Sampling strategies to create moving regions from real world observations

Author1
Author@domain.com
XX
XX

Author2
Author@domain.com
XX
XX

Author3
Author@domain.com
XX
XX

Author4
Author@domain.com
XX
XX

Author5
Author@domain.com
XX
XX

Author6
Author@domain.com
XX
XX

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 their position and shape at any time. The use of the discrete model is more straightforward in many situations, but the continuous representation has its own advantages, mostly related to the kinds of spatio-temporal operations that can be performed and in terms of data compression.

In this work, we study the use of the continuous model to represent deformable moving regions captured in discrete snapshots. We propose strategies to select the observations that should be used to define the interpolation functions and to transform data acquired at discrete steps into the continuous model. We also study how the use of geometry simplification mechanisms impact on interpolation quality.

We evaluate our proposals using a real world dataset composed by thousands of discrete representations extracted from a video of a controlled burn to prevent the spread of forest fires. We apply object simplification and evaluate the strategies proposed in this work to decompose the representation of the phenomena into slices. Then, we use moving region interpolation mechanisms to simulate in-between observations and compare them with real ones. The results prove the effectiveness of proposed method and the importance of the procedures to define the boundaries of the slices, as it impacts on the accuracy of interpolation methods.

CCS CONCEPTS

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

SAC'20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/xx.xxx/xxx_x

KEYWORDS

spatio-temporal modeling, continuous representation, moving regions

ACM Reference Format:
Author1, Author2, Author3, Author4, Author5, and Author6. 2020. Sampling strategies to create moving regions from real world observations. In *Proceedings of ACM SAC Conference (SAC'20).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/xx.xxx/xxx_x

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 [5].

We are particularly interested on using the continuous model to represent real world data, with application to studies on the propagation of forest fires. We are using movies of controlled burns captured by drones to validate moving regions interpolations. An initial challenge is to select which are the key observations to be used to generate time slices in a way that in-between observations can be adequately estimated through the use of interpolation methods. There are some works on algorithms to simulate 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 problem.

In this work, we propose a dissimilarity distance based model to select which observations should be used to decompose the representation of the moving objects into 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 the videos. We evaluate the use of different simplification algorithms and how the distance between geometries affect the definition of the slices and the quality of the interpolations.

We also compare the dissimilarity distance approach with the use of fixed size slices (whose slice boundaries are selected at certain equidistant time intervals). We compare both approaches in terms of interpolation quality and data compression rate (i.e. the use of a few time slices to represent a large number of real observations). One of the goals to be achieved with this work is to identify the parameters (in terms of time slice specification, geometry simplification and interpolation quality) that should be considered when simulating the phenomena evolution during time frames with no real information (i.e. in our case, to simulate the evolution of the burned area 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 *burned area* dataset and alternative representations obtained through the use of simplification algorithms. Section 5 presents experimental results. 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 [7, 12, 19, 20]. 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 [3], the authors propose using Abstract Data Types (ADT) to represent moving regions in spatio-temporal databases. Then, Forlizzi et al. [5] 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 [15, 17, 24], and prototypes that provide such functionalities [2, 6], to the best of our knowledge there is no detailed 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 [17], but there is an increasing demand for applications that must deal with other types of moving data, such as moving regions [17]. 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 [2, 8, 10, 11, 18], 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 the interpolation method proposed in [10] to estimate the position and shape of objects between observations, and we use geometric similarity functions to evaluate the interpolation results, based on data extracted from videos.

2.3 Geometry similarity functions

We use two 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 measures the area similarity. The Jaccard Index (JI) is defined as the ratio between the intersection and the union of two polygons, as denoted in Equation 1. The distance used is the Jaccard Distance, which 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.

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

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

The second metric is the Haussdorff Distance, which measures how far two sets of points are from each other. It is defined as the maximum distance between any pair of points where one of them is a member of A and the other is a member of B, as denoted in Equation 3. We used the Euclidean Distance to measure the distance between points.

$$HD(A, B) = max(d(a, b), d(b, a) \forall a \in A, b \in B) \tag{3}$$

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.

