Representation of continuously changing data over time and space

Modeling the shape of spatiotemporal phenomena

José Moreira, Paulo Dias
DETI / IEETA
Universidade de Aveiro
Aveiro, Portugal
{jose.moreira, paulo.dias}@ua.pt

Pedro Amaral
DETI / Retail Consult
Universidade de Aveiro, Retail Consult
Portugal
apamaral@ua.pt

***Abstract*—There are numerous technologies and tools to acquire data related to the evolution of spatial phenomena over time. These data are typically organized as sequences of 2D geometric shapes obtained from observations taken at different times. The transformation of such sequences of 2D geometric shapes into spatiotemporal data representations, which can be easily processed and interpreted, has the potential to enable novel applications in fields as diverse as environmental sciences, climate sciences, biology or medicine.**

This paper focuses on the representation of moving 2D geometric shapes acquired at discrete times using continuous models of time and space. Using morphing techniques based on compatible triangulations, issues regarding the representation of spatiotemporal data in databases, as well as the influence of different design strategies on the fidelity of the approximations with respect to the modelled phenomena, are investigated. An experimental study using synthetic and real data was performed. The findings show that the use of triangulation based interpolation is a promising approach, because it allows creating continuous spatiotemporal representations that are more realistic than those obtained using the solutions proposed in previous work. Open issues regarding the representation of spatiotemporal data in information systems are also highlighted.

Keywords—spatiotemporal databases; tracking spatial phenomena; 2D planar shape morphing; triangulation;

I. INTRODUCTION

The widespread use of novel technologies allowing to log large amounts of data concerning the location and shape of real-world phenomena over time is motivating the development of automated processes to extract knowledge from these data in several domains (e.g., earth, environmental or climate science) and applications (e.g., land use management, traffic control, urban planning or biomedical imaging). Nowadays, numerous methods and tools are available to deal with spatial data efficiently, namely spatial databases and GIS [1,2]. However, the representation and processing of spatial data changing over time is more complex, and there are several open-issues to overcome.

As for spatial data, spatiotemporal data can be represented by (1) approximating a continuous space by a discrete one, usually referred to as raster or tessellation mode, or (2) using a continuous representation, usually referred to as vector mode or half-plane representation. The representation modes for the time and space dimensions are independent, and the choice strongly depends of the applications' requirements [3]. The focus of this work is on creating spatiotemporal data using continuous representations (vector mode) over space and time dimensions that might be used to model phenomena such as the movement, melting and collapsing of icebergs, wildfire propagation, siltation in rivers, or the morphological behavior of cells and particles in biology or medicine [4-12].

The evolution of spatiotemporal phenomena is typically acquired as discrete time-ordered sequences of observations in the form of images and represented using abstractions often referred to as moving or spatiotemporal objects. Thus, to capture the continuous change of such phenomena, methods are needed to model their spatial behavior between observations. The objective is to obtain continuous spatiotemporal representations that are close to the objects' actual shapes and movements at all times. This has been concretized on different data models and query languages, firstly using constraint databases [13], and then using Abstract Data Types (ADT) [14-19]. These solutions provide a comprehensive collection of general-purpose data types and query operations to deal with different application domains. Observations are represented by 2D shapes and spatial transformations between observations are represented using linear interpolation. However, few works exist on how to create realistic representations (interpolations) of real-world phenomena for spatiotemporal databases and GIS [20].

This paper investigates two approaches for creating spatiotemporal data, and the effect of different design strategies on the fidelity of the obtained representation when compared with the modelled phenomena. In the first approach, the transformation of an object is defined by the movement of line segments using the so-called rotating plane algorithm [22-24]. Translation and rotation are represented implicitly, and so, these solutions are poor in handling large rotations between observations. The second approach relies on using morphing techniques to attempt to preserve the morphological properties of 2D planar shapes during transformations [25-28]. Although these techniques are widely used, particularly in animation or video editing software packages, their use to model spatiotemporal phenomena in scientific applications is unusual or nonexistent. The solution investigated in this work is based

on compatible triangulations [25,26,29]. The aim is to bring together solutions from different research areas in order to be able to create realistic transformations representing spatiotemporal phenomena with small errors of approximation and reasonable runtime cost to enable querying large spatiotemporal datasets efficiently.

The comparison of these approaches is based on an empirical study using real and synthetic data. Real data consist of sequences of snapshots, tracking the movement of icebergs in the Antarctic. This is an interesting case study since it involves the three basic spatial transformations, namely, translation resulting from icebergs' movement, rotation caused by ocean currents near shorelines, and deformation due to melting and fragments collapsing. The objective is to compare the spatiotemporal representations created using both approaches with respect to the real-world phenomena modeled, as well as investigating computational aspects such as, the ability to yield spatiotemporal data that are topologically valid at all times.

The remainder of this paper is organized as follows: Section II presents the two approaches investigated in this study and highlights design and implementation issues of each one; Section III presents the results of the experimental study with emphasis on functional and computational aspects; Section IV presents a discussion and a comparison of the two approaches; Section V concludes and presents insights for future research.

II. Creating spatiotemporal data from observations

This work addresses the problem of creating a continuous spatiotemporal representation of a phenomenon captured as a sequence 2D shapes at discrete times. It is assumed that the shapes have been previously extracted from a sequence of images or a video using well-known image processing tools such as ImageJ or OpenCV [30].

