A partition approach to interpolate polygon sets for animation

Alexandre Ribeiro Cajazeira Ramos, Emanuele Santos, Joaquim Bento Cavalcante Neto
Federal University of Ceara
Fortaleza - CE, Brazil
Email: alexandrecajazeira@alu.ufc.br, {emanuele, joaquimb}@dc.ufc.br

***Abstract*—The use of animation can be a good alternative to static visualizations when communicating dynamic changes. Some approaches already represent spatiotemporal phenomena using a polygon set for each time instant. However, these representations are static and not general enough to be applied for the interpolation of arbitrary polygons. Furthermore, the problem of interpolating arbitrary polygons present a set of requirements that are not satisfied by the currently available tools. For example, the polygons are arbitrary, and the interpolation should be smooth and fully automatic (not requiring user intervention). To solve this problem, we present an approach to interpolate arbitrary polygon sets that satisfy those requirements, and that can be used to visualize temporal changes of different phenomena as an animation. In the proposed approach, we appropriately divide and identify correspondences between origin and target polygon sets. Our approach is general enough so that different polygon division techniques can be used. We also performed a series of experiments comparing a few different techniques and discuss the results.**

I. Introduction

Geospatial phenomena, such as territorial disputes, agricultural exploitation, urban occupation, climatic events, endemic regions, concentration of crimes, can be represented and analyzed through geographical maps constructed by different techniques [1]–[3].

The visualization of these phenomena contributes to the understanding of their characteristics and behavior over time. In criminal analysis, for example, police agencies, analysts, and public security managers are interested in understanding how high-rate crime regions change over time [4], [5]. Currently, using the available tools, density maps representing different periods are shown side by side [6] or superposed with different colors [7]. If the number of time steps is large, these visualization techniques might not be appropriate because of the amount of screen space occupied by the images. On the other hand, if the images are too small to fit all the time steps, fewer details are visible, or if superposition is used, too many colors are not distinguishable enough. Animations, although controversial in the field of visualization [8], can be a powerful storytelling medium and are sometimes a good alternative to static maps when communicating dynamic changes [9].

Moreover, there exist situations in which the acquisition of data and the mapping of these phenomena is a complex and high-cost process [10], [11], being carried out at distant intervals of time and hampering a consistent temporal analysis. Considering this situation, Kim and Cova [10] present an approach for visualizing the temporal evolution of forest fires through interpolation, in a semi-automatic process with constraints to optimize the representation of the real changes in the studied phenomenon.

Other approaches also represent spatiotemporal phenomena using a polygon set for each time instant [2], [7], [11]. However, the representations are static and not general enough to be applied for the interpolation of arbitrary polygons.

Another constraint of the problem is that temporal changes of many phenomena are not linear. At each time instant, there can be a different number of regions that need to be displayed [2], [3]. Therefore, there may occur the division of a given region into other small regions as well as the merging of regions into a single one. Regions can also appear or disappear along the time. In the land-use context, for example, an area eventually can be divided by new roads or negotiations between owners [11].

In summary, we have identified the following requirements for the 2D polygon interpolation problem:

- R1 The polygons can be arbitrary (convex or concave) and have any number of holes.
- R2 The polygons may have a different number of vertices between time steps.
- R3 In each time step there can be a different number of polygons (many-to-many interpolation).
- R4 The interpolation should be smooth enough as to be displayed as an animation and the polygons should not intersect each other.
- R5 The process should not require user intervention (fully automatic).

Currently, there are several approaches for interpolating 2D polygon pairs [12]–[15], and for the static temporal analysis of geospatial phenomena [2], [3], [10], [16], [17]. However, to the best of our knowledge, none of the currently available methods or tools are capable of performing automatic and smooth transitions between 2D polygon sets satisfying the list of requirements above. Veltman [12], in particular, proposed a javascript library named Flubber that allows one-to-many and many-to-one interpolations between sets of 2D polygons, using a triangulation-based method for polygon division. This library does not fully support many-to-many interpolations, only a list of one-to-one interpolations, requiring users to specify each interpolation pair explicitly. Flubber ensures a good interpolation between polygon pairs, but its simple spatial

