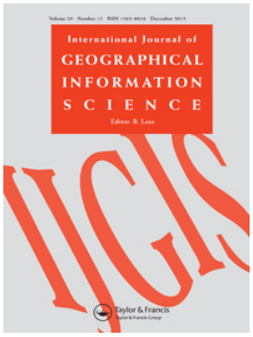

International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: http://www.tandfonline.com/loi/tgis20

Change detection for 3D vector data: a CGA-based Delaunay–TIN intersection approach

Zhaoyuan Yu, Wen Luo, Yong Hu, Linwang Yuan, A-Xing Zhu & Guonian Lü

To cite this article: Zhaoyuan Yu, Wen Luo, Yong Hu, Linwang Yuan, A-Xing Zhu & Guonian Lü (2015) Change detection for 3D vector data: a CGA-based Delaunay–TIN intersection approach, International Journal of Geographical Information Science, 29:12, 2328-2347, DOI: 10.1080/13658816.2015.1077963

To link to this article: http://dx.doi.org/10.1080/13658816.2015.1077963

Published online: 20 Aug 2015.

International Journal of Geographical Information Science, 2015
Vol. 29, No. 12, 2328–2347, http://dx.doi.org/10.1080/13658816.2015.1077963

Change detection for 3D vector data: a CGA-based Delaunay–TIN intersection approach

Zhaoyuan Yu[a,b,c], Wen Luo[a], Yong Hu[a], Linwang Yuan[a,b,c]*, A-Xing Zhu[a,b,c] and Guonian Lü[a,b,c]

[a]*Key Laboratory of Virtual Geographic Environment, Ministry of Education, Nanjing Normal University, Nanjing, P.R. China;* [b]*State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing Normal University, Nanjing, P.R. China;* [c]*Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing Normal University, Nanjing, P.R. China*

(*Received 2 April 2015; accepted 26 July 2015*)

In this paper, conformal geometric algebra (CGA) is introduced to construct a Delaunay–Triangulated Irregular Network (DTIN) intersection for change detection with 3D vector data. A multivector-based representation model is first constructed to unify the representation and organization of the multidimensional objects of DTIN. The intersection relations between DTINs are obtained using the *meet* operator with a sphere-tree index. The change of area/volume between objects at different times can then be extracted by topological reconstruction. This method has been tested with the Antarctica ice change simulation data. The characteristics and efficiency of our method are compared with those of the Möller method as well as those from the Guigue–Devillers method. The comparison shows that this new method produces five times less redundant segments for DTIN intersection. The computational complexity of the new method is comparable to Möller's and that of Guigue–Devillers methods. In addition, our method can be easily implemented in a parallel computation environment as shown in our case study. The new method not only realizes the unified expression of multidimensional objects with DTIN but also achieves the unification of geometry and topology in change detection. Our method can also serve as an effective candidate method for universal vector data change detection.

Keywords: change detection; vector data; 3D; conformal geometric algebra; TIN

1. Introduction

Change detection is one of the most important analyses in spatiotemporal data analysis. Observing the same object or phenomenon at different times to confirm its evolutionary process and then analyzing its changing character is the main goal of geographic change detection. Various methods, such as pixel-based and object-based methods, were developed for raster and point cloud data. For vector data, many change detection methods were developed based on polygon/object overlay concepts (Metternicht 2001, Agouris *et al*. 2001, Walter 2004, De Chant and Kelly 2009, Berlanga-Robles and Ruiz-Luna 2011, Frey *et al*. 2012, Aguirre-Gutierrez *et al*. 2012, Qin and Gruen 2014, Rokni *et al*. 2014).

Along with the wide application of 3D GIS and the increasing accumulation of 3D vector data (e.g. 3D Solid Earth Models, 3D cadastre), change detection of 3D vector data is becoming more and more important (Navratil 2008, Döner *et al*. 2011,

*Corresponding author. Email: yuanlinwang@njnu.edu.cn

Siejka *et al*. 2014). However, the overlay or intersection computations of 3D objects are far more complex than those of 2D objects due to the complexity in data structures, intersection judgments, and topological relationship calculations in 3D. The overlay or intersection computations of 2D objects are unified while the computations of 3D objects are not (Frankel *et al*. 2004). Special data organizations (e.g. TEN, GTP), indexes (e.g. OBB-Tree, Kd-Tree), and data structures are required (Zhang and Zhang 2010). Therefore, directly extending the exiting 2D change detection methods based on polygon overlay to the 3D situation is complex and computationally inefficient. Developing a simple and multidimensionally unified change detection mechanism, which not only inherits the advantages of 2D vector change detection methods but also is efficient for 3D computation, helps to reduce the complexity and improve the processing ability of 3D change detection.

The Delaunay–Triangulated Irregular Network (DTIN) is a simple data structure used commonly for vector data representation and computation. It is a bridge integrating 2D, 2.5D, and 3D vector GIS data in a unified framework (Domiter and Žalik 2008, Ledoux and Gold 2008). Both the 2D and 3D vector data can be easily discretized into DTINs. Most 3D/2.5D surface models are constructed based on 2D data with DTINs. Therefore, the DTIN provides a simple and unified representation for objects with different dimensions.

DTINs have excellent geometric properties that can be used for vector data change detection. The particular coordinates of control points and their relative positions influence the structure of a DTIN. Thus, the DTIN's structure will be always the same if the control points are not changed. For objects represented with a DTIN, the change of the control points affects only the shape of the neighboring triangular objects (Wu *et al*. 2011). This shape-adaptive property of the DTIN provides a simple and direct way to develop change detection methods by intersecting the DTINs of 3D objects at different times. This simplicity provides an enormous potential for developing a multidimensionally unified change detection method on the foundation of direct DTIN intersection.

Several problems should be solved in developing a change detection method based on DTIN intersection for multidimensional objects. First, intersections between different dimensional objects are required (e.g. the intersection of segment–segment, segment–triangular, and triangular–triangular). However, the intersection operation for objects in classical GIS is not multidimensionally unified. Complex judgments as well as various different data structures are required for change detection. Second, with the intersection between triangles, only intersecting lines and boundary points can be produced. Due to the lack of topological constraining information, it is difficult to reconstruct objects that have changed based on the intersection objects. Additional computations of statistical parameters of the changed objects, such as area, volume, and surface configuration, are even more complex. Third, it is necessary to traverse all the triangles to achieve the change detection of the whole object. So when dealing with large data, the efficiency will be largely reduced.

One of the major challenges in using DTINs in change detection is the lack of multidimensionally unified symbolic intersection computation. Most of the current intersection algorithms, which are based on computational geometry and/or Euclidean geometry, are related to coordinates and different dimensional objects. Special mechanisms, such as scan lines, are used to increase the efficiency (Wang *et al*. 2013). However, these mechanisms often separate the original geometric and topological structure of the original DTIN. Since the expressions of points, lines, and triangles are separated,

developing a unified intersection algorithm and reconstructing the topology becomes more difficult. The lack of the unified representation of geometries also increases the complexity of designing spatial indexes to quickly remove invariant areas. In addition, simple and unified topology computation and reconstruction are indispensable for the change of area/volume extraction.

Multidimensionally unified representation and intersection of the DTIN will be helpful to solve the above problems. The conformal geometric algebra (CGA) provides an ideal tool for the unified representation and computation of DTINs. In CGA, DTINs can be constructed easily and directly according to the Grassmann structure. The nodes, edges, and the whole triangular object can be represented as a unified and hierarchical yet single multivector. With the powerful CGA operators, the geometric metrics and topological relations can be calculated directly and symbolically (Yuan *et al*. 2014). In this paper, CGA is introduced to represent the DTINs in the multidimensionally unified way. A spatial index (*Bounding Sphere Tree*) and the *meet* product are introduced for the efficient intersection judgment and the intersection object extraction. Then a change region extraction and reconstruction algorithm is proposed. The method is demonstrated with the change detection of Antarctic ice caves.

