dist.based	2	0.65	0.91	0.98
	Uniform		0.54	0.88	0.97
	Dist.based	4	0.65	0.90	0.98
	Uniform		0.55	0.89	0.97
	Dist.based	6.9	0.65	0.89	0.98
	Uniform		0.55	0.89	0.98

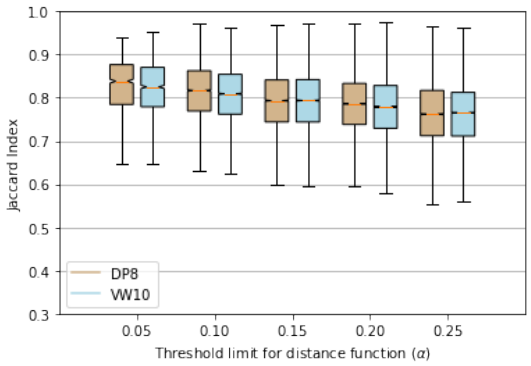

Figure 12: Jaccard Index - simulated geometries x real data - PySpatioTemporalGeom interpolation

the influence of geometry simplification algorithm, of the geometry simplification level and of the size of the intervals on the similarity between real data and Secondo interpolation results, and the overall behaviour was very similar to the one observed when using PySpatioTemporalGeom.

Figure 13 presents the Jaccard Index between geometries simulated using the Secondo DBMS and the ones of DP8 and VW10 datasets, when using the distance-based geometry selection strategy with several values of α. Table 7 presents the average Hausdorff Similarity between simulated geometries and the ones of DP8 and VW10 datasets in the same configurations.

Table 7: Average Hausdorff Similarity - simulated geometries x real data - Secondo interpolation

α	DP8	VW10
0.05	0.951	0.950
0.10	0.930	0.932
0.15	0.915	0.912
0.20	0.899	0.899
0.25	0.887	0.886

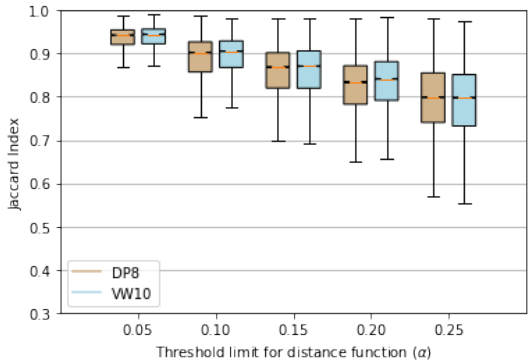

Figure 13: Jaccard Index - simulated geometries x real data - Secondo interpolation

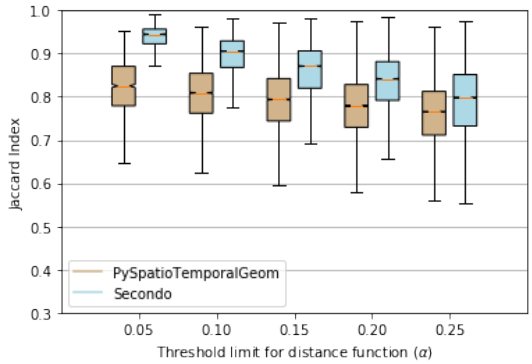

Figure 15: Jaccard Index - simulated geometries x real data - VW10 dataset

The similarity between simulated results and original ones decreases as the size of the intervals increases. Also, using the DP8 or VW10 geometries as input for Secondo interpolation does not impact significantly in terms of similarity between simulated geometries and real ones.

We may also compare the quality of simulated geometries obtained using the PySpatioTemporalGeom algorithm with the quality of geometries generated by Secondo. The Jaccard Index between simulated geometries and real (simplified) observations of datasets DP8 and VW10 are represented in figures 14 and 15, respectively.

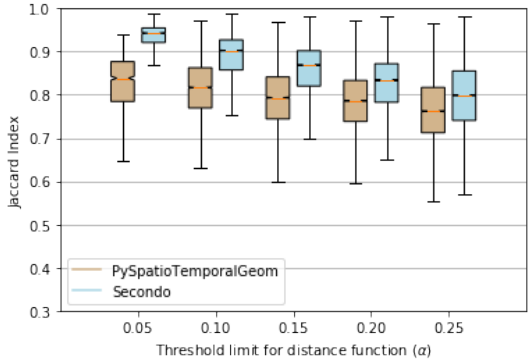

Figure 14: Jaccard Index - simulated geometries x real data - DP8 dataset

Geometries simulated using Secondo were more similar to the original ones than the geometries simulated using PySpatioTemporalGeom, both in terms of Jaccard Index (figures 14 and 15) and in terms of Haudorff Similarity (tables 6 and 7).