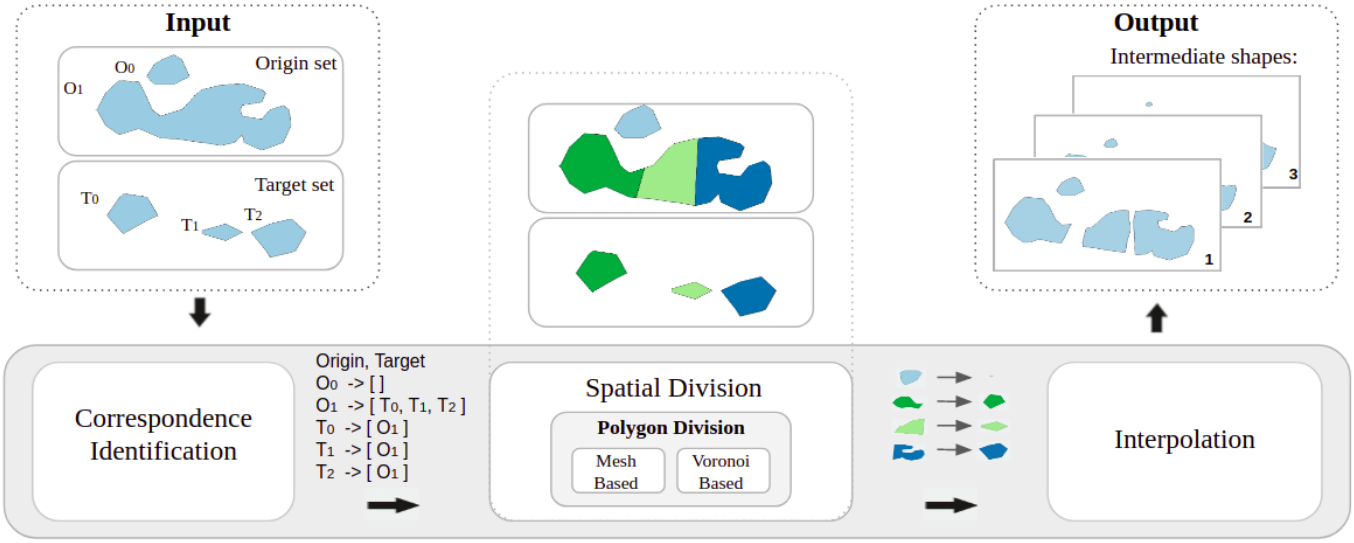

Fig. 1. The proposed approach workflow. Given two sets of polygons (Origin and Target), first the correspondences between polygon sets are identified, then we perform a spatial division to reduce the problem to a list of one-to-one interpolations; and, finally, we execute each interpolation pair, obtaining the intermediate shapes to be animated from the origin set to the target set.

division method may result in polygon overlaps during the transitions, jeopardizing a smooth animation.

To solve this problem, we present a partition approach to interpolate 2D polygon sets that satisfies all the requirements defined above and so can be used to visualize temporal changes of different phenomena as an animation. As shown in Figure 1, our approach is divided into three stages. First, we identify the correspondences between polygon sets. We then perform a spatial division to reduce the problem to a list of one-to-one interpolations; and, finally, we execute each interpolation pair, obtaining the intermediate shapes to be animated. We only require as input two sets of polygons (Origin and Target). Our approach is general enough that different techniques to perform polygon division can be used. We have also implemented and compared a few different techniques and will discuss the results.

The remainder of this paper is organized as follows. In Section II, we review the related work. In Section III, we describe the proposed approach. In Section IV, we present a few polygon division techniques that can be used in our approach. In Section V, we discuss our results. Finally, in Section VI, we present the conclusions of our work and the directions for future work.

II. Related Work

The problem of one-to-one interpolation is not new and is already well studied in the literature [11], [13], [14], [18]. Most of the works concentrated their efforts in 3D, considering it solved in 2D.

Some approaches [13], [14] focus on the problem of shape blending, especially for character animation, in which algorithms based on a physical model are required. Sederberg and Greenwood [13] propose an algorithm for the smooth interpolation between two polygons, in which the intermediate shapes are calculated with little user intervention. They present good results, but their approach requires the initial and final shapes to have an equivalent topology and generate undesired polygon deformations during interpolation. In Sederberg et al. [14], the definition of intermediate shapes is obtained by the direct interpolation of the intrinsic representations (angles and edge sizes) of the initial and final polygons, disregarding a linear trajectory between vertices. This approach improves the previous one [13] by smoothing the transitions and avoiding the deformation of regions during the interpolation, but present the same restrictions for the polygons (equivalent topology).

In [19]–[21], it is explored the deformation problem during interpolations and animations of non-rigid 3D shapes (character animation, for example), expressed in mesh sequences. These approaches present satisfactory results in reducing deformations, but the scope is different from our work, since polygon division or correspondences identification, which are crucial in our approach, are not addressed or do not have an essential role in those applications. Also, some of them assume that the connectivity is maintained, which is not the case in our problem.

Shiao et al. [15] present a technique for interpolating the contours of two slices of medical images based on the interpolation of long-axis and vertex parameters of polygon approximations for the slices' contours. Although this technique produces natural results more quickly than other shape-based techniques, it does not support many-to-many polygon interpolation.

In 3D shape reconstruction from a set of cross-sectional contours [22]–[24], we find correspondence problems (finding the correct associations between the contours of adjacent slices) and branching problems (when one contour in one slice can correspond to more than one contour in an adjacent slice),

both of which are related to the problem addressed by our approach. The main difference, however, is because their goal is to reconstruct the original surface, and so they must follow the physical constraints of the imaged object. Usually, they require that the interslice spacing of the data be fine enough to avoid certain types of topology. In our approach, on the other hand, there are no such constraints.

There are also approaches applied directly to geospatial visualization [10], [11]. Kim and Cova [10] present a semi-automatic process, restricted to five cases of correspondences between polygons for interpolation. The approach was applied with success to a specific case, the Southern California Grand Prix fire of 2003, but they support only a very limited set of interpolations. Carbunescu and Wart [11] present a temporal and interactive visualization tool to represent changes in land-use over time through interpolation, but does not support the division of polygons in the interpolation.

Other researches are devoted to discussing issues related to the spatiotemporal changes of polygons used in different contexts. Robertson et al. [2] present STAMP - an approach for the spatiotemporal analysis of polygons that are spatially distinct and experience discrete changes through time. Abhar et al. [3] explore the historical evolution of blowouts at Cape Cod National Seashore with the STAMP method. Mizutani [16], [25] presents an analytical framework for polygon-based land use transitions to understand the processes of changing regarding types of land uses and their shapes. Finally, Salamat and Zahzah [17] present a fuzzy method to define the spatiotemporal relations of objects. None of these works, however, present a practical tool for visualization and analysis of these temporal changes.

III. Proposed Approach

Our approach is an automatic solution for the smooth interpolation between arbitrary polygons sets, allowing the construction of visualizations to represent the temporal evolution of several geospatial phenomena that are suitable for animation.