The paper is organized as follows: Section 2 presents the basic idea. In Section 3, the multivector-based expressing and modeling method for a DTIN, the algorithm of triangle intersection, and the change detection are discussed; experiments and case studies are performed in Section 4; and the discussion and the conclusions are in Sections 5 and 6, respectively.

2. The basic idea

The lack of multidimensionally unified representation and computation of DTIN intersection is the key problem of DTIN-based change detection. Introducing modern mathematics that can support multidimensionally unified representation and computation will be helpful. CGA provides a unified and concise homogeneous algebraic framework for elementary geometry representation. The multidimensionally unified representation of objects with the DTIN can be addressed by CGA unified representation. In the CGA framework, geometric objects can be generated by the *outer* product according to the Grassmann structure in a coordinate-free way (Hitzer *et al*. 2013). DTINs and associated elements (points, segments, and triangles) can be uniformly represented with the multivector structure (Yuan *et al*. 2011). Then the multivector representation can support multidimensionally unified computing with CGA operators (Yuan *et al*. 2012).

Under this representational framework, the change detection can be perceived as intersection of triangles which can then be computed based on the *meet* product. The *meet* operator can be used for both the unified intersection and the topological relation computation. The adaptability of the *meet* operator is helpful for realizing the unified solution for intersection relations of different geometric objects (Hitzer 2005). The result of the *meet* operator on two multivectors is also a multivector, which indicates the geometry of the intersection result. The intersection relations of geometric objects would adaptively change as the geometry types and positions change (Yuan *et al*. 2012). Since the intersection with the *meet* product is unified and directly computed in a single mathematical framework, it can link the geometric representation and algebraic computation seamlessly. In addition, the unified intersection computation can be simply implemented and integrated with the existing GIS. To accelerate the

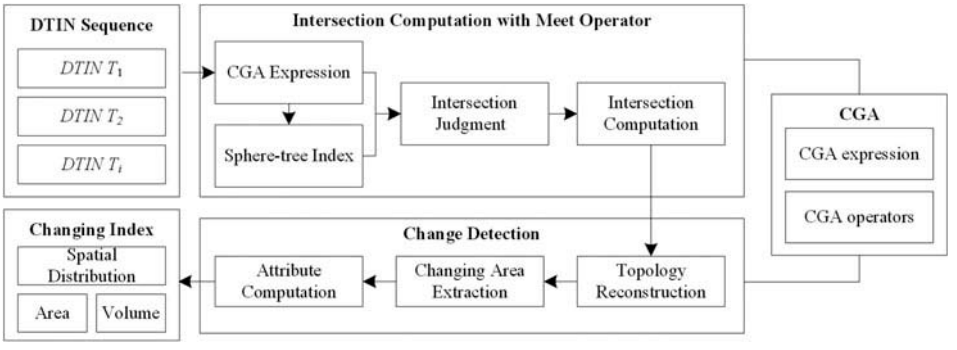

Figure 1. The basic idea of multidimensionally unified change detection method for vector data.

calculation of small changes, an efficient spatial index that removes invariant triangles is helpful.

The topological reconstruction and change of area/volume extraction can be solved by geometric reconstruction according to the Grassmann structure (Yuan *et al*. 2011). The Grassmann structure keeps not only the geometric construction structures but also topological structures (e.g. the *outer* product of two points is a line segment and the outer product of three points is a plane or circle). The *outer* product representation is data adaptive (i.e. if three points are on one line, it represents a line or two segments; if not, it presents a flat plane or a circle) and independent of dimensions (i.e. the representation will not change whether the DTIN is in 2D or 3D). The topology between points, segments, and triangles can also be uniformly computed (Yuan *et al*. 2014). Due to the unification of the *meet* operator for intersection and topological relation computation, the intersection and topological reconstruction procedure can be unified and simplified. With the help of the Grassmann structure of the CGA representation, the reconstruction of the change of area/volume can also be simplified.

The overall framework to deploy the above strategy is depicted in Figure 1. First, the DTIN of each object at each time is constructed. Therefore, objects at all times form a DTIN sequence, with a DTIN each time. Each DTIN in the sequence is represented by the CGA. With the sphere-tree index and the intersection computation algorithm constructed with the *meet* operator, the intersection judgment and computation is accomplished. The intersected components are extracted. The topology of the intersected components is reconstructed. Then the change of area/volume can be extracted and the statistical parameters can be performed to finalize the change detection analysis. All procedures are implemented under CGA, with the help of CGA expressions and operators.

3. Methods

The multidimensionally unified change detection method for vector data is composed of the following components: (1) the CGA expression of a DTIN – the DTIN is projected from geographic space to CGA and expressed as multivectors; (2) intersection judgments and computations; (3) intersection computation; (4) and change of area/volume extraction and reconstruction.

3.1. The CGA expression of the DTIN

The DTIN is a kind of typical compound object that is composed of mixed dimensions that consist of points, lines, and planes. According to the CGA 3D data model (Yuan *et al*. 2011), any DTIN can be expressed by multivectors:

$$DTIN = \{T_1, T_2, \cdots, T_m\} = T_1 \oplus T_2 \oplus \cdots \oplus T_m \tag{1}$$

where T_i is the ith triangle of a DTIN, and its CGA expression is as follows:

$$\begin{aligned} T_i &= [p_{1i} \wedge p_{2i} \wedge p_{3i} \wedge e_\infty] < p_{i1}, p_{i2}, p_{i3} > = p_{i1} \wedge p_{i2} \wedge p_{i3} \wedge e_\infty \\ &+ [p_{i1} \wedge p_{i2} \wedge e_\infty] < p_{i1}, p_{i2} > + [p_{i1} \wedge p_{i3} \wedge e_\infty] < p_{i1}, p_{i3} > + [p_{i2} \wedge p_{i3} \wedge e_\infty] < p_{i2}, p_{i3} > \\ &+ p_{i1} + p_{i2} + p_{i3} = S_i + l_{i1} + l_{i2} + l_{i3} + p_{i1} + p_{i2} + p_{i3} \end{aligned} \tag{2}$$

where '[]' represents the *GeoCarrier* components of a blade (Yu *et al*. 2015); '<>' represents a set of points sequence used to define the boundaries of elements of the DTIN. p_{i1}, p_{i2} and p_{i3} are three vertexes of the ith triangle; l_{i1}, l_{i2}, and l_{i3} are three sides of the triangle; and s_i is the plane which contains the triangle.

The CGA expression of a DTIN based on the above hierarchical relations unifies the expression of points, segments, and triangles in a single unified structure. The simplicity in the dimension structure and the object structure is represented. The anti-symmetric property of the *outer* product provides the orientation for each segment and triangle, which is very helpful for the topological relation calculation. Since the multivector representation is not only a representation structure but also a computation structure, a simple and unified data organization as well as associated data structure can be formed. Moreover, the calculability of the CGA enables it to be a powerful tool for calculating the relevant characteristic parameters of a DTIN.