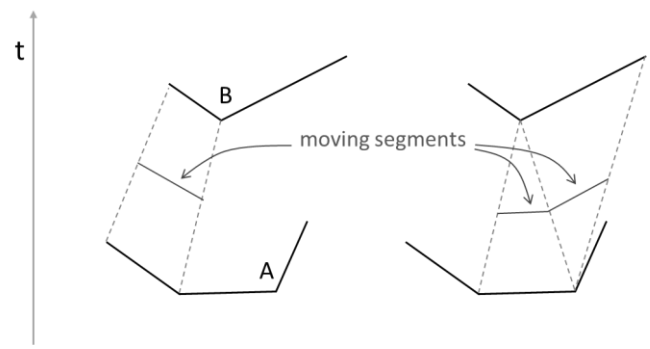
Let us consider a set of snapshots $D = (t, P)$, such that $t = \{t_i | i = 1 \cdots n\}$ denotes an ordered set of timestamps and $\{P_i | i = 1 \cdots n\}$ is a set of planar shapes. Each $P_i = \{v_j | j = 1 \cdots m\}$ consists of an ordered sequence of 2D points corresponding to the vertices of the shape i. The shape is a polygon, if $v_1 = v_m$, or a polyline, otherwise. The former may represent an object such as an iceberg or a human tissue, and the latter may represent a forest fire front. A spatiotemporal transformation between consecutive snapshots is $D_i \xrightarrow{f(t)} D_{i+1}$, where $f(t) = E$ denotes a morphing algorithm that returns an estimation of a spatiotemporal object's shape (E) for every time $t \in [t_i, t_{i+1}]$. In addition, the topology of E must be valid for every time in that interval. A good solution should reduce deformation and create realistic transformations. In the following, the source (P_i) and the target (P_{i+1}) shapes in a spatiotemporal transformation $f(t)$ are denoted A and B, respectively.

A. The rotating plane algorithm

The rotating plane is a morphing algorithm proposed in [22] to create a linear interpolation between convex or concave polygons in spatiotemporal databases. This algorithm takes as input two convex shapes and so, the angles of the edges of each shape with respect to a coordinate axis (e.g., the x-axis) are ordered. The main purpose of this algorithm is to create safe interpolations, i.e., to ensure that there are no intersecting segments during a transformation and that the topology of the resulting spatiotemporal data is valid at all times.

The algorithm takes the first segment (edge) in both shapes and executes a synchronized scan of the two arrays of segments in A and B with respect to their angles. If the angles of the selected segments $\bar{a} \in A$ and $\bar{b} \in B$ are equal, a linear interpolation between them (Fig. 1, left) is performed, and the algorithm goes to the next segment in both arrays. This yields a trapezoid in space-time. If the angles of $\bar{a}$ and b are different, for instance, the angle of $\bar{a}$ is smaller than the angle of $\bar{b}$, then the algorithm takes the first vertex v in $\bar{b}$, creates a linear interpolation between $\bar{a}$ and v (Fig. 1, right), and goes to the next segment in A. Notice that, in this case a segment degenerates into a point thus forming a triangle in space-time.

Fig. 1. Interpolation of two parallel segments (left) and a segment degeneration (right).

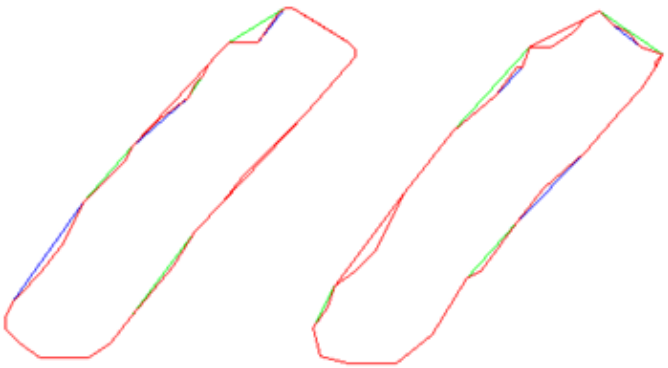


Concave shapes are split into features and organized as convex-hull trees. To build a convex-hull tree it is necessary to add extra segments to make the shape convex. Each extra segment gives origin to a new child node containing the convex-hull of the shape that should be removed from the parent node to obtain the original shape. The convex features may then be processed using the rotating plane algorithm.

Since each shape is decomposed into several convex features, it is necessary to find a matching between features in the convex-hull trees of A and B. Reference [22] recommends using criteria based on an overlap threshold between features. This strategy works well with synthetic data, but problems arise when dealing with real noisy data. For instance, consider Fig. 2, which depicts a projection of the convex-hull trees of the polygons representing the shape of an iceberg at two consecutive snapshots. The convex-hull tree in the left has three levels, and the one in the right has four levels.

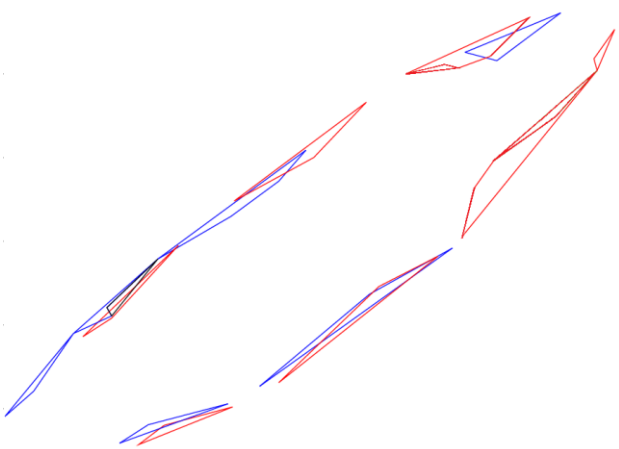
This figure shows that the external features are often thin and it is not trivial to find a match in the other shape. In our implementation, no overlap threshold was found that would allow obtaining a good matching between features of the convex-hulls trees of A and B, when using real data. When the overlap threshold is high, the algorithm may fail to match small concavities; when the overlap threshold is low, there are components that may be erroneously matched; in some cases, there is no overlap at all. Fig. 3 gives illustrative examples.

Fig. 2. Projection of the convex-hull trees for two consecutive snapshots of an iceberg.



As an alternative, a matching procedure using the minimum distance between the centroids of the features in the convex-hulls trees of A and B was implemented, but the results were worse than using an overlap threshold.

Fig. 3. Overlaping between features in the convex-hull trees of two consecutive snapshots of an Iceberg: only features in levels 2 and 3 are displayed.