As we mentioned before, Figure 1 illustrates the proposed approach, in which it receives two sets of polygons, called **Origin** and **Target**, and returns the intermediate shapes to construct a smooth animation. To make this approach flexible and general, we have divided it into three steps: correspondence identification between sets (Section III-A), spatial division (Section III-B) and interpolation between polygon pairs (Section III-C).

A. Correspondence identification between polygon sets

The first step in our approach defines the relationships between **Origin** and **Target** sets, a fundamental phase to identify which polygons will appear or disappear throughout the animation, as well as which polygons will be divided in the spatial division step.

This step receives the two polygon sets and outputs a list of many-to-many correspondences, in which, each polygon in each set points to its corresponding list of polygons in the other set. This double list will allow the correspondences to be quickly accessed.

There can be many different ways to define correspondences. To be general enough and so this approach can be applied in many application domains, we define the correspondences purely geometrically and as simple as possible. This means there will be a correspondence between a polygon O_i in the **Origin** set to another polygon T_j in the **Target** set (and vice versa), if and only if O_i and T_j spatially intersect each other. Notice that other more sophisticated functions can be used to define the correspondences, such as using a tolerance radius or by adding semantics based on other attributes of the polygons or the environment.

Therefore, at the end of this step, each polygon can have zero, one or many corresponding polygons. The three situations should be analyzed separately in the spatial division step.

B. Spatial division

The spatial division step is responsible for reducing the list of correspondences to a list of one-to-one polygons to be later interpolated. For that, it receives as input the **Origin** and **Target** polygon sets and their respective correspondence relations and outputs **SPL** - a shape pairs list.

Algorithm 1 performs the spatial division. Its input parameters are **Origin** and **Target** (the input polygon sets), **CO** and **CT** (the correspondence lists for each polygon in **Origin** and **Target**, respectively).

As mentioned earlier, each polygon's correspondence list may be empty or have one or more polygons from the other set. When the list is empty, two different cases may occur: if the polygon is in **Origin**, it will disappear during the interpolation (lines 3 to 5); otherwise, if the polygon is in **Target**, it will appear during the interpolation (lines 11 to 13). In both cases, the centroid of the polygon will be used as the correspondence shape (lines 4 and 12), that is, the polygon will collapse to its centroid or the centroid will enlarge to become the polygon.

When the lists have more than one element, we divide the polygon into a number of regions equivalent to the number of polygons in the correspondence list (lines 7 and 15). This process is done in two passes. The first pass defines the number of divisions in each polygon. The second pass (lines 17 to 19) defines the one-to-one correspondences between Origin's and Target's divisions.

For performing the polygon divisions, different techniques can be used (see Section IV). Independently of the technique used, the *Divide* function receives a polygon **P** and a list of corresponding polygons **CP**, resulting in the division of **P** in the same number of sub-regions such that each of them is paired to one element in **CP**. These pairs will be then added to SPL. When the list has only one element, the *Divide* function will do nothing, and the polygons will be added to SPL as they are.

C. Interpolation

The third step of our approach receives a list of shape pairs - **SPL** and performs the interpolation of each pair separately, generating the intermediate shapes for the animation.

Algorithm 1 Spatial division algorithm

Input: Origin, Target, CO and CT
Output: SPL

```
SPL ← ∅
for each O_i ∈ Origin do
    if CO_i = ∅ then
        c ← Centroid(O_i)
        SPL.append([O_i, c])
    else
        R_i ← Divide(O_i, CO_i)
    end if
end for
for each T_j ∈ Target do
    if CT_j = ∅ then
        c ← Centroid(T_j)
        SPL.append([c, T_j])
    else
        R_j ← Divide(T_j, CT_j)
    end if
end for
for each T_j ∈ Target do
    for each O_i in CT_j do
        SPL.append([R_ij, R_ji])
    end for
end for
return SPL
```

The interpolation of polygon pairs is based on the approaches proposed in [11]–[13] and is composed of the following steps:

1) Make sure that both polygons have an equivalent topology in the number of vertices by adding vertices to the polygon with less vertices.
2) Identify the correspondences based on the minimization of the trajectory cost, as proposed in [11].
3) Generate intermediate shapes through the interpolation of the corresponding vertices, using the parametric equation of the line.

IV. Polygon Division

One of the goals of this research was to check whether the technique used for polygon division impacted the results of the interpolation. In particular, we would like to make sure that requirement R4 was satisfied.

Thus, we implemented two types of approaches: a mesh-based technique (Section IV-A), in which any meshing technique can be used; and a technique based on Voronoi decomposition (Section IV-B). In this section we explain how the techniques are used in our approach. We discuss their results in Section V.

A. Mesh-based techniques

Polygon division is a classical problem of geometry processing that can be associated with the generation of triangular meshes [26]–[30]. Independently of the technique used, given a polygon **P** and a list of corresponding polygons **CP**, the following steps are executed (see Figure 2):

1) Generate a mesh for polygon **P** (Figure 2C).
2) Identify which elements of the mesh correspond to each polygon in the list of correspondences **CP** (Figure 2B).
3) Merge the elements with the same correspondence into a single sub-region (Figure 2D).

For the element classification in step 2, we adopt the following criteria: we associate directly the elements intersecting each corresponding polygon to that polygon, and associate the elements not intersecting any polygon to the nearest one. Figure 3 demonstrates how the classification was done for the mesh in Figure 2C.

For achieving good visual results in the interpolation, it is desirable that the final polygons are smooth. We implemented different mesh-generation techniques to verify if the quality of the underlying mesh impacted the smoothness of the resulting polygons. We decided to use a good advancing-front method [31], and we developed a method based on a quadtree and templates [32]–[34]. Figure 4 illustrates the meshes generated and a quality comparison considering these techniques.