Similar to the *MVTree* data structure (Yuan *et al*. 2014), the data structure of a DTIN can be constructed and defined as depicted in Figure 2. To support the unified calculation of different dimensional objects, the data structure is based on the dimension structure of the CGA, in which a triangle is expressed and stored by an object tree, and each triangle is

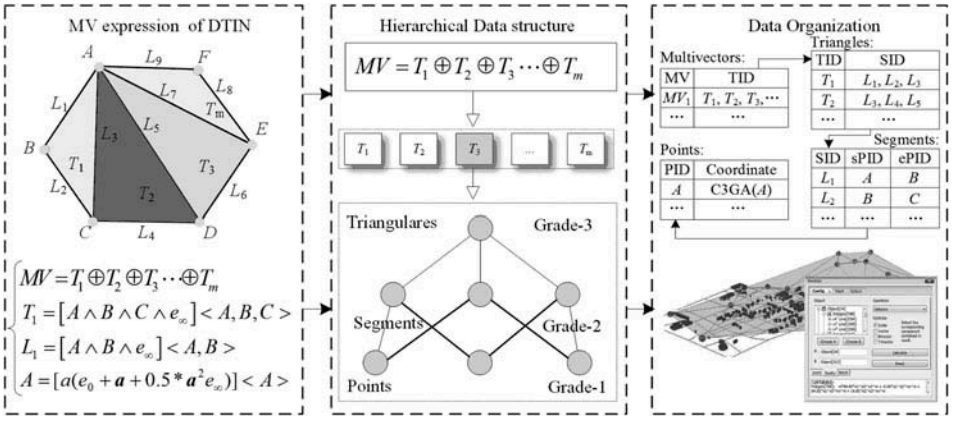

Figure 2. Multivector-based object expression of the DTIN.

identifiable by a unique ID. Each leaf node represents a vertex of the triangle; each middle node corresponds to a side formed by its leaf nodes, and the root node stores the *outer* product expression of the plane containing the triangle and the boundary constraints. The mutual operations of multidimensional objects and the computation of CGA expressions are realized by association lists. The data structure stores the DTIN and the associated elements in a hierarchical structure, with both the dimensional construction structure and the topological relation structure inherited. Therefore, the data structure can be easily processed in the multidimensionally unified computation framework with CGA operators (Yuan *et al*. 2012). The compact representation with meaningful symbolical representations also helps to reduce the space cost during the data organization and computation (Yuan *et al*. 2013)

3.2. *Intersection judgments and computations*

Owing to the advantages of the CGA in object expression and computing, we can first express the DTIN in the form of a multivector with the *outer* product representation in the CGA, and then we can construct the DTIN intersection algorithm using the *meet* operator. For small changes, most triangles will not change. These triangles are not required to be processed with the intersection algorithm to extract the intersection. Therefore, an efficient spatial index for intersection judgment needs to be developed to accelerate the calculation of the intersection. This spatial index should identify triangles that will intersect and remove unchanged triangles. After applying the intersection algorithm to the changed triangles, the intersection areas and the topological relations of the intersection should be analyzed for topological reconstruction and change of area/volume extraction.

Based on the above hierarchical expression of a DTIN, we can construct the intersecting algorithm for the DTIN by the *meet* operator (Yuan *et al*. 2014). Given any two blades A and B, the *meet* operator is defined as follows:

$$M = (A \cap B) = (I_{AB}A) \cdot B \tag{3}$$

where I_{AB} is the pseudo-scalar forming the minimal subspace which contains A and B simultaneously. Since the original *meet* operator in the CGA space cannot deal with the boundary of some geometries (e.g. the two disjoint non-parallel segments will have meaningful intersection in the CGA framework), we modified the *meet* product using the same technology that has been used in the GA-MUC to integrate the boundary of the geometries during the intersection computation (Yuan *et al*. 2012).

The sign of the square of the *meet* product (M^2) can be used to judge the intersection relations of the blades rapidly (Roa *et al*. 2011, Yuan *et al*. 2014). For line–line, line–plane, and plane–plane intersection relations, if $M^2 > 0$, there would be at least two crossover points between A and B; if $M^2 = 0$, A is tangent to B at one or more than one point; when $M^2 < 0$, A does not intersect B.

According to the whole processing flow, the following steps are constructed to deal with the DTIN intersection:

3.2.1. *Step 1: Rough-level intersection judgment with the spatial index*

If two triangles are not changed at different times, the coordinates of the three points of the triangles are not changed. With these properties, the Bounding Volume Hierarchies

(BVHs) can be utilized to construct a spatial index so as to quickly identify the triangles that are not intersected. If the BVHs coincide or are disjoint, the internal triangles will certainly coincide or be disjoint, respectively. With BVHs of a certain size and the hierarchal structure of BVHs of different sizes, the triangles that are not intersected can be easily removed.

Since the spheres are more general and easier to be represented in the CGA, we use spheres to construct the spatial index to achieve a balance between object retrieval and intersection judgment. Spheres can be directly constructed by using the *outer* product. The distance measurement and topological relation judgments can be performed on spherical objects directly with *inner* products (Rivera-Rovelo and Bayro-Corrochano 2007). The sphere tree is also easy to be maintained and updated dynamically. The hierarchy construction and partition of the sphere tree can be easily constructed according to the BRNO-ST index (Yu *et al*. 2012). Different from a BRNO-ST index that primarily pursues computing precision, we only use spheres to roughly detect where the two triangles are intersected. High compactness (the fitting precision of objects with bounding spheres) in the BRNO-ST index can be avoided. To ensure higher operational efficiency, we adopt a relatively looser bounding sphere constructing and dividing strategy to generate the bounding sphere tree:

(1) The coordinates of the center of the sphere are the median values of the coordinates of all vertexes in every coordinate axis, respectively; the radius is the maximum value of distances from all vertexes to the center of sphere.
(2) Count the number of triangles in the DTIN; if the number is 1, then terminate the algorithm; otherwise perform (3).
(3) Select the coordinate plane to which the vertical distance difference is the largest from all vertexes as the parting plane, then calculate the distances from the centroid of triangles to the parting plane and sort all the triangles according to the computing results; the first half of triangles will be stored in the left subtree and others will be stored in the right subtree.
(4) Perform (1–3) on the left subtree and right subtree, respectively.

With the above process, the construction of the boundary sphere tree can be constructed using the procedures depicted in Figure 3.

The above bounding sphere tree is a standard binary tree which could be used in the ordering and searching for complex DTIN objects. In this sphere tree, each leaf node represents a single triangle. Each middle node contains a part of the DTIN, which is also a DTIN. The root of the tree contains the whole DTIN. The obtained bounding spheres contain all the vertexes in the DTIN. In CGA, the bounding sphere tree is stored as multivector object:

$$\begin{aligned}\Sigma &= Sp_0 = Sp_{00} \oplus Sp_{01} = Sp_{000} \oplus Sp_{001} \oplus Sp_{010} \oplus Sp_{011} \\ &= \underbrace{Sp_{00\cdots0} \oplus \cdots \oplus Sp_{11\cdots1}}_{n}\end{aligned} \tag{4}$$

The algorithm begins with a relation judgment of the bounding spheres of root nodes of the two DTIN objects, as shown in Figure 4. First, we need to judge whether the two root nodes are leaf nodes. If they are not, then we can perform the *meet* operator on the bounding spheres Σ_1, Σ_2 of the two root nodes:

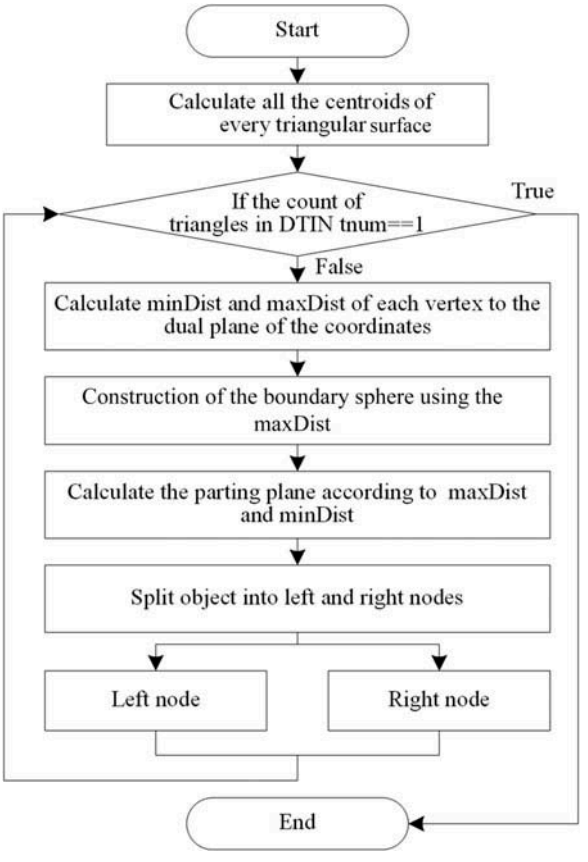

Figure 3. The flow chart of the construction of the sphere tree.

