Towards the Automatic Selection of Moving Regions Representation Methods

Rogério Luís C. Costa
rogerio.l.costa@ipleiria.pt
CIIC, Polytechnic of Leiria
IEETA, University of Aveiro
Portugal

Enrico Miranda
enrico.miranda@ua.pt
IEETA, University of Aveiro
Portugal

José Moreira
jose.moreira@ua.pt
DETI - IEETA, University of Aveiro
Portugal

ABSTRACT

Moving region is an abstraction used to represent the spatio-temporal behavior of real-world phenomena in database systems. The most common approach to model moving regions uses geometries to represent their position and shape at different times (observations), and interpolation functions to generate the evolution of the geometries between observations.

Several region interpolation methods have been proposed in the databases literature, but as there is no suitable method for all use cases, users must select the most adequate algorithm to represent each region by visual inspection. This can be infeasible when dealing with large datasets.

This paper presents the first steps towards a system that suggests which methods (and configurations) can generate representations fitting the requirements of a particular application. It includes an abstract specification of user-defined rules on the spatio-temporal evolution of moving regions to assess the suitability of region interpolation functions, a discussion on optimization strategies for efficient implementation of the rules and illustrative examples using real-world data to show how to use this approach to select the best methods to represent a spatio-temporal phenomena.

CCS CONCEPTS

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

KEYWORDS

Spatio-temporal data, continuous representation, moving regions

ACM Reference Format:
Rogério Luís C. Costa, Enrico Miranda, and José Moreira. 2020. Towards the Automatic Selection of Moving Regions Representation Methods. In *3rd ACM SIGSPATIAL International Workshop on GeoSpatial Simulation (GeoSim'20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3423335.3428170

1 INTRODUCTION

The evolution of spatio-temporal phenomena, such as the movement of icebergs, the morphological changes of living cells, or the propagation of forest fires, can be represented in databases as *moving regions*. In practice, image segmentation techniques are applied over images to extract the geometry (i.e., position and shape) of the regions of interest at some timestamps and, then, region interpolation methods are used to create a continuous spatio-temporal representation of evolution of the region.

Currently, there are some spatio-temporal DBMS prototypes and extensions (e.g. [4, 8]) to represent moving regions. These proposals simulate the spatio-temporal behavior of moving regions by means of region interpolation methods and so, they enable retrieving data about their spatial properties in-between observations, i.e., when there is no real data.

The definition of interpolation functions over moving regions is challenging. Existing methods operate over pairs of polygons and can create valid spatio-temporal transformations but often fail to generate realistic representations of the behavior of real-world objects. For instance, consider Figure 1 presenting two snapshots of an iceberg called B-15 (available at https://visibleearth.nasa.gov/) and Figure 2 presenting the corresponding geometries (in gray) of the larger fragment and three intermediate geometries (in blue) between the the two snapshots estimated using a region interpolation method. Although the geometry is valid, it is probably not a realistic representation of the evolution of the iceberg because the concavity in the initial and final snapshots becomes a hole in the intermediate representations.

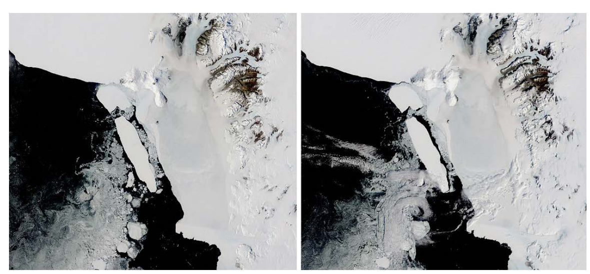

Figure 1: Real iceberg photos

As each method can create a different representation of the evolution of a moving region (e.g. in terms of deformation, rotation and the existence of holes) it is necessary to choose the one that best simulates the evolution of the phenomena to be modelled [1]. Furthermore, several aspects may influence the results estimated by a region interpolation method, e.g., the selection of the observations used to create the interpolations or the algorithms used for geometry simplification [1, 2]. In practice, quality and adherence

GeoSim'20, November 3–6, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8161-1/20/11.
https://doi.org/10.1145/3423335.3428170