The motivation to develop a quadtree and templates based method was because it generated a more regular mesh and the boundaries of the divided regions could be smoother than the ones of the advancing-front mesh. This method consists of building a quadtree, interactively classifying the cells as **In** -

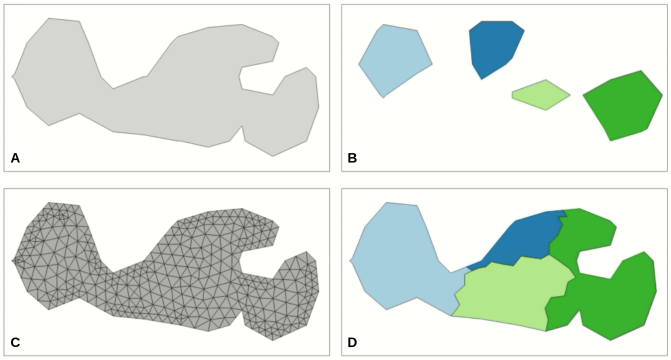

Fig. 2. Using mesh-based techniques for polygon division. (A) the input polygon, (B) the corresponding polygons, (C) the generated mesh and (D) the result after merging the cells with the same correspondence (identified by the same color): the input polygon is finally divided into the same number of regions as the number of polygons in the list of correspondences.

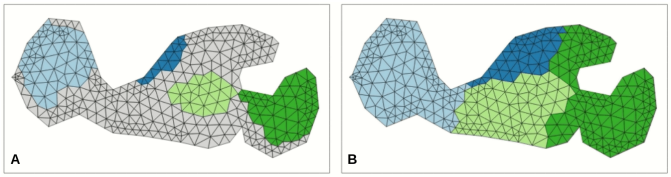

Fig. 3. Triangle classification into corresponding polygons.(A) The triangles that intersect each corresponding polygon (shown in color) are classified directly. Triangles not intersecting any polygon are shown in gray. (B) The unclassified triangles are then classified according to the nearest corresponding polygon.

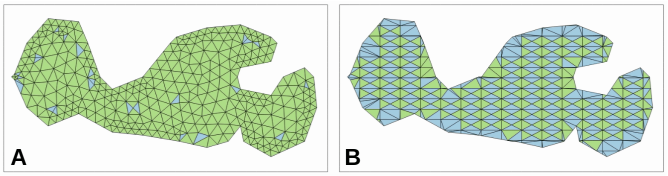

Fig. 4. Mesh generation and quality comparison. The metric used was the aspect ratio, the radius ratio of the circumscribed circle to the inscribed circle of the mesh elements. (A) The mesh generated by the advancing-front technique [31] and (B) the mesh generated by the quadtree- template method. The metric ranges from 0 to 1, where 1 corresponds to the equilateral triangles. The high-quality triangles (quality ≥ 0.8) are shown in green, while the low-quality triangles (quality < 0.8) are shown in blue. In (A), 96% are high quality triangles, while in (B) only 41% are high quality.

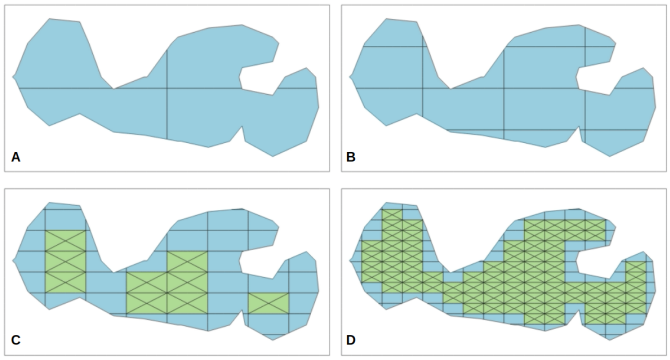

Fig. 5. The use of a quadtree and templates for mesh generation. (A) to (D) show levels one, two, three and four of the quadtree, respectively. The **In** cells, with template-based mesh are shown in green, and the clipped **On** cells based on the intersected polygon are shown in blue.

Internal to the polygon, **On** - Intersecting the border, and **Out** - External to the polygon. Provided a quadtree with a specific depth, the following steps are executed:

1) Obtain the area of interest for mesh generation: ignore the area outside the polygon by discarding **Out** cells and clipping **On** cells (Figure 5).
2) Generate the mesh: use templates for **In** cells, and Delaunay triangulation for the clipped **On** cells (Figure 4B).

The template used in this process, a rectangle divided into 4 triangles, is shown in Figure 5C and Figure 5D.

B. Voronoi-based technique

Although the use of meshes can facilitate some of the required computation, they do not produce the smoothest partition of the input polygon (depending on the size of the elements in the mesh, the borders of the sub-regions can be jagged). An alternative approach is to use a Voronoi diagram [35] to divide the input polygon. This approach consists of the following steps (see Figure 6):

1) Build the Voronoi diagram using, as seeds, one vertex from each corresponding polygon.
2) Identify the intersecting area between the polygon to be divided and each cell of the Voronoi diagram built in the previous step.

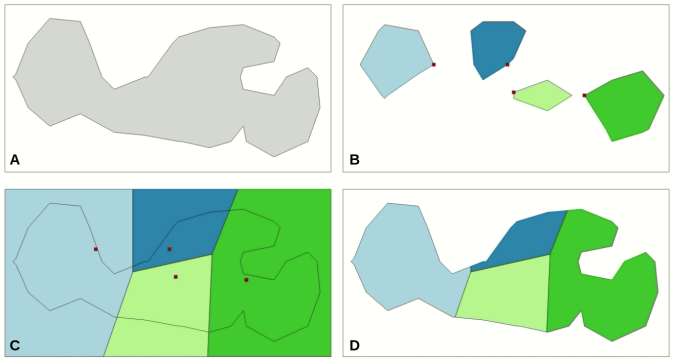

Fig. 6. Division based on a Voronoi diagram. (A) The input polygon, (B) the corresponding polygons with the vertices used to generate the Voronoi diagram indicated in red, (C) the Voronoi Diagram, and (D) the result of dividing the input polygon according to the Voronoi diagram.