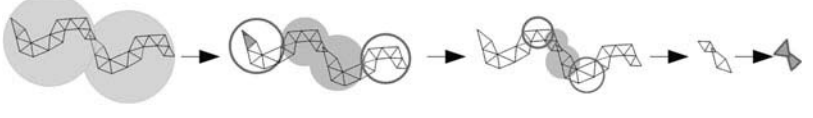

Figure 4. Sphere-tree-based interaction detecting process of the DTIN.

$$C = \Sigma_1 \cap \Sigma_2 = \left[\langle \Sigma_1 \Sigma_2 \rangle_{2n-r-s}\right]^* \tag{5}$$

where C is the result of the *meet* operation that is defined in Equation (3); n is the dimension of the space; r and s are the grades of the bounding spheres Σ_1 and Σ_2, respectively (e.g. $n = 5$, $r = 4$, and $s = 4$ for two spheres in CGA $Cl(4,1)$). If $C^2 < 0$, Σ_1, Σ_2 are disjoint, so the triangular subnetworks contained in the two spheres are also disjoint; if $C^2 >= 0$, Σ_1, Σ_2 intersect, then we need to perform the above steps on child nodes of the two nodes, respectively. If the two nodes are leaf nodes, the triangles in the nodes are possible to be intersected. Additional operations should be applied to get the detailed intersection results.

3.2.2. Step 2: Fine-intersection judgment with triangle relation classification

We use the *meet* operator to retrieve detailed intersection relations and intersection geometries. The intersection computation can be performed among vertexes, edges, and surface patches of triangles. With the multivector representation of a DTIN, the intersection with the *meet* operator can also be achieved in a hierarchical way (Yuan *et al*. 2014). Since the direct calculation of the *meet* operator is computationally inefficient, we provide the following optimized methods to determine the detailed intersection relations between a DTIN and its associated elements.

In the beginning, a pre-judging operation is performed on planes containing triangles by the *meet* operator: $M = (T_1 \cap T_2)$. If $M^2 < 0$, T_1 and T_2 are disjoint, then the triangles contained in T_1 and T_2 are also disjoint, respectively. Since the *outer* product is asymmetric, the planes have certain orientations. If the *outer* product of the three vectors pointing from any vertex of the triangle T_1 to the three vertexes (A_2, B_2 and C_2) of triangle T_2, respectively, ($Vol = A_2 \wedge B_2 \wedge C_2$) is zero, the two triangles T_1 and T_2 are coplanar and intersect, then the *meet* operation can be performed on corresponding edges of the two triangles. If not, then the *meet* operation should be performed on the surface patch of triangle T_1 and three edges of triangle T_2. Then the following three judgments need to be made and processed.

3.2.2.1. Judgment 1: Intersection relation judgment between segments. The spatial relation judgment operator for segments can be constructed based on the orientation information extracted by the *outer* products (Figure 5(a)). By calculating the *outer* product of the segment direction vector of a segment R_1 with two vectors pointing from any of the two endpoints of segment R_1 to the two endpoints of segment R_2, respectively, we obtain two 2-blades: $B_1 = \boldsymbol{v}_3 \wedge \boldsymbol{v}_1$ and $B_2 = \boldsymbol{v}_3 \wedge \boldsymbol{v}_2$. Similarly, we can obtain another two 2-blades B_3 and B_4 by the *outer* product of the two vectors pointing from segment R_2 to segment R_1. Then the intersection relation judging operator for segments can be constructed as follows:

$$O_{seg-seg} = (dir(B_1) == dir(B_2)) \| dir(B_3) == dir(B_4)) \tag{6}$$

where $dir(X)$ is the direction of X, which can be identified by the sign of the blade. $==$ is the Boolean equation determination symbol; $\|$ is the Boolean logical 'or' symbol. When the value of $O_{seg-seg}$ is 'true', the two segments are disjoint; otherwise they are intersected.

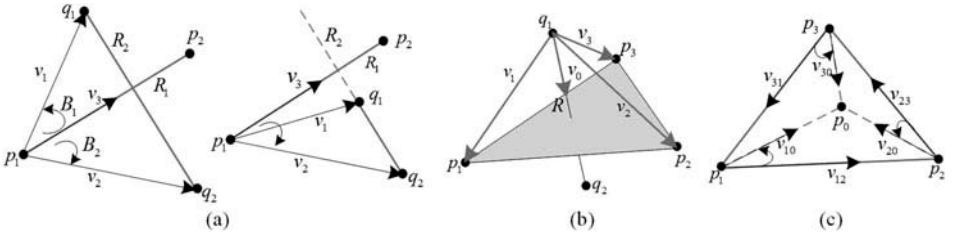

Figure 5 CGA-based spatial relation judgment for triangles. (a) Judgment of relation between segments and segments. (b) Relation between segments and triangles. (c) Relation between points and triangles.

3.2.2.2. Judgment 2: Intersection relation judgment between segment and triangle. The *outer* product of the three vectors pointing from one endpoint q_1 of the segment R to three vertexes of the triangle T is a 3-blade $Tr_1 = \boldsymbol{v}_1 \wedge \boldsymbol{v}_2 \wedge \boldsymbol{v}_3$. Similarly, there is another 3-blade Tr_2 by the *outer* product of vectors pointing from the endpoint q_2 of the segment R to the triangle T. Then the intersection relation judgment operator for segment and triangle can be constructed as follows:

$$O_{seg-tri} = (dir(Tr_1) == dir(Tr_2)) \tag{7}$$

When the value of $O_{seg-tri}$ is 'true', the segment and the triangle are disjoint; otherwise they intersect each other.

3.2.2.3. Judgment 3: Intersection relation judgment between point and triangle. By constructing the three vectors pointing from every vertex of the triangle T to the point P_0 and calculating the *outer* product of every edge of the triangle T and every vector in a clockwise or counterclockwise direction, we obtain three 2-blades $B_1 = \boldsymbol{v}_{12} \wedge \boldsymbol{v}_{10}$, $B_2 = \boldsymbol{v}_{23} \wedge \boldsymbol{v}_{20}$ and $B_3 = \boldsymbol{v}_{31} \wedge \boldsymbol{v}_{30}$. The intersection relation judging operator for points and triangles can be constructed as

$$O_{pt-tri} = (dir(B_1) \geq 0 \; dir(B_2) \geq 0 \; dir(B_3) \geq 0) \tag{8}$$

When the value of O_{pt-tri} is true, the point is within or on the boundary of the triangle; otherwise the point is outside the triangle.

3.2.3. Step 3: Intersection computation

With the above rough and fine-level judgments for non-intersecting triangles, the real intersection of the triangles can be assessed with the *meet* operator. According to the *meet* operator between two triangles, six different intersection situations can be extracted (Figure 6). In traditional algorithms, different strategies need to be designed for different situations, which cause high complexity. The *meet* operator-based triangle intersection can not only realize the unified expression and computation of objects with different dimensions, but also uniformly deal with all these peculiar situations.