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 the vector mode, we have used two well-known simplification algorithms.

The first is the Douglas-Peucker (DP) algorithm [1] 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 [22]. For every point in a polyline or a polygon 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 [16].

We use several simplification methods and configurations to study how they influence the procedures to select the observations that should be used in the representation of a moving region, and thus, the number of slices and the boundary of the time intervals in a sliced representation. The aim is not to perform a comparative study on their quality or performance (as there are works dedicated exclusively to that [23] [21] [16]).

3 IDENTIFYING TIME SLICES

In this section, we deal with identifying which time slices should be created in order to build a continuous representation from a set of discrete observations using the sliced representation. We present two algorithms, the first one is a straightforward approach, based on fixed size slices. 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 Fixed size 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) *intervalStart* (which identifies if the observation opens a time interval) and (iv) *intervalEnd* (which identifies if the observation closes a time interval). Such collection is ordered by the *time* property.

A straightforward approach to choose the observations would be to use fixed size slices (i.e. uniform sampling). The algorithm 1 presents how to identify the time slices for the *observations* collection. The variable δ represents the slice size and *numObservations* is the number of tuples in the collection.

Algorithm 1 Fixed size slices

$i \leftarrow 1$
while $i \leq numObservations - \delta$ **do**
  $observations(i).beginInterval \leftarrow true$
  $observations(i + \delta - 1).endInterval \leftarrow true$
  $i \leftarrow i + \delta$
end while
$observations(i).beginInterval \leftarrow true$
$observations(numObservations).endInterval \leftarrow true$

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* for a 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 $[t_1, t_2]$ [5]. 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.

The procedure to define the time slices to be created from the collection *observations* is detailed in algorithm 2.

Algorithm 2 Distance based slice identification

$i \leftarrow 1$
$j \leftarrow i + 1$
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++
end while
$obs(i).beginInterval \leftarrow true$
$obs(j - 1).endInterval \leftarrow true$

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

The experimental study uses data about the evolution of the burned area in a controlled forest fire. This controlled burn was filmed by drones and the videos are being used in simulations and on gas emission studies. In this paper, we use one of such videos, of approximately 15 minutes, recorded using a 25 frames per second rate. This leads to more than 22,500 observations of burned area evolution. This video is available for download at [4].

Segmentation and object detection techniques (including color-specific seeds to represent vegetation and burned areas) were used to delimit the area of interest in each video frame. After segmentation, a WKT representation of all pixels on the border of the identified burned area was generated. Then, the dataset with over than 22,500 WKT representations was loaded into PostgreSQL and stored using PostGIS's geometry data type.

Figure 1 presents an example of a video frame, depicting the segmentation of the burned area and the corresponding geometry recorded in the database.

Then, anomalies were detected in the dataset due to the presence of heavy smoke in some video frames. We removed 541 observations (about 2% of original data), which were considered outliers. An

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

observation was considered an outlier if its area was less than 85% of the area of the first observation or greater than 115% of the area of the last observation. The first and the last observations were refined manually. A detailed study on outlier and anomaly detection is considered future work.

The geometries extracted from the video using segmentation algorithms had on average almost 2,000 points per geometry, which could lead to performance issues during the simulations (the entire dataset is composed of almost 43 million points).

PostGIS has built-in functions that may be used to simplify geometries. We evaluated the use of ST_SimplifyPreserveTopology [13] and ST_SimplifyVW [14]. The former generates a simplified representation of a geometry using the Douglas-Peucker algorithm and the latter uses the Visvalingam-Whyatt algorithm. These simplification methods allow reducing the number of vertices that represent a geometry, while maintaining a good approximation of their shape in most cases.

We applied both algorithms using a *tolerance* (this parameter is a distance threshold used during simplification) equal to 1 and 10, which led us with 5 representations of each observation. From now on, we call them: *Original* (i.e. not simplified), *DP1* (simplification using Douglas-Peucker and tolerance 1), *DP10* (simplification using Douglas-Peucker and tolerance 10), *VW1* (simplification using Visvalingam-Whyatt and tolerance 1) and *VW10* (simplification using Visvalingam-Whyatt and tolerance 10). Figure 2 presents the simplified representations of the geometry displayed in Figure 1.