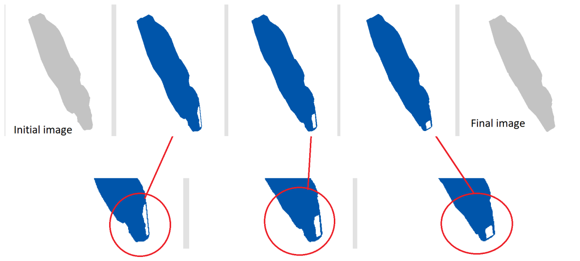

Figure 2: Iceberg simulated spatio-temporal evolution

analysis between the estimations and the expected spatio-temporal evolution of a real-world phenomena is performed visually [5]. Therefore, choosing an interpolation method may require a significant effort, which can make it unfeasible to scale this kind of procedure at a large database level. Thus, a question that arises in this scenario is how to select the method that best simulates the spatio-temporal behavior of an object or phenomenon.

A possible solution is to have a system able to evaluate several region interpolation methods and identify the ones that would be adherent to the requirements of a particular application. Users would specify the requirements in the form of rules (consistency checks) on the representation of moving regions. For instance, the user could use a function to specify that the representation of the iceberg in Figure 1 must not contain holes (e.g. in PostGIS's syntax $ST\_NumInteriorRings(region) = 0$) and the system should evaluate the simulations generated using different and select the ones that satisfy that rule at every time.

The rules may be defined on the transformation (e.g. an iceberg does not deform in the same way as an oil spill in the sea), the trajectories (e.g. a fire cannot traverse a lake), or the rate of change of the moving regions (e.g. to establish an upper bound on the rate of growth of the area or perimeter of a moving region).

The main contributions of this work are (i) a framework to evaluate the simulations generated by several region interpolation methods to represent a spatio-temporal phenomena in a database, (ii) the specification of user-defined rules and an algorithm to select best methods with respect to the requirements of a given application, (iii) a discussion of heuristics and strategies to be incorporated into a sophisticated *interpolation evaluation and execution optimizer* and (iv) examples of application over real-world data.

2 BACKGROUND AND RELATED WORK

There are two main approaches to represent spatio-temporal data in database systems: (i) to use time series, where geometries are associated to timestamps, or (ii) to use continuous models [6, 9] and abstractions, like moving points, moving lines and moving regions. In this paper, we deal with moving regions, whose representation may be decomposed into an ordered sequence of slices [7]. Each slice consists of a geometry, that denotes the position and shape of an object at a given time, and a function, that represents the spatial transformation of the geometry during the slice.

The spatial transformations in a slice are given by region interpolation methods (e.g. [4, 10–13]) that provide representations with different characteristics and behaviors, and there is no consensus on a method that provides realistic interpolations for all use cases. In fact, the evaluation the spatio-temporal transformations generated using region interpolation methods is commonly performed visually and case by case. Duarte et al. [5] present a tool that allows the execution of several methods to analyse and compare the results of interpolations by visual inspection.

3 IMPROVING REPRESENTATION QUALITY

Validating the geometries estimated in-between observations by visual inspection is an arduous task, particularly when large datasets are involved, and so it is important to develop tools to perform this task automatically.

3.1 User-defined Rules

We propose an approach to evaluate and select the most suitable region interpolation method to represent a phenomena using user-defined metadata provided as rules over shapes and trajectories. There are two main classes of rules:

I - At instant rules - Evaluate (a feature of) a geometry at each timestamp, comparing it with a (null, numeric, date, ...) value or with another geometry (looking for intersections, for example). These include features about shapes, as well as constraints on trajectory paths. For instance, one can define that a boomerang is symmetric, that a cell does not have holes or use topological predicates to prevent overlaps. These rules allow using any operator (relational, spatial, etc) provided by the DBMS, and should be validated considering the representation of the geometry at any instant.