After obtaining the intersecting segments by the intersecting operation, it is necessary to construct the topological relations of these segments. What we need to do is to figure out the sequential relations of all segments and then extract chains or circles from them. This is achieved mainly by the inclusion relation judgment between point and segment. In the CGA, for any vector x and blade $A(grade(A) \geq 1)$, the inclusion relation of them could be judged by $sign = x \wedge A = x \cdot A^*$, where A^* is the dual of A; if $sign = 0$, A contains x. Assume that the intersecting segment set of two TINs is H; the procedure of the topology reconstruction is as follows:

(1) Select any intersecting segment p_1p_2 which has not been visited in H as the starting segment, and then insert p_1p_2 into the new intersecting segment set L_i and mark it in H as 'visited'; after that, judge whether there is any segment e in H where the *outer* product of e and the endpoint of p_1p_2 is 0.

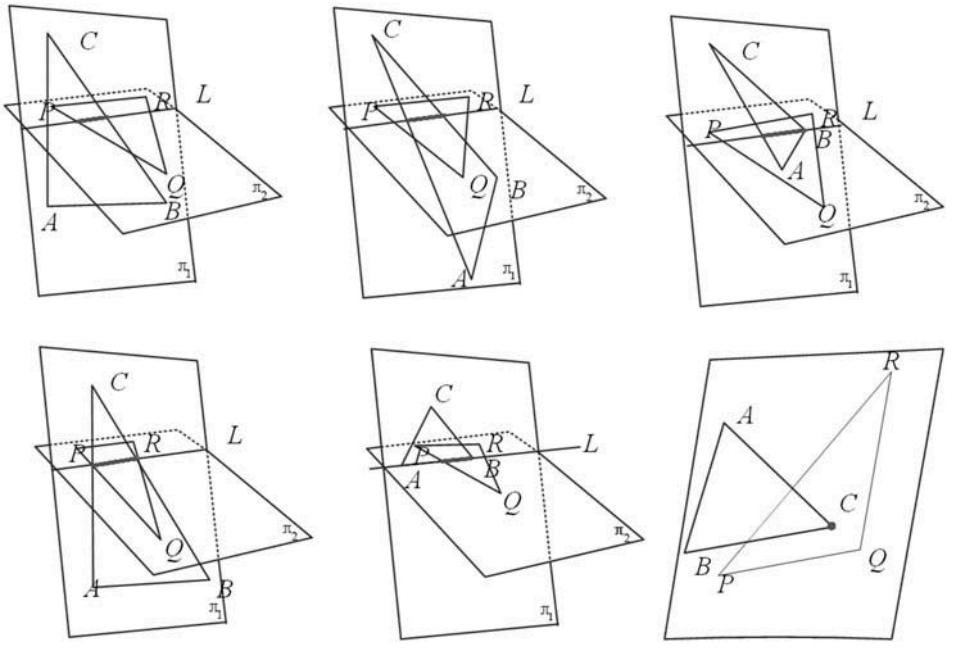

Figure 6. Different situations of triangle intersection.

(2) If e exists, insert it into the end of p_1p_2 in L_i and meanwhile mark it as 'visited' in H.
(3) Take e as the starting segment and search another segment f where the *outer* product of f and the endpoint of e is 0; if f exists, turn to step 2; otherwise perform the next step.
(4) If all the segments in H have been visited, terminate the algorithm. The sequential relations of all intersecting segments in H have been ascertained. Otherwise, return to step 1.

3.2.4. *Step 4: Change of area/volume extraction and reconstruction*

With the topological order sequences of the intersected segments, the topological reconstruction of the DTIN can be then used to reconstruct the change of area/volume.

First, the standard Delaunay triangular network should be generated based on the Delaunay algorithm. After that, we need to embed the intersecting segments into the DTIN. During the process, it is necessary to adjust the location of each segment according to the Diagonal Exchange Method so that each triangle satisfies the Empty Circle Rule (Mi and Geng 2013). We construct the circumcircle of the triangle ABC, $S = A \wedge B \wedge C$, by the *outer* product, and then we judge whether D is in S by the *inner* product of D and the *dual* of S, $\sigma = D \cdot S^*$.

The change of area/volume can be extracted according to the intersecting segments and relative topological information. As shown in Figure 7, by overlaying the original DTIN T1 with the changed DTIN T2, T2 is divided into three parts by T1: the upper side of T1, the intersection area of T1, and the underside of T1. The three parts of T2 are obtained through the intersecting segments and relative topological information. Since the

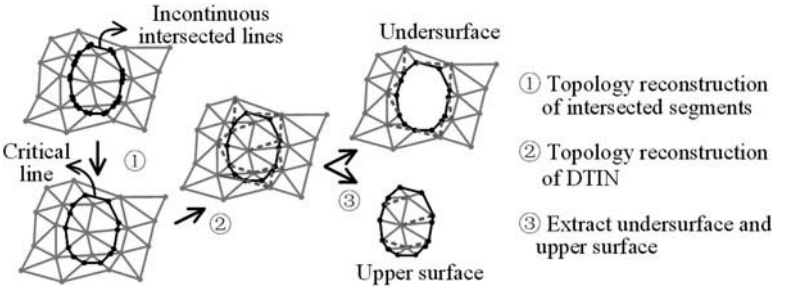

Figure 7. Topological reconstruction of the intersecting segments and triangular surfaces.

area and volume changes are calculated directly based on the sum of the exact area and volume of triangles and triangular prisms, respectively, there is no precision lost in our algorithm.

4. Case studies

4.1. Experimental environment and data

We implemented our method with the CGA-based unified spatiotemporal analysis system (CAUSTA) (Yuan *et al*. 2010, 2013). The CGA computations are implemented in C++ and optimized using the Gaalop Precomplier (Charrier *et al*. 2014). The method is implemented and tested on a Windows 7 system with the Visual Studio 2008. The experimental environment is a Lenovo T440 s notebook with a CPU of Intel Core i5-4200U, 1.60 GHz, 4 G RAM. The graphics processing unit (GPU) is the integrated HD4400 GPU, which has 20 EUs. For each EU, it has a dynamical frequency ranging from 200 MHz to 1GHz.

We use the data set simulated from a 3D dynamical ice model in an idealized simulation of ice growth starting from an ice-free continent (~34,000 kaBP) to the ice-cover formation period (~33,200 kaBP) of the Antarctica (Deconto and Pollard 2003). The ice layer area and ice-cover volume change information will be selected to reveal the evolution of the ice layer in the formation process of the Antarctica. Nine time points (33,195, 33,295, 33,395, 33,495, 33,595, 33,695, 33,795, 33,895, and 33,995 kaBP) are

Table 1. General information of data.

DTIN	Time (kaBP)	Number of nodes	Number of triangles
v1	33,995	6300	13,800
v2	33,895	6100	12,514
v3	33,795	6637	12,735
v4	33,695	7957	15,419
v5	33,595	7350	14,149
v6	33,495	7534	14,518
v7	33,395	7633	14,736
v8	33,295	7957	15,419
v9	33,195	8608	16,751

selected as experimental data. The DTIN is constructed from the original point cloud data. The numbers of nodes and triangles in each DTIN are shown in Table 1.