Secondo failed to simulate the geometries for some timestamps (mostly by generating invalid geometries), but its ratio of failure was close to the one of PySpatioTemporalGeom.

5.4 Discussion

In the dissimilarity distance-based strategy, the number of slices can vary drastically depending on the value of α. For instance, analyzing the results of section 5.1, together with the results for similarity metrics obtained when simulating geometries with Secondo (e.g. figure 13), it is possible to identify that an increase of 0.20 (from 0.05 to 0.25) on the acceptable dissimilarity (i.e. α) would reduce the number of used geometries from near 90% of the number of observations in the dataset to less than 7%, with a drop-off of just 12% of similarity (on average) between the original shapes and the generated shapes.

In terms of the overall simulation quality, the dissimilarity distance-based strategy led to better results than uniform sampling, as the former considers the geometries similarity to determine the time slices. By identifying situations where the similarity distance between two geometries is greater than a given value, it is possible to create interpolations where the transitions within the time slices is smooth, which improves the quality of the spatio-temporal data.

In section 5.1, we acknowledge the existence of slices composed of a single observation. That means such observations are significantly dissimilar to their previous and following ones, which indicates they may be outliers. We visually inspected some of them and confirmed the existence of inaccurate geometries (i.e. the object detection failed to recognize the burned area).

For instance, consider figure 16, which presents three sequential observations: 788, 789 and 790, from top to bottom, respectively. The one on the middle is clearly different from the other two. By looking back to the original frame (in figure 17), we confirm that the segmentation technique failed to correctly identify the burned area in this frame (due to the smoke in the image). Also, observation 788 is part of an interval with 10 observations and observation 790 is part of an interval with 22 observations (when using VW10 and $\alpha = 0.2$). Both time slices would be merged if observation 789 (the one with an anomaly) is discarded, and a slice representing 32 observations would be created. Thus, in this example, the proposed method was also helpful to identify an outlier observation generated by a segmentation error.

Therefore, considering the existence of single-observation slices, we plan to study, as future work, more sophisticated mechanisms to detect and remove (or correct) noisy data, outliers and anomalies.

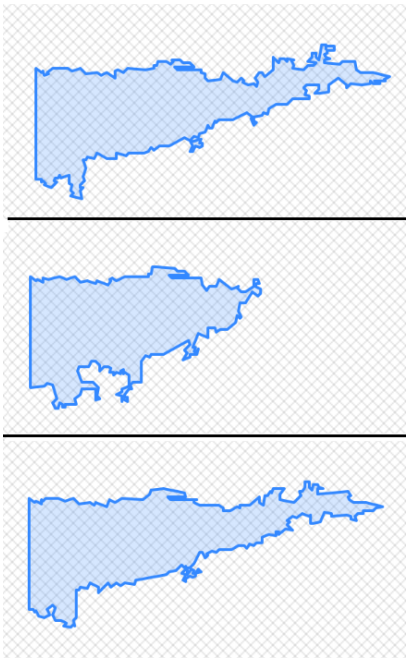

Figure 16: Sequence of observations with an outlier in the middle

Figure 17: Smoke leading to inaccurate object detection

When using the PySpatioTemporalGeom interpolation algorithm, in-between representations of datasets simplified with the Douglas-Peucker algorithm were generally better (in terms of similarity metrics) than the ones simulated using datasets simplified by the Visvalingam-Whyatt algorithm. But the geometry simplification method did not significantly influence on Secondo's interpolation quality.

The number of points in the geometries influenced the interpolation execution time and the number of faults (including invalid geometries) in the interpolations but did not significantly influence the quality of interpolation results in terms of its similarity to real data.

In the evaluated configurations, Secondo significantly outperformed PySpatioTemporalGeom in terms of considered metrics on the similarity between simulated geometries and original ones.

6. CONCLUSIONS

Most spatio-temporal data currently available is acquired and stored as a discrete sequence of snapshots. But dealing with such data discretely may not be the most efficient way in many situations. The use of the continuous representation, on the other hand, may represent a challenge, as discretely obtained observations should be used to create a sliced representation, which comprises procedures to select the most adequate observations and defining interpolation functions that adequately simulate the evolution of the considered real-world phenomena.