Let $g(t)$ represent the geometry of a moving region at time t, $f(g(t))$ a function (e.g. the PostGIS functions $ST\_NumInteriorRings$ or ST_Area) on $g(t)$, and $Value$ a (numeric, date, boolean, moving, ...) value. The rules in this family can be specified using Equation 1.

$$\forall t; f(g(t) \times Value) = True \tag{1}$$

II - Rules on temporal evolution - These rules are applied on the temporal evolution of moving regions based on metrics and calculated metadata. They are used to impose constraints on velocity and other patterns on the movement (e.g., translation) or the transformation of the shape over time. For example, it could be assumed that the area of an iceberg floating in the Atlantic Ocean decreases monotonically or that in the geometry representing the region burned in a forest fire at a time instant t is within the geometry representing the region burned at any $t' > t$.

Considering that $g(t_i)$ and $g(t_j)$ represent a geometry at instants t_i and t_j respectively, $f(g)$ is a function and $Value$ is a (null, numeric, date, boolean, moving, ...) value, the rules on the temporal evolution of moving regions can be specified using Equation 2.

$$\forall i, j; i < j; f(g(t_i) \times g(t_j) \times Value) = True \tag{2}$$

3.2 Algorithm to select methods

Let's assume that Req denotes a set of user-specified rules, RI_list a set of region interpolation methods implemented by the system, and $R1$ and $R2$ are the initial and final geometries (observations) of a moving region's slice. The evaluation process consists of applying each region interpolation method to estimate the geometry of the moving region at predefined checkpoints (time instants within a slice) and test whether the consistency rules are valid for the estimated geometries (Algorithm 1). *EvaluateRI* is a recursive procedure that defines the checkpoints according to a threshold parameter δ (the required precision in terms of time slices' size) and performs the rules checking. This procedure is called in *EvaluateRequest* for each method in *RI_list*. This procedure succeeds if the region interpolation method passes all consistency checks, otherwise the result is *null*.

Algorithm 1 Requirements evaluation

$R1$, $R2$: initial and final region representations, respectively
Req: user specified requirements
procedure EvaluateRequest(R1, R2, Req)
    $t_i \leftarrow TimestampOf(R1)$
    $t_f \leftarrow TimestampOf(R2)$
    for each Method G_j in *RI_list* **do**
        if $EvaluateRI(G_j, R1, R2, Req, t_i, t_f)$ **then**
            return G_j
        end if
    end for
    return NULL
end procedure
procedure EvaluateRI(G, R1, R2, Req, t1, t2)
    if $t2 - t1 < \delta$ **then**
        return True
    end if
    $tm \leftarrow (t2 - t1)/2$
    $P \leftarrow G(At\_Instant(tm), R1, R2)$
    return $EvaluateRequirements(P, Req)$ AND $EvaluateRI(G, R1, R2, Req, t1, tm)$ AND $EvaluateRI(G, R1, R2, Req, tm, t2)$
end procedure

As users may specify rules and queries for spatio-temporal representations in interactive mode, it is convenient to have a more sophisticated *optimizer* that uses heuristics and strategies to efficiently select the best region interpolation method, particularly:

I - Heuristics or learning methods to evaluate constraints - Evaluating the consistency rules for each geometry representing the evolution of a moving region during a time interval can be a computationally costly task in itself. Ideally, It would be up to the optimizer to decide the order of the evaluation of the rules and the order of region interpolation method evaluation. For instance, the algorithm may start by checking the least computationally demanding rules in order to eventually discard some methods and avoid performing the most costly tests on all region interpolation methods. It would also be reasonable to adopt heuristics that prioritize the evaluation of methods selected on the slices evaluated previously. Moreover, interpolation methods have distinct features that make them more prone to a particular behavior. By learning recurring behaviors of each method, the optimizer could decide to first perform the consistency checks that are most likely to fail.

II - Sampling strategies - Sampling may be used to evaluate whether a method satisfies the consistency rules at predefined checkpoints. The number of checkpoints may vary depending on desired confidence level (i.e. δ) and on the rules to be evaluated.

III - Use of caching and materialization - Since users might perform several queries on the same data, the use of a cache on interpolated geometries may significantly improve evaluation performance. Also, the use of offline checks and materialization of tests' results considering rules defined during data storage would allow costly tests to be executed even before users query for data.