4.2. *Spatiotemporal evolution analysis*

The spatiotemporal evolution of the Antarctic ice variation is extracted with our multi-dimensionally unified change detection method. First, the intersections of the DTINs of each adjacent time are calculated. The geometries of the changing area and volumes are then constructed by the intersection computation and topological reconstruction. The ice changes at each location between different adjacent times are classified into the ice expanding and shrinking according to the increase or decrease of the areas and volumes. The overall amounts of the area and volume changes of the ice expanding and shrinking are calculated and compared, respectively. Then, we can have time series of area, the

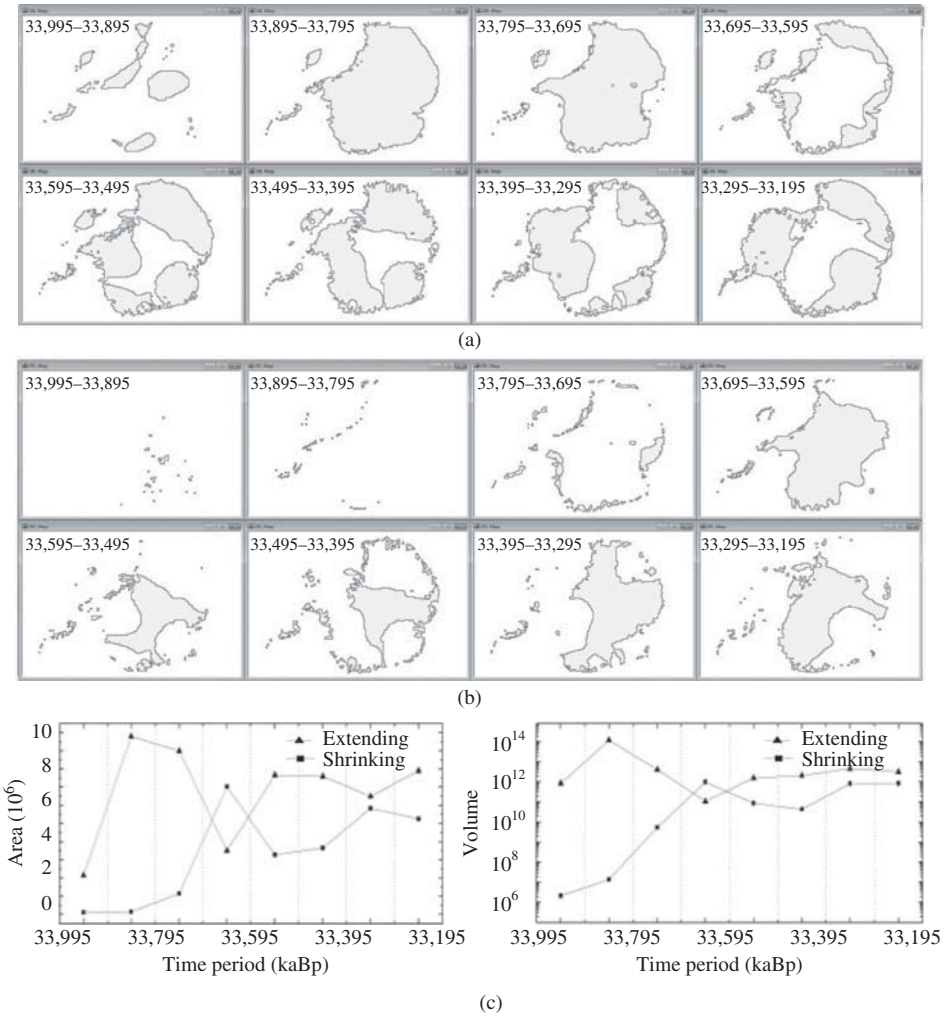

Figure 8. The result of the change detection. (a) The extending regions of different period (kaBP). (b) The shrinking regions of different period. (c) Time variation of extended and shrinked ice.

volume changes of the ice expanding and shrinking, and the spatial shape changes of the ice expanding and shrinking at different times.

By comparing the expanding area and melting area of the Antarctic ice cover in each time period, we obtained the change of area/volume at each different time. Figure 8(a) shows the expanding area changes of the ice cover in the time periods from 33,195 to 33,995 kaBP with an interval of 100 Ka. And Figure 8(b) shows the melting area change of the ice cover in each time period. In Figure 8(c), the area and volume change of the ice cover in each period is provided. Several simulation results have already suggested the Antarctic ice cape appeared 34 million years ago, grew rapidly due to Earth's orbital changes (Williams 2014), and accumulated very quickly in the first 30–40 ky. In the period of 33,995–33,895 kaBP, the expanding area was relatively large while the melting area was relatively small. In the period of 33,695–33,595 kaBP, the expanding area increased greatly; and in the period of 33,595–33,195 kaBP, both the expanding and melting areas were relatively stable. The stability of ice sheet in the period also agrees well with several simulation results (Deconto and Pollard 2003). A rapid ice-cover area melting happened during 33,795–33,695 kaBP and ice-cover area expanding happened during 33,695–33,595 kaBP. The volume of the ice is reduced during the whole period of 33,795–33,595 kaBP, which suggests a warm period. This agrees well with the studies on the Oligocene/Miocene boundary (Naish *et al*. 2001). The experimental results show that the intersecting segments extracted by our algorithm are able to reveal clearly the spatial structure and position changes of the ice-cover surface, which provides a new idea and approach for geological and morphological analysis.

4.3. *Efficiency comparison*

To our best knowledge, there are no standard and open source change detection methods for 3D vector data yet. Since the triangle intersection is the key procedure of our method, we selected two triangle intersection methods that are most commonly used, the Möller scalar method (Möller 1997) and the Guigue–Devillers vector method (Guigue and Devillers 2003), for an efficiency comparison. To make a consistent comparison, we

Table 2. The intersecting results of DTINs with different algorithms.

Type	Method	Mean	Stdev	Median
No. of intersected segments	Our method	2976*	687	3027
	Guigue–Devillers	6493	1106	6534
	Möller	6359	1103	6413
No. of redundant intersected segments	Our method	1284	586	1349
	Guigue–Devillers	4801	1000	4856
	Möller	4667	998	4735
No. of available intersected segments	Our method	1692	128	1706
	Guigue–Devillers	1692	128	1706
	Möller	1692	128	1706
Running time (second)	Our method	17.5	3.1	17.5
	Guigue–Devillers	11.4	2.2	11.4
	Möller	22.7	4.3	22.7

Notes: *The real average number is 2975.47. Since the number of segments should be integer, we transformed it into 2976 to keep the sum of the number of redundant intersected segments and the number of available intersected segments equal to the number of intersected segments.

simulated the DTIN intersection operation in our method using the Möller scalar method, the Guigue–Devillers vector method, and the *meet* operator-based method, respectively. Comparing the numbers of the intersecting segments and the running times of the three methods (Table 2), we were able to test and verify the accuracy and efficiency of the method.

From Table 1, the numbers of the available intersecting segments of the three algorithms are the same, which suggests that our method produces the same accurate results as the Möller and Guigue–Devillers methods do. The total number of intersecting segments of our algorithm was more than 50% lower than that of the two referred algorithms, which suggests that our method produces much less unnecessary judgments during the computation. This is partly because the intersection in our method is multidimensionally unified and partly because the symmetry of the *outer* product makes the triangles oriented. In our method, the intersections between segments, triangles, etc., are unified and calculated only once. However, in the two referred methods, they should be calculated separately. The results are also supported by the number of redundant intersecting segments during the calculation. In our method, the number of mean redundant intersected segments is reduced to one fifth of that of the two referred methods.

As for the computing efficiency, our method is better than the Möller method, but worse than Guigue–Devillers method. This is because the *meet* operation we used is an alternative version that is still not fully optimized. The original *meet* operation in the CGA, which is optimized by binary operations (Fontijne 2006), is inappropriate for the intersection of segments with boundaries (i.e. it can only produce the intersection between lines). The *meet* operation we use has additional computation cost in dealing with the boundary conditions (Yuan *et al*. 2012). The optimization of the intersection in parallel computation and of the *meet* product with logic-bit operation can largely improve the efficiency. In addition, we can further improve the computation efficiency using the precompile technology with automatic code optimization tools like Gaalop and Gaalop Precompiler (Hildenbrand 2013, Charrier *et al*. 2014).