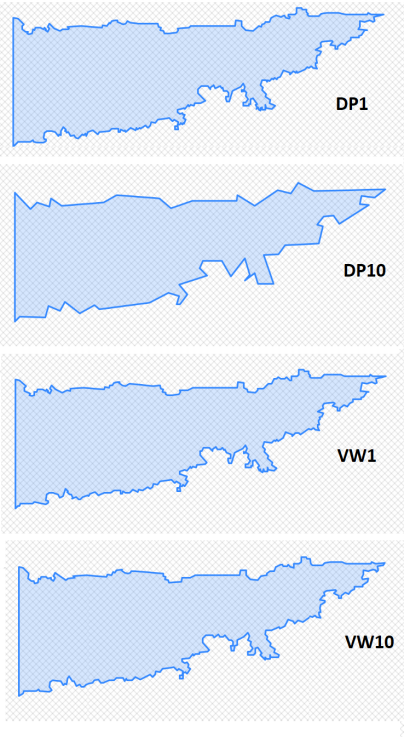

Figure 2: Simplified geometries - database representation

Table 1 presents some statistics on the number of vertices per geometry after simplification. For instance, the average number of vertices per geometry after simplification using DP10 is only 4% of the average number of vertices per geometry in the original dataset.

Table 1: Number of points per geometry after simplification

	DP1	DP10	VW1	VW10
min (% of original)	37	3	79	21
max (% of original)	41	6	79	24
avg (% of original)	40	4	79	22

Tables 2 and 3 present the similarity between each original geometry and its simplified versions using the Jaccard Index and the Hausdorff Distance.

The values of the Jaccard index are near 1.0 in all methods, which means that the similarity between original and simplified geometries is high. The results of the Hausdorff Distance when using the Visvalingam-Whyatt algorithm are significantly higher for some observations than those obtained using the Douglas-Peucker algorithm. This means that there are local deformations on the contours of the simplified geometries in the VW datasets, but these local

Table 2: Jaccard Index - Simplified dataset x Original dataset

	DP1	DP10	VW1	VW10
min	0.992	0.941	0.998	0.996
max	0.998	0.983	0.999	0.987
avg	0.996	0.967	0.999	0.993

Table 3: Hausdorff Distance - Simplified dataset x Original dataset

	DP1	DP10	VW1	VW10
min	1	8	1	3
max	1	10	49	49
avg	1	9	3	7

deformations were not big enough to influence the area similarity significantly.

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. The experiments were executed using PostgreSQL 11, PostGIS 2.5.2 and GEOS 3.6.2.

5.1 Dissimilarity distance-based approach

We experimentally created time slices for all the 5 variations (original, DP1, DP10, VW1 and VW10) of our dataset using the dissimilarity distance-based algorithm proposed in section 3. We use the *Jaccard Distance* as the $distance$ function between two observations (o_i, o_j), as defined on Section 2.3. We choose to use the Jaccard Distance as the dissimilarity distance because we want to study how the burned area evolves (i.e. grows) over time and local deformations on the geometry contours are not as important as the changes in the overall geometric area.

The algorithm was evaluated using several values for α. Figure 3 presents the results.

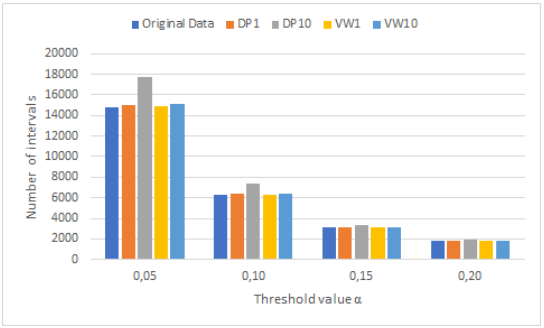

Figure 3: Number of time slices per dataset and value of α

The number of time slices created decreases significantly as the threshold value α increases. With $\alpha = 0.05$ most configurations led to a number of time slices of about 70% of the number of observations, while when using $\alpha = 0.2$, the number of time slices datasets was of about 8% of the number of original observations.