In this work we evaluate the PySpatioTemporalGeom interpolation algorithm and the region interpolation method available at Secondo DBMS. We use uniform sampling and a dissimilarity distance threshold between consecutive observations as mechanisms to identify time slices in the continuous representations. The threshold in the dissimilarity distance algorithm aims to guarantee that there is a smooth transition between all the real observations contained in a slice, which we experimentally show through the evaluation over a real-world dataset. Furthermore, we observed that in some situations the use of the dissimilarity distance may help identify potentially segmentation errors and outliers.

We also evaluate how the use of geometry simplification algorithms may impact on time slice definition and interpolation quality. We tested distinct geometry simplification algorithms and geometry simplification levels, and identified that the use of simplified objects does not impact negatively on slices, but improves slice identification execution time. Also, in the evaluated configurations, highly simplified geometries significantly reduced interpolation execution time and failures, and achieved almost the same interpolation quality (in terms of similarity to real data) than the one obtained using the geometry representations with higher number of points in the contours. We identified that Secondo's interpolation achieved the best results in terms of similarity between interpolation results and real-world data.

As future work, we plan to do a more in-depth study on how to detect and remove (or correct) noise, anomalies and outliers in the discrete observations for a better generation of their continuous representations. Outlier detection is a first step to be studied, to remove anomalies from the original datasets. We also plan to use the obtained parameters (in terms of slice specification, geometry representation and interpolation method) as part of our study on burned area evolution, specially to simulate such evolution during time frames on which there is no real data.

7. ACKNOWLEDGMENTS

This work is partially funded by National Funds through the FCT (Foundation for Science and Technology) in the context of the projects UID/CEC/00127/2013 and POCI-01-0145-FEDER-032636.

8. REFERENCES

[1] R. L. C. Costa, E. Miranda, P. Dias, and J. Moreira. Sampling strategies to create moving regions from real world observations. In *Proceedings of the 35th Annual*

ACM Symposium on Applied Computing, SAC '20, page 609–616, 2020.

[2] DETI - IEETA, University of Aveiro. MoST-IEETA - GitHub, 2020. https://github.com/most-ieeta/ - Last accessed: 2020-06-06.

[3] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[4] J. Duarte, P. Dias, and J. Moreira. A framework for the management of deformable moving objects. In A. Mansourian, P. Pilesjö, L. Harrie, and R. van Lammeren, editors, *Geospatial Technologies for All*, pages 327–346, Cham, 2018. Springer International Publishing.

[5] J. Duarte, B. Silva, J. Moreira, P. Dias, E. Miranda, and R. L. C. Costa. Towards a qualitative analysis of interpolation methods for deformable moving regions. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, page 592–595, 2019.

[6] M. Erwig, R. H. Güting, M. Schneider, and M. Vazirgiannis. Abstract and discrete modeling of spatio-temporal data types. In *Proceedings of the 6th ACM International Symposium on Advances in Geographic Information Systems*, GIS '98, pages 131–136, New York, NY, USA, 1998. ACM.

[7] R. H. G. et al. SECONDO: an extensible database system, 2020. http://dna.fernuni-hagen.de/secondo/ - Accessed: 2020-06-06.

[8] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 319–330, New York, NY, USA, 2000. ACM.

[9] R. H. Guting, V. Almeida, D. Ansorge, T. Behr, Z. Ding, T. Hose, F. Hoffmann, M. Spiekermann, and U. Telle. SECONDO: an extensible dbms platform for research prototyping and teaching. In *21st International Conference on Data Engineering (ICDE'05)*, pages 1115–1116, April 2005.

[10] R. H. Güting, T. Behr, and C. Düntgen. SECONDO: A platform for moving objects database research and for publishing and integrating research implementations. *IEEE Data Engineering Bulletin*, 33(2):56–63, 2010.

[11] H. Hayashi, A. Asahara, N. Sugaya, Y. Ogawa, and H. Tomita. Composition of simulation data for large-scale disaster estimation. In *Proceedings of the Second ACM SIGSPATIALInternational Workshop on the Use of GIS in Emergency Management*, EM-GIS '16, pages 4:1–4:8, New York, NY, USA, 2016. ACM.

[12] F. Heinz and R. H. Güting. Robust high-quality interpolation of regions to moving regions. *GeoInformatica*, 20(3):385–413, Jul 2016.

[13] J. Henrikson. Completeness and total boundedness of the hausdorff metric. *MIT Undergraduate Journal of Mathematics*, 1:69–80, 1999.

[14] M. McKenney. pyspatiotemporalgeom - pypi, 09 2019. https://pypi.org/project/pyspatiotemporalgeom/ - Last accessed: Sept, 2019.