There are several special conditions in the intersection computation. For example, the two corresponding triangles may be tangent to each other; the triangle may degenerate into a segment. Both the Guigue–Devillers and the Möller methods choose artificial intersecting rules to distinguish these conditions, which makes the intersecting algorithm more complicated. For example, topological relations between different triangles should be calculated to distinguish the redundant segment intersections or the triangles with one common edge (Möller 1997, Guigue and Devillers 2003). In our algorithm, the anti-symmetric and directionally oriented properties of the *outer* product largely reduced the need of such judgments. The algorithm proposed in this paper not only can set intersecting rules flexibly according to application needs but is also self-adaptive in computing the intersection relations in dynamic scenes. The case study based on the analog data of the Antarctic ice-cover evolution shows that our algorithm is concise and efficient and may even provide references for expressing and analyzing complex geometric objects in the geometrically meaningful and multidimensionally unified way.

5. Discussion

5.1. Possibility to support parallel computation

Vector data involve many basic elements such as points, lines, planes, and spheres. Unlike point cloud or raster data, the different elements in vector data do not have definite orders.

Thus, vector data cannot be processed in a definite sequence, which is the major obstacle for designing a unified framework for parallel algorithms. For these reasons, the change detection for 3D vector data develops very slowly, which to some degree influences the development, application, and popularization of GIS change detection algorithms.

The CGA has the inherited ability to support the parallel computation. In our method, the expression of a DTIN and the operations of the DTIN are all multi-dimensionally unified. The computation of intersections is independent of different objects, thus providing a very suitable way to design parallel computation. For example, the sphere–sphere intersection detection procedures in our method, which are obviously order independent at each level, can be simply implemented in parallel computation. Here, we tested the performance of the sphere–sphere intersection detection procedures with the sequential computation by a CPU and parallel computation by a GPU. The GPU code of the sphere–sphere intersection is implemented in OpenCL using the Gaalop Precomplier and the Intel OpenCL SDK. Since the change of area varies at different situations, the intersections between the two different groups of spheres may require to be intersected with only the corresponding sphere (i.e. in the case where the changes are very small) or intersected with nearly all the other spheres (i.e. in the case where the changes are very large). Therefore, we demonstrate two different extreme experiments. In the first experiment, we simulate two groups of spheres, with each group randomly generating a constant number of spheres. Each sphere is judged with any other spheres. In the second experiment, the spheres are only judged with the corresponding sphere. Due to the memory limits, in the first case, the number of spheres in each sphere group is changing from 1 to 50 K; while in the second experiment, the number of spheres in each group is changing from 1 to 30 M. The time costs of both the CPU and GPU implementations according to the different number of spheres are logged in Figure 9. Clearly, the GPU parallel intersection detecting algorithm improves the efficiency greatly; and the larger the number of spheres is, the higher ratio of efficiency improvement will be. Even in the worst case where each sphere is intersected with any other spheres, the computational time is less than 24 seconds using the GPU with 50 K spheres in each group. If well-defined algorithms that can pre-define the correspondences between spheres and the sphere numbers are not changed in the change detection, the efficiency can be largely improved. As shown in Figure 9 (b), the intersection time cost is less than 8 seconds using the GPU with 30 M spheres in each group. The result indicates that our method has large potential to support the large-scale data analysis. Although only the sphere–sphere intersection is tested here, the intersections between triangles also have the possibility to be parallelized (Hildenbrand 2013).

5.2. Directions for future improvements

Vector data change detection is much more complicated compared to images or point cloud data change detection. In our method, both of the geometric structure and the topological relations between elements are considered. The *outer* product representation of the DTIN logs the topological relations. Due to the asymmetric properties of the *outer* products, the topological changes can be easily identified directly with the sign of the *outer* products. Since the data are organized as a hierarchical object-based structure, the identification of the attribution change can also be simply compared with a hierarchical search index. All these implementations can be direct and simple, and do not require frequent data updates and complex spatial analysis.

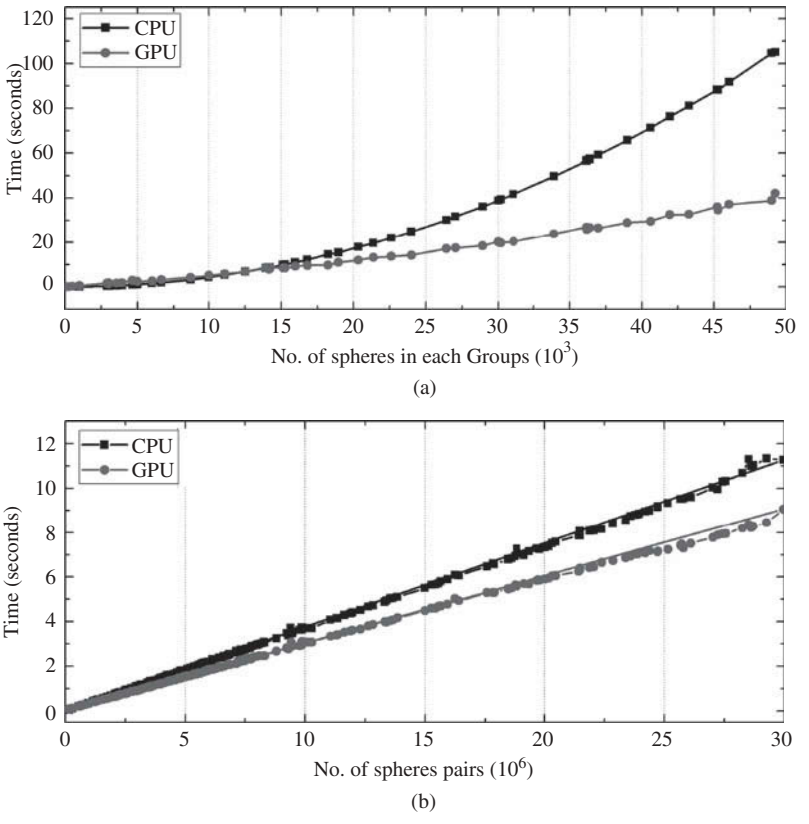

Figure 9. Comparison of the efficiency of GPU- and CPU-based algorithm in CGA. (a) Time cost of the first experiment (intersection between any other spheres). (b) Time cost of the second experiment (intersection between only corresponding spheres).

The change detection in the spherical coordinates for global scale objects and phenomena will be of great importance for geography and associated discipline research. Expressing geometric objects in the CGA framework can not only reflect their intrinsic geometric features but also obtain coordinate-independent geometric relations of objects. The inherent support from Euclidean, homogeneous, and spherical space provides an ideal tool for coordinate-independent and dimension-independent change detection across different scales. In the CGA representation, we can compute topological relations of geometric objects by direct operation computing because the CGA multivector expressions contain semantic information such as direction, position, and value in the CGA expression. This provides a beneficial precondition for self-adaptive expression, relation judgment, and relation computation for geometric objects.

6. Conclusions

This paper presents a multidimensional unified change detection method for 3D vector data. The multidimensionally unified DTIN representation, intersection detection, change of area/volume extraction, and topological reconstruction are developed with CGA

operations. The method can clearly reveal the dynamic changes of the complex multidimensional vector data. Since both geometry and topology are considered in the multidimensionally unified change detection, it provides the possibility to construct a universal change detection method for vector data.

Acknowledgements

We thank anonymous reviewers and Mr. Mingsong Xu for their helpful comments.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This study was supported by the National Natural Science Foundation of China [grant number 41231173], [grant number 41471319], [grant number 41431177]; the NCET program of the Ministry of Education of the People's Republic of China [grant number NCET-12-0735]; and the Priority Academic Program Development of Jiangsu Higher Education Institutions. The support received by A-Xing Zhu through the Vilas Associate Award, the Hammel Faculty Fellow, and through the 'One-Thousand Talents' Program of China is greatly appreciated.