Paper [23] presents a solution to compute the morphing of two convex or concave shapes, which also uses the rotating plane algorithm, but the strategy to deal with concavities is different. In addition, the authors claim that it is not always possible to use a single interpolation to describe the transformation of a source polygon into a target polygon. This claim is supported with an example representing the transformation of two polygons with a snail shell configuration, to illustrate that some configurations require splitting the interpolation into at most three interpolations.

The strategy to deal with concavities consists of mapping all segments forming a concavity in A into a single vertex in the convex-hull of B, or vice-versa. This means that a concavity in A degenerates into a point in B or a point in A expands to a concavity in B.

The last step consists in detecting edge intersections during the interpolation. These cases may occur for complex shapes with concavities that resemble spirals, i.e., when there are points that are surrounded by a concavity. The solution proposed in [23] consists of splitting an interpolation into smaller ones, so that, surrounding concavities degenerate into a single point and then expand again. Note that, this strategy, as well as the previous one used to deal with simple concavities, causes deformation during transformations. Indeed, the main purpose of the rotating plane algorithm and the other procedures described above is creating spatiotemporal data that are topologically valid at all times, i.e., to avoid segment intersections during spatial transformations.

The movement of the segments during a spatial transformation yields a triangle or a trapezoid, which are planar faces. This is an important feature because this kind of representation can be smoothly integrated into current spatiotemporal data models.

B. Rigid shape interpolation

The aim of rigid shape interpolation is to preserve the shape's physical properties during the transformation of a source into a target, i.e., to perform a spatial transformation that keeps deformation as low as possible. This work uses compatible triangulations, and is based on the approach initially proposed by [26] and followed in several subsequent works [25,27,28]. This approach is considered to handle rotation naturally and has low runtime cost.

It is assumed that a correspondence between the vertices of A and B is created beforehand. The vertex correspondence is a mapping where each vertex in A has a corresponding vertex in B. If the number of vertices in A and B differs (a normal situation with real data) it is necessary to add extra vertices in A or B. This is a well-known problem in morphing, which is commonly referred to as vertex correspondence problem and several solutions exist to perform this task [30,31]. The first step consists of creating a compatible triangulation to generate an isomorphic meshing of the interiors of A and B, denoted ($\hat{A}$) and ($\hat{B}$), respectively. The algorithm presented in [29] was selected, because it allows to create compatible triangulations yielding a small number of Steiner vertices (extra vertices).

This algorithm tends to produce meshes that are not optimized. Therefore, a smoothing procedure is needed to obtain an approximately uniform spatial distribution of the vertices in a mesh. Two criteria for mesh smoothing were used: angle optimization and area optimization. Both use an iterative algorithm to generate alternative positions for each Steiner vertex in order to find the position that maximizes the minimum angle (or the minimum area) of the triangles sharing that vertex. Fig. 4 shows an example of a mesh before (top) and after (bottom) smoothing.

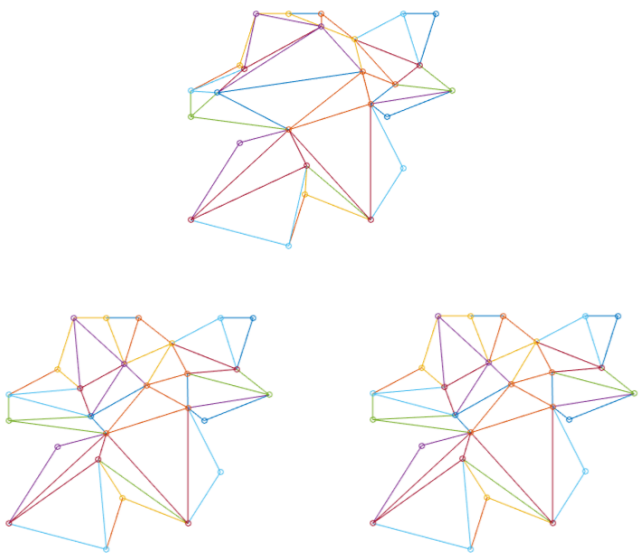
The second step consists of the interpolation between $\hat{A}$ and $\hat{B}$. The transformation of a triangle $X \in \hat{A}$ into a triangle $Y \in \hat{B}$ is a matrix M such that:

$$Y_i = M \cdot X_i + L, \quad i \in \{1,2,3\} \qquad (1)$$

where M is an affine matrix and L is a vector denoting the translation of the centroid of A towards the centroid of B. Since the motivation of this work is modeling the extent of spatial phenomena during time, we assume that the translation

component can be easily interpolated and so, the focus is on the evaluation of the affine matrix that minimizes deformation.

Fig. 4. Polygon triangulation before optimization (top), after minimum angle optimization (bottom-left) and after minimum area optimization (bottom-right).



The simplest solution to evaluate the affine matrix is to use linear interpolation. However, this is not a good solution because when triangles rotate, they tend to shrink until the middle of the interpolation and then they expand again, causing large deformation. An alternative solution proposed in [26] is a factorization of the affine matrix using Single Value Decomposition (SVD). Several types of decomposition derived from SVD have been proposed in literature [27]. In this work, the decomposition proposed in [25] was selected, since it ensures symmetry, i.e., the interpolation from source to target is equal to the interpolation from target to source. The affine matrix is a factorization $M = R_\gamma \cdot S$, such that, $R_\gamma = R_1 \cdot \mathrm{R}_2$ and $S = R_2^T \cdot D \cdot R_2$. The matrices R_1 ,D and R_2 are obtained by decomposition of the affine matrix M using SVD, and represent a rotation, a deformation (scaling and shear) and another rotation, respectively. So, the intermediate shape of a triangle at a time instant $t \in [t_i, t_{i+1}]$ is:

$$V_i(t) = M(t) \cdot X_i + L(t), \quad i \in \{1,2,3\} \qquad (2)$$

where $V(t)$ is the estimation of the coordinates of the vertices of triangle X at time t. However, applying this formula to each triangle in a mesh can generate different positions for shared vertices since each triangle has its own transformation formula (Fig. 5).