4 APPLICATIONS ON REAL-WORLD DATA

This section presents how to express user-defined rules using SQL and illustrative examples over real-world data. Three region interpolation methods are used: Secondo [10], PySpatioTemporalGeom [11] and SPTMesh [4]. The source images and WKT representations of geometries are available at http://most.web.ua.pt/datasamples.zip.

4.1 Using SQL to express rules

We propose the use of the SQL language extension *REQUIREMENTS* [3] to express user-defined rules. This clause allows a flexible definition that can be used either when storing a specific representation or when querying stored data. For instance, in the following SQL command the rules are (i) the perimeter of the region must be smaller than 100 units at any time instant and (ii) the estimated geometries should not contain holes during the interpolation.

```
SELECT ST_Get_AtPeriod(region,'PERIOD(1100 4500)')
FROM db.regions WHERE id=5
Requirements
   ST_Perimeter(region) < 100
   AND ST_NumInteriorRings(region) = 0;
```

The main idea is to be able to use the statements in the REQUIREMENTS clause as the rules to be used by the optimizer to look for valid spatio-temporal evolution representations.

4.2 Iceberg Evolution

Consider the B-15A iceberg represented in Figure 1 and the result of the interpolation obtained using Secondo in Figure 2. Now, we may compare the results of this method with the ones obtained using PySpatioTemporalGeom and SPTMesh (see the statement below). The first rule is to guarantee that the geometries (iceberg) cannot have holes during the interpolation. Moreover, as real-world data is noisy, we allowed for a tolerance of 1% on the area of the geometries relatively to the initial and final representations of a moving region, as well as on the area of the geometries estimated for two consecutive checkpoints (we use the LEAD window function to access the next representation on continuous time):

```
ST_NumInteriorRings(region) = 0
AND ST_Area(region) <= initial_area * 1.01
AND ST_Area(region) >= final_area * 0.99
AND ST_Area(region) >= LEAD(ST_Area(region)*1.01)
                        OVER (ORDER BY Instant);
```

SPTMesh is the only region interpolation method satisfying all the requirements specified in the statement, and after inspection the results were deemed acceptable. Otherwise, we should adjust the parameters to define constraints that are less restrictive and run the procedure again.

4.3 Forest Fire Evolution

We now consider the evolution of the burned area in a controlled fire, carried out at Pinhão Cel, Portugal on June 2019. Figure 3 presents two snapshots of the field at two different times, the initial and the final states in our interpolation. The geometries represent the burned area in both images.

Figure 3: Controlled Fire - Segmented Burnt Area

First we consider that the burned area should always grow during the interpolation. The evaluation of the three region interpolation methods shown that they all meet this requirement. However, as seen in Figure 4, the shape of the line segment in the left border, which should always be a vertical line, is highly variable when using two of the interpolation methods. Thus, to handle this issue, we may also specify that the region's left boundary should contain the vertical line that delimits the left border of the burned area in the initial image. After that, the interpolation generated using Secondo is the only one that satisfies both requirements. Actually, by visual inspection it may be verified that the evolution generated by this method was also more realistic.

```
ST_Area <= LEAD (ST_Area(region))
            OVER (ORDER BY Instant))
AND ST_HausdorffDistance(region,
            'LINESTRING(0 190, 0 480)') = 0;
```

Although SPTMesh obtained the most appropriate results for the iceberg's requirements, it was Secondo that met the rules for the burned area evolution. These examples also show how to use rule-based selection to identify suitable interpolation methods to simulate the behaviour of real-world spatio-temporal events.

5 CONCLUSION

The representation of the spatio-temporal behavior of objects using moving regions requires defining interpolation functions over sequences of observations. However, existing region interpolation methods often create unrealistic transformations regarding the spatio-temporal behaviour of the objects or events in the real-world.

In this paper we propose using user-defined rules over the continuous representation of moving regions (i.e. interpolation results) to automatically select the most suitable representation. Two families of consistency checks are proposed, an evaluation algorithm is presented and the main challenges on implementing a sophisticated optimizer for rules evaluation are discussed. We also present a solution to define rules using SQL and examples on using the proposed strategies over real-world data.