References

Agouris, P., Stefanidis, A., and Gyftakis, S., 2001. Differential snakes for change detection in road segments. *Photogrammetric Engineering and Remote Sensing*, 67 (12), 1391–1399.

Aguirre-Gutierrez, J., Seijmonsbergen, A.C., and Duivenvoorden, J.F., 2012. Optimizing land cover classification accuracy for change detection, a combined pixel-based and object-based approach in a mountainous area in Mexico. *Applied Geography*, 34, 29–37. doi:10.1016/j.apgeog.2011.10.010

Berlanga-Robles, C.A. and Ruiz-Luna, A., 2011. Integrating remote sensing techniques, Geographical Information Systems (GIS), and stochastic models for monitoring land use and land cover (LULC) changes in the northern coastal region of Nayarit, Mexico. *GIScience & Remote Sensing*, 48 (2), 245–263. doi:10.2747/1548-1603.48.2.245

Charrier, P., *et al*., 2014. Geometric algebra enhanced precompiler for C++, OpenCL and Mathematica's OpenCLLink. *Advances in Applied Clifford Algebras*, 24, 613–630. doi:10.1007/s00006-014-0443-7

De Chant, T. and Kelly, M., 2009. Individual object change detection for monitoring the impact of a forest pathogen on a hardwood forest. *Photogrammetric Engineering & Remote Sensing*, 75 (8), 1005–1013. doi:10.14358/PERS.75.8.1005

Deconto, R. and Pollard, D., 2003. Rapid Cenozoic glaciation of Antarctica induced by declining atmospheric CO_2. *Nature*, 421, 245–249. doi:10.1038/nature01290

Domiter, V. and Žalik, B., 2008. Sweep-line algorithm for constrained Delaunay triangulation. *International Journal of Geographical Information Science*, 22 (4), 449–462. doi:10.1080/13658810701492241

Döner, F., *et al*., 2011. Solutions for 4D cadastre – with a case study on utility networks. *International Journal of Geographical Information Science*, 25 (7), 1173–1189. doi:10.1080/13658816.2010.520272

Fontijne, D., 2006. Gaigen 2: a geometric algebra implementation generator. *In*: S. Jarzabek, D. Schmidt, and T. Veldhuizen, eds., *5th international conference on generative programming and component engineering*, 22–26 October. New York: ACM Press, 141–150. doi:10.1145/1173706.1173728

Frankel, A., Nussbaum, D., and Sack, J.R., 2004. Floating-point filter for the line intersection algorithm. *In*: M.J. Egenhofer, C. Freksa, and H.J. Miller, eds. *Geographic information science, proceedings*. Heidelberg: Springer, 94–105.

Frey, D., Butenuth, M., and Straub, D., 2012. Probabilistic graphical models for flood state detection of roads combining imagery and DEM. *IEEE Geoscience and Remote Sensing Letters*, 9 (6), 1051–1055. doi:10.1109/LGRS.2012.2188881

Guigue, P. and Devillers, O., 2003. Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of Graphics Tools*, 8 (1), 25–42.

Hildenbrand, D., 2013. *Foundations of geometric algebra computing*. Heidelberg: Springer.

Hitzer, E., 2005. Conic sections and meet intersections in geometric algebra. *In*: H. Li, P.J. Olver, and G. Sommer, eds. *Computer algebra and geometric algebra with applications*. Berlin: Springer, 350–362.

Hitzer, E., Nitta, T., and Kuroe, Y., 2013. Applications of Clifford's geometric algebra. *Advances in Applied Clifford Algebras*, 23 (2), 377–404.

Ledoux, H. and Gold, C.M., 2008. Modelling three-dimensional geoscientific fields with the Voronoi diagram and its dual. *International Journal of Geographical Information Science*, 22 (5), 547–574.

Metternicht, G., 2001. Assessing temporal and spatial changes of salinity using fuzzy logic, remote sensing and GIS. Foundations of an expert system. *Ecological Modelling*, 144 (2–3), 163–179.

Mi, H. and Geng, X., 2013. Research on the management of large-scale terrain data. *Journal of Theoretical and Applied Information Technology*, 49 (2), 803–806.

Möller, T., 1997. A fast triangle–triangle intersection test. *Journal of Graphics Tools*, 2, 25–30.

Naish, T., *et al*., 2001. Orbitally induced oscillations in the East Antarctic ice sheet at the Oligocene/Miocene boundary. *Nature*, 413, 719–723.

Navratil, G., 2008. 3D cadastre in an international context. *International Journal of Geographical Information Science*, 22 (9), 1033–1034.

Qin, R. and Gruen, A., 2014. 3D change detection at street level using mobile laser scanning point clouds and terrestrial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90, 23–35.

Rivera-Rovelo, J. and Bayro-Corrochano, E., 2007. Medical image segmentation, volume representation and registration using spheres in the geometric algebra framework. *Pattern Recognition*, 40 (1), 171–188.

Roa, E., *et al*., 2011. GPU collision detection in conformal geometric space. ed. *V Ibero-American Symposium in Computer Graphics SIACG*, 2011, 153–157.

Rokni, K., *et al*., 2014. Water feature extraction and change detection using multitemporal Landsat imagery. *Remote Sensing*, 6 (5), 4173–4189.

Siejka, M., Slusarski, M., and Zygmunt, M., 2014. 3D+time Cadastre, possibility of implementation in Poland. *Survey Review*, 46 (335), 79–89.

Walter, V., 2004. Object-based classification of remote sensing data for change detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 58 (3–4), 225–238.

Wang, Y.F., *et al*., 2013. Parallel scanline algorithm for rapid rasterization of vector geographic data. *Computers & Geosciences*, 59, 31–40.

Williams, T., 2014. Climate science how Antarctic ice retreats. *Nature*, 510 (7503), 39–40.

Wu, H.Y., Guan, X.F., and Gong, J.Y., 2011. ParaStream: a parallel streaming Delaunay triangulation algorithm for LiDAR points on multicore architectures. *Computers & Geosciences*, 37 (9), 1355–1363.

Yu, Z.-Y., *et al*., 2012. Boundary restricted non-overlapping sphere tree for unified multidimensional solid object index. *Ruan Jian Xue Bao/Journal of Software*, 23 (10), 2746–2759.

Yu, Z.-Y., *et al*., 2015. Geometric algebra model for geometry-oriented topological relation computation. *Transactions In GIS*, in press. doi:10.1111/tgis.12154

Yuan, L., *et al*., 2010. CAUSTA: Clifford algebra-based unified spatio-temporal analysis. *Transactions in GIS*, 14 (S1), 59–83.

Yuan, L., *et al*., 2011. A 3D GIS spatial data model based on conformal geometric algebra. *Science China Earth Sciences*, 54 (1), 101–112.

Yuan, L., *et al*., 2012. Geometric algebra method for multidimensionally-unified GIS computation. *Chinese Science Bulletin*, 57 (7), 802–811.

Yuan, L., *et al*., 2013. Geometric algebra for multidimension-unified geographical information system. *Advances in Applied Clifford Algebras*, 23 (2), 497–518.

Yuan, L., *et al*., 2014. Multidimensional-unified topological relations computation: a hierarchical geometric algebra-based approach. *International Journal of Geographical Information Science*, 28 (12), 2435–2455.

Zhang, S.Q. and Zhang, J.Y., 2010. Theoretical analytics of stereographic projection on 3D objects' intersection predicate. *International Journal of Geographical Information Science*, 24 (1), 25–46.