The number of time slices created using the original and the simplified datasets was very similar, with the original dataset leading to just a few less slices than the simplified ones. 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. This indicates that it is possible to use the simplified geometries to generate time slices and obtain a representation that is close to the one obtained using the original geometries. Also, as we will present later, the use of the simplified datasets significantly reduce the computational costs.

The average of the time slices length of was similar in all configurations, varying from just 1.5 observations for very low values of α to 12.5 observations (i.e. half a second) when $\alpha = 0.2$. Also, when $\alpha = 0.2$, some large slices were defined, being the largest one of almost 41 seconds (i.e. 1,015 observations) for the VW10 dataset, and of 25 seconds (more than 600 observations) for the other datasets.

There also are some time slices with just one observation. 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 due to noise or other anomalies in the data, like exemplified in Section 5.3.

After identifying the time slices to be created, we used the implementation of PySpatioTemporalGeom [10] algorithm available at [9] to estimate the shape of the intermediate geometries within each slice. Then, we computed the Jaccard Index to determine the similarity between each geometry generated using the interpolation algorithm and its corresponding one in the original dataset.

Figure 4 presents the average Jaccard Index for all simplified datasets using two configurations (i.e. two values of α). The results show that there is no significant variation on the Jaccard Index for each value of α, which means that the interpolations for all simplification methods (and tolerances) are similar.

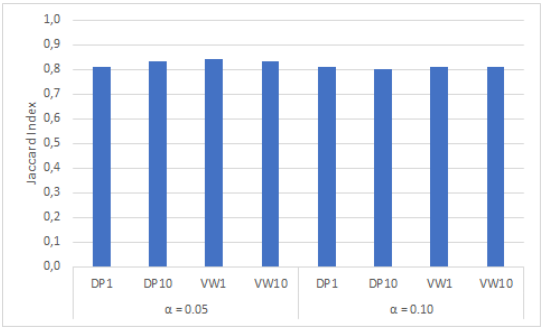

Figure 4: Jaccard Index - Comparing simplification methods and tolerance

To evaluate how the dissimilarity distance (i.e. α) influences the interpolation, we must use the same dataset. Figure 5 presents the Jaccard Index for the VW10 dataset and four values of α. For the

lowest α, the average of the Jaccard Index was 0.83, while for the largest value of α (0.20), the average of the Jaccard Index was 0.78. This means that the dissimilarity distance between observations used to define the time slices influenced the interpolation.

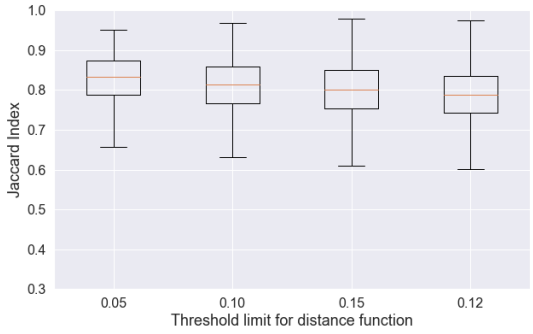

Figure 5: Jaccard Index - simulated geometries x original ones - VW10 dataset and several values of α

5.2 Using fixed-size slices

We evaluated the use of the fixed-size time slice strategy, considering 5 distinct sizes: 3 observations, 7 observations, 12 observations (which are almost 1 observation per half a second in the original movie), 25 observations (1 per second of the original movie) and 1,500 observations (1 per minute of the original movie).

The shape of the intermediate geometries within a time slice was generated using the PySpatioTemporalGeom implementation and the Jaccard Index was used to compare the shapes generated using that algorithm with the corresponding ones in the original dataset, in the same way we did for the distance-based approach. Figure 6 presents the Jaccard Index obtained when using the VW10 dataset. The lowest values of Jaccard Index were obtained for the smallest and the largest slices. For the three mid-size slices the values of the Jaccard Index are similar.