Two methods were implemented to deal with this problem. The simplest method was to consider that the position of a shared vertex is the centroid of the positions estimated in each transformation. The second method uses normal equations as proposed in [25] and the coordinates of the vertices of all triangles in a mesh are:

$$\hat{V}(t) = \left(\hat{P}^T \cdot \hat{P}\right)^{-1} \cdot \hat{P}^T \cdot M_\gamma(t) \qquad (3)$$

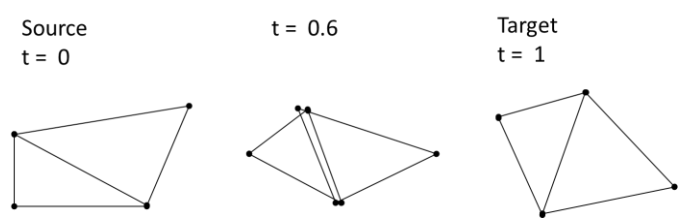
such that,

$$M_\gamma(t) = [M(t)_1^T \cdots M(t)_k^T]^T \qquad (4)$$

$$\hat{P} = \left[\hat{P}_1^T \cdots \hat{P}_m^T\right]^T \qquad (5)$$

where k stands for the number of triangles in the mesh and $\hat{P}_j$ is a $2 \times m$ sparse matrix (m is the number of vertices of the source polygon) with all elements equal to zero, except the values in the columns corresponding to the indices of the vertices of triangle j. Each P_j denotes the coordinates of the vertices of triangle j. The results using the centroid of the coordinates were equal or worse than the results obtained using normal equations, and so, hereafter we only consider the latter.

Fig. 5. Shared vertices with non-coincident positions during the interpolation of two adjacent triangles.



Using the equations above, the rotation angle is between $[-180,180]$ degrees. However, this may not be the best overall angle to ensure consistency, since the rotation of adjacent triangles must have the same orientation. So, a post-processing step is needed to deal with rotational consistency [25]. First, the rotation matrix R_γ is analyzed to check for adjacent triangles with different orientation. A discontinuity exists when the difference between the rotation angles of adjacent triangles is greater than 180 degrees. In such cases, multiples of 360 degrees are added or subtracted to make the absolute value of the difference between the rotation angles of adjacent triangles less or equal to 180 degrees. This creates consistent rotations but does not guarantee that the rotation of all triangles is minimal. This problem was mitigated by computing the average of all rotation angles, and adding or subtracting multiples of 360 degrees to make this average close to zero.

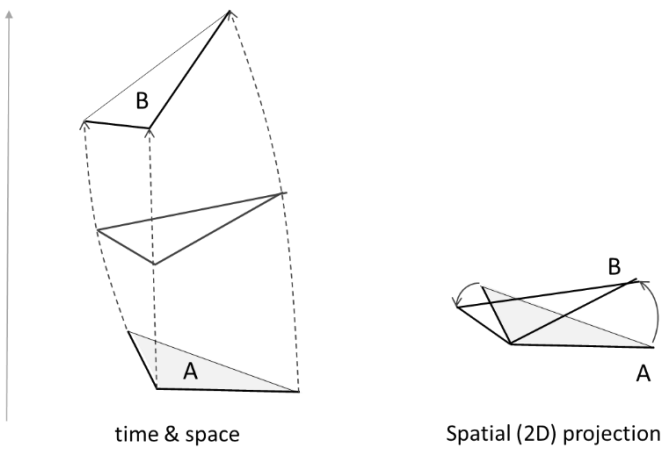
It is important to emphasize that the faces produced by the movement of the edges of a shape in space-time are non-planar, as depicted in Fig. 6.

III. Experimental Study

This section presents the results of an empirical study to evaluate and compare the methods presented in section II. The main assumptions considered during the design of the experimental study were: (1) the spatiotemporal data representations should minimize deformation and create a realistic transformation of the objects between observations; and (2) the spatial projection of a spatiotemporal value at any time instant between observations must be a valid shape (edge intersections are not allowed). The former is referred to as

fidelity of spatiotemporal data representations. This study uses synthetic and real data. The former are used to put in evidence qualitative features that could be difficult to find in real data.

Fig. 6. The rotation of a shape yields non-planar faces.



A. Experimental data and algorithms

Synthetic data are composed of twelve user-defined 2D planar shapes, including convex polygons such as triangles or pentagons and concave polygons like stars and other shapes resembling letters ('U' and 'T'), a table, a chair and a dog, with at most 20 vertices.

Real data were obtained from two sequences of satellite images tracking the location and the shape of icebergs over time:

- *Ross:* a sequence of thirty images tracking Iceberg *B-15* during 2000-2001. The time intervals between consecutive observations are irregular and vary from 1 to 29 days. The movement of the iceberg is predominantly translational but there are some parts where rotation is also significant.
- *B-15:* a sequence of ten images of two large blocks of iceberg *B-15* captured between November 19, 2004 and December 20, 2004. The predominant movement of the largest block (*B-15a*) is translation, while for the smaller one (*B-15j*) is rotation.

The original images, from which were extracted the data used in this study, are available in [33]. The procedures used to extract the icebergs' shape and to create the vertex correspondences between shapes is described in [29]. Some images have been discarded due to the presence of clouds.

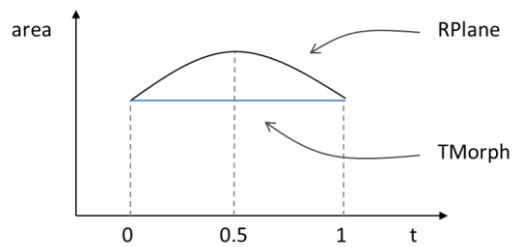
Two morphing algorithms are compared:

- The first implements the rotating plane *(RPlane)* algorithm, as proposed in [23]. The algorithm proposed in [22] was excluded because of the matching problem between concavities described in section II.A.
- The second implements the algorithm based on compatible triangulations (*TMorph*) presented in Section II.B, which uses normal equations to compute the position of shared vertices.