The diagram generated in step 1 will have the same number of cells as the number of corresponding polygons (see Figure 6C). At first we used the centroid of each corresponding polygon as seeds but this generated overlaps in a few cases. To avoid these overlaps, we chose as seed the position of the vertex closest to the centroid of the polygon to be divided.

V. Results

To verify that our approach satisfies all the requirements defined in the Introduction, we implemented it in JavaScript and performed a series of experiments. Their results are discussed in the following subsections.

A. Experiment 1: Interpolation between arbitrary polygon pairs

The main goal of this simple experiment is to show that our approach satisfies requirements R1 and R2. Furthermore, one-to-one interpolation is the basic type of interpolation used in our approach: all the other operations culminate in a series of one-to-one interpolations. Figure 7 illustrates three examples of interpolating the same polygon to other three different polygons with arbitrary geometry.

B. Experiment 2: Comparison of polygon division techniques

As we briefly mentioned in Section IV, one of our concerns about satisfying requirement R4 was with the technique used for dividing the polygons. We used two mesh-based and a Voronoi-based techniques described in Sections IV-A and IV-B.

In this experiment, we performed a comparative analysis of these approaches. Figure 8 illustrates applying the three techniques for interpolating the same polygon. By looking at the zoomed-in regions in the last row of Figure 8, we can notice a jagged pattern on the boundaries of the polygons divided by the mesh-based techniques (the advancing-front technique being slightly better) while the Voronoi's boundaries are much smoother.

When comparing the quality of the generated meshes of the two mesh-based techniques in the example of Figure 8,

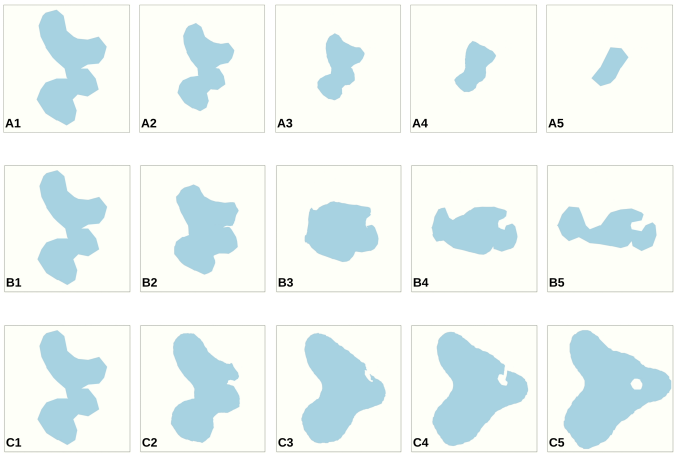

Fig. 7. Interpolation between arbitrary polygon pairs satisfying requirements R1 and R2. The same Origin polygon (A1, B1 and C1) is interpolated to three other polygons with arbitrary geometry (A5, B5 and C5). The intermediate shapes of each interpolation are displayed in the middle frames (A2-A4, B2-B4 and C2-C4).

the advancing-front technique [31] generated a higher quality mesh (96% good triangles) than the quadtree + template technique (41% good triangles). More details are available in Figure 4.

In several of our tests, the meshes generated by the advancing-front technique [31] produced satisfactory results. However, in some cases, we observed that the spatial organization of the triangles would influence the division results, compromising sufficiently smooth transitions.

The developed mesh-based method using quadtree and templates maximized the area generated with templates and reduced the unstructured area (see Figure 5).

We also noticed that the computational cost of the mesh-based techniques was too high and that their results did not produce smooth transitions for animation (R4). The results of polygon division approach based on Voronoi diagrams (see Section IV-B) show that using the edges of the diagram produced smooth and continuous divisions, which represent the closest regions to the corresponding polygon. This approach also avoided overlaps and generated the divisions fast enough to be animated in a interactive tool (R4).

C. Experiment 3: Interpolation between arbitrary sets of polygons

In this experiment, we performed one-to-many and many-to-one as well as many-to-many polygon interpolations to verify that our approach satisfies all the requirements R1 to R5. The results showed in the section already use the Voronoi-based approach for polygon division (Section IV-B).

In Figure 9, we show the results of a one-to-many interpolation comparison between Flubber and our approach. Notice the presence of overlaps in the interpolation generated by Flubber in Figure 9 (A2-A4). This is because its spatial division method does not take into consideration the relative position of the corresponding polygons (it performs a simple triangulation of the polygon). Our approach does not generate overlaps (Figure 9 (B2-B4)).

Figures 10 and 11 illustrate other examples including many-to-many interpolations with arbitrary geometry. Figure 11 shows how our approach can be used to visualize temporal changes of crime hotspots (regions where there was crime above a certain threshold) as an animation. Figure 11 (A1) displays the hotspots statically using superposition of polygons with two different colors. Each color represents a different time period. In Figure 11 (B2-B5) we show the animated interpolation of the hotspots from one time period to the other, keeping the Origin polygon set as a reference during all interpolation steps. This final example demonstrates that the proposed approach generates smooth transitions of arbitrary polygon sets without overlapping, satisfying all the established requirements R1 to R5. It also illustrates the potential of our approach to be used for visualizing geospatial phenomena represented as polygon sets.