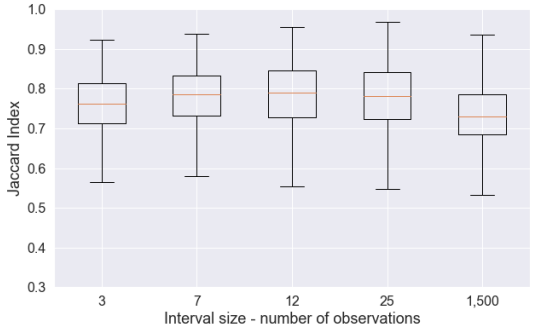

Figure 6: Jaccard Index - simulated geometries x original ones - VW10 dataset and fixed size slices

When using the fixed size strategy with slices of length 7 and the distance-based strategy with $\alpha = 0.85$, on the VW10 dataset, we could remove 60.9% and 67.9% of the samples, respectively, and so the compression rates are similar. We calculated the Jaccard Index comparing the interpolated samples with the original frames, and our dissimilarity distance method is significantly better than when using the fixed-size strategy (Mann-Whitney U test, $p<0.05$), even though we removed slightly more samples. The results of the Jaccard Index are presented on Figure 7, on the left side.

The fixed sampling interval 12 and the distance-based $\alpha = 0.80$ removed respectively 74.0% and 75.8% of the samples, again with similar compression rates. We calculated the Jaccard Index comparing the interpolated samples with the original frames, and our method is again significantly better (Mann-Whitney U test, $p<0.05$). The results of the Jaccard Index are presented on Figure 7, on the right side.

These results show that for a similar amount of samples, the dissimilarity distance-based strategy proposed in this work are better than using uniform sampling.

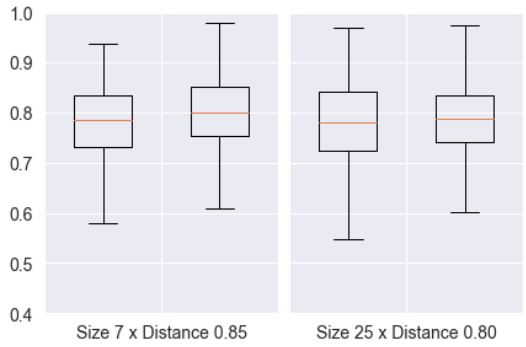

Figure 7: Jaccard Index - simulated geometries x original ones - VW10 dataset

5.3 Discussion

When using the fixed size (uniform sampling) strategy, the user is, in fact, specifying a compression rate to be used and defining the final number of slices to be created. The dissimilarity distance-based strategy, on the other hand, provides no guarantee on the compression rate and depending on the value of α, the number of slices can vary drastically. For instance, analyzing the results of section 5.1, together with the results of the Jaccard Index (figure 5), it is possible to identify that an increase of 0.15 (from 0.05 to 0.20) on the acceptable dissimilarity (i.e. α) would reduce the number of slices from 69% to just 8% of the number of original observations, with a drop off of just 5% (on average) on the similarity 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 representing a single observation. That means such observations are

significantly dissimilar to their previous and following ones, which indicates that these observations 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 8, which presents 3 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 9), 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.