B. Evaluation of qualitative features using synthetic data

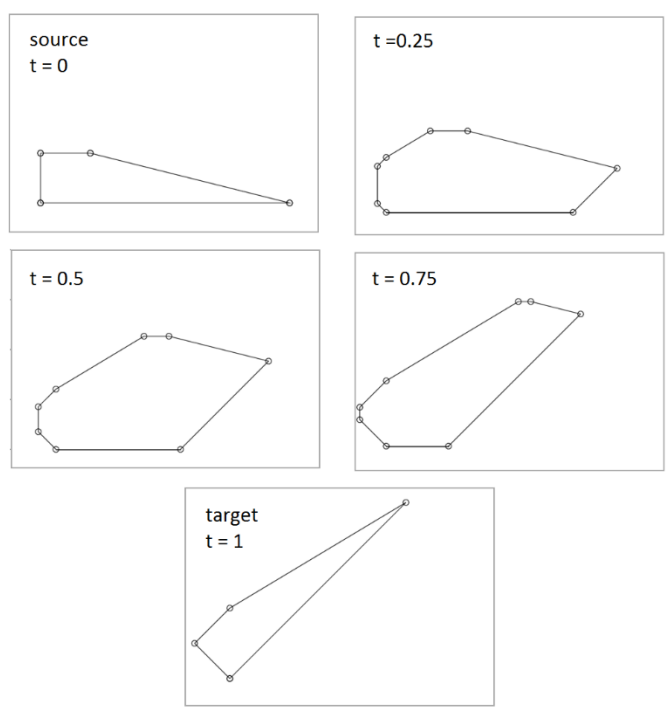
The design choice underlying the approaches presented in section II is quite different. While the approach presented in II.A represents translation and rotation implicitly, these spatial transformations are represented explicitly in II.B. Both design choices have no impact on the representation of the objects' movement (translation). However, the implicit representation of rotation through linear movements of the shapes' edges used in *RPlane* algorithm may cause important deformations. Fig. 7 displays the variation of the area of a rigid object during a rotation between consecutive observations.

Fig. 7. Variation of the area during the rotation of a rígid object between consecutive observations.



In the case of *RPlane*, the area of the shape increases until the middle and decreases during the second half of the interpolation. The horizontal line represents the object's area during an interpolation created using the *TMorph* algorithm. The area does not change, as expected for a rigid object. This is the typical behavior of *RPlane* algorithm with rotating objects, since they tend to get a squared shape in the middle of interpolation as illustrated in Fig. 8.

Fig. 8. Deformation of a rigid object during a rotation using the rotating plane algorithm.



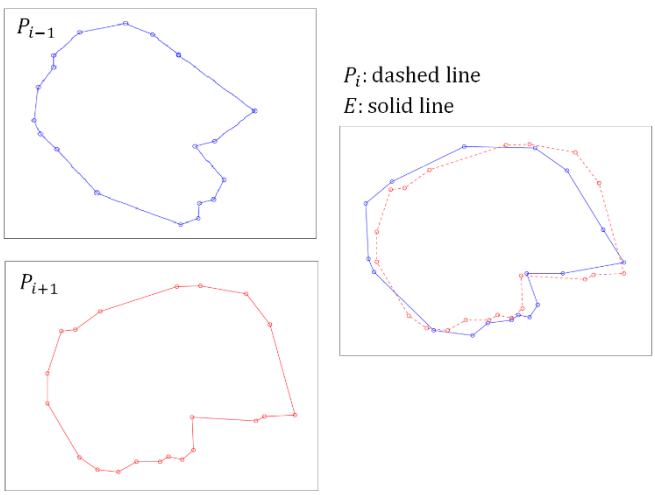
C. *Experiments with real data*

To evaluate whether the spatiotemporal data representations are realistic, we use the following similarity measure:

$$Similarity = 1 - \frac{A(P_i \cup E) - A(P_i \cap E)}{(P_i \cup E)} \qquad (6)$$

where $A(x)$ is a function that returns the area of a polygon x; P_i is one a polygon in a sequence of observations that represents the shape of the object (iceberg) captured at time t_i; and E is the object's shape at t_i, estimated using one of the algorithms investigated in this work from observations $D_{i-1} = (t_{i-1}, P_{i-1})$ and $D_{i+1} = (t_{i+1}, P_{i+1})$. Since we are interested in investigating the deformation of spatial objects with extent, the shapes P_i and E are aligned using a translation so that the centroids coincide, followed by a rotation that is calculated using a function that minimizes the sum of squared errors considering the distance between the corresponding vertices in both polygons. This procedure is illustrated in Fig. 9.

Fig. 9. Similarity between a captured P_i and an estimated E shape.



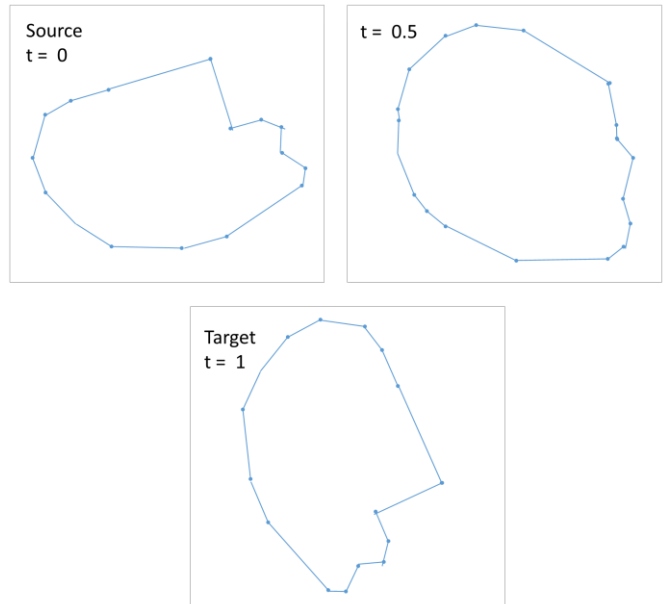
The results are summarized in TABLE I. The similarity values are the averages by algorithm and dataset.