[15] M. Mckenney and R. Frye. Generating moving regions from snapshots of complex regions. *ACM Trans. Spatial Algorithms Syst.*, 1(1):4:1–4:30, July 2015.

[16] M. McKenney and J. Webb. Extracting moving regions from spatial data. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 438–441. ACM, 2010.

[17] X. Meng and J. Chen. *Moving Objects Management: Models, Techniques and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[18] MoST Project. Controlled burn dataset - pinhão cel, portugal, 2020. http://most.web.ua.pt/ - Last accessed: 2020-06-06.

[19] OSGeo. JTS Topology Suite - OSGeo, 2020. https://www.osgeo.org/projects/jts/ - Last accessed: 2020-06-06.

[20] M. Santos and J. Peña. Representing, storing and mining moving objects data. *Lecture Notes in Engineering and Computer Science*, 2192, 07 2011.

[21] W. Shi and C. Cheung. Performance evaluation of line simplification algorithms for vector generalization. *The Cartographic Journal*, 43(1):27–44, 2006.

[22] W. Siabato, C. Claramunt, S. Ilarri, and M. A. Manso-Callejo. A survey of modelling trends in temporal gis. *ACM Comput. Surv.*, 51(2):30:1–30:41, Apr. 2018.

[23] E. Tøssebro and R. H. Güting. Creating representations for continuously moving regions from observations. In C. S. Jensen, M. Schneider, B. Seeger, and V. J. Tsotras, editors, *Advances in Spatial and Temporal Databases*, pages 321–344, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[24] S. Villarroya, J. R. Viqueira, M. A. Regueiro, J. A. Taboada, and J. M. Cotos. Soda: A framework for spatial observation data analysis. *Distrib. Parallel Databases*, 34(1):65–99, Mar. 2016.

[25] J. R. R. Viqueira and N. A. Lorentzos. Sql extension for spatio-temporal data. *The VLDB Journal*, 16(2):179–200, Apr. 2007.

[26] M. Visvalingam. Explorations in digital cartography. 2015.

[27] M. Visvalingam and J. D. Whyatt. Line generalisation by repeated elimination of points. *The cartographic journal*, 30(1):46–51, 1993.

[28] M. Visvalingam and P. J. Williamson. Simplification and generalization of large scale data for roads: a comparison of two filtering algorithms. *Cartography and Geographic Information Systems*, 22(4):264–275, 1995.

[29] J. Xu and R. H. Güting. Manage and query generic moving objects in secondo. *Proc. VLDB Endow.*, 5(12):2002–2005, Aug. 2012.

ABOUT THE AUTHORS:

Rogério Costa received his PhD in Computer Engineering from University of Coimbra in 2011. With more than 15 years of experience in teaching and in industry projects, and he is currently a researcher at University of Aveiro working on the project MoST. His research interests include spatiotemporal databases, big data, data analysis and performance tuning.

Enrico Mirada graduated from the State University of Campinas (Brazil) in 2009 and started working in the private sector. In 2017, he concluded his Master's degree at the Federal University of Maranhão Brazil) with the thesis “Meta Learning applied to the MaxSAT Problem”. He has experience in applied data science, machine learning and optimization on big scale logistic environments. During 2019, he was part of Institute of Electronics and Informatics Engineering of Aveiro (IEETA) on the project "MoST - Modeling, querying and interactive visualization of spatiotemporal data", focusing on data acquisition and pre-processing.

Paulo Dias graduated from the University of Aveiro Portugal in 1998 and started working in 3D reconstruction at the European Joint research Centre in Italy. In September 2003, he concluded his PhD with the thesis “3D Reconstruction of real World Scenes Using Laser and Intensity Data”. He is currently an assistant professor within the Department of Electronics Telecommunications and Informatics (DETI) and is involved in several works and projects within the Institute of Electronics and Informatics Engineering of Aveiro (IEETA) related to 3D Reconstruction, Virtual Reality, Computer Vision, Computer Graphics, Visualization and Combination and Fusion of data from multiple sensors.

José Moreira received his Ph.D. in Computer Science and Networks from the École Nationale Supérieure des Télécommunications de Paris (France), currently known as Telecom Paristech, and the Faculdade de Engenharia da Universidade do Porto (Portugal) in 2001. He is Assistant Professor at the Department of Electronics, Telecommunications and Informatics of the Universidade de Aveiro and a researcher at IEETA, a non-profit R&D institute affiliated to the same university. His main research interests are on spatial and spatiotemporal database systems and GIS Science.