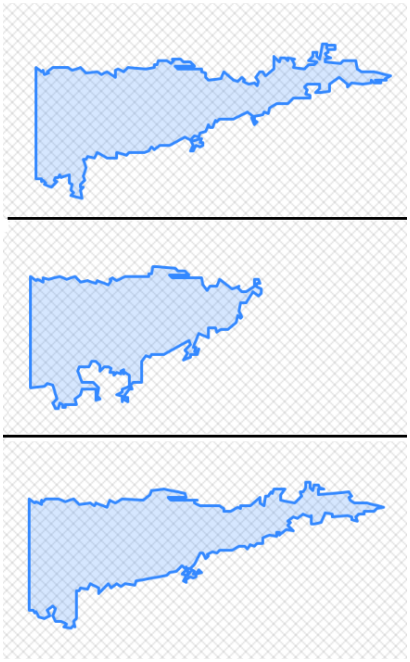

Figure 8: Sequence of observations with an outlier in the middle

There also were time instants to which the interpolation method could not generate simulations. In our experiments, the number of failures varied according to the number of points in each geometry, as represented in Figure 10. This does not imply that the interpolation method has any limitation on the number of points per geometry, but geometries with thousands of points are more likely to have features (like holes or self-intersections) that prevent the interpolation algorithm from generating simulations. On the other hand, it is important to notice that each dataset in Figure 10 has its own simplified version of each original geometry, and that those simplified representations may remove (or create) self-intersections and other anomalies. This means that in certain cases, it may be possible to use interpolation method on some original geometries and not on the corresponding simplified geometries, or vice-versa.

Figure 9: Smoke leading to inaccurate object detection

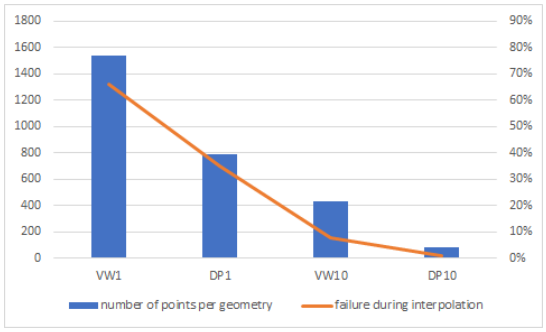

Figure 10: Interpolation failure

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

We also measured the execution time in several configurations and identified it was influenced by the number of points in input geometries. For instance, when using time slices generated by the distance-based algorithm (with $\alpha = 0.1$), the number of simulations to be done in each dataset was relatively close, but the average execution time for VW1 and DP1 was almost 10 times greater than the average execution time for DP10 and VW10. This was expected due to the algorithmic complexity of the distance functions used.

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 original (discreetly obtained) observations should be used to create a sliced representation, which comprises procedures to select the most adequate observations and defining interpolation functions that represent well the real world phenomena.

In this work we propose using a dissimilarity distance threshold between consecutive observations as a mechanism to identify time slices in the continuous representations. Such threshold aims to guarantee that there is a smooth transition between all the real observations contained in a slice. We experimentally evaluated

the distance based proposal, testing several configurations and dissimilarity limits. Experimental results prove the dissimilarity limit improves the quality of in-between observations generated by interpolation algorithms. We also studied how the use of geometry simplification algorithms may impact on time slice definition, and identified that the use of simplified objects does not impact negatively on slices, but improves interpolation execution time. We also compared our distance based proposal with the use of fixed size time slices, using real world data, and proved the benefits of using strategies that take into account the evolution of historical data to define the boundaries of the slices when creating continuous representations of moving objects in databases.

As future work, we plan to do a more detailed study on how to detect and remove (or correct) noises, anomalies and outliers in the discrete observations while generation 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 quality) as part of our study on burned area evolution, specially to simulate such evolution during time frames on which there is no real data.

ACKNOWLEDGMENTS

This work was partially funded by REMOVED DUE TO DOUBLE BLIND REVIEW.

REFERENCES

[1] David H Douglas and Thomas K Peucker. 1973. 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 (1973), 112–122.

[2] José Duarte, Paulo Dias, and José Moreira. 2018. A Framework for the Management of Deformable Moving Objects. In *Geospatial Technologies for All*, Ali Mansourian, Petter Pilesjö, Lars Harrie, and Ron van Lammeren (Eds.). Springer International Publishing, Cham, 327–346.