TABLE I. SIMILARITY VALUES

Algorithm	Dataset		
	Ross	***B-15a***	***B-15j***
RPlane	0.86	0,92	0,83
TMorph	0,89	0,90	0,89

The best results were obtained for *B-15a*. This is a large iceberg, which has no important concavities and the rotation between snapshots is nearly inexistent. The worst results occurred for *B-15j*. This is an iceberg with a large concavity and significant rotation between snapshots. In this case, the results obtained using *TMorph* are close to the results obtained for *B-15a*, but the similarity values using *RPlane* show that there is significant deformation during interpolation. This deformation is mainly caused by difficulties in handling rotation and by the method used to deal with concavities. The data used in this study did not allow to investigate the relative importance of each of these factors in deformation, but Fig. 10 shows an illustrative example.

Fig. 10. Estimation of the shpe of *B-15j* using the *rotating plane* algorithm.



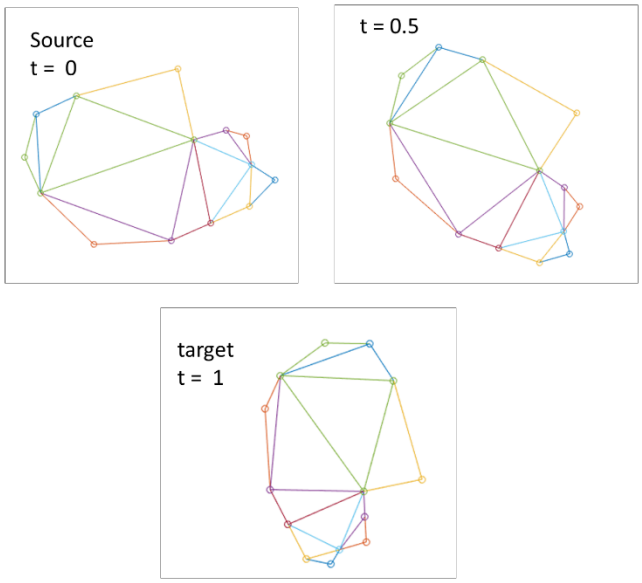
The shape estimated in the middle of the transformation ($t = 0.5$) has a larger size than the source and target shapes, and the concavity tends to vanish. The former is due to the rotation of the shape and the latter comes from the mapping of all vertices in a concavity to a single vertex in the convex-hull of the other shape. A similar transformation created using *TMorph* is illustrated in Fig. 11.

As depicted in previous figures, *TMorph* can handle rotation and concavities better than *RPlane*, and is able to create transformations that are visually more natural. This is also in accordance with the numerical results just presented.

The results obtained for *Ross* are in between *B-15a* and *B-15j*, since there are periods where the rotation of the iceberg between observations is considerable and almost inexistent in other periods.

In absolute terms, the margin of improvement for *TMorph* is between 10 and 11%, and for *RPlane* is between 8 and 17%. There are several factors contributing to these errors, such as, noise in source images, approximation errors due to the segmentation of source images or bad vertex correspondences, besides the interpolation algorithms studied here. For instance, in the case of icebergs, there are numerous ice fragments around the icebergs and in certain images it is difficult to distinguish the boundaries of the main block unambiguously. In addition, a large fragment of the *B-15a* collapsed during the sampling period, and so, the similarity value obtained for the time interval between the observations immediately before and after the collapsing was lower than in the other cases.

Fig. 11. Estimation of the shpe of B15j using compatible triangulations.



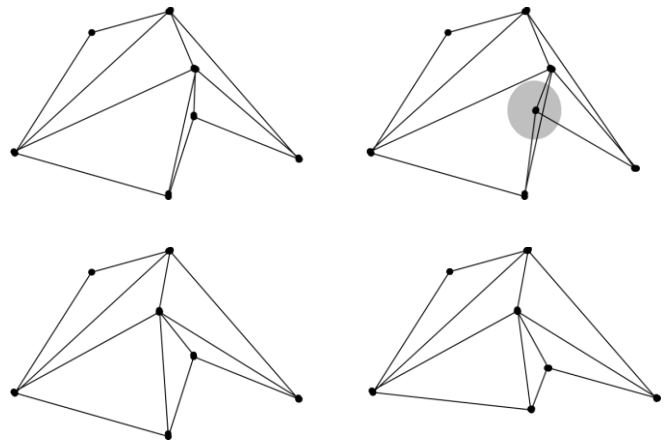
D. *Topological and geomtric features.*

Several constraints must hold on the representation of spatiotemporal data. Specifically, the segments defining the geometry of a shape must not intersect during transformations, because this is a topological anomaly in view of spatial data systems currently in use. Only intersections between the segments' end points of are allowed.

The spatiotemporal representations created using the algorithms studied in this work were topologically valid at all times, even for *B-15j*, which is the iceberg with the most complex geometry. For synthetic data, there also were no segment intersections in the spatiotemporal data representations created using *RPlane*. This means that mapping all edges in a concavity into a single point in the convex-hull of another shape was enough to create topologically safe spatial transformations at all times. However, this strategy may cause important deformations.

The interpolations created using *TMorph* are not completely safe, and topological anomalies may arise in the interpolation of very dissimilar shapes. The mesh structure is important because topological anomalies occurred more often when there are thin triangles, highlighting the importance of the post-processing step to improve the mesh of source and target shapes. The incidence of topological anomalies also tends to decrease with the number of vertices, i.e., the results indicate that it is possible to increase the robustness of the algorithm adding Steiner vertices into the triangulated shapes. Note that, the occurrence of these topological anomalies depends on several factors. For instance, it is difficult to define a vertex correspondence between highly dissimilar shapes (e.g., a table and a car), and so, the results may be ambiguous. As it is difficult to create a coherent suite of test cases, we just present an example of a topological anomaly generated using a mesh before and after refinement (Fig. 12), rather than numerical results.