The next steps are a formal definition of the language to specify the user-defined rules and the use of learning methods to improve the system's capacity to identify object behaviour over past decisions.

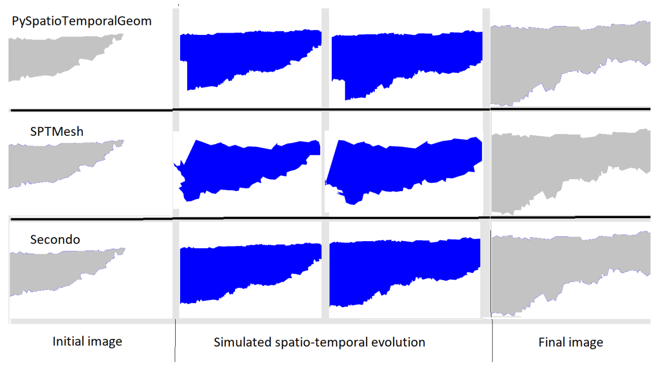

Figure 4: Simulation on Burned Area Evolution

ACKNOWLEDGMENTS

This work is partially funded by National Funds through the FCT (Foundation for Science and Technology) in the context of the projects UIDB/04524/2020, UID/CEC/00127/2013 and POCI-01-0145-FEDER-032636.

REFERENCES

[1] Rogério Luís C. Costa, Enrico Miranda, Paulo Dias, and José Moreira. 2020. Evaluating Preprocessing and Interpolation Strategies to Create Moving Regions from Real-World Observations. *SIGAPP Appl. Comput. Rev.* 20, 2 (July 2020), 46–58.
[2] Rogério Luís C. Costa, Enrico Miranda, Paulo Dias, and José Moreira. 2020. Sampling Strategies to Create Moving Regions from Real World Observations. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC '20)*. Association for Computing Machinery, New York, NY, USA, 609–616.
[3] Rogério Luís de Carvalho Costa and Pedro Furtado. 2011. Quality of experience in distributed databases. *Distributed and Parallel Databases* 29, 5 (2011), 361–396.
[4] José Duarte, Paulo Dias, and José Moreira. 2018. A Framework for the Management of Deformable Moving Objects. In *Geospatial Technologies for All - Selected Papers of the 21st AGILE Conference on Geographic Information Science (Lecture Notes in Geoinformation and Cartography)*. Springer, 327–346.
[5] José Duarte, Bruno Silva, José Moreira, Paulo Dias, Enrico Miranda, and Rogério L. C. Costa. 2019. 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)*. Association for Computing Machinery, New York, NY, USA, 592–595.
[6] 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.
[7] 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)*. Association for Computing Machinery, New York, NY, USA, 319–330.
[8] Ralf Hartmut Guting, Victor Almeida, Dirk Ansorge, Thomas Behr, Zhiming Ding, Thomas Hose, Frank Hoffmann, Markus Spiekermann, and Ulrich Telle. 2005. SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching. In *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*. IEEE Computer Society, USA, 1115–1116.
[9] Sofie Haesevoets and Bart Kuijpers. 2000. Closure Properties of Classes of Spatio-Temporal Objects for Boolean Set Operations. In *Procs 7th International Workshop on Temporal Representation and Reasoning. TIME 2000*. IEEE, USA, 79–86.
[10] Florian Heinz and Ralf Hartmut Güting. 2016. Robust high-quality interpolation of regions to moving regions. *GeoInformatica* 20, 3 (Jul 2016), 385–413.
[11] Mark Mckenney and Roger Frye. 2015. Generating Moving Regions from Snapshots of Complex Regions. *ACM Trans. Spatial Algorithms Syst.* 1 (2015), 4:1–4:30.
[12] 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.
[13] Erlend Tøssebro and Ralf Hartmut Güting. 2001. Creating Representations for Continuously Moving Regions from Observations. In *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001 (Lecture Notes in Computer Science)*, Vol. 2121. Springer, 321–344.