[3] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. 1998. 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)*. ACM, New York, NY, USA, 131–136. https://doi.org/10.1145/288692.288716

[4] Omitted for double-blind review. 2019. Omitted for double-blind review. Retrieved Sept, 2019 from https://Omitted_for_double_blind_review

[5] Luca Forlizzi, Ralf Hartmut Güting, Enrico Nardelli, and Markus Schneider. 2000. 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)*. ACM, New York, NY, USA, 319–330. https://doi.org/10.1145/342009.335426

[6] R. H. Guting, V. Almeida, D. Ansorge, T. Behr, Z. Ding, T. Hose, F. Hoffmann, M. Spiekermann, and U. Telle. 2005. SECONDO: an extensible DBMS platform for research prototyping and teaching. In *21st International Conference on Data Engineering (ICDE'05)*. 1115–1116. https://doi.org/10.1109/ICDE.2005.129

[7] Hideki Hayashi, Akinori Asahara, Natsuko Sugaya, Yuichi Ogawa, and Hitoshi Tomita. 2016. 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)*. ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3017611.3017615

[8] Florian Heinz and Ralf Hartmut Güting. 2016. Robust high-quality interpolation of regions to moving regions. *GeoInformatica* 20, 3 (01 Jul 2016), 385–413. https://doi.org/10.1007/s10707-015-0240-z

[9] Mark McKenney. 2019. pyspatiotemporalgeom - PyPI. Retrieved Sept, 2019 from https://pypi.org/project/pyspatiotemporalgeom/

[10] Mark Mckenney and Roger Frye. 2015. Generating Moving Regions from Snapshots of Complex Regions. *ACM Trans. Spatial Algorithms Syst.* 1, 1, Article 4 (July 2015), 30 pages.

[11] Mark McKenney and James Webb. 2010. Extracting Moving Regions from Spatial Data. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10)*. ACM, New York, NY, USA, 438–441. https://doi.org/10.1145/1869790.1869856

[12] Xiaofeng Meng and Jidong Chen. 2011. *Moving Objects Management: Models, Techniques and Applications* (1st ed.). Springer Publishing Company, Incorporated.

[13] PostGis-Team. 2019. ST_SimplifyPreserveTopology - PostGIS dev Manual. Retrieved Sept 9, 2019 from https://postgis.net/docs/ST_SimplifyPreserveTopology.html

[14] PostGis-Team. 2019. ST_SimplifyVW - PostGIS dev Manual. Retrieved Sept 9, 2019 from https://postgis.net/docs/ST_SimplifyVW.html

[15] Maribel Santos and Jorge PeÃśa. 2011. Representing, Storing and Mining Moving Objects Data. *Lecture Notes in Engineering and Computer Science* 2192 (07 2011).

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

[17] Willington Siabato, Christophe Claramunt, Sergio Ilarri, and Miguel Angel Manso-Callejo. 2018. A Survey of Modelling Trends in Temporal GIS. *ACM Comput. Surv.* 51, 2, Article 30 (April 2018), 41 pages. https://doi.org/10.1145/3141772

[18] Erlend Tøssebro and Ralf Hartmut Güting. 2001. Creating Representations for Continuously Moving Regions from Observations. In *Advances in Spatial and Temporal Databases*, Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 321–344.

[19] Sebastián Villarroya, José R. Viqueira, Manuel A. Regueiro, José A. Taboada, and José M. Cotos. 2016. SODA: A Framework for Spatial Observation Data Analysis. *Distrib. Parallel Databases* 34, 1 (March 2016), 65–99. https://doi.org/10.1007/s10619-014-7165-7

[20] Jose R. Rios Viqueira and Nikos A. Lorentzos. 2007. SQL Extension for Spatio-temporal Data. *The VLDB Journal* 16, 2 (April 2007), 179–200. https://doi.org/10.1007/s00778-005-0161-9

[21] Mahes Visvalingam. 2015. Explorations in Digital Cartography. (2015).

[22] Maheswari Visvalingam and James D Whyatt. 1993. Line generalisation by repeated elimination of points. *The cartographic journal* 30, 1 (1993), 46–51.

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

[24] Jianqiu Xu and Ralf Hartmut Güting. 2012. Manage and Query Generic Moving Objects in SECONDO. *Proc. VLDB Endow.* 5, 12 (Aug. 2012), 2002–2005. https://doi.org/10.14778/2367502.2367558