Fig. 12. Source polygon with a thin triangle (top-left) and topological anomaly (edges intersection) during a transformation (top-right) ; the same transformation after mesh refinement (bottom).



From a geometrical point of view, the projection of a spatiotemporal value at any time instant returns a planar shape using any of the algorithms studied in this work. However, while the projection of a spatiotemporal value created using the *RPlane* algorithm during a time interval between two consecutive observations returns always a planar face (Fig. 1), *TMorph* may produce non-planar faces (Fig. 6). Although the latter allows creating spatiotemporal representations that are more realistic, the algorithms to deal with this kind of geometries are more complex.

IV. Discussion

The results presented in previous section provide interesting insights on the development of solutions to represent spatiotemporal phenomena in information systems. Two main approaches are presented. The first, implements the rotating plane algorithm [22,23]. This algorithm is able to create spatial transformations that are safe at all times but is poor in handling rotations. There is no previous matching between shapes in a sequence of observations and the correspondence between the first pair of edges is based on simple rules, such as, selecting the edges with lowest angles relatively to a given coordinate axis. Thus, to ensure that the results are realistic it is necessary that the time interval between snapshots is sufficiently small so that rotation between them is also small, which might not be possible in some use cases.

The second approach used in this work is based on [25,26,29], which enables using a vertex correspondence between a source and a target shape defined beforehand. The interpolation is split into components, and so, rotation is represented explicitly in the model, allowing to create spatiotemporal representations that are more realistic than with the previous approach. In addition, the use of triangulation is an interesting approach, because there are many algorithms in computational geometry based on triangulation and the implementation of complex spatiotemporal operations using this kind of decomposition can be simplified. In this work, we assumed that translation and rotation are linear and uniform, and so, the focus was on the morphology of objects with spatial

extent. However, splitting spatial transformations into components, enables the modeling of translation and rotation using special-purpose functions, which may take into account external factors such as ocean currents, or wind direction and speed.

The results of this study show that there is not best solution to represent the evolution of spatiotemporal phenomena over time. While the solution using the *RPlane* tries to ensure that data is topologically valid at all times, *TMorph* generates more realistic representations of spatiotemporal phenomena. It is also important to note that there is no formal proof that the algorithms are safe, i.e., that the spatial transformations created using an algorithm are topologically valid at all times for any pair of source and target shapes. Thus, validation is usually made by example.

The combination of both strategies is possible: first, we can use *TMorph* to create an interpolation based on the observations; second, we split the spatial transformation into smaller time intervals to create pseudo-observations that are closer to each other; and third, we can use *RPlane* and the pseudo-observations to create the interpolations. The first step would allow creating a more realistic representation of the real-world phenomena and the last step would replace a representation producing non-planar faces by another algorithm producing planar faces only. As pseudo-observations are closer to each other, large rotations between observations were broken into several smaller rotations, thus reducing deformation caused by the *RPlane* algorithm.

V. Conclusion

This paper focuses on the representation of the evolution of spatial phenomena over time, using continuous models of space and time. Although the case study is about monitoring icebergs in the Antarctic, the aim is to investigate generic solutions that may be used in several application domains, e.g., earth sciences, medicine or biology.

Tackling the problem of creating spatiotemporal data from a sequence of observations taken at discrete times, this paper investigates two interpolation algorithms. The first uses a rotating plane algorithm to represent spatial transformations between consecutive observations, as proposed in [23]. Although this is the main approach to represent spatiotemporal phenomena in databases, the experiments performed using real data put into evidence issues on dealing with rotation and concavities. The other algorithm splits the representation of spatial transformations between observations into components [25], namely, translation, rotation and deformation, and uses compatible triangulations of source and target polygons [29]. This algorithm is better when dealing with rotation and concavities, producing a more realistic morphing, but the interpolations are not safe since topological anomalies may occur during morphing. These anomalies occurred only with experiments using synthetic data, where source and target polygons were highly dissimilar. It is also important to note that only the first algorithm is compatible with current data models proposed in spatiotemporal databases.

Despite being an important tool in several application domains, there still are important open issues regarding spatiotemporal data processing. The support for developing applications is still limited, and it is often necessary to make great efforts on the development of special-purpose algorithms to implement complex spatiotemporal operations. In the future, it would be interesting to have generic tools, where approximation errors are bounded, and high-level query languages to make the development of automated processes for storage, management, interpretation and analysis of spatiotemporal data easier. The use of morphing techniques sounds a promising approach to achieve these goals. A good solution should create realistic transformations that are topologically valid at all times, using compact representations and simple procedures to enable retrieving large amounts of spatiotemporal data efficiently. Morphing techniques based on computationally intensive algorithms, e.g., iterative algorithms, are not a good choice.

The use of real and synthetic data was important, since both have enabled finding different issues on the representation of spatiotemporal data. The creation of benchmarks to enable a comprehensive evaluation of the essential properties of spatiotemporal data management systems, is also an interesting research topic in this area.

Acknowledgment

This work is partially funded by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UID/CEC/00127/2013.

References

[1] N. Mamoulis, Spatial data management, Morgan & Claypool Publishers, Synthesis Lectures on Data Management, 2012.

[2] P. Rigaux, M. Scholl and A. Voisard, Spatial databases with applications. The Morgan Kaufmann Series in Data Management Systems, 2002.

[3] M. Koubarakis et al., Spatiotemporal databases: The ChoroChronos Approach, Springer–Verlag, Lecture Notes in Computer Science, 2003.

[4] M. Bügelmayer, D. M. Roche1 and H. Renssen, Representing icebergs in the iLOVECLIM model (version 1.0) – a sensitivity study, Geoscientific Model Development, vol. 8, pp 2139–2151, July 2015.

[5] Z. Zhao, S.-L. Shaw and D. Wand A Space-Time raster GIS data model for spatiotemporal analysis of vegetation responses to a freeze event, Transactions in. GIS, vol. 19, pp 151–168, February 2015.