VI. Conclusions and Future Works

In this paper, we proposed an approach for automatic interpolation between arbitrary sets of polygons. We showed that the approach was flexible and general enough to be used in many application domains. Our approach can be used with different spatial division techniques, and we found out that using a Voronoi-based approach for polygon division provided the best solution that satisfied the set of requirements of the interpolation problem. We validated the approach with a series of experiments with several types of interpolation.

There are a few areas of future work that we would like to pursue. First and foremost, we would like to integrate our approach into a visualization tool and perform a design study [36]. We are particularly interested in comparing static and dynamic visualizations of temporal changes. We also plan to explore using more sophisticated functions for generating the corresponding polygons instead of simple geometric intersections. These sophisticated functions can be dependent on the application domain. We would also like to investigate what kind of semantic information can we extract from the interpolation and use to automatically annotate the polygons during the interpolation. For example, we could detect when regions of interest contracted, expanded, divided or moved to another location. That could help users to better understand the temporal changes.

Acknowledgment

Our work has been funded by FUNCAP under the SPI grant.

References

[1] M.-J. Kraak and F. Ormeling, *Cartography: Visualization of Geospatial Data*, 3rd ed. Prentice-Hall, 2010.

[2] C. Robertson, T. A. Nelson, B. Boots, and M. A. Wulder, "STAMP: spatial–temporal analysis of moving polygons," *Journal of Geographical Systems*, vol. 9, no. 3, pp. 207–227, Sep 2007.

[3] K. C. Abhar, I. J. Walker, P. A. Hesp, and P. A. Gares, "Spatialtemporal evolution of aeolian blowout dunes at Cape Cod," *Geomorphology*, vol. 236, pp. 148 – 162, 2015.

[4] R. B. Santos, *Crime Analysis With Crime Mapping*, 3rd ed. Los Angeles, California: SAGE Publications, 2013.

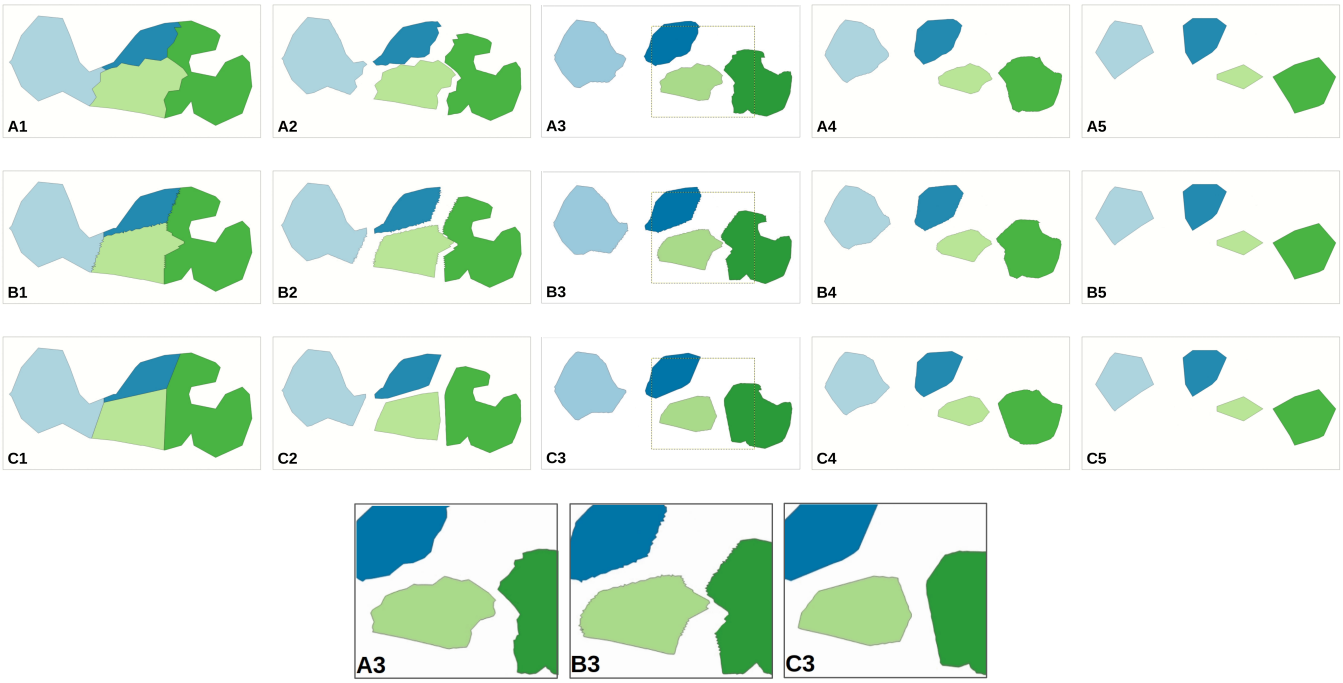

Fig. 8. Comparison of three different techniques for polygon division during interpolation. Line A1 - A5 illustrates the interpolation steps for the advancing-front technique [31]. In line B1 - B5 it was used the method based on a quadtree and templates, and in line C1 - C5 it was used a Voronoi-based technique.