[6] E. Meijering, O. Dzyubachyk and I. Smal, Methods for cell and particle tracking, Methods Enzymol vol. 504, 183–200, Elsevier, 2012.

[7] X. Silvani, F. Morandini and J.L.Dupuy, Effects of slope on fire spread observed through video images and multiple-point thermal measurements, Experimental Thermal and Fluid Science, vol. 41, 99–111, September 2012.

[8] D. Casbeer, D. Kingston, R. Beard and T. McLain, Cooperative forest fire surveillance using a team of small unmanned air vehicles, International Journal of Systems Science, vol. 37, pp. 351–360, June 2006.

[9] L. Merino et al., An unmanned aircraft system for automatic forest fire monitoring and measurement. Journal of Intelligent, Robotics Systems, vol. 65, pp. :533–548, January 2012.

[10] P. Naden, Siltation in Rivers: A Review of Monitoring Techniques. Conserving Natura 2000 Rivers Conservation Techniques, Series no. 6, March 2015.

[11] T. Nelson, J. Long, K. Laberee and B. Stewart, A time geographic approach for delineating areas of sustained wildlife use, Annals of GIS, vol. 21, pp 81–90, May 2015.

[12] T. Nelson and J. Long, Home range and habitat analysis using dynamic time geography, Journal of Wildlife Management, vol 79, pp. 481–490, April 2015.

[13] S. Grumbach, P. Rigaux and Luc Segoufin, Manipulating interpolated data is easier than you thought, in Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00), Cairo, Egypt, pp 156–165, 2000.

[14] R. Güting, M. Böhlen, M. Erwig, C. S. Jensen, N. Lorentzos, M. Schneider, and M. Vazirgiannis, A foundation for representing and querying moving objects, ACM Transactions on Database Systems, vol. 25, pp 1–42, March 2000.

[15] J. A. Lema, L. Forlizzi, R. Güting, E. Nardelli, M. Schneider. Algorithms for Moving Objects Databases, Computer Journal, vol. 46, pp 680–712, June 2003.

[16] X. Meng and J. Chen, Moving objects management: models, techniques and applications, Springer–Verlag, 2010.

[17] L. Matos, J. Moreira, A. Carvalho. A spatiotemporal extension for dealing with moving objects with extent in Oracle 11*g*, ACM SIGAPP Applied Computing Review, vol. 12, pp 7–17, June 2012.

[18] L. Zhao, P. Jin, X. Zhang, L. Zhang and H. Wang. STOC: extending oracle to support spatiotemporal data management, In Proceedings of the 13th Asia-Pacific web conference on Web technologies and applications, pp 393–397, 2011.

[19] N. Pelekis, Y. Theodoridis, S. Vosinakis and T. Panayiotopoulos, Hermes: A Framework for Location-Based Data Management, in Proceedings of the 10th International Conference on Advances in Database Technology, Saint-Petersburg, Russian Federation, pp 1130-1134, 2006.

[20] S. Gebberta and E- Pebesma, A temporal GIS for field based environmental modelling, Environmental Modelling & Software, vol. 53, pp 1–12, March 2014.

[21] W. Siabato, C. Claramunt, M. Manso-Callejo and M. Bernabé-Poveda, TimeBliography: a dynamic and online bibliography on temporal GIS, Transactions in GIS, vol. 18, pp 799–816, March 2014.

[22] E. Tøssebro, R. Güting. Creating Representations for Continuously Moving Regions from Observations, in Proceedings of 7th Symposium on Advances in Spatial and Temporal Databases, Redondo Beach, CA, USA, 321–344, 2001.

[23] M. McKenney and R. Frye, Creating moving regions from snapshots of complex regions, ACM Transactions on Spatial Algorithms and Systems, vol 1, pp 4.1–4.30, July 2015.

[24] H. Liu and M. Schneider, Tracking continuous topological changes of complex moving regions, in Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11), Taichung, Taiwan , pp 833–838, 2011.

[25] W. Baxter, P. Barla and K-I. Anjyo, Rigid shape interpolation using normal equations, in Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering (NPAR '08), Annecy, France, pp 59–64., 2008

[26] M. Alexa, D. Cohen-Or and D. Levin, As-rigid.as possible shape interpolation, in Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'00, New Orleans, Louisiana, USA, pp 157–164, 2000.

[27] D. Xu, H. Zhang, Q. Wang and H. Bao. Poisson shape interpolation, in proceedings of the ACM Symposium on Solid and Physical Modelling, Boston, MA, USA, pp 267–274, 2005.

[28] R. Summer, M. Zwicker, C. Gostman and J. Popović, Mesh-based inverse kinematics, in Proceedings of the 32nd International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'05, Los Angeles, CS, USA, pp 488–495, 2005.

[29] V. Surazhsky and C. Gotsman, High quality compatible triangulations, Engineering. with Computers, vol. 20, pp 147–156, July 2004.

[30] J. Moreira, P. Dias and L. Paulo, Creating moving objects representations for spatiotemporal databases, Encyclopedia of Information Science and Technology, Third Edition, pp 1703–1709, 2015.

[31] O. Kaick, H. Zhang, G. Hamarneh and D. Cohen-Or, A Survey on Shape Correspondence, Computer Graphics Forum vol. 30, pp 1681–1707, September 2011.

[32] L. Liu, G. Wang, B. Zhang, B. Guo and H.-Y. Shum, Perceptually based approach for planar shape morphing, in Proceedings. 12th Pacific Conference on Computer Graphics and Applications, Seoul, South Korea, pp. 111–120, 2004.

[33] Rutherford Appleton Laboratory, The ATSR project [online], available URL: http://www.atsr.rl.ac.uk/images/sample/ross/index.shtml, accessed May 2016.

[34] Eosdis earthdata, Ross sea subset, terra 1km true color, [online], available https://lance.modaps.eosdis.nasa.gov/imagery/subsets/?project=antarctica&subset=RossSea.2004324.terra.1km, accessed May 2016.