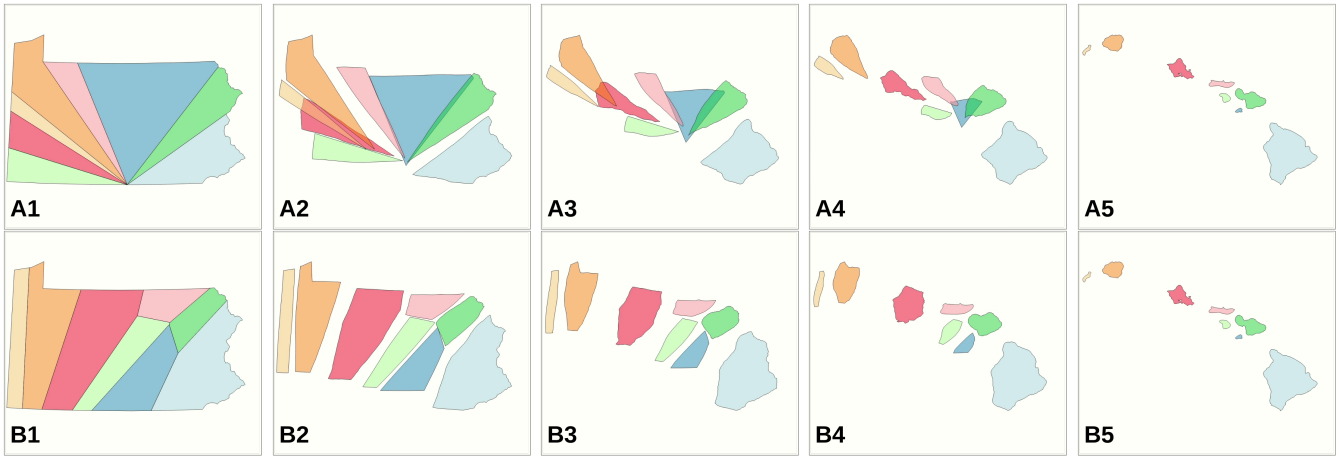

Fig. 9. Comparison of our approach with the Flubber library [12] on one-to-many interpolation. In this example the state of Pennsylvania is interpolated to the state of Hawaii. Line A1-A5 was generated with Flubber and Line B1-B5 was generated with our approach.

[5] K. J. Bowers, S. D. Johnson, and K. Pease, "Prospective Hot-Spotting: The Future of Crime Mapping?" *The British Journal of Criminology*, vol. 44, pp. 641–658, 2004.

[6] A. Malik, R. Maciejewski, E. Hodgess, and D. S. Ebert, "Describing Temporal Correlation Spatially in a Visual Analytics Environment," *In Proceedings of the Hawaii International Conference on System Sciences*, pp. 1–8, 2011.

[7] J. F. Queiroz, Neto, E. M. d. Santos, and C. A. Vidal, "MSKDE: Using Marching Squares to Quickly Make High Quality Crime Hotspot Maps," in *SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2016, pp. 305–312.

[8] B. Tversky, J. B. Morrison, and M. Betrancourt, "Animation: can it facilitate?" *International journal of human-computer studies*, vol. 57, no. 4, pp. 247–262, 2002.

[9] B. Lee, N. H. Riche, P. Isenberg, and S. Carpendale, "More Than Telling a Story: Transforming Data into Visually Shared Stories," *IEEE Computer Graphics and Applications*, vol. 35, no. 5, pp. 84–90, Sep. 2015.

[10] T. H. Kim and T. J. Cova, "Tweening Grammars: Deformation Rules for Representing change between Discrete Geographic Entities," *Computers, Environment and Urban Systems*, vol. 31, pp. 317–336, 2007.

[11] R. C. Carbunescu and S. V. Wart, "Polygon Vertex Set Matching Algorithm for Shapefile Tweening," http://vis.berkeley.edu/courses/cs294-10-fa08/wiki/images/d/d3/Finalpaper_rc_sv.pdf, University of California at Berkeley, 2008.

[12] N. Veltman, "Flubber: Tools for smoother shape animations," https://github.com/veltman/flubber, 2017.

[13] T. W. Sederberg and E. Greenwood, "A Physically Based Approach to 2-D Shape Blending," in *SIGGRAPH '92 Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, vol. 26. ACM New York, 1992, pp. 25–34.

[14] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, "2-D Shape Blending:

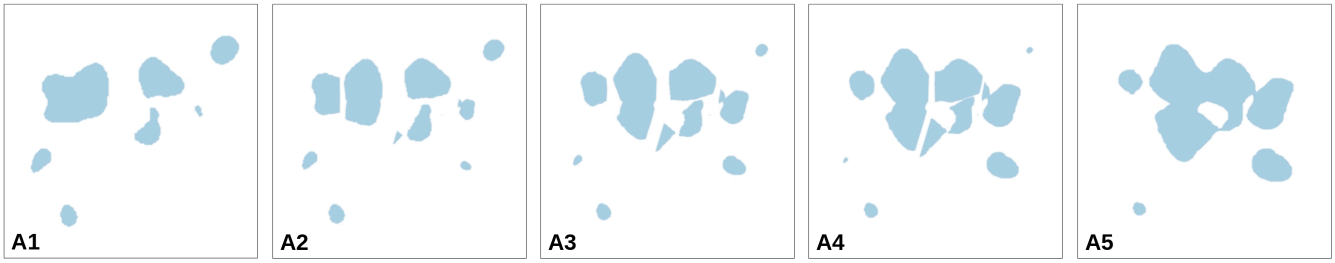

Fig. 10. Examples of many-to-many polygon interpolation using our approach with the Voronoi-based polygon division. A1 is the Origin polygon set, A5 is the Target polygon set. A2 to A4 are the intermediate shapes of the interpolation.

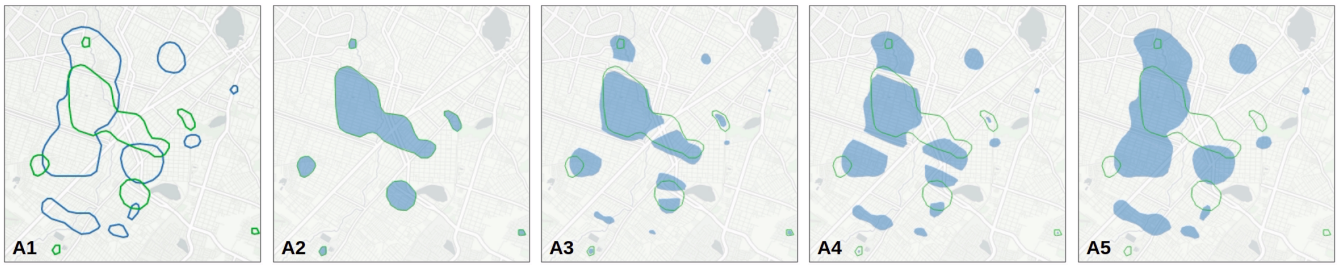

Fig. 11. Using our approach to visualize, as an animation, temporal changes of crime hotspots (regions where there was crime above a certain threshold). (A1) the hotspots are superposed polygons with two different colors (green and blue). Each color represents a different time period. (A2-A5) The animated interpolation of the hotspots showing the Origin time period (in green) as a reference during the animation.

An Intrinsic Solution to the Vertex Path Problem," in *SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM Nwe York, 1993, pp. 15–18.

[15] Y.-H. Shiao, K.-S. Chuang, T.-J. Chen, and C.-Y. Chen, "Polygon interpolation for serial cross sections," *Computers in Biology and Medicine*, vol. 37, pp. 1241–1251, 2007.

[16] C. Mizutani, "Land use transition process analysis using polygon events and polygon status: A case study of Tsukuba Science City," in *2009 17th International Conference on Geoinformatics*, Aug 2009, pp. 1–6.

[17] N. Salamat and E. Zahzah, "Fuzzy Spatio-temporal Relations Analysis," in *2010 Seventh International Conference on Information Technology: New Generations*, April 2010, pp. 301–306.

[18] M. Alexa, "Recent Advances in Mesh Morphing," *Computer Graphics forum*, vol. 21, pp. 173–197, 2002.

[19] T. Cashman and K. Hormann, "A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape," *Computer Graphics Forum*, vol. 31, pp. 735–744, 05 2012.

[20] P. von Radziewsky, E. Eisemann, H.-P. Seidel, and K. Hildebrandt, "Optimized subspaces for deformation-based modeling and shape interpolation," *Computers & Graphics*, vol. 58, pp. 128 – 138, 2016, shape Modeling International 2016.

[21] C. Brandt, C. von Tycowicz, and K. Hildebrandt, "Geometric Flows of Curves in Shape Space for Processing Motion of Deformable Objects," *Computer Graphics Forum*, vol. 35, no. 2, pp. 295–305, 2016.

[22] C. L. Bajaj, E. J. Coyle, and K.-N. Lin, "Arbitrary Topology Shape Reconstruction from Planar Cross Sections," *Graphical Models and Image Processing*, vol. 58, no. 6, pp. 524 – 543, 1996.

[23] J.-D. Boissonnat, "Shape reconstruction from planar cross sections," *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 1 – 29, 1988.

[24] C. Giertsen, A. Halvorsen, and P. R. Flood, "Graph-directed modelling from serial sections," *The Visual Computer*, vol. 6, no. 5, pp. 284–290, Sep 1990.

[25] C. Mizutani, "Construction of an analytical framework for polygon-based land use transition analyses," *Computers, Environment and Urban Systems*, vol. 36, no. 3, pp. 270 – 280, 2012.

[26] M. Berg, M. v. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

[27] F. P. Preparata and M. I. Shamos, *Computational Geometry: an Introduction*. Springer-Verlag, 1985.

[28] A. El-Hamalawi, "A 2d combined advancing front-delaunay mesh generation scheme," *Finite Elements in Analysis and Design*, pp. 967–989, 2004.

[29] D. J. Mavriplis, "An advancing front Delaunay triangulation algorithm designed for robustness," *Journal of Computational Physics*, pp. 90–101, 1995.

[30] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry*, vol. 22, no. 1, pp. 21 – 74, 2002, 16th ACM Symposium on Computational Geometry.

[31] A. C. Miranda, J. B. Cavalcante Neto, and L. F. Martha, "An algorithm for two-dimensional mesh generation for arbitrary regions with cracks," *SIBGRAPI 99: proceedings of the XII Brazilian symposium on computer graphics and image processing*, pp. 29–38, 1999.

[32] A. Tabarraei and N. Sukumar, "Adaptive computations on conforming quadtree meshes," *Finite Elements in Analysis and Design*, vol. 41, no. 7, pp. 686 – 702, 2005, the Sixteenth Annual Robert J. Melosh Competition.

[33] E. Ooi, H. Man, S. Natarajan, and C. Song, "Adaptation of quadtree meshes in the scaled boundary finite element method for crack propagation modelling," *Engineering Fracture Mechanics*, vol. 144, pp. 101 – 117, 2015.

[34] A. Tabarraei and N. Sukumar, "Extended finite element method on polygonal and quadtree meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 5, pp. 425 – 438, 2008, enriched Simulation Methods and Related Topics.

[35] F. Aurenhammer, "Voronoi Diagrams - a Survey of a Fundamental Geometric Data Structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, Sep. 1991.

[36] M. Sedlmair, M. Meyer, and T. Munzner, "Design Study Methodology: Reflections from the Trenches and the Stacks," *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, vol. 18, no. 12, pp. 2431–2440, 2012.